Consistency driven Sequential Transformers Attention Model
for Partially Observable Scenes

Samrudhdhi B. Rangreja, Chetan L. Srinidhi, James J. Clark
aMcGill University, Canada. bSunnybrook Research Institute, University of Toronto, Canada.
samrudhdhi.rangrej@mail.mcgill.ca, chetan.srinidhi@utoronto.ca, james.clark1@mcgill.ca

Abstract

Most hard attention models initially observe a complete scene to locate and sense informative glimpses, and predict class-label of a scene based on glimpses. However, in many applications (e.g., aerial imaging), observing an entire scene is not always feasible due to the limited time and resources available for acquisition. In this paper, we develop a Sequential Transformers Attention Model (STAM) that only partially observes a complete image and predicts informative glimpse locations solely based on past glimpses. We design our agent using DeiT-distilled [44] and train it with a one-step actor-critic algorithm. Furthermore, to improve classification performance, we introduce a novel training objective, which enforces consistency between the class distribution predicted by a teacher model from a complete image and the class distribution predicted by our agent using glimpses. When the agent senses only 4% of the total image area, the inclusion of the proposed consistency loss in our training objective yields 3% and 8% higher accuracy on ImageNet and fMoW datasets, respectively. Moreover, our agent outperforms previous state-of-the-art by observing nearly 27% and 42% fewer pixels in glimpses on ImageNet and fMoW.

1. Introduction

High-performing image classification models such as EfficientNet [42], ResNet [15] and Vision Transformers (ViT) [13] assume that a complete scene (or image) is available for recognition. However, in many practical scenarios, a complete image is not always available at once. For instance, an autonomous agent may acquire an image only partially and through a series of narrow observations. The reasons may include a small field of view, high acquisition cost, limited time for acquisition, or limited bandwidth between the sensor and the computational unit. Often, an agent would have partially acquired an image, and the system must perform recognition based on incomplete information. Models trained on complete images prove inefficient while classifying incomplete images. For example, the accuracy of DeiT-Small [44] drops by around 10% when 50% of the image regions are unavailable [25]. Moreover, they cannot perform autonomous sensing.

Many developed autonomous agents that acquire a series of most informative sub-regions from a scene to perform classification from partial observations [5, 14, 24, 27]. Most existing scalable approaches [14, 27, 45] initially glance at an entire scene to locate the informative sub-regions. However, in practice, glancing at an entire scene is not always feasible. Examples include time-sensitive rescue operations using aerial imagery, an autonomous car driving in a new territory, and a medical expert probing a tissue to find abnormalities. We develop an autonomous agent that predicts locations of the most informative regions, called glimpses, without observing the entire scene initially. Starting from observing a glimpse at a random location, the autonomous agent decides which location to attend next solely based on the partial observations made so far.

We design our autonomous agent using a transformer architecture [13, 44, 47] and call it Sequential Transformers Attention Model (STAM). Transformers efficiently model long-range dependencies and are ideal for aggregating information from distant glimpses. At any given time, our
agent predicts an optimal location for the next glimpse and class-label of an image based on the glimpses collected so far. As glimpse acquisition is a discrete and non-differentiable process, we train our agent using reinforcement learning (RL). Further, we propose an additional training objective where the agent is required to predict a class distribution from a set of glimpses consistent with the class distribution predicted from a complete image. To do so, we employ a teacher transformers model to predict the class distribution from a complete image and our agent (a student model) tries to reproduce this distribution using partial observations. We perform experiments on two large-scale real-world datasets, namely, ImageNet [34] and fMoW [10].

Our main contributions are as follows.

1. We develop a transformers-based RL agent called STAM, which actively senses glimpses from a scene and predicts class-label based on partial observations. Instead of locating informative glimpses by observing an entire image, our agent sequentially predicts the next most informative glimpse location based on past glimpses.

2. We propose a consistency-based training objective, where the agent must predict a class distribution consistent with the complete image using only partial observations. With only 4% of the total image area observed, our proposed objective yields \( \sim 3\% \) and \( \sim 8\% \) gain in accuracy on ImageNet and fMoW, respectively.

3. Our agent that never observes a complete image outperforms previous methods that initially glance at an entire image to locate informative glimpses. It starts exceeding the previous state-of-the-art while sensing 27% and 42% fewer pixels in glimpses on ImageNet and fMoW, respectively.

