Consistency driven Sequential Transformers Attention Model
for Partially Observable Scenes

Samrudhdhi B. Rangrej^a, Chetan L. Srinidhi^b, James J. Clark^a

^{a}McGill University, Canada. ^{b}Sunnybrook 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 [45] 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 classify-
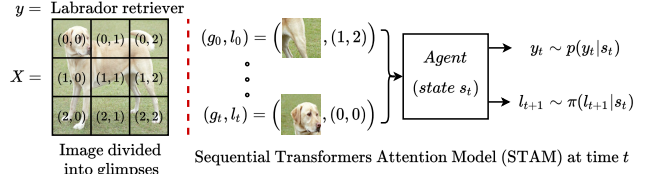

Figure 1. Schematic diagram of Sequential Transformers Attention Model (STAM). We divide an image (X) into equally-sized non-overlapping glimpses. STAM sequentially observes informative glimpses (g_t) from an image. While never observing an image entirely, STAM predicts the class-label of an image (y) based on glimpses. At each t, our agent encodes past glimpses and their locations ($g_{0:t}, l_{0:t}$) into a Markov state s_t. It uses state s_t to predict class distribution $p(y_t|s_t)$ and attention policy $\pi(l_{t+1}|s_t)$. We sample the next glimpse location l_{t+1} from $\pi(l_{t+1}|s_t)$.

ing incomplete images. For example, the accuracy of DeiT-Small [45] 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, 46] 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, 45, 48] 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

1

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.

- 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.

- 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 ∼3% and ∼8% gain in accuracy on ImageNet and fMoW, respectively.

- 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 [53], 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 [5,14,24,27,49,53].

Most previous hard attention models initially glance at a complete image to locate the most informative glimpses. For instance, Xu *et al*. [53] and Saccader [14] analyze the

complete image at original resolution; whereas DRAM [6], TNet [27] and GFNet [49] observes the complete image at low resolution. Furthermore, TNet [27] and GFNet [49] 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 [46]. 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 [55], Dynamic-ViT [31] and IA-RED^2 [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 [8,20,39,51]. Consistency learning acts as a regularizer that enforces the model output to be invariant to the perturbations in the input image [8,36,39,51] 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 [9,52]. 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.
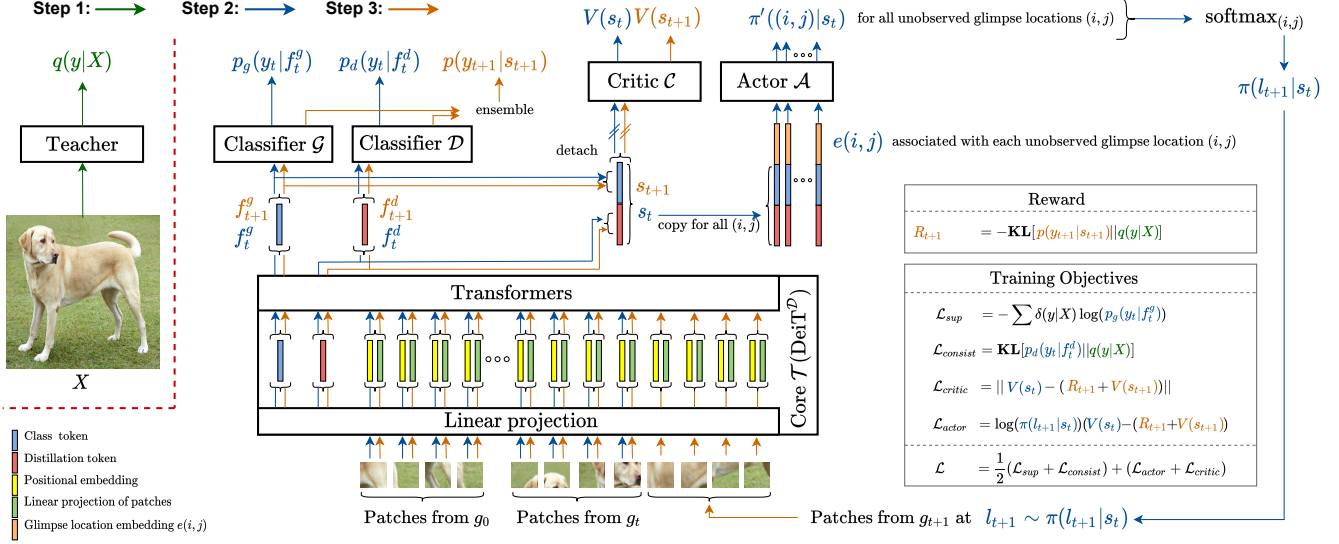
Step 1: ⟶ **Step 2:** ⟶ **Step 3:** ⟶

$q(y|X)$ | $p_g(y_t|f_t^g)$ $p_d(y_t|f_t^d)$ $p(y_{t+1}|s_{t+1})$ | $V(s_t)$ $V(s_{t+1})$ | $\pi'((i,j)|s_t)$ for all unobserved glimpse locations (i,j) ⟶ $\text{softmax}_{(i,j)}$

Teacher | Classifier $\mathcal{G}$ Classifier $\mathcal{D}$ — ensemble | detach | Critic $\mathcal{C}$ Actor $\mathcal{A}$ | $\pi(l_{t+1}|s_t)$

$e(i,j)$ associated with each unobserved glimpse location (i,j)

f_{t+1}^g / f_t^g f_{t+1}^d / f_t^d s_{t+1} / s_t copy for all (i,j)

Reward
$R_{t+1} = -\mathbf{KL}[p(y_{t+1}|s_{t+1})||q(y|X)]$

Training Objectives
$\mathcal{L}_{sup} = -\sum \delta(y|X)\log(p_g(y_t|f_t^g))$
$\mathcal{L}_{consist} = \mathbf{KL}[p_d(y_t|f_t^d)||q(y|X)]$
$\mathcal{L}_{critic} = ||V(s_t) - (R_{t+1}+V(s_{t+1}))||$
$\mathcal{L}_{actor} = \log(\pi(l_{t+1}|s_t))(V(s_t)-(R_{t+1}+V(s_{t+1})))$
$\mathcal{L} = \frac{1}{2}(\mathcal{L}_{sup}+\mathcal{L}_{consist}) + (\mathcal{L}_{actor}+\mathcal{L}_{critic})$

