Spot and Learn: A Maximum-Entropy Patch Sampler for Few-Shot Image Classification

Wen-Hsuan Chu[1*], Yu-Jhe Li[2,3], Jing-Cheng Chang[2,3], Yu-Chiang Frank Wang[2,3]
[1] Carnegie Mellon University, Pittsburgh, PA, USA
[2] National Taiwan University, Taipei, Taiwan
[3] MOST Joint Research Center for AI Technology and All Vista Healthcare
[1] chuwenhsuan@cmu.edu, [2] {yujheli, b04901138, ycwang}@ntu.edu.tw

Abstract

Few-shot learning (FSL) requires one to learn from object categories with a small amount of training data (as novel classes), while the remaining categories (as base classes) contain a sufficient amount of data for training. It is often desirable to transfer knowledge from the base classes and derive dominant features efficiently for the novel samples. In this work, we propose a sampling method that decorrelates an image based on maximum entropy reinforcement learning, and extracts varying sequences of patches on every forward-pass with discriminative information observed. This can be viewed as a form of "learned" data augmentation in the sense that we search for different sequences of patches within an image and performs classification with aggregation of the extracted features, resulting in improved FSL performances. In addition, our positive and negative sampling policies along with a newly defined reward function would favorably improve the effectiveness of our model. Our experiments on two benchmark datasets confirm the effectiveness of our framework and its superiority over recent FSL approaches.

1. Introduction

Deep neural networks have achieved extraordinary performance in supervised visual learning tasks [13, 23, 10] in recent years. However, these supervised learning methods often require a large amount of training data and annotations to achieve such performance. This significantly limits the problems they can be applied to, as it may be expensive to acquire enough annotations for the training data, or worse, the training data may be difficult to acquire themselves. In comparison, humans are surprisingly good at learning new concepts using very little supervised information. For example, humans (both adult and children) can learn to recognize a new animal just from a couple of pictures in books or from online sources. On the other hand, neural networks would suffer from the issue of severe data over-fitting, resulting in poor generalization results during inference. This has motivated researchers to propose different methods for *few-shot learning* [12, 27, 20, 1, 21, 24, 5, 9, 25, 6, 29, 7, 19, 28]. Few-shot learning aims to classify novel visual classes from very few labeled samples. Contemporary methods usually tackle this challenge using meta-learning approaches [5, 20] or metric-learning approaches [27, 24]. Another branch of algorithms focus on data hallucination to generate more training samples [9, 29, 32]. There has also been some works on using soft attention for Few-Shot Learning using attention generated from semantic information [28, 3]. Current "attention" mechanisms are largely inspired by the human visual systems, where we only focus on small regions located at the center of our view or gaze, while the areas further away from the center are in fact very "blurry". As a result, we build our understanding of a scene by aggregating infor-

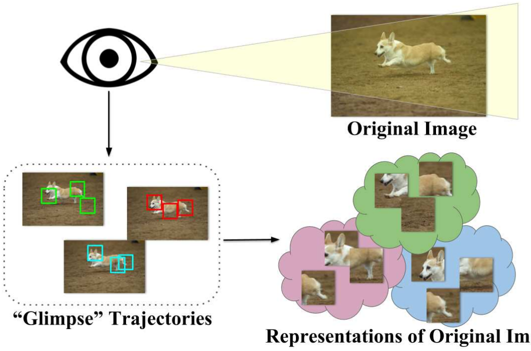


Figure 1: Illustration of our proposed patch sampling strategy for FSL. If varying glimpse trajectories can be obtained on each forward-pass, one can create a variety of input patch sequences for training from the same input image.

[*] Work done while at National Taiwan University.

mation from different regions of the scene as needed over time. It is believed that this behaviour allows us to ignore the “clutter” that is outside of the salient regions of interest [17], hence the name of “attention”.

Inspired by the aforementioned concept, we present an alternative view to why humans exhibit this behaviour: in addition to being able to ignore “clutter” during inference, decomposing or de-correlating an image or a scene into sequences of patches can allow us to increase the input variety for any given image during training. For example, when we stare at the same image or scene twice, our gazes would likely follow any possible trajectory that allows us to understand the image or the scene. This motivates us to make one crucial observation: if we use the whole image or scene as the input, we only get one possible input variety for training; on the contrary, if we could model a more human-like behavior and sample randomly from any possible regions of interest for training, we would increase the input variety, which may lead to better generalization results (see Fig. 1).