2. Related Works

**Hard Attention.** As opposed to soft attention [52], which attends to all regions of an image but with a varying degree of importance, hard attention [24] sequentially attends to only the most informative subregions in an image. Hard attention was first introduced by Minh et al. [24] and later studied by many others. Different techniques are used for hard attention such as expectation maximization [30], majority voting [1], wake-sleep algorithm [6], sampling from self-attention or certainty maps [37, 38], and Bayesian optimal experiment design [29]. The most successful hard attention models learn to sample glimpses using policy gradient RL algorithms [5, 14, 24, 27, 48, 52].

Most previous hard attention models initially glance at a complete image to locate the most informative glimpses. For instance, Xu et al. [52] and Saccader [14] analyze the complete image at original resolution; whereas DRAM [6], TNet [27] and GFNet [48] observes the complete image at low resolution. Furthermore, TNet [27] and GFNet [48] uses the low resolution gist of an image to predict the class-label. In contrast, our model does not look at the entire image at low resolution or otherwise. We predict the attention-worthy glimpse locations and the class of the complete image solely based on partial observations. From this perspective, RAM [24], which also operates under partial observability, is the closest related approach. While RAM could not scale beyond MNIST dataset [21], our approach scales to large-scale real-world datasets.

**Patch Selection.** Many approaches observe an entire image to select all informative sub-regions at once. For example, region proposal networks [33], top-K patch selection [2, 11], multiple-instance learning [18], attention sampling [19] and PatchDrop [45]. Unlike these approaches, our model does not observe the entire image, and it predicts the location of informative sub-regions sequentially. Among vision transformers, methods such as PS-ViT [54], Dynamic-ViT [31] and IA-RED [26] start with observing a complete image and progressively (re-)sample most discriminative patches from each successive transformer block. In contrast, we sample and input only informative patches to the transformers. Moreover, our model is sequential in nature, sensing only one additional patch at each step.

**Consistency Learning.** The idea of consistency learning was initially proposed by Sajjadi et al. [36] and has become an important component in many recent semi-supervised learning (SSL) algorithms [8, 20, 39, 50]. Consistency learning acts as a regularizer that enforces the model output to be invariant to the perturbations in the input image [8, 36, 39, 50] or hidden state [7, 20] or model parameters [17, 40]. The consistency is achieved by using the predictions made from one perturbation as pseudo-targets for the predictions made from another perturbation.

Another closely related idea in SSL is the pseudo-labeling [4, 22, 28], where a trained model, known as ‘teacher model’, is used to generate soft (continuous distributions) or hard (one-hot distributions) pseudo-labels for the unlabeled data. These pseudo-labels are later used as targets while training a student model with unlabeled examples [9, 51]. This approach is closely linked to Knowledge Distillation [9, 16], where the student is trained to reproduce the teacher output.

In this work, we develop a consistency training objective based on these concepts. We train our agent to be invariant to a specific type of input perturbations, i.e., the partial and the complete observations. Furthermore, we use a teacher model to produce soft pseudo-labels from a complete image and use them as targets while training our agent (a student model) using partial observations.
3. Sequential Transformers Attention Model (STAM)

Given an unobserved scene $X$, the agent actively captures a series of non-overlapping glimpses and, while never observing $X$ completely, it predicts the class-label $y$ of $X$ based on glimpses. A schematic diagram of our agent is shown in Figure 1. At time $t$, the agent senses a glimpse $g_t$ at location $l_t$ from an image $X$. Using the glimpses observed up to time $t$, our agent predicts: i) $y_t$, an approximation of label $y$, and ii) $l_{t+1}$, the location of the next glimpse.

We model the sequential attention mechanism of our agent as a Partially Observable Markov Decision Process (POMDP). In the POMDP, our agent encodes a history of partial observations, \{$(g_0, l_0), \ldots, (g_t, l_t)$\}, in a Markov state $s_t$ and maps it to: i) a class distribution $p(y_t|s_t)$ and ii) an attention policy $\pi(l_{t+1}|s_t)$ - a distribution over candidate glimpse locations for $t+1$.

