Learning When and Where to Zoom with Deep Reinforcement Learning

Burak Uzkent
Department of Computer Science
Stanford University
buzkent@cs.stanford.edu

Stefano Ermon
Department of Computer Science
Stanford University
ermon@cs.stanford.edu

Abstract

While high resolution images contain semantically more useful information than their lower resolution counterparts, processing them is computationally more expensive, and in some applications, e.g. remote sensing, they can be much more expensive to acquire. For these reasons, it is desirable to develop an automatic method to selectively use high resolution data when necessary while maintaining accuracy and reducing acquisition/run-time cost. In this direction, we propose PatchDrop a reinforcement learning approach to dynamically identify when and where to use/acquire high resolution data conditioned on the paired, cheap, low resolution images. We conduct experiments on CIFAR10, CIFAR100, ImageNet and fMoW datasets where we use significantly less high resolution data while maintaining similar accuracy to models which use full high resolution images.

1. Introduction

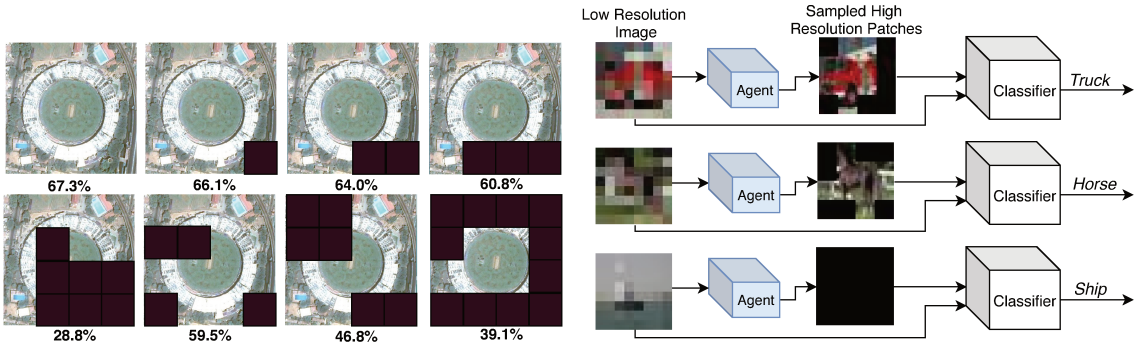
Deep Neural Networks achieve state-of-the-art performance in many computer vision tasks, including image recognition [4], object detection [23], and object tracking [18]. However, one drawback is that they require high quality input data to perform well, and their performance drops significantly on degraded inputs, e.g., lower resolution images [49], lower frame rate videos [28], or under distortions [38]. For example, [45] studied the effect of image resolution, and reported a 14% performance drop on CIFAR10 after downsampling images by a factor of 4.

Nevertheless, downsampling is often performed for computational and statistical reasons [51]. Reducing the resolution of the inputs decreases the number of parameters, resulting in reduced computational and memory cost and mitigating overfitting [2]. Therefore, downsampling is often applied to trade off computational and memory gains with accuracy loss [25]. However, the *same* downsampling level is applied to *all* the inputs. This strategy can be suboptimal because the amount of information loss (e.g., about a label) depends on the input [7]. Therefore, it would be desirable to build an *adaptive* system to utilize a minimal amount of high resolution data while preserving accuracy.

In addition to computational and memory savings, an adaptive framework can also benefit application domains where acquiring high resolution data is particularly expensive. A prime example is remote sensing, where acquiring a high resolution (HR) satellite image is significantly more expensive than acquiring its low resolution (LR) counterpart [26, 32, 9]. For example, LR images with 10m-30m spatial resolution captured by Sentinel-1 satellites [8] are publicly and freely available whereas an HR image with 0.3m spatial resolution captured by DigitalGlobe satellites can cost in the order of 1,000 dollars [6]. Similar examples arise in medical and scientific imaging, where acquiring higher quality images can be more expensive or even more harmful to patients [16, 15].

In all these settings, it would be desirable to be able to adaptively acquire only specific parts of the HR quality input. The challenge, however, is how to perform this selection automatically and efficiently, i.e., *minimizing the number of acquired HR patches while retaining accuracy*. As expected, naive strategies can be highly suboptimal. For example, randomly dropping patches of HR satellite images from the functional Map of the World (fMoW) [3] dataset will significantly reduce accuracy of a trained network as seen in Fig. 1a. As such, an adaptive strategy must learn to identify and acquire useful patches [27] to preserve the accuracy of the network.

To address this challenges, we propose *PatchDrop*, an adaptive data sampling-acquisition scheme which only samples patches of the full HR image that are required for inferring correct decisions, as shown in Fig. 1b. PatchDrop uses LR versions of input images to train an agent in a reinforcement learning setting to sample HR patches only if necessary. This way, the agent learns *when* and *where* to zoom in the parts of the image to sample HR patches. PatchDrop is extremely effective on the functional Map of the World (fMoW) [3] dataset. Surprisingly, we show that we can use only about 40% of full HR images without any



(a) (b)

Figure 1: **Left:** shows the performance of the ResNet34 model trained on the fMoW original images and tested on images with dropped patches. The accuracy of the model goes down with the increased number of dropped patches. **Right:** shows the proposed framework which dynamically drops image patches conditioned on the low resolution images.

significant loss of accuracy. Considering this number, we can save in the order of 100,000 dollars when performing a computer vision task using expensive HR satellite images at global scale. We also show that PatchDrop performs well on traditional computer vision benchmarks. On ImageNet, it samples about 50% of HR images on average with a minimal loss in the accuracy. On a different task, we then increase the run-time performance of patch-based CNNs, BagNets [1], by 2× by reducing the number of patches that need to be processed using PatchDrop. Finally, leveraging the learned patch sampling policies, we generate hard positive training examples to boost the accuracy of CNNs on ImageNet and fMoW by 2-3%.

