Efficient Classification of Very Large Images with Tiny Objects

Supplementary Material

A. Memory Cost Analysis

In Memory requirements Section 2.2, we empirically analyze the memory requirements of the proposed Zoom-In network and the closely related ATS model. Below we study the memory usage of these models using the colon cancer data.

The memory usage for the one-stage ATS and the proposed two-stage model are $\mathcal{O}(s^2HW + Nh_2w_2)$ and $\mathcal{O}(s_1^2HW + N's_1^2s_2^2HW + Nh_2w_2)$ respectively. Notations is inherited from the main paper, Zoom-In Network Section 2.2. In the colon cancer experiment, $s = 0.2$, $s_1 = 0.1$, $s_2 = 0.2$ and $N = 10$. Then, if we want the memory usage order for the two-stage model to be smaller than that of the one-stage model, we set $N' < 7.5$. In the experiment, $N' < N/2$ when the two-stage model has converged. It should be noted that in the beginning of training, the initial attention map α is approximately uniform because the weights of $a_\Theta(\cdot)$ are initialized at random. The number selected tiles N' is close to N, which implies that the memory consumption of the two-stage model is slightly larger than the one-stage model. However, after the attention network $a_\Theta(\cdot)$ is optimized, the number of selected tiles N' drops dramatically. At which time, the proposed two-stage model consumes much less memory than one-stage model as shown in Figure 4. Note that for very large images (gigapixel in size), the number of selected tiles is much smaller than the size of sample space $|C|$, which means that even at the beginning of training, the two-stage model does not need to instantiate all tiles in an image and thus requires substantially less memory than the one-stage model (see right plot in Figure 4).

The above memory usage analysis of input entries can be reflected on the counts of FLOP, since the FLOPs is dominated by the size of the feature maps and model parameters, that is computed in the following way in agreement with [10, 36]:

$$FLOPs = C_{in} \times k^2 \times H_{out} \times W_{out} \times C_{out} \tag{10}$$

where C_{in} is the number of channels of the input tensor, k^2 is the size of the convolution kernels in this layer, H_{out}, W_{out} and C_{out} are the heights, width and number of channels of the output tensor.

B. Derivations

Below we complete the proof for the equation (8) in the main paper, by showing the equivalence between $\mathbb{E}_{c'\sim b_\Theta(V(x,s_2,c))}[f_\Theta(T_{s_2}(T_{s_1}(x,c),c'))]$ and $\sum_{c'\in C'}\sum_{i\neq c'}\beta^c_{c'}\frac{\beta^c_i}{1-\beta^c_{c'}}(\beta^c_{c'}f_\Theta(T_{s_2}(T_{s_1}(x,c),c') + (1-\beta^c_{c'})f_\Theta(T_{s_2}(T_{s_1}(x,c),i))$ as follows,

$$\begin{aligned}
&\sum_{c'\in C'}\sum_{i\neq c'}\beta^c_{c'}\frac{\beta^c_i}{1-\beta^c_{c'}}(\beta^c_{c'}f_\Theta(T_{s_2}(T_{s_1}(x,c),c') + (1-\beta^c_{c'})f_\Theta(T_{s_2}(T_{s_1}(x,c),i))\\
&= \sum_{c'\in C'}[\sum_{d\in C'}\beta^c_{c'}\frac{\beta^c_d}{1-\beta^c_{c'}}(\beta^c_{c'}f_\Theta(T_{s_2}(T_{s_1}(x,c),c') + (1-\beta^c_{c'})f_\Theta(T_{s_2}(T_{s_1}(x,c),d))-\\
&\beta^c_{c'}\frac{\beta^c_{c'}}{1-\beta^c_{c'}}(\beta^c_{c'}f_\Theta(T_{s_2}(T_{s_1}(x,c),c') + (1-\beta^c_{c'})f_\Theta(T_{s_2}(T_{s_1}(x,c),c'))]\\
&= \sum_{c'\in C'}[\frac{{\beta^c_{c'}}^2}{1-\beta^c_{c'}}f_\Theta(T_{s_2}(T_{s_1}(x,c),c') + \sum_{d\in C'}\beta^c_d\beta^c_{c'}f_\Theta(T_{s_2}(T_{s_1}(x,c),d)\\
&- \frac{{\beta^c_{c'}}^2}{1-\beta^c_{c'}}f_\Theta(T_{s_2}(T_{s_1}(x,c),c'))]\\
&= \sum_{c'\in C'}\beta^c_{c'}\sum_{d\in C'}\beta^c_d f_\Theta(T_{s_2}(T_{s_1}(x,c),d)\\
&= \sum_{d\in C'}\beta^c_d f_\Theta(T_{s_2}(T_{s_1}(x,c),d)\\
&= \mathbb{E}_{c'\sim b_\Theta(V(x,s_2,c))}[f_\Theta(T_{s_2}(T_{s_1}(x,c),c'))].
\end{aligned} \tag{11}$$