In this paper, we propose an end-to-end trainable framework to model this novel interpretation of the human visual system. Our model aims to produce possible patch sequences from an input that would lead to correct classification by applying the maximum entropy reinforcement learning objective [26]. Moreover, we utilize a negative trajectory sampler for non-interesting regions. Combined with the proposed positive trajectory sampler enforcing correct classifications, this positive/negative sampling strategy would further help regularize the network. Finally, the experiments on two open datasets demonstrate the effectiveness of our proposed model. We note that, to the best of our knowledge, we are among the first to advance reinforcement learning for few-shot learning as a form of “learned” data augmentation, which is orthogonal to many of the other contemporary few-shot learning methods.

The contributions of this paper are highlighted below:

- We propose a novel deep reinforcement learning based approach for few-shot learning.
- During training, our model samples varying candidate patch sequences from an input image, which would satisfy FSL and result in improved performances.
- Our proposed sampling mechanism jointly utilizes both positive and negative sampling policies, which are able to determine “patches of interests” and “background”.
- Experimental results on two open datasets confirm that our method performs favorably against other existing few-shot learning approaches.

2. Related Works

Few-Shot Learning Given abundant labeled training samples from some “base” classes, few-shot learning aims to learn to classify samples from “novel” classes using only a small amount of labeled samples from the novel classes. One category of algorithms tackle this using meta-learning approaches by *learning to learn*, such as learning to initialize [5] or optimize [20] for few-shot learning settings. Another category of algorithms explore metric-learning based approaches, which can be viewed as *learning to compare*. For examples, siamese networks [12], cosine similiarity [27], Euclidean distance to the mean [24], CNN-based relation modules [25], or Graph Neural Networks [6] have all been explored in literature. Approaches that learn to “hallucinate” or generate new data samples for novel classes [9, 29, 32] have also been explored recently. We note that this is different from our work as we’re not explicitly generating any new data. Finally, there has also been works that predict weights [1] or the novel class classifier [7] directly, or uses novel class features as weights [19].

Attention Models Visual attention has been studied extensively and can be broadly categorized into two categories. The first category is referred to as hard attention, where cropped patches of the original image is returned [17, 30]. The second category is referred to as soft attention, where an “attention map” corresponding to the entire image is returned [30]. Soft attention models have the advantage of being fully differentiable, which makes it easier to train, while hard attention models have some form of stochasticity in them, and has to be trained using reinforcement learning methods (like policy gradients) due to the non-differentiability in cropping. Spatial Transformer Networks [11], while designed for general image transformations, can also be used for “cropping” images and therefore, a form of attention. The work with the closest motivation to ours is [31], in which they employ a soft attention model that aims to extract all the important regions of an image by minimizing the correlation between multiple attention maps and restricts the overlap across attention maps to be lower than some threshold. While their method also wishes to find all areas of interest in an image, they imposed strong constraints and there is no stochasticity between different forward passes. In contrast, we explicitly maximize the variety of patches extracted from a given input image in different forward passes, allowing us to deal with the scarcity of data in FSL settings. While soft attention schemes have also been applied in few-shot learning very recently [28, 3], they typically require semantic information and the lack of stochatiscity would be a concern.

3. Preliminaries

To make our paper more self-contained, we briefly review reinforcement learning (RL) algorithms related to our work. This section will serve as a theoretical basis to our framework in the following section.

Reinforcement Learning. To favorably sample a sequence of patches for an image through RL, we would like to find a sequence of actions, $(a_1, a_2, ..., a_N)$ given an image x. These actions indicate the location of the patch that will be extracted from the image, specifically, they correspond to a normalized 2D coordinate in the image x. These taken actions aim to maximize the total amount of RL discounted rewards $\sum_{t=1}^{N} \gamma^{t-1} r_t$, where the discount factor $\gamma \leq 1$ is a constant. In the classification tasks, a common choice for the reward function is often set as $r_N = 1$ if the classification is correct after the Nth time-step and 0 otherwise [17]. In addition, these actions are often sampled from a learned policy π, i.e. $a_i \sim \pi_\theta(a|s)$, where the policy is parameterized by θ, which could be modeled as neural networks. Our objective in reinforcement learning can be written as:

$$\arg\max_{\pi} \sum_{t=1}^{N} \mathrm{E}[\gamma^{t-1} r_t]. \tag{1}$$

Standard ways to solve for the policy π include Policy Gradient [22] and Q-Learning [18] methods. Policy Gradient based methods aim at learning desirable policies directly, and have been explored in attention models [17], where a family of distributions is considered for the policy π (e.g. a Gaussian policy). On the other hand, Q-Learning methods learn a Q-function corresponding to the "value" of an action (Q-value) given some state s, and the best scoring action is then chosen at every time step. However, Policy Gradient methods require one to pre-define a form for the policy (e.g., Gaussian) which implies that the optimal behavior is unimodal. As for Q-Learning, one only takes the action with the maximum Q-value, resulting a single "good" behavior [9] without any stochasticity. This would cause the sampling policy to collapse into a single mode, i.e., locations near the peak of the Gaussian or the highest Q-value, neither of which would be satisfactory in FSL settings.