Transformers — Linear projection — Core $\mathcal{T}$ (DeiT$^{\mathcal{D}}$)

X

Class token
Distillation token
Positional embedding
Linear projection of patches
Glimpse location embedding $e(i,j)$

Patches from g_0 Patches from g_t Patches from g_{t+1} at $l_{t+1} \sim \pi(l_{t+1}|s_t)$

Figure 2. An overview of our **Sequential Transformers Attention Model** (**STAM**). The STAM consists of a core $\mathcal{T}$, classifiers $\mathcal{G}$ and $\mathcal{D}$, an actor $\mathcal{A}$, and a critic $\mathcal{C}$ (only used during training). We discuss the working principles of these modules in Section 3.1, except for the critic $\mathcal{C}$ which we discuss in Section 4. We update model parameters T times per batch using the objectives shown in the right and discussed in Section 4. Each training iteration consists of three steps: **Step 1 (green path)**: Given a complete image X, the teacher model predicts a soft pseudo-label $q(y|X)$. **Step 2 (blue path)**: Given glimpses $g_{0:t}$, the core $\mathcal{T}$ predicts features f_t^g and f_t^d. The classifiers $\mathcal{G}$ and $\mathcal{D}$ predict class distributions $p_g(y_t|f_t^g)$ and $p_d(y_t|f_t^d)$ from features f_t^g and f_t^d, respectively. Given a state $s_t = [f_t^g; f_t^d]$, the critic $\mathcal{C}$ predicts value $V(s_t)$ and the actor $\mathcal{A}$ predicts attention policy $\pi(l_{t+1}|s_t)$. The actor predicts logits $\pi'((i,j)|s_t)$ for all unobserved glimpse locations (i,j) in a conditionally independent manner and applies softmax to the logits resulting in $\pi(l_{t+1}|s_t)$. **Step 3 (orange path)**: A glimpse g_{t+1} at $l_{t+1} \sim \pi(l_{t+1}|s_t)$ is sensed. Using the glimpses $g_{0:t+1}$, the ensemble class distribution $p(y_{t+1}|s_{t+1})$ and value $V(s_{t+1})$ are computed following the same path as Step 2. The model parameters are updated using the gradients from Step 2. In practice, Step 1 is performed once per batch at $t=0$, whereas, Steps 2-3 are performed T times per batch.

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, $\{(g_{t'}, l_{t'})| \ t' \in \{0,\ldots,t\}\}$, 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 [45], referred to as DeiT$^{\mathcal{D}}$ in the rest of the paper. Briefly, DeiT$^{\mathcal{D}}$ is a type of ViT trained using knowledge distillation. The transformers in DeiT$^{\mathcal{D}}$ 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$^{\mathcal{D}}$ 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 ($\mathcal{T}$). 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$^{\mathcal{D}}$ 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.

3

Algorithm 1 Inference using STAM

1: Initialize l_0 randomly;
2: **for** $t \in \{0, \dots, T-1\}$ **do**
3:     Sample g_t at l_t from an image   ▷ **Sensor**
4:     $f_t^g, f_t^d = \mathcal{T}(g_{0:t}, l_{0:t}); \;\; s_t = [f_t^g; f_t^d]$ ▷ **Core**
5:     $p_g(y_t|\cdot) = \mathcal{G}(f_t^g); \;\; p_d(y_t|\cdot) = \mathcal{D}(f_t^d)$ ▷ **Classifiers**
6:     $\pi'(l'|\cdot) = \mathcal{A}(s_t, l'), \;\; \forall l' \in \{\{1, .., N\}\}^2 - l_{0:t}$ ▷ **Actor**
7:     $y_t = argmax(p_g(y_t|\cdot) + p_d(y_t|\cdot))$
8:     $l_{t+1} = argmax(\pi'(l'|\cdot))$
9: **end for**

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

Classifiers ($\mathcal{G}$ and $\mathcal{D}$). As in DeiT$^{\mathcal{D}}$, we use two linear classifiers to predict two class distributions $p_g(y_t|f_t^g)$ and $p_d(y_t|f_t^d)$ from f_t^g and f_t^d, respectively. We treat the predicted distributions independently during training and average them to form an ensemble distribution during inference [45]:

$$p(y_t|s_t) = \frac{1}{2}(p_g(y_t|f_t^g) + p_d(y_t|f_t^d)). \tag{1}$$

Actor ($\mathcal{A}$). 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 logits $\{\pi'((i,j)|s_t)\}$, where (i,j)s are unobserved glimpse locations. An actor predicts $\pi'((i,j)|\cdot)$ 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 pretrained DeiT$^{\mathcal{D}}$. We use $l_{t+1} \sim \pi(l_{t+1}|s_t)$ during training and $l_{t+1} = argmax(\pi(l_{t+1}|s_t))$ during inference.

We provide the inference steps in Algorithm 1.

4. Training Objectives

We train the parameters of the core ($\theta_{\mathcal{T}}$), the classifiers ($\theta_{\mathcal{G}}$ and $\theta_{\mathcal{D}}$) and the actor ($\theta_{\mathcal{A}}$) using the training objectives discussed next. Figure 2 illustrates the training steps of our model, and Algorithm 2 in the Appendix presents the corresponding pseudocode.

4.1. Learning Classification

Our agent predicts two class distributions based on input glimpses, namely, p_g and p_d; where, p_g 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_g and p_d using the following two objectives:

Supervised Loss. As our goal is to predict y from partial observations, we learn the parameters $\{\theta_{\mathcal{T}}, \theta_{\mathcal{G}}\}$ by minimiz-

ing a cross-entropy between $p_g(y_t|s_t)$ and $\delta(y|X)$ given by

$$\mathcal{L}_{sup} = -\sum \delta(y|X) \log(p_g(y_t|s_t)), \tag{2}$$

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. *Ideally*, for each t, we require our agent to produce $p_d(y_t|s_t)$ that minimize $\mathbf{KL}[p_d(y_t|s_t)||p(y|X)]$; 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 [16,52]. The resultant consistency objective to train $\{\theta_{\mathcal{T}}, \theta_{\mathcal{D}}\}$ is given by

$$\mathcal{L}_{consist} = \mathbf{KL}[p_d(y_t|s_t)||q(y|X)]. \tag{3}$$

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 [50] to learn an attention policy. These methods run an agent for $t = 0$ to $T-1$ steps to achieve R_1 to R_T and compute G_0 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 ($\mathcal{C}$) with parameters υ. A critic learns a value function $V(s_t)$ that estimates the expected return given the current state of the agent, i.e., $\mathbb{E}_\pi[G_t]$. As $\mathbb{E}_\pi[G_t] = \mathbb{E}_\pi[R_{t+1} + G_{t+1}]$, $V(s_t)$ should be equal to $\mathbb{E}_\pi[R_{t+1} + V(s_{t+1})]$. Hence, the critic parameters υ are learned by minimizing the difference between the two quantities. In practice, we estimate the expectation with respect to π using a single Monte-Carlo sample, yielding

$$\mathcal{L}_{critic} = ||V(s_t) - (R_{t+1} + V(s_{t+1}))||. \tag{4}$$

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 $(V(s_t) - (R_{t+1} + V(s_{t+1})))$; 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 $\{\theta_\mathcal{T}, \theta_\mathcal{A}\}$ by minimizing

$$\mathcal{L}_{actor} = \log(\pi(l_{t+1}|s_t))(V(s_t) - (R_{t+1} + V(s_{t+1}))). \tag{5}$$

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

$$R_t = -\mathbf{KL}[p(y_t|s_t)||q(y|X)], \tag{6}$$

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.

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_{sup} + \mathcal{L}_{consist}) + (\mathcal{L}_{actor} + \mathcal{L}_{critic}) \tag{7}$$

5. Experiment Setup

Datasets. We experiment with two large-scale real-world datasets, namely, ImageNet [34] and fMoW [10]. ImageNet consists of natural images from 1000 categories. It includes $\sim$1.3M training images and 50K validation images. We resize the images to size 224×224. The fMoW contains satellite images from 62 categories. It holds $\sim$0.36M, $\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 224×224. Unless stated otherwise, we implement and optimize STAM with the same default setting on both datasets.

Implementation.[1] We divide the images into non-overlapping glimpses of size 32×32, yielding a 7×7 grid

of glimpses. As required by DeiT$^\mathcal{D}$, we further divide each glimpse into four non-overlapping patches of size 16×16.

We use DeiT$^\mathcal{D}$-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$^{\mathcal{D}2}$ 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 τ 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 [47] to the predicted values.

We use DeiT$^\mathcal{D}$ and DeiT as teacher models for ImageNet and fMoW, respectively. For the ImageNet teacher model, we use publicly available weights. The teacher model for fMoW is first initialized with the DeiT model pretrained on ImageNet, followed by fine-tuning on the fMoW dataset for 100 epochs using the default hyperparameter setting from [45] with an additional vertical flip augmentation.

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 τ 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 Rand-Augment scheme [12] and follow the same setting as Touvron *et al.* [45]. 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-

---

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

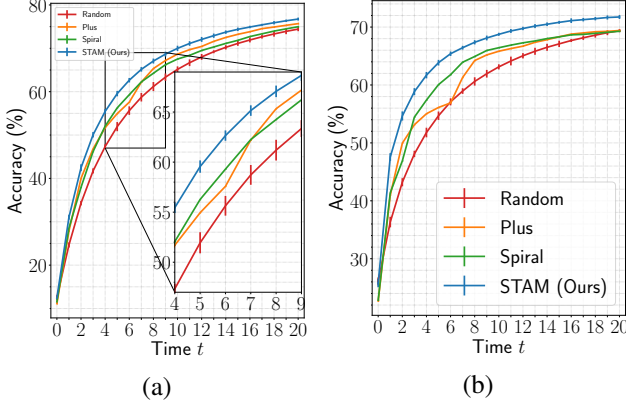[2]https://github.com/facebookresearch/deit

Figure 3. Baseline comparison of various attention policies. (a) ImageNet; (b) fMoW. The Random selects glimpses in random order. The Plus and the Spiral select glimpses in the order shown in Figure 4 (c). Starting from a random glimpse location, our STAM uses an RL agent to predict next glimpse location. Results for the Random and STAM are presented as mean±5×std computed from ten independent runs.
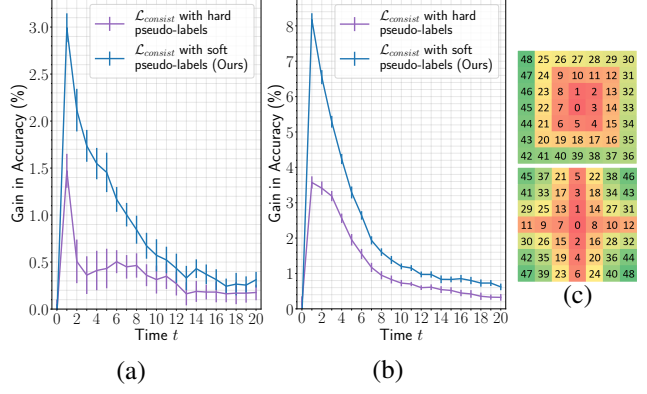

Figure 4. The gain in the accuracy of STAM when trained by including the consistency loss (with soft vs hard pseudo-labels) over excluding it from the training objectives. (a) ImageNet; (b) fMoW. Results are presented as mean ± std computed using ten different runs. (c) Given an image, (top) the Spiral and (bottom) the Plus baselines select glimpses in the order shown.

domly from a set of unobserved glimpses. In contrast, the Plus and the Spiral agents account for the object-centric nature of vision datasets and select glimpses in the order shown in Figure 4 (c). For a fair comparison, all baseline agents begin with the first glimpse at a random location. The model architecture of the baseline agents is similar to our proposed agent, except that the baseline agents do not have an actor module. We train the baseline agents following the same procedure as our agent using the losses from Equation 2 and 3.

Results are shown in Figure 3. Among the three baselines, the Spiral and the Plus agents outperform the Random agent. For $t \geq 8$, the Plus achieves higher accuracy than the Spiral on ImageNet, whereas, on fMoW, the Spiral outperforms the Plus. This inconsistent behavior is mainly due to the different orientations of objects in the two datasets. While the objects are mainly aligned vertically or horizontally in ImageNet, the landmarks in fMoW have no specific orientation. Finally, our agent outperforms all baseline agents across the two datasets both at initial ($t < 8$) and later ($t \geq 8$) time-steps. At $t = 8$, it achieves 1.8% higher accuracy on ImageNet and 2.3% higher accuracy on fMoW than the top-performing baselines for the respective datasets.