C. Supplementary Figures

Here we present the supplementary figures mentioned in the main paper.

Figure 5 is the error bar plot for the results of colon cancer dataset.

Figure 6 illustrates the interpretability of the proposed attention model for the Colon Cancer, NeedleCamelyon, Traffic Sign Recognition, fMoW and Camelyon16 experiments. We find that areas of high attention of the Zoom-In network and extracted ROI patches are highly consistent with the ground-truth, manually annotated, segmentation masks.

Figure 7 quantitatively examines the quality of the attention of the Zoom-In network. The plots illustrate the correlation of the attention weights generated by the Zoom-In network with the ground-truth metastases-to-tile ratio obtained from pixel annotations. The Spearman correlation coefficient for all the tiles with pixel-level annotations is $\rho = 0.3570$, indicating a good agreement. Interestingly, from the plot, we see that many tiles contain different size of ROIs but has the same magnitude of attention weights, indicating that the attention mechanism seems to attend to the presence of ROIs but not the proportion of ROIs in a tile.

D. Details of Generating NeedleCamelyon Dataset

We present the details of the NeedleCamelyon dataset not included in the main paper. We first split the unique metastases in the original Camelyon16 dataset for positive

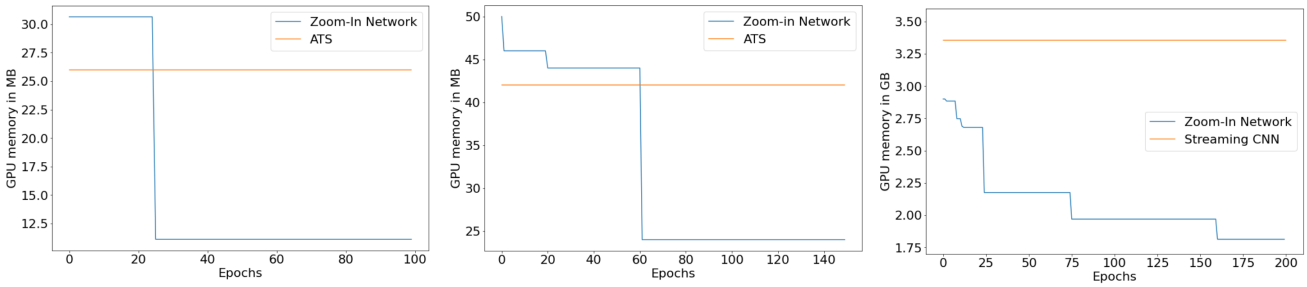

Figure 4. GPU memory usage (y-axis) versus training epoch (x-axis). We plot the GPU memory usage of the Zoom-In network and the one-stage ATS for the (left) colon cancer dataset, (middle) NeedleCamelyon dataset, and (right) Camelyon16 dataset.