### 3.1. Model of Our Agent

We build our agent using DeiT-distilled [44], referred to as DeiT$^D$ in the rest of the paper. Briefly, DeiT$^D$ is a type of ViT trained using knowledge distillation. The transformers in DeiT$^D$ transform an input sequence of a class token, a distillation token, and patch tokens (linear projections of the image patches added to the positional embeddings) to an output sequence; with the outputs corresponding to the class and the distillation tokens, the two classifiers predict the ground truth and the teacher’s prediction, respectively. In our approach, we adapt the DeiT$^D$ to predict labels from glimpses and use the distillation token to impose consistency. Figure 2 shows the model of our agent. Our agent is composed of the following components.

**Sensor.** We consider a sensor that captures non-overlapping glimpses from a scene. To model this sensor, we divide the image $X$ into $N \times N$ equally sized non-overlapping blocks, $X = \{X(i, j) | i, j \in \{1, \ldots, N\}\}$. Given a location $l_t = (i, j)$, a sensor senses a glimpse $g_t = X(i, j)$, as shown in Figure 1.

**Core ($T$).** At time $t$, we extract $M \times M$ patches from each glimpse observed up to $t$, forming a set of $t \times M \times M$ patches. We feed these patches, the positional embeddings, the class token, and the distillation token to the DeiT$^D$ model. The positional embedding represents the position of a patch in an image. We derive the position of a patch from the location of a parent glimpse in an image.
Algorithm 1 Inference using STAM

1: Initialize \( l_0 \) randomly;
2: for \( t \in \{0, \ldots, T-1\} \) do
3: \( f^a_t, f^d_t = \mathcal{T}(y_{t-1}; s_t) \); \( s_t = [f^a_t; f^d_t] \) \( \triangleright \) Sensor
4: \( p_y(y_t|s_t) = G(f^a_t); \quad p_d(y_t|s_t) = D(f^d_t) \) \( \triangleright \) Classifiers
5: \( \pi'(t') = A(s_t, t'), \forall t' \in \{1, \ldots, N\}^2 - l_{t-1} \) \( \triangleright \) Actor
6: \( y_t = \text{argmax} \{p_y(y_t|s_t) + p_d(y_t|s_t)\} \)
7: \( l_{t+1} = \text{argmax} (\pi'(t')) \)
8: end for

Among the outputs of the final transformer block, let us define the ones corresponding to the class token as \( f^d_t \) and the distillation token as \( f^a_t \). We then form a Markov state \( s_t \) by concatenating \( f^a_t \) and \( f^d_t \), which an actor module will later use to predict an attention policy.

Classifiers \((G \text{ and } D)\). As in DeiT, we use two linear classifiers to predict two class distributions \( p_y(y|f^a_t) \) and \( p_d(y|f^d_t) \) from \( f^a_t \) and \( f^d_t \), respectively. We treat the predicted distributions independently during training and average them to form an ensemble distribution during inference [44]:

\[
p_y(y_t|s_t) = \frac{1}{2} \{p_y(y_t|f^a_t) + p_d(y_t|f^d_t)\}.
\]

Actor \((A)\). An actor MLP predicts attention policy \( \pi(l_{t+1}|s_t) \). The distribution \( \pi(l_{t+1}|s_t) \) is computed by applying softmax over logs \( \pi'((i,j)|s_t) \), where \((i,j)\)'s are unobserved glimpse locations. An actor predicts \( \pi'(i,j)|s_t \) for each \((i,j)\) in a conditionally independent manner [3, 43]. For any \((i,j)\), the actor accepts a concatenation of glimpse location embeddings \( e(i,j) \) and a Markov state \( s_t \), and outputs \( \pi'(i,j)|s_t \). Here, \( e(i,j)\)'s are learnable embeddings initialized by interpolating positional embeddings of a pre-trained DeiT. We use \( l_{t+1} \sim \pi(l_{t+1}|s_t) \) during training and \( l_{t+1} = \text{argmax}(\pi(l_{t+1}|s_t)) \) during inference.

We provide the inference steps in Algorithm 1.

4. Training Objectives

We train the parameters of the core \((\theta_T)\), the classifiers \((\theta_C \text{ and } \theta_D)\) and the actor \((\theta_A)\) using the training objectives discussed next. Figure 2 illustrates the training steps of our model, and Algorithm 1 in the supplementary material presents the corresponding pseudocode.