2. Related Work

Dynamic Inference with Residual Networks Similarly to DropOut [36], [13] proposed a stochastic layer dropping method when training the Residual Networks [11]. The probability of survival linearly decays in the deeper layers following the hypothesis that low-level features play key roles in correct inference. Similarly, we can decay the likelihood of survival for a patch w.r.t its distance from image center based on the assumption that objects will be dominantly located in the central part of the image. Stochastic layer dropping provides only training time compression. On the other hand, [43, 47] proposes reinforcement learning settings to drop the blocks of ResNet in both training and test time conditionally on the input image. Similarly, by replacing *layers with patches*, we can drop more patches from easy samples while keeping more from ambiguous ones.

Attention Networks Attention methods have been explored to localize semantically important parts of images [42, 31, 41, 39]. [42] proposes a Residual Attention network that replaces the residual identity connections from [11] with residual attention connections. By residually learning feature guiding, they can improve recognition accuracy on different benchmarks. Similarly, [31] proposes a differentiable saliency-based distortion layer to spatially sample input data given a task. They use LR images in the saliency network that generates a grid highlighting semantically important parts of the image space. The grid is then applied to HR images to magnify the important parts of the image. [21] proposes a perspective-aware scene parsing network that locates small and distant objects. With a two branch (coarse and fovea) network, they produce coarse and fine level segmentations maps and fuse them to generate final map. [50] adaptively resizes the convolutional patches to improve segmentation of large and small size objects. [24] improves object detectors using pre-determined fixed anchors with adaptive ones. They divide a region into a fixed number of sub-regions recursively whenever the zoom indicator given by the network is high. Finally, [30] proposes a sequential region proposal network (RPN) to learn object-centric and less scattered proposal boxes for the second stage of the Faster R-CNN [33]. These methods are tailored for certain tasks and condition the attention modules on HR images. On the other hand, we present a general framework and condition it on LR images.

Analyzing Degraded Quality Input Signal There has been a relatively small volume of work on improving CNNs' performance using degraded quality input signal [20]. [37] uses knowledge distillation to train a student network using degraded input signal and the predictions of a teacher network trained on the paired higher quality signal. Another set of studies [45, 48] propose a novel method to perform domain adaptation from the HR network to a LR network. [29] pre-trains the LR network using the HR data and finetunes it using the LR data. Other domain adaptation methods focus on person re-identification with LR

images [14, 22, 44]. All these methods boost the accuracy of the networks on LR input data, however, they make the assumption that the quality of the input signal is fixed.

3. Problem statement

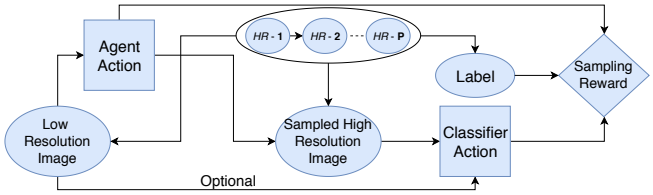


Figure 2: Our Bayesian Decision influence diagram. The LR images are observed by the agent to sample HR patches. The classifier then observes the agent-sampled HR image together with the LR image to perform prediction. The ultimate goal is to choose an action to sample a masked HR image to maximize the expected utilities considering the accuracy and the cost of using/sampling HR image.

We formulate the *PatchDrop* framework as a two step episodic Markov Decision Process (MDP), as shown in the influence diagram in Fig. 2. In the diagram, we represent the random variables with a circle, actions with a square, and utilities with a diamond. A high spatial resolution image, x_h, is formed by equal size patches with zero overlap $x_h = (x_h^1, x_h^2, \cdots, x_h^P)$, where P represents the number of patches. In contrast with traditional computer vision settings, x_h is latent, *i.e.*, it is *not observed by the agent*. $y \in \{1, \cdots, N\}$ is a categorical random variable representing the (unobserved) label associated with x_h, where N is the number of classes. The random variable *low spatial resolution image*, x_l, is the lower resolution version of x_h. x_l is initially observed by the agent in order to choose the binary action array, $\mathbf{a_1} \in \{0, 1\}^P$, where $\mathbf{a_1^p} = 1$ means that the agent would like to sample the p-th HR patch x_h^p. We define the patch sampling policy model parameterized by θ_p, as

$$\pi_1(\mathbf{a_1}|x_l; \theta_p) = p(\mathbf{a_1}|x_l; \theta_p), \tag{1}$$

where $\pi_1(x_l; \theta_p)$ is a function mapping the observed LR image to a probability distribution over the patch sampling action $\mathbf{a_1}$. Next, the random variable *masked HR image*, x_h^m, is formed using $\mathbf{a_1^p}$ and x_h^p, with the masking operation formulated as $x_h^m = x_h \bigodot \mathbf{a_1}$. The first step of the MDP can be modeled with a joint probability distribution over the random variables, x_h, y, x_h^m, and x_l, and action $\mathbf{a_1}$, as

$$\begin{aligned} p(x_h, x_h^m, x_l, y, \mathbf{a_1}) = p(x_h)p(y|x_h)p(x_l|x_h) \\ \cdot p(\mathbf{a_1}|x_l; \theta_p)p(x_h^m|\mathbf{a_1}, x_h). \end{aligned} \tag{2}$$

In the second step of the MDP, the agent observes the random variables, x_h^m and x_l, and chooses an action $\mathbf{a_2} \in \{1, \cdots, N\}$. We then define the class prediction policy as follows:

$$\pi_2(\mathbf{a_2}|x_h^m, x_l; \theta_{cl}) = p(\mathbf{a_2}|x_h^m, x_l; \theta_{cl}), \tag{3}$$

where π_2 represents a classifier network parameterized by θ_{cl}. The overall objective, J, is then defined as maximizing the expected utility, R represented by

$$\max_{\theta_p, \theta_{cl}} J(\theta_p, \theta_{cl}) = \mathbb{E}_p[R(\mathbf{a_1}, \mathbf{a_2}, y)], \tag{4}$$

where the utility depends on $\mathbf{a_1}$, $\mathbf{a_2}$, and y. The reward penalizes the agent for selecting a large number of high-resolution patches (e.g., based on the norm of $\mathbf{a}_1$) and includes a classification loss evaluating the accuracy of $\mathbf{a}_2$ given the true label y (e.g., cross-entropy or 0-1 loss).