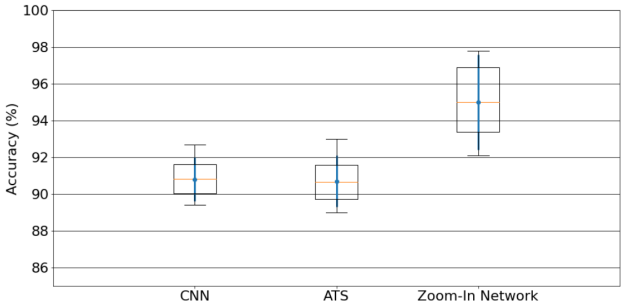

Figure 5. Standard deviations (error bars) for the results on the colon cancer dataset. Each error bar is obtained as the result of the 5 repetitions with the same training hyper-parameters and different random seeds. Standard deviations for CNN, ATS and Zoom-In network are 1.2%, 1.4%, 2.6% respectively. We see that though the variation of the Zoom-In network is larger than for the CNN and ATS, the differences, accounting for variation are still significant in favor of the proposed model.

samples in training set, validation set and test set. The number of unique metastases region for training set, validation set and test set are 122, 41 and 41, respectively. Then we crop these metastases area to build the NeedleCamelyon dataset. Negative images are taken by randomly cropping normal whole-slide images and filtering image crops that mostly contain background. We ensure the class balance by sampling an equal amount of positive and negative crops. Finally, we obtain 6000 crops for training, 2000 crops for validation and 2000 crops for testing. Each crop has a size of 1024×1024.

E. Details of the Functional Map of the World Subset

In order to facilitate the experiments on fMoW dataset, we extract a subset from the original fMoW dataset. We randomly choose 10 categories from the original 63 categories. The chosen categories are airport, amusement, aquaculture, archaeological, barn, border, burial, car, construction and crop. Then, we randomly select 1,500 high-resolution images from the training set within the above classes as our training set and use all data belongs to the above categories in the validation set of orginal fMoW as our test set.

F. More Implementation Details of Attention α and β

Here, we describe more implementation details of attention α and β not covered in Zoom-In Network Section 2.

In order to avoid additional computations on overlapped area in the image, each value in the first stage attention α represents the attention on a unique non-overlapped tile in the whole image. The tile sizes are 250×250 for colon cancer, 256×256 for NeedleCamelyon, 250×250 for Functional Map of the World and 3200×3200 for Camelyon16 dataset. For the second stage attention β, the attention maps are computed same as [20]. Each attention value in β is the attention of each input entry.

G. Complete Training Details

Here, we describe the training details not included in Experiments Section 4, such as learning rate, parameters of the optimizer, *etc*.

For the colon cancer dataset, we train our Zoom-In network using the Adam optimizer with a batch size of 5, β_1 of 0.9, β_2 of 0.999 and a learning rate of 0.001 for 100 epochs. We use N of 10 and λ of $1e^{-5}$. The contrastive learning strategy kicks in after training for 10 epochs.

For NeedleCamelyon dataset, we set the learning rate to 0.0001, N to 30 and training epochs to 150. Other settings are the same as for the colon cancer experiment.

For Traffic Sign Recognition, we set the learning rate to 0.001, batch size to 32, N to 10 and training epochs to 150. Since the number of samples for each class is imbalanced, we use a weighted cross-entropy loss that the weights for class empty, 50 limit sign, 70 limit sign and 80 limit sign are $[0.1843, 2.3639, 1.5183, 3.1653]$. Other settings are conformed with the colon cancer experiment.

For Functional Map of the World dataset, we set the learning rate to 0.0001, batch size to 32, N to 30 and train-