Maximum Entropy Reinforcement Learning. To address the aforementioned issue, a maximum entropy reinforcement learning objective can be employed, which additionally maximizes the entropy of the action distribution $H(\pi(\cdot|s))$ given the state s we are in. Intuitively speaking, we would like to maximize the variety of our actions while also obtaining a high reward (i.e. making the correct classification). The objective of maximum entropy reinforcement learning can be written as:

$$\arg\max_{\pi_{MaxEnt}} \sum_{t=1}^{N} \mathrm{E}[\gamma^{t-1} r_t + \alpha H(\pi(\cdot|s))], \tag{2}$$

where α is a constant that balances the importance of the entropy term relative to the rewards. Soft Q-Learning [8] has recently been proposed to solve this objective function by using an fixed-point iteration method:

$$Q_{soft}(s_t, a_t) \leftarrow r_t + \mathrm{E}[V_{soft}(s_{t+1})] \tag{3}$$

$$V_{soft}(s_t) \leftarrow \alpha \log \int_{\mathrm{A}} exp(\frac{1}{\alpha} Q_{soft}(s_t, a^{'}))da^{'}, \tag{4}$$

where $Q_{soft}(s, a)$ is the Q-function and $V_{soft}(s)$ represents the value function, a measure of how high a state's value is. The maximum entropy policy then can be calculated as a softmax over the advantage function, which is a measure of how good an action is *relative* to the other actions:

$$\pi_{MaxEnt}(a_t|s_t) \propto exp(\frac{1}{\alpha} Q_{soft}(s_t, a_t) - V_{soft}(s_t)). \tag{5}$$

Thus, we are able to derive a multimodal policy without pre-defining its form from particular distributions that maximize the total reward (i.e., correct classifications), while exhibitng varying behaviours during sampling. Please see [8] for thorough derivations of how the algorithm could be approximated using deep neural networks.

4. Proposed Framework

Given a set of K input images $X = \{x_j\}_{j=1}^{K}$ and its corresponding label set $Y = \{y_j\}_{j=1}^{K}$, where $x_j \in \mathbb{R}^{H \times W \times 3}$ and $y_j \in \mathbb{R}$ are the j^{th} image and its label respectively, our goal is to correctly predict the labels given these input images, especially in few-shot scenarios. To achieve this, we propose a Maximum Entropy RL-based framework which learns to sample sequences of patches from the input image x_j, denoted as $P^j = \{p_i^j\}_{i=1}^{N}$. In this section, we first describe our model architecture and provide details for each component. We then further explain the sampling mechanism governed by the designed reward function, which helps enhance and regulate our model. Finally, we provide the details about the training objective and the inference process of our model.

4.1. Architecture Overview

As depicted in Fig. 2, our model consists of five components: feature extractor, action context encoder, state encoder, maximum entropy sampler, and a final classifier. We now describe the details of each component.

Feature Extractor f_e. To encode the patches p_i[1] of the input image x, we introduce the feature extractor f_e to extract the feature embedding e_i of the patch p_i at every time step using a CNN. The feature extractor only has access to the local patches given by the maximum entropy sampler (described in later subsections) in the form of cropped windows of the original image x.

[1] For simplicity, we omit the subscript j, and represent their corresponding input image as x and label as y