4. Proposed Solution

4.1. Modeling the Policy Network and Classifier

In the previous section, we formulated the task of *PatchDrop* as a two step episodic MDP. Here, we detail the action space and how the policy distributions for $\mathbf{a_1}$ and $\mathbf{a_2}$ are modelled. To represent our discrete action space for $\mathbf{a_1}$, we divide the image space into equal size patches with no overlaps, resulting in P patches, as shown in Fig. 3. In this study, we use $P = 16$ regardless of the size of the input image and leave the task of choosing variable size bounding boxes as a future work. In the first step of the two step MDP, the policy network, f_p, *outputs the probabilities for all the actions at once* after observing x_l. An alternative approach could be in the form of a Bayesian framework where $\mathbf{a_1^p}$ is conditioned on $\mathbf{a_1^{1:p-1}}$ [7, 30]. However, the proposed concept of *outputting all the actions at once* provides a more efficient decision making process for patch sampling.

In this study, we model the action likelihood function of the policy network, f_p, by multiplying the probabilities of the individual high-resolution patch selections, represented by patch-specific Bernoulli distributions as follows:

$$\pi_1(\mathbf{a_1}|x_l, \theta_p) = \prod_{p=1}^{P} s_p^{\mathbf{a_1^p}} (1 - s_p)^{(1 - \mathbf{a_1^p})}, \tag{5}$$

where s_p represents the prediction vector formulated as

$$s_p = f_p(x_l; \theta_p). \tag{6}$$

To get probabilistic values, $s_p \in [0, 1]$, we use a sigmoid function on the final layer of the policy network.

The next set of actions, $\mathbf{a_2}$, is chosen by the classifier, f_{cl}, using the sampled HR image x_h^m and the LR input x_l. The upper stream of the classifier, f_{cl}, uses the sampled HR images, x_h^m, whereas the bottom stream uses the LR images, x_l, as shown in Fig. 3. Each one outputs probability

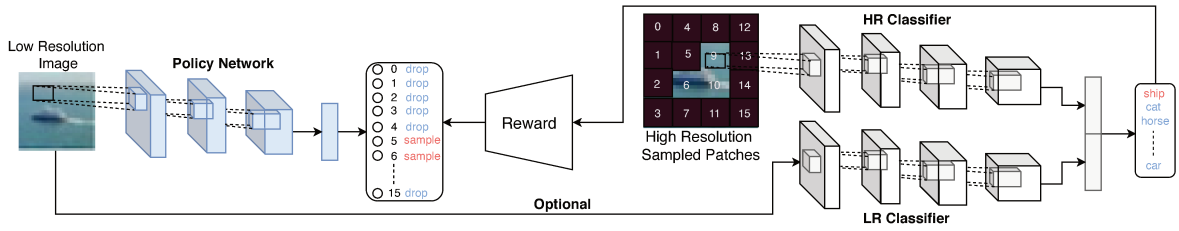


Figure 3: The workflow of the *PatchDrop* formulated as a two step episodic MDP. The agent chooses actions conditioned on the LR image, and only agent sampled HR patches together with LR images are jointly used by the two-stream classifier network. We note that the LR network can be disconnected from the pipeline to only rely on selected HR patches to perform classification. When disconnecting LR network, the policy network samples more patches to maintain accuracy.

distributions, s_{cl}^{l} and $s_{cl}^{h^m}$, for class labels using a softmax layer. We then compute the weighted sum of predictions via

$$s_{cl} = (S/P)s_{cl}^{h^m} + (1 - S/P)s_{cl}^{l}, \tag{7}$$

where S represents the number of sampled patches. To form $\mathbf{a_2}$, we use the maximally probable class label: *i.e.*, $\mathbf{a_2^j} = 1$ if $s_{cl}^{j} = \max(s_{cl})$ and $\mathbf{a_2^j} = 0$ otherwise where j represents the class index. In this set up, if the policy network samples *no HR patch*, we completely rely on the LR classifier, and the impact of the HR classifier increases linearly with the number of sampled patches.

4.2. Training the PatchDrop Network

After defining the two step MDP and modeling the policy and classifier networks, we detail the training procedure of *PatchDrop*. The goal of training is to learn the optimal parameters of θ_p and θ_{cl}. Because the actions are discrete, we cannot use the reparameterization trick to optimize the objective w.r.t. θ_p. To optimize the parameters θ_p of f_p, we need to use model-free reinforcement learning algorithms such as Q-learning [46] and policy gradient [40]. Policy gradient is more suitable in our scenario since the number of unique actions the policy network can choose is 2^P and increases exponentially with P. For this reason, we use the $REINFORCE$ method [40] to optimize the objective w.r.t θ_p using

$$\nabla_{\theta_p} J = \mathbb{E}[R(\mathbf{a_1}, \mathbf{a_2}, y) \nabla_{\theta_p} \log \pi_{\theta_p}(\mathbf{a_1}|x_l)]. \tag{8}$$