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.

In Figure 4 (blue curve), we display the *difference* in the accuracy of STAM when trained by including and excluding the consistency loss in the training objectives (Equation 7). With only two glimpses (i.e., one random and one selected by the agent at $t = 1$), the proposed consistency loss results in significant improvement in accuracy with a gain of $\sim 3\%$ on ImageNet and $\sim 8\%$ on fMoW datasets.

To benchmark the improvement offered by our proposed consistency loss, we study the effect of an alternative consistency loss using hard pseudo-labels, $\mathcal{L}_{consist} = -\sum \delta(\hat{y}|X) \log(p_d(y_t|s_t))$ where $\hat{y}$ are the hard pseudo-labels predicted by the teacher model from a complete image, i.e., $\hat{y} = \operatorname{argmax}(q(y|X))$, and $\delta(\hat{y}|X)$ is a delta distribution. The results are shown in Figure 4 (purple curve). An agent trained using a consistency loss with hard pseudo-labels achieves a gain of $\sim 1.5\%$ on ImageNet and $\sim 3.5\%$ on fMoW for the first two glimpses.

Overall, the consistency loss improves the performance of STAM. The gain in accuracy with consistency loss evaluated using soft pseudo-labels is greater than the hard pseudo-labels. In the Appendix (Section A.1), we demonstrate that the consistency loss also improves the performance of the Random, the Plus, and the Spiral agents. There we observe that the gain in performance is higher for STAM over the baseline agents.

Additional results in the Appendix. In Section A.2, 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 A.3, we show that an increase in model capacity can improve the performance of STAM. In Section A.4, we show that longer training on ImageNet also improves STAM's performance further.

$y = $ Umbrella $y_0 = $ Flowerpot ← $y_{1:2} = $ Park bench → ← $y_{3:6} = $ Lakeside → ←

$y_{7:13} = $ Park bench → ← $y_{14:15} = $ Umbrella →

(a)

$y = $ Factory or Powerplant $y_0 = $ Crop field ← $y_{1:7} = $ Electric substation →
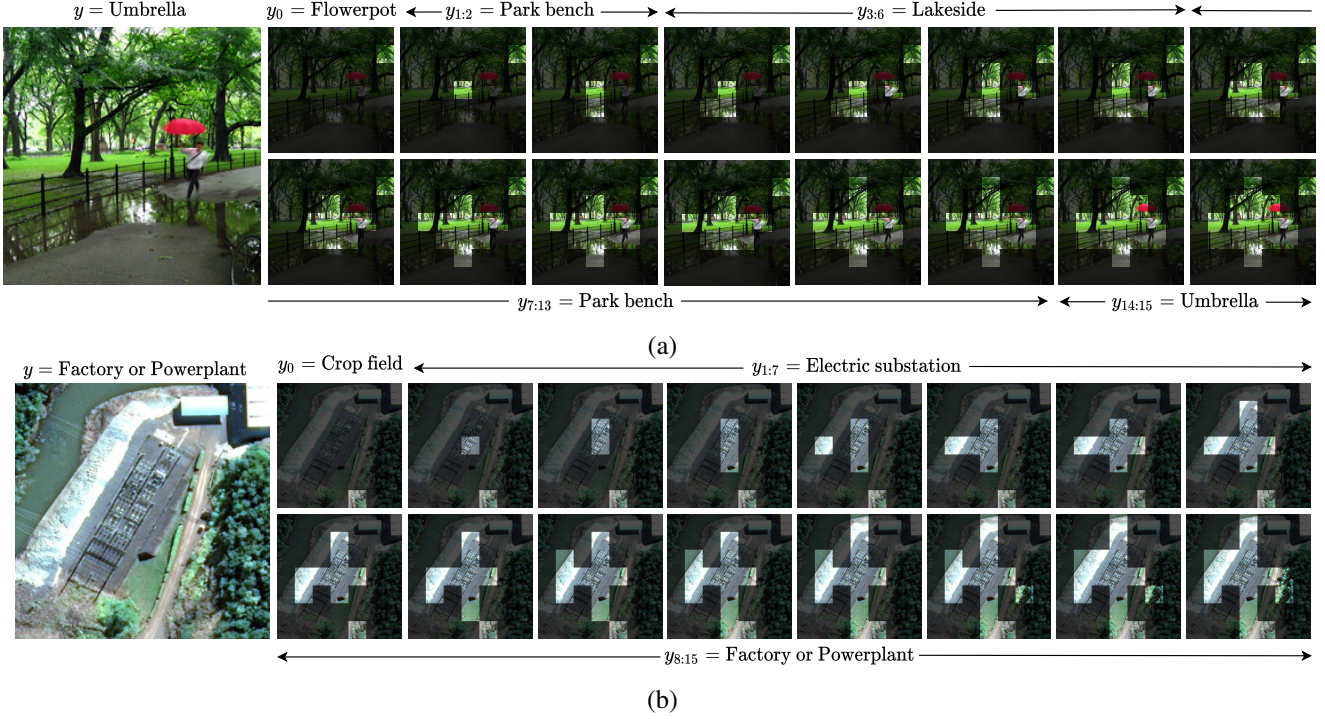
$y_{8:15} = $ Factory or Powerplant →

(b)

Figure 5. Visualization of glimpses selected by STAM on example images from $t = 0$ to 15. (a) ImageNet; (b) fMoW. Complete images are shown for reference only. STAM does not observe a complete image. Zoom-in to observe details.

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 12-13 in the Appendix.

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.
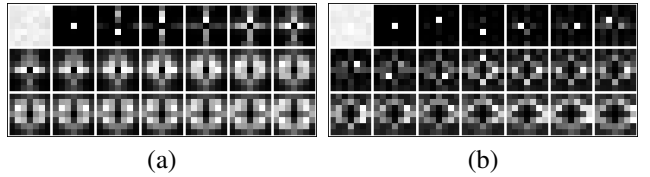

(a) (b)

Figure 6. Histograms of glimpse locations sensed by STAM on (a) ImageNet and (b) fMoW. The first, second, and third rows of each panel display histograms for $t = 0$ to 6, 7 to 13, and 14 to 20. At $t = 0$, STAM observes a glimpse at a random location. At $t > 0$, STAM senses glimpses at the locations predicted by an RL agent.