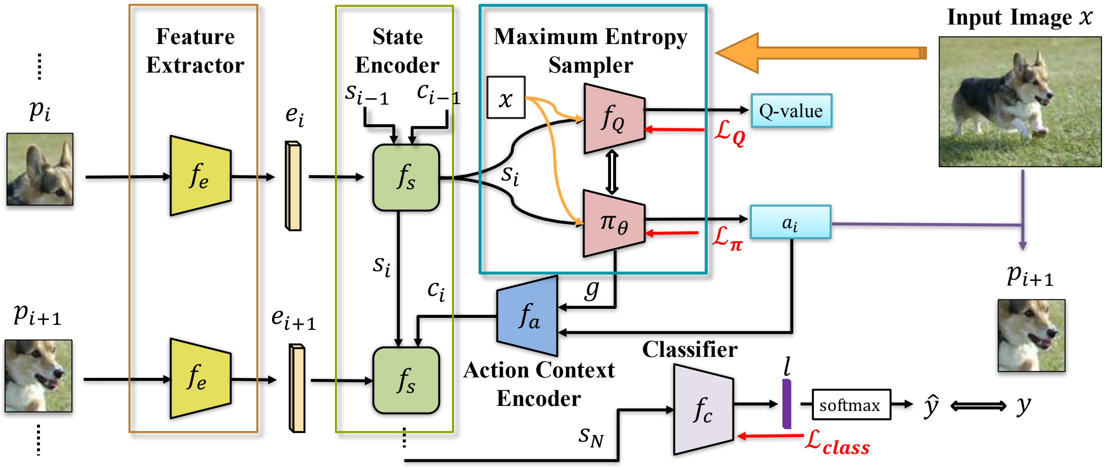


Figure 2: Our framework mainly comprises of five components: feature extractor, action context encoder, state encoder, maximum entropy sampler, and a final classifier. The feature extractor f_e takes an input patch p_i to derive the feature e_i, which is used by the state encoder f_s to produce current state s_i. Next, the maximum entropy sampler (with the Q-function f_Q and the actual policy π_θ) takes the input image x with s_i to sample an action a_i, that produces the next patch p_{i+1}. The action context encoder f_a then encodes the context c_i of the current action using a_i and the features of the image g (extracted by π_θ). Finally, the state encoder f_s takes the newly extracted feature e_{i+1} (extracted from p_{i+1} by f_e) and action context c_i to produce the next state s_{i+1} (not depicted in figure). The final state s_N is fed into the classifier to determine its output vector l and the predicted label $\hat{y}$. We note that N is selected as a hyperparameter.

State Encoder f_s. To aggregate the features of prior extracted patches for the input image x, a recurrent neural network (RNN) is used as the state encoder f_s to encode the state s_i from the previous state s_{i-1} and the sampled patch p_i. The state encoder f_s also takes the current action context c_{i-1} (produced by the module f_a in the Fig. 2 and will be detailed later when introducing f_a) and derives the current state s_i.

More specifically, we implement this RNN with a GRU [2]. On each forward-pass, we initialize the GRU with the features extracted from a random patch and the initial action context c_0 set to a zero vector.

Maximum Entropy Sampler f_Q & π_θ. Since our goal is to generate varying patch trajectories from the input image x, we employ a maximum entropy sampler to sample the next candidate patch p_{i+1}. The Maximum Entropy Sampler is built based on the Soft Q-Learning algorithm in [8]. Generally, the sampler takes the whole image x and utilizes the current state s_i (aggregated information from all prior patches) to produce a 2D action vector a_i, which corresponds to the coordinates of the center of the next patch p_{i+1}. More specifically, the sampler itself contains two components: the Q-function f_Q and the actual sampling policy π_θ, which are related by (5). The architectures of both components are identical, containing a small CNN with fully connected layers. The Q-function f_Q evaluates how "good" actions are, while the sampling policy π_θ outputs the actual 2D action a_i and the features of the image x, denoted by g. We note that only the sampling policy π_θ is used during inference. More details about the sampling mechanism can be found in Sect. 4.2.

Action Context Encoder f_a. The action context encoder f_a in Fig. 2 takes the output features g of the image x from the convolution layers in the aforementioned policy π_θ and the sampled 2D action a_i to produce a context c_i. Intuitively speaking, the action context encoder aims to account for the global information produced by π_θ, which is utilized by f_s.

Classifier f_c. Since our goal is to correctly classify the input image x as its corresponding label y, we introduce the classifier f_c which takes the final state s_N and produces the output vector l and the label prediction $\hat{y}$.

4.2. Sampling Mechanism

In this section, we present the sampling mechanism for our maximum entropy sampler (f_Q and π_θ), which is based on the Maximum Entropy RL algorithm in Sect. 3. While the maximum entropy sampler alone would sample patches that is beneficial for classification, we choose to further regulate both the maximum entropy sampler and the feature extractor for performance guarantees. To achieve this, we introduce the concept of *negative sampling* into our model. For example, a negative sequence sample may be a sequence of patches that land outside of the object or regions

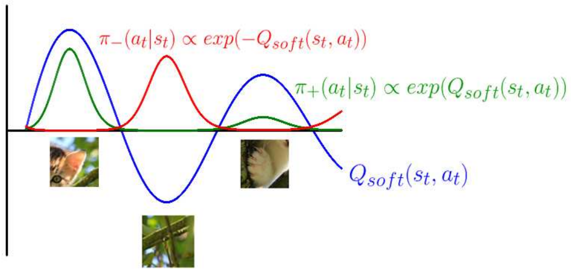


Figure 3: Illustration of the relationship between the Q-function Q_{soft} (blue), the positive policy π_+ (green) and the negative policy π_- (red) satisfying (5) and (6), respectively. The Q-function is expected to output higher values for patches within the regions of interest, and lower values for irrelevant regions like the background. If the traditional RL objective is applied, one would only sample from one of the peaks. Note that the two overlapping policies are shown in different scales.