Averaging across a mini-batch via Monte-Carlo sampling produces an unbiased estimate of the expected value, but with potentially large variance. Since this can lead to an unstable training process [40], we replace $R(\mathbf{a_1}, \mathbf{a_2}, y)$ in Eq. 8 with the advantage function to reduce the variance:

$$\nabla_{\theta_p} J = \mathbb{E}[A \sum_{p=1}^{P} \nabla_{\theta_p} \log(s_p \mathbf{a_1^p} + (1 - s_p)(1 - \mathbf{a_1^p}))], \tag{9}$$

$$A(\mathbf{a_1}, \hat{\mathbf{a}}_\mathbf{1}, \mathbf{a_2}, \hat{\mathbf{a}}_\mathbf{2}) = R(\mathbf{a_1}, \mathbf{a_2}, y) - R(\hat{\mathbf{a}}_\mathbf{1}, \hat{\mathbf{a}}_\mathbf{2}, y), \tag{10}$$

where $\hat{\mathbf{a}}_\mathbf{1}$ and $\hat{\mathbf{a}}_\mathbf{2}$ represent the baseline action vectors. To get $\hat{\mathbf{a}}_\mathbf{1}$, we use the most likely action vector proposed by the policy network: *i.e.*, $\mathbf{a_1^i} = 1$ if $s_p^i > 0.5$ and $s_p^i = 0$ otherwise. The classifier, f_{cl}, then observes x_l and $\hat{x}_h^m$ sampled using $\hat{\mathbf{a}}_\mathbf{1}$, on two branches and outputs the predictions, $\hat{s}_{cl}$, from which we get $\hat{\mathbf{a}}_\mathbf{2}^\mathbf{j}$: *i.e.*, $\hat{\mathbf{a}}_\mathbf{2}^\mathbf{j} = 1$ if $\hat{s}_{cl}^{j} = \max(\hat{s}_{cl})$ and $\hat{\mathbf{a}}_\mathbf{2}^\mathbf{j} = 0$ otherwise where j represent the class index. The advantage function assigns the policy network *a positive value* only when the action vector sampled from Eq. 5 produces higher reward than the action vector with maximum likelihood, which is known as a self-critical baseline [34].

Finally, in this study we use the temperature scaling method [40] to encourage exploration during training time by bounding the probabilities of the policy network as

$$s_p = \alpha s_p + (1 - \alpha)(1 - s_p), \tag{11}$$

where $\alpha \in [0, 1]$.

Pre-training the Classifier After formulating our reinforcement learning setting for training the policy network, we first pre-train the two branches of f_{cl}, f_{cl}^h and f_{cl}^l, on $x_h \in \mathcal{X}_h$ and $x_l \in \mathcal{X}_l$. We assume that $\mathcal{X}_h$ is observable in the training time. The network trained on $\mathcal{X}_h$ can perform reasonably (Fig. 1a) when the patches are dropped at test time with a *fixed policy*, forming x_h^m. We then use this observation to pre-train the policy network, f_p, to dynamically learn to drop patches while keeping the parameters of f_{cl}^h and f_{cl}^l fixed.

Pre-training the Policy Network (Pt) After training the two streams of the classifier, f_{cl}, we pre-train the policy network, f_p, using the proposed reinforcement learning setting while fixing the parameters of f_{cl}. In this step, we only use f_{cl}^h to estimate the expected reward when learning θ_p. This is because we want to train the policy network to understand which patches contribute most to correct decisions made by the HR image classifier, as shown in Fig. 1a.

Finetuning the Agent and HR Classifier (Ft-1) To further boost the accuracy of the policy network, f_p, we jointly finetune the policy network and HR classifier, f_{cl}^h. This way, the HR classifier can adapt to the sampled images, x_h^m,

Input: Input($\mathrm{X}_l, \mathcal{Y}, \mathcal{C}$) $\quad \mathrm{X}_l = \{x_l^1, x_l^2, ..., x_l^N\}$
for $k \leftarrow 0$ *to* K_1 **do**
  $s_p \leftarrow f_p(x_l; \theta_p)$
  $s_p \leftarrow \alpha + (1 - s_p)(1 - \alpha)$
  $\mathbf{a_1} \sim \pi_1(a_1|s_p)$
  $x_h^m = x_h \bigodot \mathbf{a_1}$
  $\mathbf{a_2} \leftarrow f_{cl}^h(x_h^m; \theta_{cl}^h)$
  Evaluate Reward $R(\mathbf{a_1}, \mathbf{a_2}, y)$
  $\theta_p \leftarrow \theta_p + \nabla\theta_p$
end
for $k \leftarrow 0$ *to* K_2 **do**
  Jointly Finetune θ_{cl}^h and θ_p using f_{cl}^h
end
for $k \leftarrow 0$ *to* K_3 **do**
  Jointly Finetune θ_{cl}^h and θ_p using f_{cl}^h and f_{cl}^l
end

Algorithm 1: PatchDrop Pseudocode

while the policy network learns new policies in line with it. The LR classifier, f_{cl}^l, is not included in this step.

Finetuning the Agent and HR Classifier (Ft-2) In the final step of the training stage, we jointly finetune the policy network, f_p, and f_{cl}^h *with the addition* of f_{cl}^l into the classifier f_{cl}. This way, the policy network can learn policies to drop further patches with the existence of the LR classifier. We combine the HR and LR classifiers using Eq. 7. Since the input to f_{cl}^l does not change, we keep θ_{cl}^l fixed and only update θ_{cl}^h and θ_p. The algorithm for the *PatchDrop* training stage is shown in Alg. 1. Upon publication, we will release the code to train and test PatchDrop.