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, 46, 49]. 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, 46, 49]. 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

Method	Note	Is complete image used for		ImageNet		fMoW	
		Attention?	Classification?	#pixels for classification	Accuracy (%)	#pixels for classification	Accuracy (%)
DRAM [5]	results from [27]	Yes	No	47.4K	67.50	—	—
GFNet [49]		Yes	Yes	46.1K	75.93	—	—
Saccader [14]	results from [27]	Yes	No	35.6K	70.31	—	—
TNet [27]		Yes	Yes	35.6K	74.62	—[†]	—
STN [32,46]	developed in [46] based on [32]	Yes	Yes	28.2K	71.40	22.0K	64.8
PatchDrop [46]		Yes	Yes	27.9K	76.00	19.4K	68.3
STAM (DeiT$^{\mathcal{D}}$-Small (default))	*equivalent	No	No	**20.5K**$(t = 19)$	**76.35**	**11.3K**$(t = 10)$	**68.8**
STAM (DeiT$^{\mathcal{D}}$-Base)	accuracy to [46]	No	No	**14.3K**$(t = 13)$	**76.13**	—	—
STAM (DeiT$^{\mathcal{D}}$-Small (default))	°equivalent	No	No	**27.7K**$(t = 26)$	**78.25**	**19.5K**$(t = 18)$	**71.5**
STAM (DeiT$^{\mathcal{D}}$-Base)	sensing to [46]	No	No	**27.7K**$(t = 26)$	**80.78**	—	—

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. [†]TNet [27] use 896×896 resolution images; hence, we do not include their results in the above comparison.

include the pixels of the complete image in the above count.

To achieve the same level of accuracy as PatchDrop [46] (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$^{\mathcal{D}}$-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, $\mathcal{S}_t = max(p(y_t|s_t))$. The agent stops sensing more glimpses if $\mathcal{S}_t$ is greater than threshold γ. In Figure 7, we show the average number of glimpses observed per image vs. the accuracy of our agent for γ 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 pre-
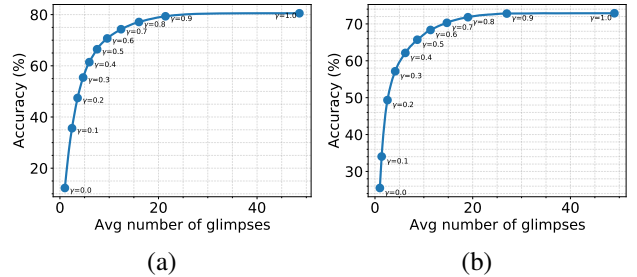

(a)   (b)

Figure 7. Average number of glimpses observed per image vs the accuracy achieved by STAM on (a) ImageNet and (b) fMoW datasets with early termination scheme. STAM suspends sensing once the probability of predicted class is higher than threshold γ.

dicts 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 [46] 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$^{\mathcal{D}}$ 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, 54].

References

[1] Bogdan Alexe, Nicolas Heess, Yee W Teh, and Vittorio Ferrari. Searching for objects driven by context. In *Advances in Neural Information Processing Systems*, pages 881–889, 2012. 2

[2] Baptiste Angles, Simon Kornblith, Shahram Izadi, Andrea Tagliasacchi, and Kwang Moo Yi. Mist: Multiple instance spatial transformer network. *arXiv preprint arXiv:1811.10725*, 2018. 2

[3] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 14278–14287, 2021. 4

[4] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. 2

[5] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015. 1, 2, 4, 7, 8

[6] Jimmy Ba, Russ R Salakhutdinov, Roger B Grosse, and Brendan J Frey. Learning wake-sleep recurrent attention models. In *Advances in Neural Information Processing Systems*, pages 2593–2601, 2015. 2

[7] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. *Advances in Neural Information Processing Systems*, 27:3365–3373, 2014. 2

[8] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019. 2

[9] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. *arXiv preprint arXiv:2106.05237*, 2021. 2

[10] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018. 2, 5

[11] Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. Differentiable patch selection for image recognition. *arXiv preprint arXiv:2104.03059*, 2021. 2

[12] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 5

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[14] 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. 1, 2, 4, 7, 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 4

[17] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661, 2016. 2

[18] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International Conference on Machine Learning*, pages 2127–2136. PMLR, 2018. 2

[19] Angelos Katharopoulos and Francois Fleuret. Processing megapixel images with deep attention-sampling models. In *International Conference on Machine Learning*, pages 3282–3291, 2019. 2

[20] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 2

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[22] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 896, 2013. 2

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 5

[24] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014. 1, 2, 4

[25] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *arXiv preprint arXiv:2105.10497*, 2021. 1

[26] Bowen Pan, Yifan Jiang, Rameswar Panda, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red ^{2}: Interpretability-aware redundancy reduction for vision transformers. *arXiv preprint arXiv:2106.12620*, 2021. 2, 8

[27] Athanasios Papadopoulos, Paweł Korus, and Nasir Memon. Hard-attention for scalable image classification. *arXiv preprint arXiv:2102.10212*, 2021. 1, 2, 4, 7, 8

[28] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021. 2

[29] Samrudhdhi B. Rangrej and James J. Clark. A probabilistic hard attention model for sequentially observed scenes. *arXiv preprint arXiv:2111.07534*, 2021. 2

[30] Marc'Aurelio Ranzato. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014. 2

[31] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *arXiv preprint arXiv:2106.02034*, 2021. 2, 8

[32] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018. 8

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 91–99, 2015. 2

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 5

[35] Michael S Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *2011 International Conference on Computer Vision*, pages 1036–1043. IEEE, 2011. 8

[36] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171, 2016. 2

[37] Soroush Seifi, Abhishek Jha, and Tinne Tuytelaars. Glimpse-attend-and-explore: Self-attention for active visual exploration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 16137–16146, 2021. 2

[38] Soroush Seifi and Tinne Tuytelaars. Attend and segment: Attention guided active semantic segmentation. In *Computer Vision–ECCV: 16th European Conference, Glasgow, UK, Proceedings, Part XXV 16*, pages 305–321, 2020. 2