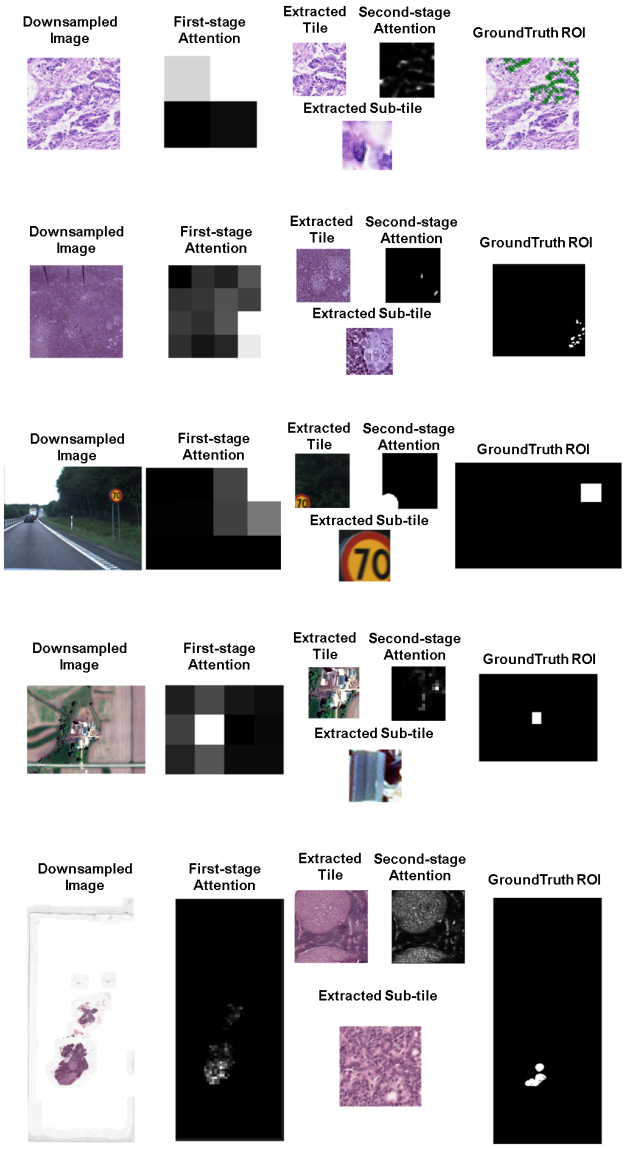

Figure 6. Intermediate results of the proposed Zoom-In network. For each row, we exhibit the visualization of a) colon cancer, b) NeedleCamelyon, c) Traffic Sign Recognition, d) fMoW and e) Camelyon16 datatsets. In each panel we show the downsampled original image, the ground truth ROI mask, the attention masks, the extracted tiles and sub-tiles with the highest first-stage and second-stage attention respectively.

ing epochs to 200. Other settings are the same as for the colon cancer experiment. For EfficientNet-B0, the input images are resized to 896×896 and all other hyper-parameters are consistent with [36].

For Camelyon16 dataset, we set the learning rate to 0.0001, N to 100 and training epochs to 200. Other settings are the same as for the colon cancer experiment. For the baseline models, we only use $5\times$ magnifications for CLAM and MIL. For MRMIL, we use $5\times$ magnifications at the first stage and $20\times$ magnifications at the second stage.

All of our experiments ran on an NVIDIA TITAN Xp 12GB with CUDA version 10.2.

H. Time/Memory-Accuracy Trade-off

In our model, the main hyper-parameter that varies time and memory consumption is the sample size (N). Although the time/memory - accuracy trade-off can be inferred from the ablation study of sample size (N) in Table 4, we show additional details concerning the time-memory trade-off in the following table:

Table 3. Time/Memory-Accuracy Trade-off of Zoom-In Network compared with ATS in the colon cancer dataset experiment.

Method	Sample Sze (N)	Accuracy (%)	Memory (Mb)	Time (ms)
Zoom-In Network	5	93.2	2.43	3.04
	10	78.0	2.55	3.20
	50	79.9	5.03	4.33
ATS	10	90.7	15.83	2.81
	50	90.7	25.77	4.03

I. Ablation Study

In Table 4, we present and ablation study to evaluate the effects of the entropy regularization λ, sample size N and contrastive learning in our Zoom-In network. We examine these hyper-parameters on the colon cancer dataset to show the effects of varying N and λ, as well as to demonstrate the usefulness of the contrastive learning objective. Then, we further justify the contribution of contrastive learning on NeedleCamelyon and Camelyon16 datasets a similar ablation strategy.

Table 4. Ablation study results of λ, N and using contrastive learning. The first, seventh, and ninth rows are the standard hyper-parameter settings used in our experiments and the others are selected to show performance variations for different settings.