5. Experiments

5.1. Experimental Setup

Datasets and Metrics We evaluate *PatchDrop* on the following datasets: (1) CIFAR10, (2) CIFAR100, (3) ImageNet [4] and (4) functional map of the world (fMoW) [3]. To measure its performance, we use image recognition accuracy and the number of dropped patches (cost).

Implementation Details In CIFAR10/CIFAR100 experiments, we use a ResNet8 for the policy and ResNet32 for the classifier networks. The policy and classifier networks use 8×8px and 32×32px images. In ImageNet/fMoW, we use a ResNet10 for the policy network and ResNet50 for the classifier. The policy network uses 56×56px images whereas the classifier uses 224×224px images. We initialize the weights of the LR classifier with HR classifier [29] and use Adam optimizer in all our experiments [17]. Finally, initially we set the exploration/exploitation parameter, α, to 0.7 and increase it to 0.95 linearly over time.

Reward Function We choose $R = 1 - \left(\frac{|\mathbf{a_1}|_1}{P}\right)^2$ if $y = \hat{y}(\mathbf{a_2})$ and -σ otherwise as a reward. Here, $\hat{y}$ and y represent the predicted class by the classifier after the observation of x_h^m and x_l and the true class, respectively. The proposed reward function quadratically increases the reward w.r.t the number of dropped patches. To adjust the trade-off between accuracy and the number of sampled patches, we introduce σ and setting it to a large value encourages the agent to sample more patches to preserve accuracy.

5.2. Baseline and State-of-The-Art Models

No Patch Sampling/No Patch Dropping In this case, we simply train a CNN on LR or HR images with cross-entropy loss without any domain adaptation and test it on LR or HR images. We call them LR-CNN and HR-CNN.

Fixed and Stochastic Patch Dropping We propose two baselines that sample central patches along the horizontal and vertical axes of the image space and call them Fixed-H and Fixed-V. We list the sampling priorities for the patches in this order 5,6,9,10,13,14,1,2,0,3,4,7,8,11,15 for *Fixed-H*, and 4,5,6,7,8,9,10,11,12,13,14,15,0,1,2,3 for *Fixed-V*. The patch IDs are shown in Fig. 3. Using a similar hypothesis, we then design a stochastic method that decays the survival likelihood of a patch w.r.t the euclidean distance from the center of the patch p to the image center.

Super-resolution We use SRGAN [19] to learn to upsample LR images and use the SR images in the downstream tasks. This method only improves accuracy and increases computational complexity since SR images have the same number of pixels with HR images.

Attention-based Patch Dropping In terms of the state-of-the art models, we first compare our method to the Spatial Transformer Network (STN) by [31]. We treat their saliency network as the policy network and sample the top S activated patches to form masked images for classifier.

Domain Adaptation Finally, we use two of the state-of-the-art domain adaptation methods by [45, 37] to improve recognition accuracy on LR images. These methods are based on Partially Coupled Networks (PCN), and Knowledge Distillation (KD) [12].

The LR-CNN, HR-CNN, PCN, KD, and SRGAN are standalone models and always use full LR or HR image. For this reason, we have same values for them in Pt, Ft-1, and Ft-2 steps and show them in the upper part of the tables.

5.3. Experiments on fMoW

One application domain of the PatchDrop is remote sensing where LR images are significantly cheaper than HR images. In this direction, we test the PatchDrop on functional Map of the World [3] consisting of HR satellite images. We use 350,000, 50,000 and 50,000 images as training, validation and test images. After training the classifiers, we pre-

Figure 4: Policies learned on the fMoW dataset. In columns 5 and 8, Ft-2 model does not sample any HR patches and the LR classifier is used. Ft-1 model samples more patches as it does not utilize LR classifier.

train the policy network for 63 epochs with a learning rate of 1e-4 and batch size of 1024. Next, we finetune (Ft-1 and Ft-2) the policy network and HR classifiers with the learning rate of 1e-4 and batch size of 128. Finally, we set σ to 0.5, 20, and 20 in the pre-training, and fine-tuning steps.

As seen in Table 1, PatchDrop samples only about 40% of each HR image on average while increasing the accuracy of the network using the full HR images to 68.3%. Fig. 4 shows some examples of how the policy network chooses actions conditioned on the LR images. When the image contains a field with uniform texture, the agent samples a small number of patches, as seen in columns 5, 8, 9 and 10. On the other hand, it samples patches from the buildings when the ground truth class represents a building, as seen in columns 1, 6, 12, and 13.

	Acc. (%) (Pt)	S	Acc. (%) (Ft-1)	S	Acc. (%) (Ft-2)	S
LR-CNN	61.4	0	61.4	0	61.4	0
SRGAN [19]	62.3	0	62.3	0	62.3	0
KD [37]	63.1	0	63.1	0	63.1	0
PCN [45]	63.5	0	63.5	0	63.5	0
HR-CNN	67.3	16	67.3	16	67.3	16
Fixed-H	47.7	7	63.3	6	64.9	6
Fixed-V	48.3	7	63.2	6	64.7	6
Stochastic	29.1	7	57.1	6	63.6	6
STN [31]	46.5	7	61.8	6	64.8	6
PatchDrop	**53.4**	**7**	**67.1**	**5.9**	**68.3**	**5.2**

Table 1: The performance of the proposed *PatchDrop* and baseline models on the fMoW dataset. S represents the average number of sampled patches. Ft-1 and Ft-2 represent the finetuning steps with single and two stream classifiers.

