SpotTune : Transfer Learning through Adaptive Fine-tuning

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, Rogerio Feris

IBM Research & MIT-IBM Watson AI Lab, University of California, San Diego, The University of Texas at Austin

CVPR 2019

2020.04.16

최영제

Index

1. Introduction

2. Proposed Approach

3. Experiments

Transfer learning overview

- Transfer learning이란 source task를 해결하면서 얻어진 knowledge를 target task로 전이하여 사후적으로 학습하는 개념을 말함

- Donahue et al.은 pre-trained AlexNet을 이용하여 image의 특성을 추출 후 SVM(support vector machine)의 input으로 활용 [CVPR 2014]

- Yosinski et al.은 pre-trained AlexNet을 target task에 맞게 fine-tuning하는 방법을 사용 [NIPS 2014]
  → feature 추출 후 다른 classifier를 사용하는 것 보다 좋은 성능을 보임

- Azizpour et al은 전체 parameter를 갱신하는 것은 target dataset이 작고, model capacity가 큰 경우 overfitting을 경고

  → 전체 layer가 아닌 특정 layer를 freeze, 소수 layer를 fine-tuning하는 방법을 제안

어떤 layer를 freeze시키고 어떤 layer를 fine-tuning 시켜야 하는가

Proposed Approach

SpotTune Overview

- Model performance를 증가시키는 방향으로 fine-tuning strategy를 사람이 아닌 model이 결정

  → ~ in order to improve the accuracy of the model in the target domain

- Image classification 문제에서 target task의 instance마다 policy network가 layer block(in ResNet)을 fine-tuning or freeze을 선택

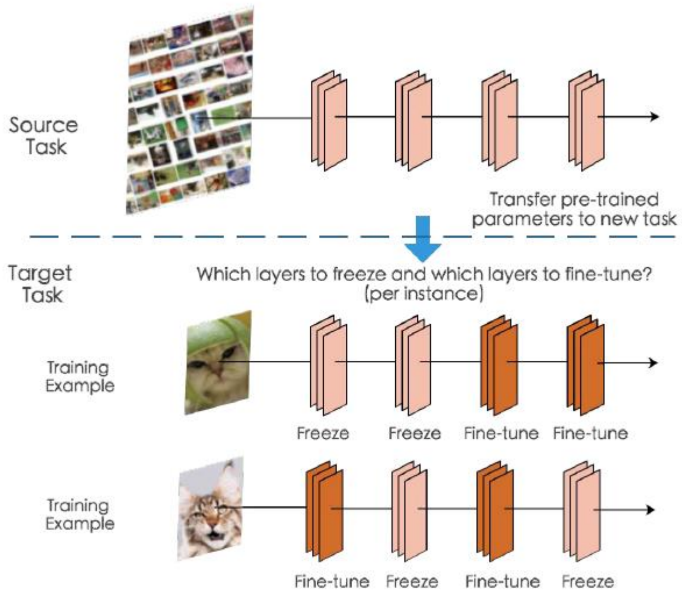


Figure 1: Given a deep neural network pre-trained on a source task, we address the question of *where to fine-tune* its parameters with examples of the target task. We propose a novel method that decides, per training example, which layers of the pre-trained model should have their parameters fixed, i.e., shared with the source task, and which layers should be fine-tuned to improve the accuracy of the model in the target domain.

Proposed Approach

SpotTune Overview

- Consider the l-th residual block in a pre-trained ResNet model

$$x_l = F_l(x_{l-1}) + x_{l-1} \tag{1}$$

- Policy network가 freeze, fine-tuning을 결정하기에 original block(F_l)과 trainable block($\hat{F}_l$) 을 구분

$$x_l = I_l(x)\hat{F}_l(x_{l-1}) + \big(1 - I_l(x)\big)F_l(x_{l-1}) + x_{l-1} \tag{2}$$

→ $I_l(x)$는 binary random variable로써 residual block이 freeze($I_l(x) = 0$)될지 fine-tuning($I_l(x) = 1$)될지 결정

→ trainable block($\hat{F}_l$)은 F_l의 weight를 초기값으로 할당받으며 target task를 학습하며 optimized

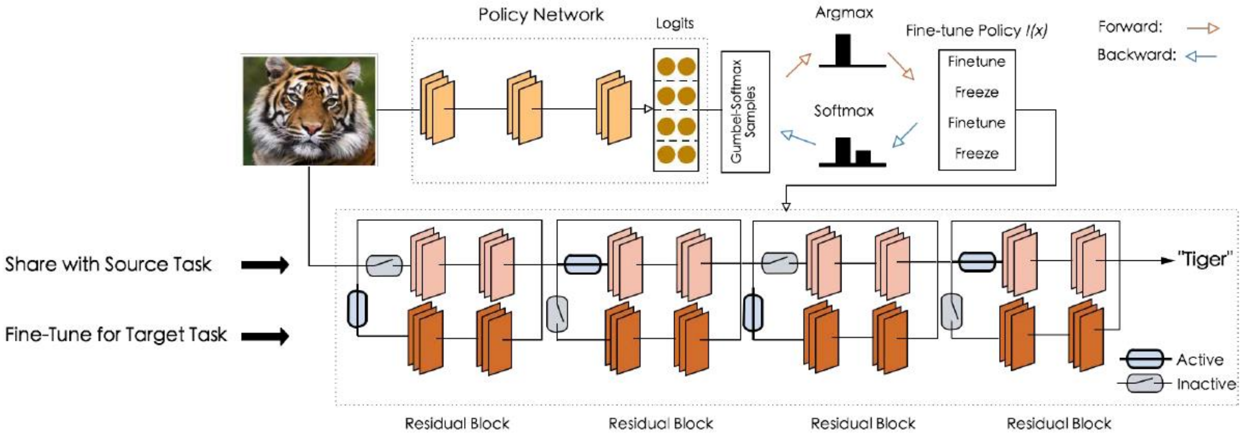
- Binary random variable $I_l(x)$는 standard gumbel distribution을 따름