Dataset	Entropy Regularization (λ)	Sample Size (N)	Contrastive Learning	Test Accuracy (%)
Colon Cancer	1e-5	10	Yes	95.0
	0	10	Yes	94.0
	1.0	10	Yes	95.0
	1e-5	10	No	94.0
	1e-5	5	Yes	93.2
	1e-5	50	Yes	96.0
NeedleCamelyon	1e-5	30	Yes	76.0
	1e-5	30	No	74.3
Camelyon16	1e-5	100	Yes	81.3
	1e-5	100	No	80.6

J. Limitations: when there is no discriminative information at lower scales

We also examine the behavior of the proposed Zoom-In network on NeedleMNIST dataset introduced by [37]. In

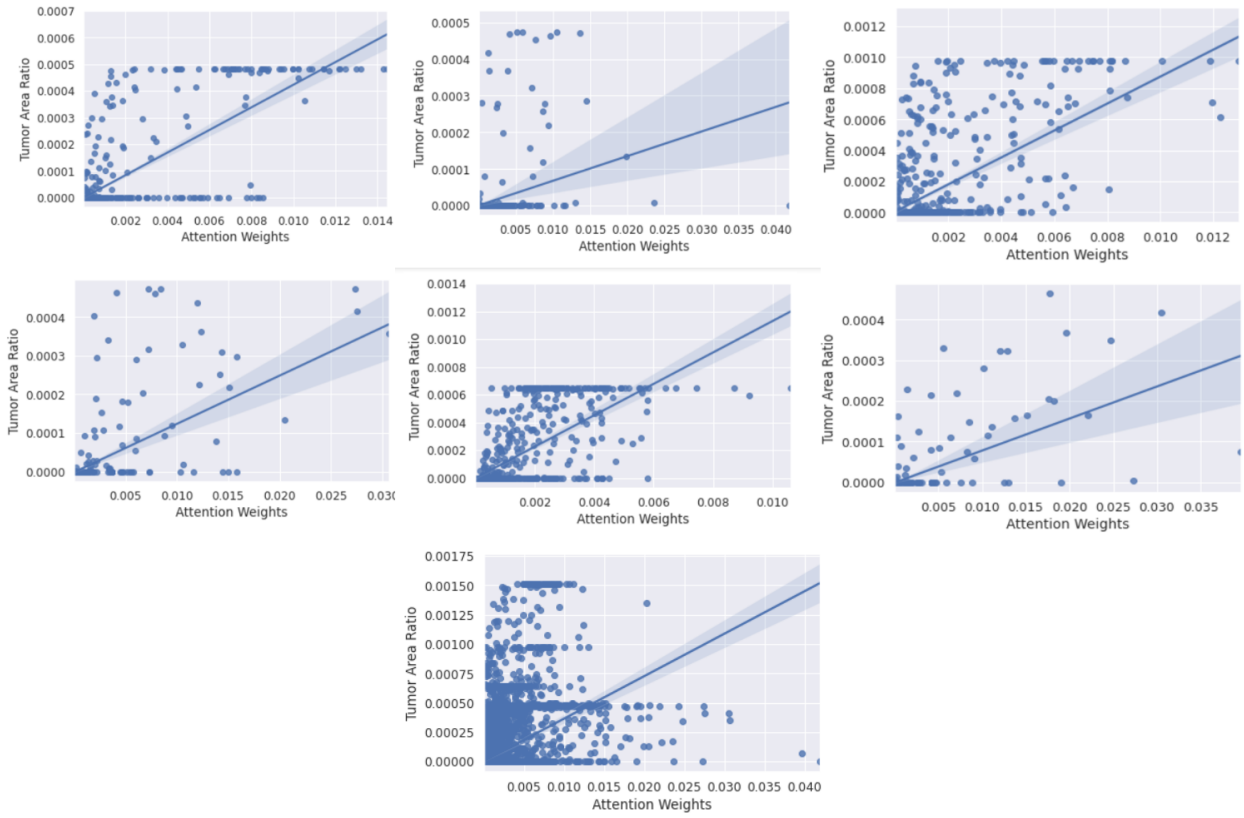

Figure 7. Correlation of attention with ROI abundance, where the x-axis shows the attention weight α_c on each tile and the y-axis corresponds to the ratio of the ROI in a tile, *i.e.*, the franction of annotated ROI relative to the tile area. The plots are produced using images from the Camelyon16 dataset, for which detailed per-pixel annotations for cancer metastases are available. A linear regression fit is estimated for each image and overlayed on the plot to highlight the linear trend. (Top): 6 scatter plots created from 6 individual samples of the Camelyon16 test set. (Bottom): scatter plot created from the attention weight points and metastases ratio points in the whole dataset.

this dataset, images with a size of 1024×1024 are generated via randomly placing 401 MNIST digits on a black image canvas. The task for this dataset is to classify whether there is a digit 3 in a given image. The positive samples in this dataset contain only one digit 3 and 400 distracting digits, that is, any MNIST digits belong to a set of labels $\{0, 1, 2, 4, 5, 6, 7, 8, 9\}$. The negative samples in this dataset have 401 distracting digits. We use the same split for training, validation and test set as described in [37]. The Zoom-In network fails to handle this dataset appropriately because the discriminative information is washed out when downsampling the view of the image. As shown Figure 8, we can hardly recognize the ROI of the downsampled view of NeedleMNIST images at scale $s_1 = 0.25$, which means that the attention network is unable to learn good attention weights that correlate with the ROI. The test accuracy of the Zoom-In network is 0.503, that is just slightly better than random guessing. There are a few possible solutions to address this problem: i) increasing the value of N, s_1 or s_2, at the cost of increased memory consumption; ii) pre-processing the images by a network pre-trained on ROI objects with labels (inferred from pixel-level annotations), in order to obtain feature maps with higher ROI-to-image ratio; and iii) randomly cropping the whole image so that some input images have higher ROI-to-image ratio.