of interest, like the background. To be more precise, we define the negative sampling policy π_- as:

$$\pi_-(a_t|s_t) = exp(-\frac{1}{\alpha}Q_{soft}(s_t,a_t) + V_{soft}(s_t)) \quad (6)$$

This allows us to derive a policy that samples from undesirable points given a Q-function (see Fig. 3). Intuitively, we train the positive policy π_+ to match the Q-function $Q_{soft}(s_t,a_t)$, which has a higher value in the regions of interest, and train the negative policy to match the negative of the Q-function $-Q_{soft}(s_t,a_t)$, which now outputs a higher value in non-relevant regions. Note that the two policies are both conditioned on the same Q-function, and can be viewed as the "inverse" of the other policy.

To jointly apply both policies, we create an artificial "background" class and randomly select one policy on each forward-pass, i.e., the positive policy π_+ or the negative policy π_-. We encourage the classifier to regress to the ground truth y if the positive policy is chosen and the background class if the negative policy is selected. This motivates us to assign a reward value of -1 if the predicted label corresponds to the background class. Thus, the reward function is defined as:

$$R_i = \begin{cases} 1, & \text{if } i = N \text{ and } \hat{y} = y \\ -1, & \text{if } i = N \text{ and } \hat{y} = \text{background} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

The reward function R_i can be interpreted as a ranking of the three possible outcomes: for the positive policy π_+ (corresponding to the normal Q-function f_Q), an action sequence that results in the correct label will be preferred, followed by an action sequence that results in a wrong label, and an action sequence that causes the predicted label to be "background" class would generally be avoided. For the negative policy π_- (corresponding to the inverted Q-function $-f_Q$), this ranking is inverted, and the best action sequences for π_- would result in a predicted label of the "background" class.

It is worth noting that, the benefits of this joint sampling policy is two-fold. First, it allows the encoders to "see" and learn to encode poorly chosen patches. Second, this helps pull the values of different actions apart, which avoids extreme scenarios like when all action scores are about the same, which might occur when the model overfits and classifies everything correctly regardless of the quality of the sampled patches.

4.3. Training Objective and Evaluation

Training In the training stage, we first randomly select the policy "mode" of our network: the positive policy π_+, regressing towards the ground truth label y, or the negative policy π_-, regressing towards the artificial "background" class y_b. We then obtain the predicted label $\hat{y}$ and use the reward function in (7) to update the Maximum Entropy Sampler. This is done by updating the Q-function f_Q based on the equations (3) and (4), and the two policies (π_+ and π_-) such that they follow the formulas in (5) and (6) by applying Soft Q-Learning [8] with a Replay Buffer [15]. To better optimize the framework, we incorporate the classification loss, i.e. negative log-likelihood, into our training objective along with RL-based training following [17]. The classification loss L_{class} is calculated to jointly update feature extractor f_e, the action context encoder f_a, the state encoder f_s, and the classifier f_c:

$$L_{class} = \frac{1}{|M_1|}\sum_{j \subset M_1} y_j^T \log \hat{y}_j + \alpha\frac{1}{|M_2|}\sum_{j \subset M_2} y_j^T \log \hat{y}_j \quad (8)$$

where y_j^T and $\hat{y}_j$ denotes the (transposed) ground truth label (augmented with the "background" class) and the predicted label at the final time step N for the j-th image in the sampled batch, respectively. M_1 and M_2 denotes the set of indexes in the batch that were selected by the two "modes", i.e. the positive policy π_+ or the negative policy π_-. In the same equation, α is a scaling constant to balance the importance of the two loss terms. We summarize our training algorithm in the pseudo-code in Algorithm 1.

Inference During inference, we only select the positive policy π_+ to extract patches for a given test image because we aim to produce the correct label for the input image instead of the background label.

Algorithm 1 Training algorithm

```
Input: Data, label tuples $\{(x_j, y_j)\}$; Replay Buffer D; parameter $\beta$
Output: Network modules $f_e$, $f_s$, $f_a$, $f_c$, $f_Q$, $\pi_+$, $\pi_-$
  for number of training iterations do
      Sample $k$ from uniform distribution between [0, 1]
      if $k < \beta$ then
          Sample action sequences $(a_1, a_2, ...., a_N)$ using the positive policy $\pi_+$, extract the patches $(p_1, p_2, ...., p_N)$ and compute the predicted label $\hat{y_j}$
      else
          Sample action sequences $(a_1, a_2, ...., a_N)$ using the negative policy $\pi_-$, extract the patches $(p_1, p_2, ...., p_N)$ and compute the predicted label $\hat{y_j}$
      end if
      Store all transitions ($x_j$, $s_i$, $s_{i+1}$, $a_i$, $r_i$, $i$) in replay buffer D
      Calculate the classification loss $L_{class}$ with $y_j$ and $\hat{y_j}$ according to (8)
      Update $f_e$, $f_s$, $f_a$, $f_c$ using $L_{class}$
      Sample transition batch ($x_j$, $s_i$, $s_{i+1}$, $a_i$, $r_i$, $i$) from replay buffer D
      Update $f_s$, $f_Q$, $\pi_+$, $\pi_-$ using Soft Q-Learning [8]
  end for
```