4.1. Learning Classification

Our agent predicts two class distributions based on input glimpses, namely, \( p_y \) and \( p_d \); where, \( p_y \) is an estimation of the ground truth class distribution associated with a complete image and \( p_d \) is an approximation of the class distribution predicted by a teacher model from a complete image. We learn \( p_y \) and \( p_d \) using the following two objectives:

**Supervised Loss.** As our goal is to predict \( y \) from partial observations, we learn the parameters \( \{\theta_T, \theta_C\} \) by minimiz-

\[
\mathcal{L}_{\text{sup}} = - \sum \delta(y|X) \log(p_y(y|s_t)),
\]

where, \( \delta(y|X) \) is a delta distribution indicating the ground truth label of a complete image.

**Consistency Loss.** To improve the performance of our agent, we enforce that the predictions made from the glimpses are consistent with the predictions made from a complete image. Furthermore, the above predictions should also be the same irrespective of the number and location of the glimpses observed so far. \textit{Ideally}, for each \( t \), we require our agent to produce \( p_d(y_t|s_t) \) that minimize \( \text{KL} [p_d(y_t|s_t)||p(y|X)] \); where \( p(y|X) \) is the agent’s prediction after observing all glimpses from an image.

The direct optimization of the above KL divergence is difficult as the target \( p(y|X) \) keeps shifting during training. To circumvent this issue, we rely on a separate teacher model to provide a stable target. Our teacher model predicts the class distribution \( q(y|X) \) from a complete image; where \( q(y|X) \) is commonly referred to as soft pseudo-label for \( X \) in the literature [16, 51]. The resultant consistency objective to train \( \{\theta_T, \theta_D\} \) is given by

\[
\mathcal{L}_{\text{consist}} = \text{KL} [p_d(y_t|s_t)||q(y|X)].
\]

4.2. Learning Attention Policy

We consider attention to be a POMDP. After observing a glimpse at location \( l_{t+1} \sim \pi(l_{t+1}|s_t) \), we award our agent a reward \( R_{t+1} \) indicating the utility of the observed glimpse. Our training objective is to learn \( \pi(l_{t+1}|s_t) \) that maximizes the sum of future rewards, also known as return, \( G_t = \sum_{t'=t+1}^{T} (R_{t'}) \). A majority of the existing works [5, 14, 24, 27] use REINFORCE algorithm [49] to learn an attention policy. These methods run an agent for \( t = 0 \) to \( T - 1 \) steps to achieve \( R_t \) to \( R_T \) and compute \( G_t \) to \( G_{T-1} \).

At the end, the parameters of the agent are updated once to maximize the returns. Due to the quadratic complexity of the transformers, running our agent for \( T \) steps and updating the parameters just once at the end is expensive. Instead, we adopt one-step actor-critic algorithm [41] to update the parameters at each time step.

**Critic loss.** To train our agent using the one-step actor-critic algorithm, we introduce a critic MLP \((C)\) with parameters \( v \). A critic learns a value function \( V(s_t) \) that estimates the expected return given the current state of the agent, i.e., \( \mathbb{E}_v[G_t] \). As \( \mathbb{E}_v[G_t] = \mathbb{E}_v[R_{t+1} + G_{t+1}] \), \( V(s_t) \) should be equal to \( \mathbb{E}_v[R_{t+1} + V(s_{t+1})] \). Hence, the critic parameters \( v \) are learned by minimizing the difference between the two quantities. In practice, we estimate the expectation with respect to \( \pi \) using a single Monte-Carlo sample, yielding

\[
\mathcal{L}_{\text{critic}} = ||V(s_t) - (R_{t+1} + V(s_{t+1}))||.
\]
We run our agent for one additional time step to compute \( V(s_{t+1}) \). Note that the quantity \( (R_{t+1} + V(s_{t+1})) \) acts as a target and does not contribute to the parameter update. We use the critic MLP only during training and discard it once the training is over.