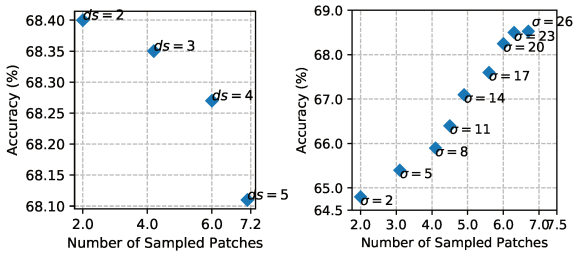


Figure 5: **Left:** The accuracy and number of sampled patches by the policy network w.r.t downsampling ratio used to get LR images for the policy network and classifier. **Right:** The accuracy and number of sampled patches w.r.t to σ parameter in the reward function in the joint finetuning steps (ds=4). It is set to 0.5 in the pre-training step.

Also, we perform experiments with different downsampling ratios and σ values in the reward function. This way, we can observe the trade-off between the number of sampled patches and accuracy. As seen in Fig. 5, as we increase the downsampling ratio we zoom into more patches to maintain accuracy. On the other hand, with increasing σ, we zoom into more patches as larger σ value penalizes the policies resulting in unsuccessful classification.

Experiments on CIFAR10/CIFAR100 Although CIFAR datasets already consists of LR images, we believe that conducting experiments on standard benchmarks is useful to characterize the model. For CIFAR10, after training the classifiers, we pre-train the policy network with a batch size of 1024 and learning rate of 1e-4 for 3400 epochs. In the joint finetuning stages, we keep the learning rate, reduce the batch size to 256, and train the policy and HR classifier networks for 1680 and 990 epochs, respectively. σ is set to -0.5 in the pre-training stage and -5 in the joint finetuning stages whereas α is tuned to 0.8. Our CIFAR100 methods are similar to the CIFAR10 ones, including hyper-parameters.

As seen in Table 2, *PatchDrop* drops about 56% of the patches in the original image space in CIFAR10, all the while with minimal loss in the overall accuracy. In the case of CIFAR100, we observe that it samples 2.2 patches more than the CIFAR10 experiment, on average, which might be due to higher complexity of the CIFAR100 dataset.

Experiments on ImageNet Next, we test the PatchDrop on ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [35]. It contains 1.2 million, 50,000, and 150,000 training, validation and test images. For augmentation, we use randomly cropping 224×224px

Figure 6: Policies learned on ImageNet. In columns 3 and 8, Ft-2 model does not sample any HR patches and the LR classifier is used. Ft-1 model samples more patches as it does not use the LR classifier.

	CIFAR10				CIFAR100				ImageNet			
	Acc. (%) (Pt)	Acc. (%) (Ft-1)	Acc. (%) (Ft-2)	S (Pt,Ft-1,Ft-2)	Acc. (%) (Pt)	Acc. (%) (Ft-1)	Acc. (%) (Ft-2)	S (Pt,Ft-1,Ft-2)	Acc. (%) (Pt)	Acc. (%) (Ft-1)	Acc. (%) (Ft-2)	S (Pt,Ft-1,Ft-2)
LR-CNN	75.8	75.8	75.8	0,0,0	55.1	55.1	55.1	0,0,0	58.1	58.1	58.1	0,0,0
SRGAN [19]	78.8	78.8	78.8	0,0,0	56.1	56.1	56.1	0,0,0	63.1	63.1	63.1	0,0,0
KD [37]	81.8	81.8	81.8	0,0,0	61.1	61.1	61.1	0,0,0	62.4	62.4	62.4	0,0,0
PCN [37]	83.3	83.3	83.3	0,0,0	62.6	62.6	62.6	0,0,0	63.9	63.9	63.9	0,0,0
HR-CNN	92.3	92.3	92.3	16,16,16	69.3	69.3	69.3	16,16,16	76.5	76.5	76.5	16,16,16
Fixed-H	71.2	83.8	85.2	9,8,7	48.5	65.8	67.0	9,10,10	48.8	68.6	70.4	10,9,8
Fixed-V	64.7	83.4	85.1	9,8,7	46.2	65.5	67.2	9,10,10	48.4	68.4	70.8	10,9,8
Stochastic	40.6	82.1	83.7	9,8,7	27.6	63.2	64.8	9,10,10	38.6	66.2	68.4	10,9,8
STN [31]	66.9	85.2	87.1	9,8,7	41.1	64.3	66.4	9,10,10	58.6	69.4	71.4	10,9,8
PatchDrop	**80.6**	**91.9**	**91.5**	**8.5,7.9,6.9**	**57.3**	**69.3**	**70.4**	**9,9.9,9.1**	**60.2**	**74.9**	**76.0**	**10.1,9.1,7.9**

Table 2: The results on CIFAR10, CIFAR100 and ImageNet datasets. S represents the average number of sampled patches per image. The Pt, Ft-1 and Ft-2 represent the pre-training and finetuning steps with single and two stream classifiers.

area from the 256×256px images and perform horizontal flip augmentation. After training the classifiers, we pre-train the policy network for 95 epochs with a learning rate of 1e-4 and batch size of 1024. We then perform the first fine-tuning stage and jointly finetune the HR classifier and policy network for 51 epochs with the learning rate of 1e-4 and batch size of 128. Finally, we add the LR classifier and jointly finetune the policy network and HR classifier for 10 epochs with the same learning rate and batch size. We set σ to 0.1, 10, and 10 for pre-training and fine-tuning steps.

As seen in Table 2, we can maintain the accuracy of the HR classifier while dropping 56% and 50% of the patches with the Ft-1 and Ft-2 model. Also, we show the learned policies on ImageNet in Fig. 6. The policy network decides to sample no patch when the input is relatively easier as in column 3, and 8.