[39] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. 2

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2

[41] Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984. 4

[42] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019. 1

[43] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 4

[44] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, volume 33, pages 6827–6839, 2020. 11

[45] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021. 1, 3, 4, 5, 11

[46] Burak Uzkent and Stefano Ermon. Learning when and where to zoom with deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12345–12354, 2020. 1, 2, 7, 8

[47] Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in Neural Information Processing Systems*, 29:4287–4295, 2016. 5

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1

[49] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 7, 8

[50] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. 4

[51] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019. 2

[52] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. 2, 4

[53] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 2

[54] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2678–2687, 2016. 8

[55] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 387–396, 2021. 2

A. Additional Results

A.1. Analysis of Consistency Loss with Baseline Attention Policies

In the main paper, we analyzed the gain in accuracy of STAM when the proposed consistency loss (Equation 3 in the main paper) is included in the training objectives. Here, we analyze the same for the agents with baseline attention policies, namely, the Random, the Plus, and the Spiral. We train baseline agents with and without the proposed consistency objective and plot the difference in their accuracy in Figure 8. We observe that the consistency training objective yields a positive gain in the accuracy for all baseline agents.

Furthermore, the gain achieved with learned policy (i.e., STAM) is higher than the heuristics-based baseline policies. The gain in accuracy is highest for STAM as it learns to attend to the most discriminative glimpses early in time. These results align with the recent findings showing that minimizing the distance between the predictions made from two views of the same image improves model performance the most when the views optimally share the task-specific information [44].
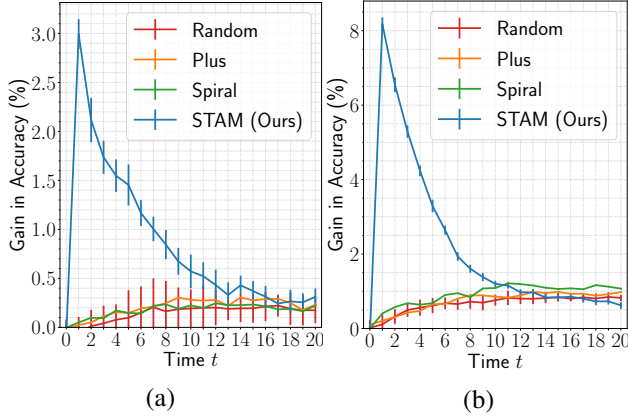

Figure 8. Comparison of gain in accuracy of various baseline agents with inclusion of consistency loss in their training objectives. (a) ImageNet; (b) fMoW. Results for the Random and STAM are presented as mean ± std computed across ten independent runs.

A.2. Effect of Glimpse Size

We compare the performance of our agents with glimpses of sizes 32×32, 48×48, and 64×64. To extract the non-overlapping glimpses, we resize the image to 224×224, 240×240, and 256×256 for the three glimpse sizes stated above, respectively.

For the image-size 224×224, we use the teacher models as discussed in the main paper. To train teacher models for images of sizes 240×240 and 256×256, we finetune the pretrained DeiT on images of respective sizes, following the
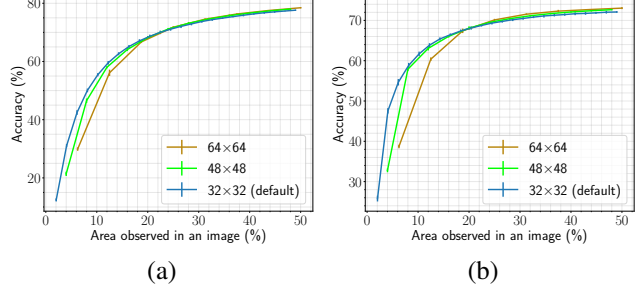

Figure 9. Accuracy of STAM with different glimpse sizes presented as a function of % area observed in an image (a) ImageNet; (b) fMoW. The results are presented as mean±5×std computed across ten independent runs.

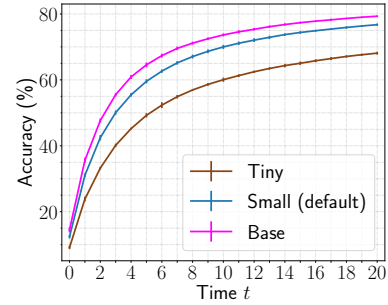Figure 10. Accuracy of STAM with core of different capacity. We compare DeiT$^{\mathcal{D}}$-Tiny, DeiT$^{\mathcal{D}}$-Small, DeiT$^{\mathcal{D}}$-Base architectures for the core module. The results are presented as mean±5×std computed across ten independent runs.

procedure suggested by Touvron *et al.* [45]. We train all agents following the same experimental setup discussed in the main paper, except for the following. We train the agents for image sizes 240×240 and 256×256 using batch sizes of 2000 and 1600, and they observe a maximum of 16 and 7 glimpses per image.

As the glimpse and the image sizes are different, we compare the accuracy of the three agents as a function of the area observed in the image (see Figure 9). Initially, when an area observed in an image is less than 20%, the agent with smaller glimpses achieves higher accuracy than the agent with larger glimpses. The reason is that the agent explores more regions using smaller glimpses than the larger ones while sensing the same amount of area. Once the agents have observed sufficient informative regions (nearly 20% of the total image area), their performance converges. We use glimpse size 32×32 with image size 224×224 as our default setting.

A.3. Effect of Model Capacity

To study the effect of model capacity on the performance, we compare DeiT$^{\mathcal{D}}$-Tiny, DeiT$^{\mathcal{D}}$-Small, and DeiT$^{\mathcal{D}}$-Base architectures as the core of our agent. The three agents are trained using the same procedure as dis-
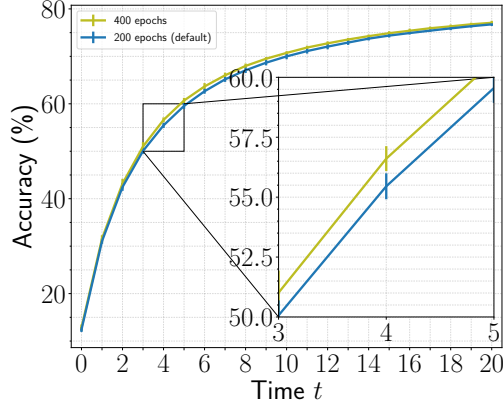
Figure 11. Accuracy of STAM on ImageNet when trained for 200 and 400 epochs. The results are presented as mean$\pm5\times$std computed across ten independent runs.

cussed in the main paper expect for the following. We train agent with DeiT$^{\mathcal{D}}$-Base core using batch size of 512. We use pretrained DeiT$^{\mathcal{D}}$ of respective capacity as the teacher model. Results for ImageNet are presented in Figure 10. We observe increasing accuracy with increasing model capacity. However, training an agent with DeiT$^{\mathcal{D}}$-Base is computationally expensive. To achieve a good trade-off between efficiency and accuracy, we use DeiT$^{\mathcal{D}}$-Small as a default architecture for our agent.