5. Experiments

We first highlight the experimental datasets and provide our experimental settings in the first two subsections. Afterwards, the evaluation results are presented in the third subsection followed by the ablation studies with respects to different voting strategies in the fourth subsection. Finally, we present the sampling trajectory analysis on our trained sampling policy. More experiments are provided in the supplementary materials.

5.1. Datasets

We tested our model on two widely adopted datasets for few-shot learning: the Omniglot [14] and the miniImagenet [27]. We describe the two datasets below.

Omniglot Omniglot [14] contains 1623 different characters from 50 different alphabets. Each character contains images of hand-drawn characters by 20 different people. We follow the same evaluation strategy in [27, 24, 5, 21], where 1200 random character classes are sampled (independent of alphabet) as "base" classes, and the remaining 423 character classes are considered the "novel" classes, i.e. only a small amount of labeled samples per class is available. The images are all resized to 28×28, and we perform the common practice of class augmentation using rotations of 90, 180, and 270 degrees randomly, which results in a total of $1200 + 3600$ base classes and $423 + 1269$ novel classes.

miniImagenet miniImagenet was first proposed by [27], with 80 base classes and 20 novel classes sampled from the original Imagenet dataset [4], but recent work uses the splits proposed by [20], where there are 64 base classes, 16 validation classes, and 20 novel classes. We follow this split so our results can be compared to other work. Each class contains 600 images. The images are resized to 84×84, and we perform standard data augmentation techniques: color jittering, random left-right flips, and random crops. Only the 64 base classes were used for training and the 16 validation classes were used for modeling generalization performance and for choosing hyperparameters (e.g. number of finetuning iterations).

5.2. Experimental Setting

Implementation for few-shot setting. To ensure a fair comparison with other proposed methods, we employ a Conv-4 backbone structure for our feature extractor f_e, which is identical to the one used by [24]. For baselines, we experiment with two different kinds of classifiers following the same Conv-4 backbone. The first one, which we denote as **Baseline-FC**, uses a standard fully connected layer followed by a softmax activation to output the label prediction. The second one, which we denote as **Baseline-CS**, applies a cosine similarity measure instead of a dot product in standard fully connected layers. We would like to clarify that using cosine similarity as an alternative classifier for few-shot learning is not our contribution. Cosine similarity layers has been recently explored in [16] and has been applied to few-shot learning in [7, 19]. We set the number of extracted patches to be 4 (i.e. $N = 4$), with an additional patch that is randomly sampled independent of the sampling policies π_+ and π_- to initialize the state encoder GRU (f_s) for all experiments.

Inference. During inference, we apply a *Best-of-N Voting* method to obtain the final prediction label $\hat{y}_{nway}$. We note that due to the stochastic properties of the policy π_+, there is a chance (albeit small) that it will select an "irrelevant" region at any time-step. A simple workaround for this is to repeat the classification a number of times, i.e. N times, and aggregate the prediction results before outputting the class with the highest prediction probability/score. We first explain two ways of performing this aggregation: **Hard Voting** and **Soft Voting** as follows, while the study of the voting behavior can be found in our ablation studies in Section 5.4.

For hard voting, we take the argmax of the N predicted labels before aggregating them and see which label has the most votes. This can be seen as a form of "discrete" vot-

Table 1: Results on Omniglot. FC denotes a fully-connected classifier and CS denotes a cosine similiarity classifier. The number in bold indicates the best result.

	5-Way	
	1-Shot	5-Shot
Baseline-FC	91.95±0.48	98.97±0.10
Baseline-CS	93.30±0.44	99.33±0.09
Ours-FC	97.43±0.28	99.51±0.07
Ours-CS	**97.56±0.31**	**99.65±0.06**

ing and it discards the "uncertainty" information in the predicted labels. For example, if we had a binary label, the prediction (0.6, 0.4) and (0.99, 0.01) would both reduce to (1, 0). For soft voting, we aggregate the N predicted labels without performing the argmax and see which label has the highest accumulated probability.