**Actor loss.** The goal of an agent is to learn a policy that achieves the maximum return. When the agent achieves lower than the expected return by sensing a glimpse at location \( l_{t+1} \), \( \pi(l_{t+1}|s_t) \) must reduce proportional to the deficit. In other words, \( \pi(l_{t+1}|s_t) \) must reduce by the factor of \( \frac{V(s_t) - (R_{t+1} + V(s_{t+1}))}{V(s_t)} \); where \( V(s_t) \) is an estimation of the expected return for \( s_t \), and \( (R_{t+1} + V(s_{t+1})) \) is the estimation of the expected return following glimpse at \( l_{t+1} \). We optimize the parameters \( \{\theta_T, \theta_A\} \) by minimizing

\[
\mathcal{L}_{actor} = \log(\pi(l_{t+1}|s_t))(V(s_t) - (R_{t+1} + V(s_{t+1}))).
\]

(Note that \( (V(s_t) - (R_{t+1} + V(s_{t+1}))) \) acts as a scaling factor and does not contribute to the parameter update.)

**Reward.** We use a reward that incentivizes the agent to predict \( y_t \) that is consistent with the label predicted by the teacher model based on a complete image. Our reward is

\[
R_t = -KL[p(y_t|s_t)\|q(y|X)],
\]

where \( p(y_t|s_t) \) is computed using Equation 1. We expect the accuracy of the predictions made from a complete image to provide an upper bound for the accuracy of the predictions made from partial observations. The above reward encourages the agent to reach for the upper bound. Our overall final training objective is as follows.

\[
\mathcal{L} = \frac{1}{2}(\mathcal{L}_{sup} + \mathcal{L}_{consist}) + (\mathcal{L}_{actor} + \mathcal{L}_{critic})
\]

5. **Experiment Setup**

**Datasets.** We experiment with two large-scale real-world datasets, namely, ImageNet [34] and IMoW [10]. ImageNet consists of natural images from 1000 categories. It includes \( \sim 1.3 \)M training images and 50K validation images. We resize the images to size \( 224 \times 224 \). The fMoW contains satellite images from 62 categories. It holds \( \sim 0.36 \)M, \( \sim 53K \), and \( \sim 64K \) images for training, validation, and test, respectively. We crop the images based on the bounding boxes provided with the dataset and resize the cropped images to \( 224 \times 224 \). Unless stated otherwise, we implement and optimize STAM with the same default setting on both datasets.

**Implementation.** We divide the images into non-overlapping glimpses of size \( 32 \times 32 \), yielding a \( 7 \times 7 \) grid

---

1. Our code is available at: https://github.com/samrudhdhirangrej/STAM-Sequential-Transformers-Attention-Model

2. https://github.com/facebookresearch/deit

---

We use DeiT\textsuperscript{D}-Small architecture for our agent unless stated otherwise. The actor and the critic MLPs are of the form \( \{3 \times \text{(FC-BN-ReLU)} - \text{FC}\} \) with hidden dimensions of 2048 and 512, respectively. We initialize the core and the classifiers using a pretrained DeiT\textsuperscript{D} and initialize the actor and the critic at random. We normalize the logits \( \pi'(\cdot) \) with \( l_2 \) norm for training stability and multiply them with \( \tau \) before applying a softmax. We stop the gradients from the critic to our agent. We normalize the rewards to have zero mean and unit variance in each training iteration. The magnitude of the value \( V(\cdot) \) varies from one time-step to the next, as it approximates the expected sum of future rewards. To learn \( V(\cdot) \) of varying magnitude, we apply PopArt-style normalization [46] to the predicted values.

**Optimization.** Our agent runs for \( T = 21 \) time-steps per image, capturing one glimpse at a time. We update the model parameters \( T \) times per batch, once after each time step. To account for \( T \) updates per batch, we allow the agent to see only \( 1/T^{th} \) of the data during one epoch. We train our agents with the batch size (\( B \)) of 4096 for 200 epochs on ImageNet and \( B \) of 600 for 400 epochs on fMoW. The hyperparameter \( \tau \) is increased linearly from 1 to 4 for the first 100 epochs and fixed to 4 for the remaining training.

We augment the training images using the RandAugment scheme [12] and follow the same setting as Touvron et al. [44]. Additionally, for fMoW, we also use random vertical flip augmentation. We train our agents using an AdamW optimizer [23] with a weight decay of 0.05. We adapt a cosine learning schedule with an initial learning rate of \( lr_{base} \times B/512 \) and a minimum learning rate of 1e-6. The base learning rate \( lr_{base} \) is set to 1e-3 for the critic module. For the remaining modules, \( lr_{base} \) is set to 1e-6 for ImageNet and 1e-5 for fMoW. We train our agents on four V100 GPUs in less than a day, using 32GB memory per GPU for ImageNet and 16GB for fMoW.