Analyzing Policy Network's Actions To better understand the sampling actions of policy network, we visualize the accuracy of the classifier w.r.t the number of sampled patches as shown in Fig. 7 (left). Interestingly, the accuracy of the classifier is *inversely* proportional to the number of sampled patches. We believe that this occurs because the policy network samples more patches from the challenging and ambiguous cases to ensure that the classifier successfully predicts the label. On the other hand, it successfully learns when to sample *no patches*. However, it samples no patch (S=0) 7% of the time on average in comparison to sampling 4≤S≤7 50% of the time. Increasing the ratio for S=0 is a future work of this study. Finally, Fig. 7 (right) displays the probability of sampling a patch given its position. We see that the policy network learns to sample the central patches more than the peripheral patches as expected.

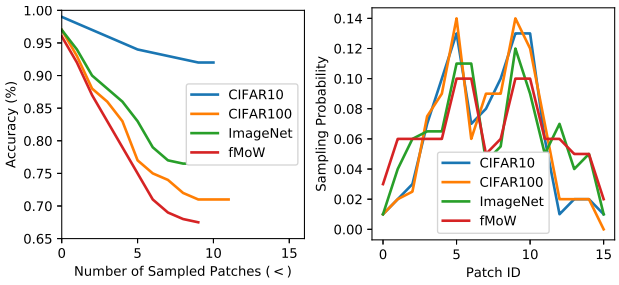


Figure 7: **Left:** The accuracy w.r.t the average number of sampled patches by the policy network. **Right:** Sampling probability of the patch IDs (See Fig. 3 for IDs).

6. Improving Run-time Complexity of BagNets

Next, we use PatchDrop to decrease the run-time complexity of local CNNs, such as BagNets. The BagNets have recently been proposed as a novel image recognition architecture [1]. They run a CNN on image patches independently and sum up class-specific spatial probabilities. Surprisingly, the BagNets perform similarly to CNNs that process the full image in one shot. This concept fits perfectly to PatchDrop as it learns to select semantically useful local patches which can be fed to a BagNet. This way, the BagNet is not trained on all the patches from the image but only on *useful patches*. By dropping redundant patches, we can then speed it up and improve its accuracy. In this case, we first train the BagNet on all the patches and pre-train the policy network on LR images ($4\times$) to learn patches important for BagNet. Using LR images and a shallow network (ResNet8), we reduce the run-time overhead introduced by the agent to 3% of the CNN (ResNet32) using HR images. Finally, we jointly finetune (Ft-1) the policy network and BagNet. We illustrate our apporoach in Fig. 8.

We perform experiments on CIFAR10 and show the results in Table 3. The proposed Conditional BagNet using PatchDrop improves the accuracy of BagNet by 7% closing the gap between global CNNs and local CNNs. Additionally, it decreases the run-time complexity by 50%, significantly reducing the gap between local CNNs and global CNNs in terms of run-time complexity[1]. The increase in the speed can be further improved by running different GPUs on the selected patches in parallel at test time.

Finally, utilizing learned masks to avoid convolutional operations in the layers of global CNN is another promising direction of our work. [10] drops spatial blocks of the feature maps of CNNs in training time to perform stronger regularization than DropOut [36]. Our method, on the other hand, can drop blocks of the feature maps dynamically in both training and test time.

[1]The run-times are measured on Intel i7-7700K CPU@4.20GHz

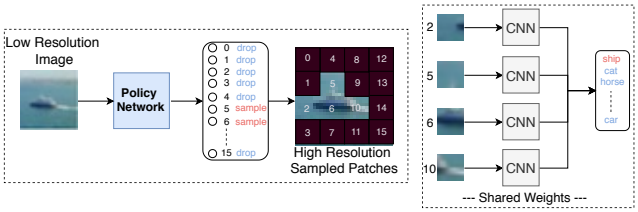


Figure 8: Dynamic BagNet. The policy network processes LR image and sample HR patches to be processed independently by CNN. More details on BagNet can be found in [1].

	Acc. (%) (Pt)	S	Acc. (%) (Ft-1)	S	Run-time. (%) (ms)
BagNet (No Patch Drop) [1]	85.6	16	85.6	16	192
CNN (No Patch Drop)	92.3	16	92.3	16	77
Fixed-H	67.7	10	86.3	9	98
Fixed-V	68.3	10	86.2	9	98
Stochastic	49.1	10	83.1	9	98
STN [19]	67.5	10	86.8	9	112
BagNet (PatchDrop)	**77.4**	**9.5**	**92.7**	**8.5**	98

Table 3: The performance of the PatchDrop and other models on improving BagNet on CIFAR10 dataset. We use a similar set up to our previous CIFAR10 experiments.

7. Conditional Hard Positive Sampling

PatchDrop can also be used to generate hard positives for data augmentation. In this direction, we utilize the masked images, $\mathcal{X}_h^m$, learned by the policy network (Ft-1) to generate hard positive examples to better train classifiers. To generate conditional hard positive examples, we choose the number of patches to be masked, M, from a uniform distribution with minimum and maximum values of 1 and 4. Next, given s_p by the policy network, we choose M patches with the highest probabilities and mask them and use the masked images to train the classifier. Finally, we compare our approach to CutOut [5] which randomly cuts/masks image patches for data augmentation. As shown in Table 4, our approach leads to higher accuracy in all the datasets when using original images, $\mathcal{X}_h$, in test time. This shows that the policy network learns to select informative patches.

	CIFAR10 (%) (ResNet32)	CIFAR100 (%) (ResNet32)	ImageNet (%) (ResNet50)	fMoW (%) (ResNet34)
No Augment.	92.3	69.3	76.5	67.3
CutOut [5]	93.5	70.4	76.5	67.6
PatchDrop	**93.9**	**71.0**	**78.1**	**69.6**

Table 4: Results with different augmentation methods.