Evaluation protocol. We evaluate using a K-Shot 5-Way evaluation protocol with Best-of-7 Soft Voting, where K is the number of labeled samples we have per novel class. For each testing episode, we randomly select 5 classes from all the novel classes, and out of these 5 classes, we sample $5K$ labeled examples and 15 testing samples. The $5K$ labeled sample are first used for finetuning our model, then we perform inference by calculating the predicted label $\hat{y}_{nway}$ based on the distance of the output vector l of the query image to the output vectors of the $5K$ labeled samples l_s. This is akin to a nearest neighbor method.

$$\hat{y}_{nway} = \frac{exp(-dist(l - l_s))}{\sum_{s \in 5K} exp(-dist(l - l_s)))} \tag{9}$$

We perform 600 test episodes and report the mean and 95% confidence interval for all evaluation settings.

5.3. Evaluation Results and Comparisons

Omniglot For Omniglot, we select the sampling patch size to be 16×16 during experiments. We perform 1-shot and 5-shot 5-Way experiments on the novel classes, and then compare with the two baseline methods: Baseline-FC and Baseline-CS. The results on Omniglot are presented in Table 1. For 1-shot, the performance gain over Baseline-FC is 5.5% while the gain over Baseline-CS is 4.2%. For 5-shot, the performance gain over Baseline-FC is 0.5%, while the gain over Baseline-CS is 0.3%. Here we observed that our model shows effectiveness in few-shot settings even if it's presented one image per class (one-shot setting) compared to the baseline methods. We also find that simply replacing the fully connected classifier with a cosine similarity based one yields a noticeable improvement.

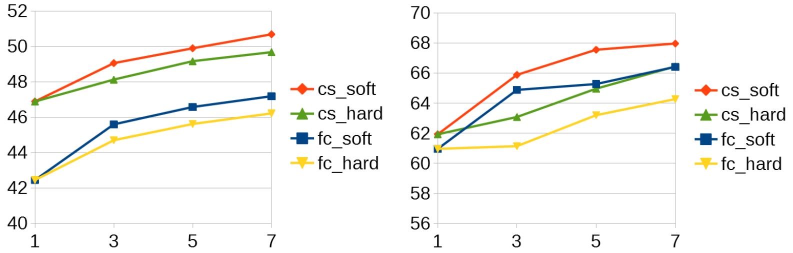
We note that, the images in Omniglot consist of a single character occupying a big portion of the image, and the "local" feature in patches simply reduces to oriented lines, which is why we select a larger patch size (16×16 to input image size 28×28), thereby allowing some global information to be captured and encoded.

Table 2: Results on miniImagenet. FC denotes a fully-connected classifier and CS denotes a cosine similiarity classifier. ProtoNet# denotes the training method (30-Way for 1-shot and 20-Way for 5-shot) in the original paper. ProtoNet denotes a 5-Way training strategy (as other methods do). The results of Matching Network is cited from [24] (denoted with a star). Bold and underlined numbers indicate top two scores, respectively.

	5-Way	
	1-Shot	5-Shot
Baseline-FC	42.02±0.73	61.54±0.68
Baseline-CS	46.84±0.77	64.13±0.69
Matching Network* [27]	46.61±0.78	60.97±0.67
ProtoNet [24]	46.14±0.77	65.77±0.70
ProtoNet# [24]	49.42±0.78	**68.20±0.66**
MAML [5]	48.07±1.75	63.15±0.91
RelationNet [25]	50.44±0.82	65.32±0.70
Ours-FC	47.18±0.83	66.41±0.67
Ours-CS	**51.03±0.78**	67.96±0.71

miniImagenet Compare with Omniglot, miniImagenet is more realistic dataset. We select the patch size as 24×24 where the input size is 64×64 (after performing data augmentation). We compare our proposed model with existing methods [27, 24, 5, 25] which also fairly applies the similar setup with the same Conv-4 backbone. The comparison results can be obtained in Table 2. Similar to the results reported in Omniglot, we see a cosine similarity classifier provides better results compared to a fully connected layer. For 1-shot, the performance gain over Baseline-FC is 5.1% while the gain over Baseline-CS is 4.2%. For 5-shot, the performance gain over Baseline-FC is 4.9%, while the gain over Baseline-CS is 3.8%. Our proposed method performs favorably against the state-of-the-art methods in 1-shot learning. For 5-shot, our proposed model outperforms the best competitor ProtoNet [24] under fair comparisons (equal training and evaluation schemes).

We note that ProtoNet# [24] used a slightly more different training scheme, where they performed training using 30-way episodes (i.e. 30 training classes) for 1-shot and 20-way episodes (i.e. 20 training classes) for 5-shot during meta-training, which they reported that resulted in better performance than just training from samples from 5 classes. Compared to this, we fall slightly behind by 0.28%, but there is a high overlapping of the confidence intervals. We thereby include the performance of their model under 5-way meta-training episodes for completeness.