6. **Results**

6.1. **Comparison with Baseline Attention Policies**

We compare the policy learned by our agent with three baseline policies, namely, the Random, the Plus, and the Spiral. The Random agent selects the next glimpse ran-
6.2. Analysis of Consistency Loss

To quantify the gain achieved with a consistency loss from Equation 3, we compare our agents trained with and without this loss. For a fair comparison, when training our agent without the consistency loss, we evaluate the cross-entropy loss between the ensemble distribution $p(y_t|s_t)$ from Equation 1 and the ground truth. The remaining training setup is the same for both agents.

Additional results in the supplementary material (SM). In Section 1.2 of SM, we show that when the area observed in an image is $< 20\%$, STAM achieves higher accuracy with smaller glimpses than the larger ones. Also, in Section 1.3 of SM, we show that an increase in model capacity can improve the performance of STAM. In Section 1.4 of SM, we show that longer training on ImageNet also improves STAM’s performance further.
6.3. Glimpse Visualization

In Figure 5, we display the glimpses selected by STAM and the predicted labels on example images from ImageNet and fMoW. In the ImageNet example, STAM locates and identifies the umbrella at \( t = 14 \). In the fMoW example, STAM locates the transmission line and identifies the powerplant at \( t = 8 \). Note that, while not observing a complete image, STAM predicts the location of informative glimpses solely based on past glimpses. For more examples, refer to Figures 5-6 in the supplementary material.

In Figure 6, we display the histograms of glimpse locations selected by our agent with increasing \( t \). At \( t = 0 \), the agent observes a glimpse at a random location, and at \( t = 1 \), the agent learns to observe mainly a glimpse centered on an image, perhaps due to the object-centric nature of the dataset. For the subsequent glimpses, the agent prefers to attend vertically and horizontally centered glimpses in ImageNet. While for fMoW, it attends to glimpses with minimum distance from the center. Note in ImageNet the object of interest frequently appears at the center and is aligned vertically or horizontally; whereas in fMoW, the object of interest appears at the center but with no specific orientation. With time, the agent attends to different locations away from the center based on the content observed through previous glimpses as shown in Figure 5.

6.4. State-of-the-Art Comparison

A fair comparison between our method and the previous works is challenging due to the following reasons. Most previous works observe an entire image, occasionally at low resolution, to locate the most informative glimpses \([5, 14, 27, 45, 48]\). Furthermore, if they observe an entire image at low resolution, they optionally use the image along with the glimpses to predict the class-label \([27, 45, 48]\). In contrast, our agent operates only under partial observability. We present the state-of-the-art comparison in Table 1 and indicate which method uses an entire image and for what reasons. As different methods use different glimpse sizes, we compare them based on the number of pixels sensed per image for classification. If the method senses a complete image but does not use it for classification \([5, 14]\), we do not
partially