8. Conclusion

In this study, we proposed a novel reinforcement learning setting to train a policy network to learn *when* and *where* to sample high resolution patches conditionally on the low resolution images. Our method can be highly beneficial in domains such as remote sensing where high quality data is significantly more expensive than the low resolution counterpart. In our experiments, on average, we drop a 40-60$\%$ portion of each high resolution image while preserving similar accuracy to networks which use full high resolution images in ImageNet and fMoW. Also, our method significantly improves the run-time efficiency and accuracy of BagNet, a patch-based CNNs. Finally, we used the learned policies to generate hard positives to boost classifiers' accuracy on CIFAR, ImageNet and fMoW datasets.

Acknowledgements

This research was supported by Stanford's Data for Development Initiative and NSF grants 1651565 and 1733686.

References

[1] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019. 2, 8

[2] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 1

[3] 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. 1, 5

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 5

[5] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 8

[6] Jonathan RB Fisher, Eileen A Acosta, P James Dennedy-Frank, Timm Kroeger, and Timothy M Boucher. Impact of satellite imagery spatial resolution on land use classification accuracy and modeled water quality. *Remote Sensing in Ecology and Conservation*, 4(2):137–149, 2018. 1

[7] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6926–6935, 2018. 1, 3

[8] Dirk Geudtner, Ramón Torres, Paul Snoeij, Malcolm Davidson, and Björn Rommen. Sentinel-1 system capabilities and applications. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 1457–1460. IEEE, 2014. 1

[9] Pedram Ghamisi and Naoto Yokoya. Img2dsm: Height simulation from single imagery using conditional generative adversarial net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):794–798, 2018. 1

[10] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018. 8

[11] 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. 2

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5

[13] 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. Springer, 2016. 2

[14] Jiening Jiao, Wei-Shi Zheng, Ancong Wu, Xiatian Zhu, and Shaogang Gong. Deep low-resolution person re-identification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[15] Eunhee Kang, Junhong Min, and Jong Chul Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical physics*, 44(10):e360–e375, 2017. 1

[16] Justin Ker, Lipo Wang, Jai Rao, and Tchoyoson Lim. Deep learning applications in medical image analysis. *Ieee Access*, 6:9375–9389, 2017. 1

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1

[19] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 5, 6, 7, 8

[20] Pei Li, Loreto Prieto, Domingo Mery, and Patrick J Flynn. On low-resolution face recognition in the wild: Comparisons and new techniques. *IEEE Transactions on Information Forensics and Security*, 14(8):2000–2012, 2019. 2

[21] Xin Li, Zequn Jie, Wei Wang, Changsong Liu, Jimei Yang, Xiaohui Shen, Zhe Lin, Qiang Chen, Shuicheng Yan, and Jiashi Feng. Foveanet: Perspective-aware urban scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 784–792, 2017. 2

[22] Xiang Li, Wei-Shi Zheng, Xiaojuan Wang, Tao Xiang, and Shaogang Gong. Multi-scale learning for low-resolution person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3765–3773, 2015. 3

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[24] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2351–2359, 2016. 2

[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 1

[26] K Malarvizhi, S Vasantha Kumar, and P Porchelvan. Use of high resolution google earth satellite imagery in landuse map preparation for urban related applications. *Procedia Technology*, 24:1835–1842, 2016. 1

[27] Silviu Minut and Sridhar Mahadevan. A reinforcement learning model of selective visual attention. In *Proceedings of the fifth international conference on Autonomous agents*, pages 457–464. ACM, 2001. 1

[28] Matthias Mueller, Neil Smith, and Bernard Ghanem. Context-aware correlation filter tracking. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1396–1404, 2017. 1

[29] Xingchao Peng, Judy Hoffman, X Yu Stella, and Kate Saenko. Fine-to-coarse knowledge transfer for low-res image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3683–3687. IEEE, 2016. 2, 5

[30] Aleksis Pirinen and Cristian Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6945–6954, 2018. 2, 3

[31] 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. 2, 5, 6, 7

[32] Felix Rembold, Clement Atzberger, Igor Savin, and Oscar Rojas. Using low resolution satellite imagery for yield prediction and yield anomaly detection. *Remote Sensing*, 5(4):1704–1733, 2013. 1

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[34] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024, 2017. 4

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec. 2015. 6

[36] 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, 8

[37] Jong-Chyi Su and Subhransu Maji. Adapting models to signal degradation using distillation. *arXiv preprint arXiv:1604.00433*, 2016. 2, 5, 6, 7

[38] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 769–777, 2015. 1

[39] Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. Multi-attention multi-class constraint for fine-grained image recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–821, 2018. 2

[40] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 4

[41] 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. 2

[42] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017. 2

[43] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018. 2

[44] Yan Wang, Lequn Wang, Yurong You, Xu Zou, Vincent Chen, Serena Li, Gao Huang, Bharath Hariharan, and Kilian Q Weinberger. Resource aware person re-identification across multiple resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8042–8051, 2018. 3

[45] Zhangyang Wang, Shiyu Chang, Yingzhen Yang, Ding Liu, and Thomas S Huang. Studying very low resolution recognition using deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4792–4800, 2016. 1, 2, 5, 6

[46] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992. 4

[47] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018. 2

[48] Yuan Yao, Xutao Li, Yunming Ye, Feng Liu, Michael K Ng, Zhichao Huang, and Yu Zhang. Low-resolution image categorization via heterogeneous domain adaptation. *Knowledge-Based Systems*, 163:656–665, 2019. 2

[49] Linwei Yue, Huanfeng Shen, Jie Li, Qiangqiang Yuan, Hongyan Zhang, and Liangpei Zhang. Image super-resolution: The techniques, applications, and future. *Signal Processing*, 128:389–408, 2016. 1

[50] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2031–2039, 2017. 2

[51] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 1