(a) One-shot

(b) Five-shot

Figure 4: Effects of hard and soft voting and the number of votes N on miniImagenet on (a) 1-shot and (b) 5-shot classification. FC denotes a fully connected classifier, and CS denotes a cosine similarity classifier.

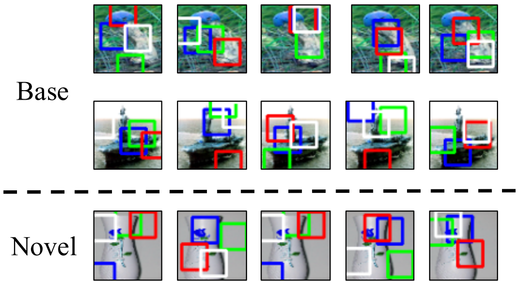


Figure 5: Sampling trajectories on miniImagenet. The top two rows are from base classes and the bottom row is from a novel class. The order of sampled patches is: blue, green, red, and white. Note that the sampled patch trajectories differ in every feed-forward pass.

Our performance gain can be ascribed to incorporating maximum entropy samplers and our unique design of model for aggregating features from patches.

5.4. Ablation Studies

To explore the effects of voting strategies, we experiment the hard and soft voting strategies during inference with N as 1, 3, 5, and 7, using both the fully connected classifier and the cosine similarity classifier. We evaluate on miniImagenet and compare the results in Fig. 4. For soft voting, the increase in performance comes from the addition of voting (N=1 to N=3), with an accuracy increase of 3% to 4% for 5-shot settings and 2% to 3% for 1-shot settings. We observe the same trend for hard voting with 1-shot settings. However, for the 5-shot settings with hard voting, the highest increase in accuracy is achieved at going from 3 to 5 votes, with an increase of 2%. The preserved uncertainty information also seems preferable, as the soft voting scheme outperforms to hard voting scheme (1% to 2% for 5-shot settings, around 1% for 1-shot settings).

5.5. Sampling Trajectory Analysis

Here we plot some of the sampling trajectories from the base classes and novel classes from the miniImagenet dataset in Fig. 5. Note that the sampling policies were *not* finetuned on the novel classes, thus the policies must be able to generalize beyond the seen classes.

At first glance, we can see that the sampling policy learns to sample on the regions of interest and may also sometimes choose to sample on background patches. We would like to clarify that this is, in fact, the intended behavior, and is the results of our main objective function in (2), where we aim to maximize the action variety of the sampling policy. Consider the scenario where, after seeing the first couple of patches, we are already certain of the object present in the image. In this case, we can sample anywhere on the image to maximize the entropy term in the objective function (2).

6. Conclusion

We presented a deep learning framework employing the maximum entropy reinforcement learning objective. The novelty of our model lies in the incorporation of maximum entropy reinforcement learning and soft Q-Learning for the sampling policies, with applications to few-shot learning. We utilize both positive and negative sampling policies to determine the favorable regions in an image and regularize the learning process. Thus, our approach is able to increase the input variety for the feature extractors (CNN) during training, which can be seen as a form of "learned" data augmentation. Moreover, during inference, the sampling policies would be able to "attend" to the relevant regions of the test images, which allows us to elegantly deal with any potential clutter in test images. Experiments on two FSL datasets demonstrated that our model is able to improve FSL performances and performs favorably against the state-of-the-art methods.

Acknowledgements. This work is supported by the Ministry of Science and Technology of Taiwan under grant MOST 108-2634-F-002-018.

References

[1] Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip H. S. Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2

[2] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. 4

[3] Wen-Hsuan Chu and Yu-Chiang Frank Wang. Learning semantics-guided visual attention for few-shot image classification. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2018. 1, 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 6

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1, 2, 6, 7

[6] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 1, 2

[7] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6

[8] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 3, 4, 5, 6

[9] Bharath Hariharan and Ross B. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3

[10] 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 (CVPR)*, 2016. 1

[11] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 2

[12] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 1, 2

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2012. 1

[14] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci*, 2011. 6

[15] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993. 5

[16] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, 2018. 6

[17] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2, 3, 5

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015. 3

[19] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6

[20] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 6

[21] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016. 1, 6

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 3

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, 2014. 1

[24] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1, 2, 6, 7

[25] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 7

[26] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009. 2

[27] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2, 6, 7

[28] Peng Wang, Lingqiao Liu, Chunhua Shen, Zi Huang, Anton van den Hengel, and Heng Tao Shen. Multi-attention network for one shot learning. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 1, 2

[29] Yu-Xiong Wang, Ross B. Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[30] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 2

[31] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Trans. Multimedia*, 2017. 2

[32] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2