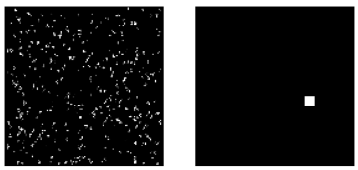

Figure 8. Left: downsampled views of one image in NeedleMNIST dataset at scale $s_1 = 0.25$. Right: the corresponding ROI mask.

Table 5. The architecture of the Zoom-In Network using a LeNet Structure.

$a_\Theta(\cdot)$

Layer	Type
1	Conv(3, 1, 1, 8) + Tanh()
2	Conv(3, 1, 1, 8) + Tanh()
3	Conv(3, 1, 1, 1) + Tanh()
4	GlobalAveragePooling2D()
5	SoftMax()

$b_\Theta(\cdot)$

Layer	Type
1	Conv(3, 1, 1, 8) + Tanh()
2	Conv(3, 1, 1, 8) + Tanh()
3	Conv(3, 1, 1, 1) + SoftMax()

$f_\Theta(\cdot)$

Layer	Type
1	Conv(7, 1, 3, 32) + ReLU()
2	Conv(3, 1, 1, 32) + ReLU()
3	Conv(3, 1, 1, 32) + ReLU()
4	Conv(3, 1, 1, 32) + ReLU()
5	GlobalAveragePooling2D()

$g_\Theta(\cdot)$

Layer	Type
1	fc-n_{class}

K. Model Components

The components of the proposed Zoom-In network are summarized in Figure 2. We consider the LeNet and ResNet16 architecutres for the feature function $f_\Theta(\cdot)$. The choice of LeNet is consistent to the one used in [16,42]. The mapping $f_\Theta(\cdot)$ is implemented by a LeNet5-like architecture; consistent to the one used in [16, 42]. We also considered a ResNet architecture to show our zoom-in strategies is also compatible with modern network architectures. The details of each subnetwork is listed in Table 5 and Table 6. In the table, the convolutional layer is denoted as "Conv" and the kernel size, stride, padding and number of filters are provided in the following brackets. "fc" means the fully-connected layer and the output hidden units is provided after the dash. n_{class} is the number of classes in the task that the model is solving. "Tanh", "ReLU" and "SoftMax" represent the non-linear functions. "GlobalAveragePooling2D" is the global average pooling operation in the spatial dimension of the tensors, functioning the same as https://www.tensorflow.org/api_docs/python/tf/keras/layers/GlobalAveragePooling2D. "ResBlock" is the standard ResNet block [15]. In the brackets, we provide the kernel size, stride, and number of filters.