A.4. Longer Training on ImageNet

We demonstrate that longer training improves the performance of STAM on ImageNet. We compare performance of STAM trained for 200 and 400 epochs in Figure 11. When STAM is allowed to observe only five glimpses, longer training yields 1.15% improvement in the accuracy. In contrast, we observe overfitting and reduced performance with longer training on fMoW.
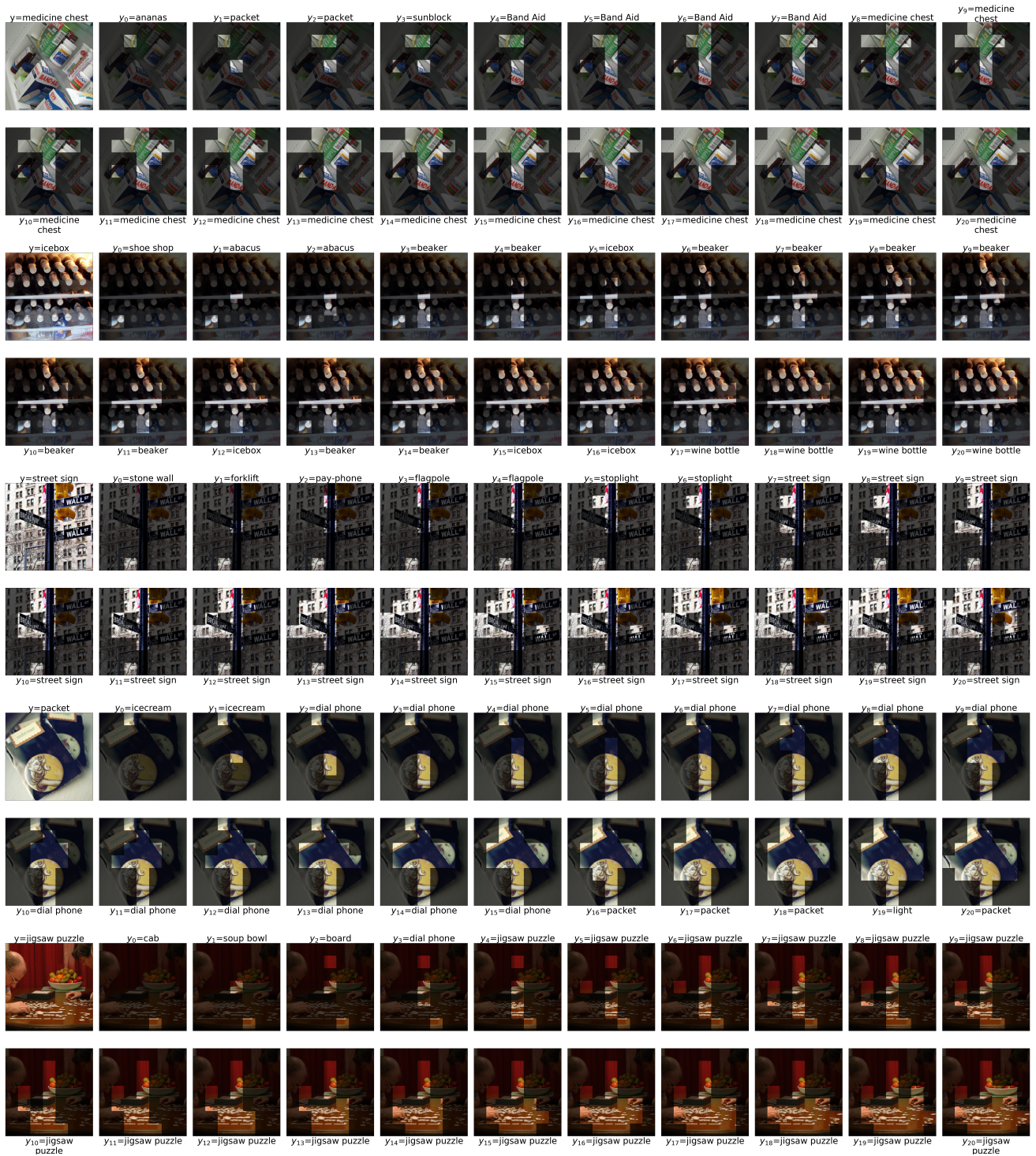
Figure 12. Glimpses selected by STAM on example images from the ImageNet dataset and the predicted labels. Complete images are shown for reference only. Note that STAM does not observe the complete image. Ground truth labels are displayed above complete images.
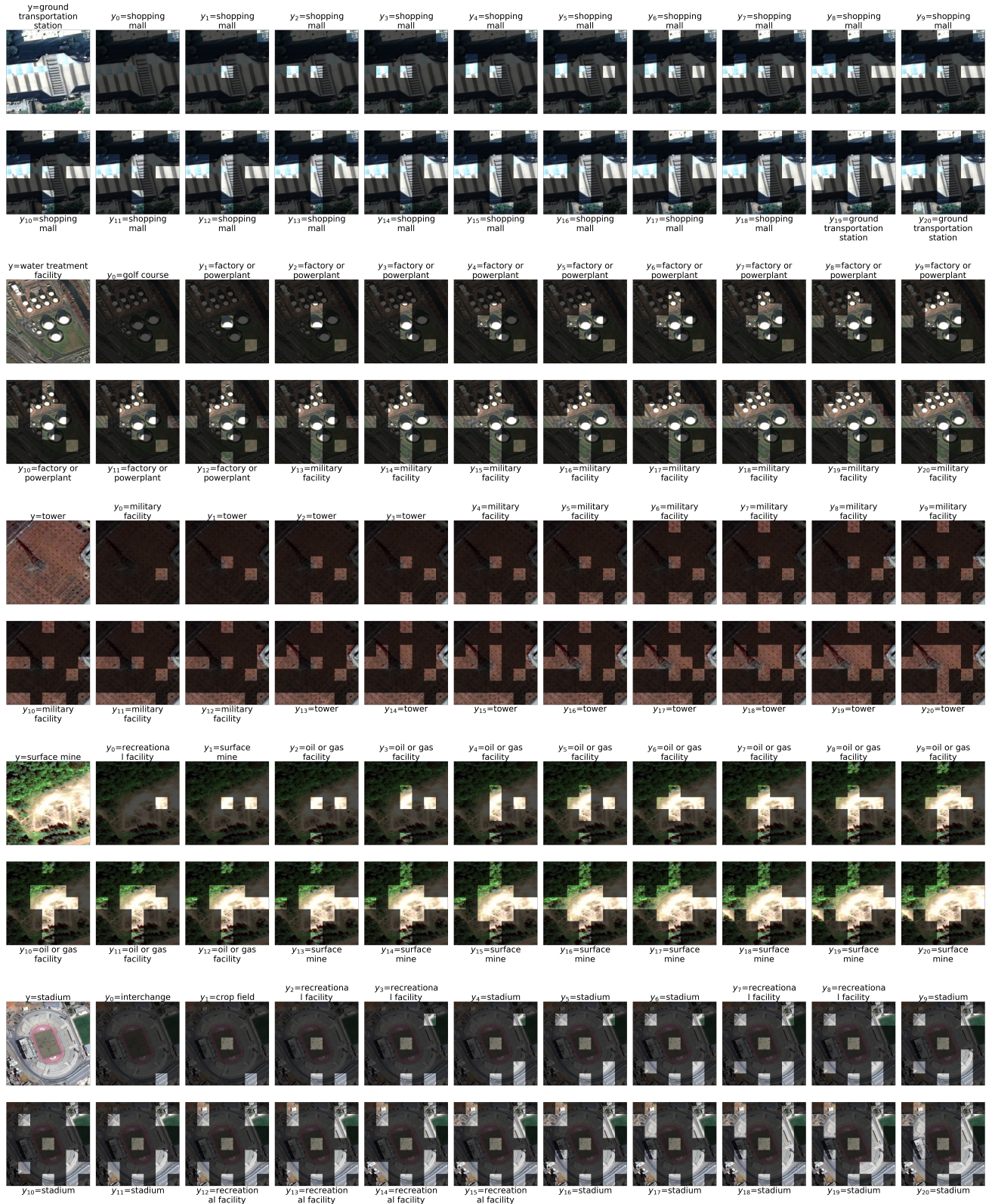
Figure 13. Glimpses selected by STAM on example images from the fMoW dataset and the predicted labels. Complete images are shown for reference only. Note that STAM does not observe the complete image. Ground truth labels are displayed above complete images.