<table>
<thead>
<tr>
<th>Method</th>
<th>Note</th>
<th>Is complete image used ?</th>
<th>Attention ?</th>
<th>Classification ?</th>
<th>#pixels for classification</th>
<th>Accuracy (%)</th>
<th>#pixels for classification</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM [5]</td>
<td>results from [27]</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GFNet [48]</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saccader [14]</td>
<td>results from [27]</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNet [27]</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STN [32, 45]</td>
<td>developed in [45]</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PatchDrop [45]</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAM (DeiT&lt;sup&gt;D&lt;/sup&gt;-Small (default))</td>
<td>*equivalent</td>
<td>No</td>
<td>No</td>
<td>20.5K&lt;sup&gt;(t = 19)&lt;/sup&gt;</td>
<td>76.35</td>
<td>11.3K&lt;sup&gt;(t = 10)&lt;/sup&gt;</td>
<td>68.8</td>
<td></td>
</tr>
<tr>
<td>STAM (DeiT&lt;sup&gt;D&lt;/sup&gt;-Base)</td>
<td>accuracy to [45]</td>
<td>No</td>
<td>No</td>
<td>14.3K&lt;sup&gt;(t = 13)&lt;/sup&gt;</td>
<td>76.13</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>STAM (DeiT&lt;sup&gt;D&lt;/sup&gt;-Small (default))</td>
<td>*equivalent</td>
<td>No</td>
<td>No</td>
<td>27.7K&lt;sup&gt;(t = 26)&lt;/sup&gt;</td>
<td>78.25</td>
<td>19.5K&lt;sup&gt;(t = 18)&lt;/sup&gt;</td>
<td>71.5</td>
<td></td>
</tr>
<tr>
<td>STAM (DeiT&lt;sup&gt;D&lt;/sup&gt;-Base)</td>
<td>sensing to [45]</td>
<td>No</td>
<td>No</td>
<td>27.7K&lt;sup&gt;(t = 26)&lt;/sup&gt;</td>
<td>80.78</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. State-of-the-Art comparison. We report the number of pixels sensed per image for classification and the resultant accuracy. If a method uses a low-resolution gist of a complete image for classification, we include the pixels of the gist in the above count. Our results are presented as an average computed over ten runs. We present our results at two different time steps, (*first two rows) when the accuracy achieved by our method is equivalent to PatchDrop, and (**last two rows) when the number of pixels sensed by our agent for classification is equivalent to PatchDrop. †TNNet [27] use 896 × 896 resolution images; hence, we do not include their results in the above comparison.

To achieve the same level of accuracy as PatchDrop [45] (the best performing method), our default agent observes 7.4K fewer pixels per image from ImageNet and 8.1K fewer pixels per image from fMoW. Moreover, while observing a similar number of pixels as PatchDrop, our default agent achieves 2.25% higher accuracy on ImageNet and 3.2% higher accuracy on fMoW. We also train our agent with DeiT<sup>D</sup>-Base as the core module on ImageNet. This agent requires 13.6K fewer pixels per image to achieve the same accuracy as PatchDrop, and achieves 4.78% higher accuracy than PatchDrop while sensing the same number of pixels per image. We emphasize that our agent does not observe a complete image to locate informative glimpses or to perform classification, whereas PatchDrop does.

6.5. Early Termination

In practice, we can save time and resources by terminating sensing when the agent confidently concludes a class for the image. To this end, we devise a simple mechanism to decide when to stop sensing. Let us define a scoring function based on the probability of the predicted class, \(S_t = max(p(y_t|s_t))\). The agent stops sensing more glimpses if \(S_t\) is greater than threshold \(\gamma\). In Figure 7, we show the average number of glimpses observed per image vs. the accuracy of our agent for \(\gamma\) varying from 0 to 1. Remarkably, with \(\gamma = 0.5\), STAM suspends sensing on ImageNet after observing on an average 7.5 glimpses per image and achieves 66.47% accuracy. Similarly, on fMoW with \(\gamma = 0.5\), STAM suspends sensing after observing on an average 8.7 glimpses per image and achieves 65.71% accuracy.

7. Discussion and Conclusions

We introduced a novel Sequential Transformers Attention Model (STAM) that progressively observes a scene only partially using glimpses to predict its label. It predicts future informative glimpse locations solely based on past glimpses. STAM is applicable in scenarios where a complete image is not observable due to reasons including a small field of view or limited time and resources, e.g., aerial imaging. We trained STAM using a one-step actor-critic algorithm; and proposed a novel consistency training objective which further improves its accuracy by 3% on ImageNet and 8% on fMoW with only two glimpses.

While never sensing a complete image, our agent outperforms the previous state-of-the-art [45] that observes an entire image by sensing nearly 27% and 42% fewer pixels in glimpses per image on ImageNet and fMoW, respectively. Finally, to save the inference time and resources, we devise a simple scheme to terminate sensing when STAM has predicted a label with sufficient confidence. With a confidence score \(> 0.5\), STAM correctly classifies nearly 65% images on both datasets by observing on an average < 9 glimpses per image (i.e., < 18% of the total image area). However, STAM is limited by its quadratic computational cost. One way to overcome this limitation is to replace DeiT<sup>D</sup> with sparse transformers [26, 31]. Lastly, note that while we focus on static images only, STAM could also potentially be applied to dynamic scenes for early recognition [35, 53].
References