Table 6. The architecture of the Zoom-In Network using a ResNet16 Structure.

$a_\Theta(\cdot)$

Layer	Type
1	Conv(3, 1, 1, 8) + ReLU()
2	Conv(3, 1, 1, 16) + ReLU()
3	Conv(3, 1, 1, 32) + ReLU()
4	Conv(3, 1, 1, 1) + ReLU()
5	GlobalAveragePooling2D()
6	SoftMax()

$b_\Theta(\cdot)$

Layer	Type
1	Conv(3, 1, 1, 8) + ReLU()
2	Conv(3, 1, 1, 16) + ReLU()
3	Conv(3, 1, 1, 32) + ReLU()
4	Conv(3, 1, 1, 1) + SoftMax()
5	SoftMax()

$f_\Theta(\cdot)$

Layer	Type
1	Conv(3, 1, 1)-32 + ReLU()
2	ResBlock(3, 1, 32)
3	ResBlock(3, 2, 32)
4	ResBlock(3,2, 32)
5	ResBlock(3,2, 32)
6	BatchNorm()+ReLU()
7	GlobalAveragePooling2D()

$g_\Theta(\cdot)$

Layer	Type
1	fc-n_{class}

The attention function $a_\theta(\cdot)$ in (1) is a smaller neural network consisting of a three-layer convolutional network with 8 kernels and ReLu activations, followed by average pooling and a softmax activation to obtain the matrix of attention weights. The attention function $b_\theta(\cdot)$ in (4) is defined similarly.

Finally, the classifier $g_\Theta(\cdot)$ is specified as a single fully connected layer with sigmoid activation. The complete objective for the Zoom-In network is

$$\mathcal{L}(y, x; \Theta) = \mathcal{L}_{\text{ce}}(y, x) + \mathcal{L}_{\text{con}}(x, y = 1) + \mathcal{L}_{\text{er}}(\alpha) + \textstyle\sum_{c \in Q} \mathcal{L}_{\text{er}}(\beta^c), \tag{12}$$

where $\mathcal{L}_{\text{ce}}(y, x)$ is the cross-entropy loss for the image-level binary classification, $\mathcal{L}_{\text{con}}(x, y = 1)$ is the contrastive loss introduced above, and $\mathcal{L}_{\text{er}}(\cdot)$ is the entropy regularization loss for attention matrices α and $\{\beta^c\}_c$. The regularization term [20, 34], $\mathcal{L}_{\text{er}}(p) = -\lambda H(p) = \lambda \sum_i p_i \log(p_i)$, where $H(\cdot)$ is the entropy of a discrete distribution and λ is the trade-off coefficient, is included in the overall objective

to prevent overly sparse attention matrices that may result from overfitting or early converging during training.

L. Leveraging Pixel-Level Annotations

In some tiny object image classification datasets, pixel-level annotations of the ROIs are provided; usually in the form of manually-created segmentation masks. In Table 2, we also report the results of our model using the pixel-level annotations provided in the Camelyon16 dataset. Below we describe how to leverage pixel-level annotations, when available, to further improve the performance of the proposed model.