14

Algorithm 2 Pseudo code for training our Sequential Transformers Attention Model (STAM)

```
'''
Inputs:
    X = complete image X
    y = ground truth for X
'''
def process_one_batch(X,y):
    # STAM collects series of T glimpses from X
    # Parameters of STAM are updated after each additional glimpse
    q = step_one(X)
    l_t = initial_random_location() # Initial glimpse should be captured at a random location
    g_t = extract_glimpse(X, l_t)
    g_upto_t = [g_t]                # A list of all glimpses
    l_upto_t = [l_t]                # A list of all glimpse locations
    for t in range(T):
        # Perform step 2
        p_g_t, p_d_t, V_t, pi_of_l_tplus1, l_tplus1 = step_two(g_upto_t, l_upto_t)
        # Extract one additional glimpse and append it to previous glimpses
        g_tplus1 = extract_glimpse(X, l_tplus1)
        g_upto_t.append(g_tplus1)
        l_upto_t.append(l_tplus1)
        # Perform step 3
        p_tplus1, V_tplus1 = step_three(g_upto_t, l_upto_t)
        # Evaluate losses
        loss = evaluate_losses(y, q, p_g_t, p_d_t, V_t, pi_of_l_tplus1, p_tplus1, V_tplus1)
        # Update model parameters
        loss.backward()
        optimizer.step()

def step_one(X):
    # Teacher predicts soft pseudo-label from a complete image
    with no_grad():
        q = teacher(X)
    return q

def step_two(g_upto_t, l_upto_t):
    # STAM predicts class distributions, state value, attention policy and next glimpse location
    f_g_t, f_d_t, s_t = core(g_upto_t, l_upto_t)        # Core
    p_g_t, p_d_t = classifiers(f_g_t, f_d_t)            # Classifiers
    V_t = critic(s_t)                                   # Critic
    l_unobserved = find_unobserved_locations(l_upto_t)  # Find yet unobserved locations
    pi_of_l_tplus1, l_tplus1 = actor(s_t, l_unobserved) # Actor
    return p_g_t, p_d_t, V_t, pi_of_l_tplus1, l_tplus1

def step_three(g_upto_tplus1, l_upto_tplus1):
    # STAM computes ensemble class distribution and the state value one step ahead
    with no_grad():
        f_g_tplus1, f_d_tplus1, s_tplus1 = core(g_upto_tplus1, l_upto_tplus1)    # Core
        p_g_tplus1, p_d_tplus1 = classifiers(f_g_tplus1, f_d_tplus1)             # Classifiers
        p_tplus1 = (p_g_tplus1 + p_d_tplus1)/2                                   # Ensemble
        V_tplus1 = critic(s_tplus1)                                             # Critic
    return p_tplus1, V_tplus1

def evaluate_losses(y, q, p_g_t, p_d_t, V_t, pi_of_l_tplus1, p_tplus1, V_tplus1):
    # Evaluate losses
    L_sup = cross_entropy(p_g_t, y)                              # Supervised classification loss
    L_consist = kl_div(p_d_t, q)                                # Consistency loss
    R_tplus1 = - kl_div(p_tplus1, q)                                # Reward
    L_critic = l1_loss(V_t, R_tplus1 + V_tplus1)                    # Critic loss
    L_actor = pi_of_l_tplus1 * (V_t-(R_tplus1 + V_tplus1)).detach()     # Actor loss
    L_final = (L_sup + L_consist)/2 + L_critic + L_actor            # Final loss
    return L_final
```