Incorporating pixel-level annotations via pre-training A good attention function and a feature extractor can be obtained by training the model components as tile-level classifiers [39, 48]. Specifically, assuming pixel-level annotations are available for (a subset of) the images, we can obtain patches consistent in size and scale with view $V(x, s_1)$ for $a_\Theta(V(x, s_1),)$ (of size $h \times w$), $V(x, s_2, c)$ for $b_\Theta(V(x, s_1, c))$ (of size $u \times v$), and with sub-tiles $T_{s_2}(T_{s_1}(x, c)c')$ for $f_\Theta(\cdot)$ (of size $h_2 \times w_2$). Then, we can pair them with labels obtained from the pixel-level annotation so that the extracted patch is assigned $y = 1$ is any of the annotation pixels is of a class different than background and $y = 0$ if all the patch consists of background pixels. Subsequently, we can proceed to pre-train $a_\Theta(V(x, s_1),)$, $b_\Theta(V(x, s_1, c))$ and $f_\Theta(\cdot)$. For the attention functions we convert their output to a scalar prediction using a global average pooling layer, and for $f_\Theta(\cdot)$ we use a single fully connected layer similar to $g_\Theta(\cdot)$, but whose parameters we discard after pre-training.

Incorporating pixel-level supervision Recently, [43] introduced the body mask approach to guide the attention map, by adding a mean squared error (MSE) loss between the attention map for the positive class and the corresponding body segmentation mask to improve the model performance on person re-identification tasks, which is a detection task. The segmentation mask (pixel-level annotation) is represented by a binary matrix of the same size as the original image. If a pixel of the image is in a ROI (not background), the corresponding value in the mask is set to 1, alternatively the value is set to 0 (background). Here, we optimize both attention networks using pixel-level annotations by adding MSE losses to the outputs of $a_\theta(V(x, s_1))$ and $b_\Theta(V(x, s_2, c))$. The MSE losses used in our experiment are

$$\begin{aligned} \mathcal{L}_\alpha(\alpha) &= \textstyle\sum_{c \in C} ||M(V(x, s_1), c) - \alpha_c||_2^2, \\ \mathcal{L}_\beta(\beta^c) &= \textstyle\sum_{c' \in C'} ||M(V(x, s_2), c') - \beta^c_{c'}||_2^2, \end{aligned} \tag{13}$$

where $M(\cdot)$ is a function that returns the binary segmentation mask value for a specific view and location of the image, $\mathcal{L}_\alpha$ is the MSE loss for $\alpha = a_\theta(V(x, s_1))$ and $\mathcal{L}_\beta$ is the MSE loss for $\beta^c = b_\Theta(V(x, s_2, c))$. These two losses are added to (12) when pixel-level annotations are available.

M. Engineering Details for Camelyon16 Experiment

In the Camelyon16 experiment, the input images have irregular size and they are too large to be loaded into RAM directly. The conventional coding implementation of neural networks are not compatible with the above situation. Here, we briefly explain some engineering details about the implementation of our Zoom-In model for Camelyon16 experiment.

Irregular input size. To process the irregular size images with GPU parallelization, we implement a patch generator (*e.g.*, tf.image.extract_patches) to load input images patch by patch, also including the location information of the extracted patches. When training the model, we use indices to keep track of the location of these patches to ensure that our model tracks them correctly;

Image cannot be directly loaded into RAM. OpenSlide is a library that allows to access the local patches of a large image (WSIs) in a variety of resolutions. Our patch generator integrates with OpenSlide, which allows us to access the tiles for computing attentions without instantiating the whole image.

It should be noted that our model is amenable to current GPU computing frameworks like CUDA and we implement our model in Pytorch.

N. Code and Data Availability

The source code of our project will be uploaded at https://github.com/timqqt/pytorch-zoom-in-network.

Colon cancer dataset can be downloaded at https://github.com/MuniNihitha/cancer-detection/tree/master/data/CRCHistoPhenotypes_2016_04_28.

The source code to reproduce NeedleCamelyon and NeedleMNIST dataset is at https://github.com/facebookresearch/Needles-in-Haystacks.

Traffic Sign dataset is avaliable at https://www.cvl.isy.liu.se/research/trafficSigns/.

Functional Map of the world(fMoW) dataset can be found at https://github.com/fMoW/dataset.

Camelyon16 dataset can be found at https://camelyon16.grand-challenge.org/.