$$G = -\log\big(-\log(U)\big) \text{ with } U \sim \text{Unif}\,[0,1] \tag{3}$$

- Gumbel distribution은 극치 분포의 일종, 극치분포는 정규분포에서 최소 또는 최대구간에 밀집되어있는 데이터 분포를 모사

Proposed Approach

SpotTune Overview



- 4개의 residual block의 ResNet의 경우 (4,2)의 행렬이 policy network의 output(logits = $\log\alpha_i$)이 됨
- (Output + G_i)의 argmax값으로 fine-tuning strategy를 구성
- 그 후 back propagation을 통해 target task의 accuracy가 최대화 되는 방향으로 학습

policy $I_l(x)$ is discrete makes the network non-differentiable(sampling node)

Proposed Approach

Optimizing stochastic computation graph

- (a)는 deterministic한 neural net의 forward & backward pass, (b)는 stochastic한 node를 포함

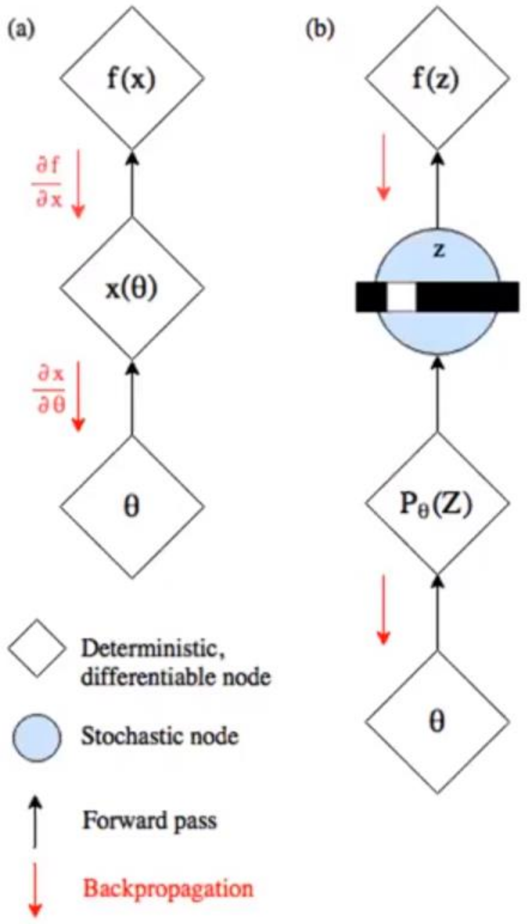
- (b)의 경우 sampling한 것을 back propagation을 진행할 때 일반적인 방법으로 불가

- Optimization target은 θ(neural net의 parameter)와 φ(distribution의 parameter)

$$L(\theta, \varphi) = E_{x \sim p_{\varphi}(x)}[f_{\theta}(x)]$$

- Neural net의 parameter는 gradient를 쉽게 구할 수 있으나 $\varphi(\mu, \sigma)$의 경우 분포에 dependent하여 미분 불가

- Score function estimator와 Re-parameterization trick으로 해결할 수 있음



Proposed Approach

Re-parameterization trick

- Re-parameterization trick을 통해서 stochastic part를 back-prop 과정에서 분리

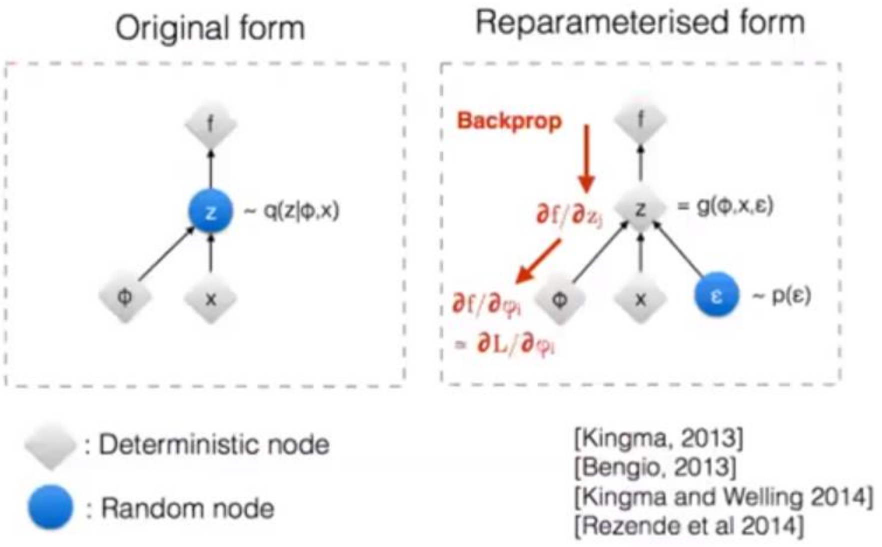
- 그 후 gradient를 deterministic part로 보내 back propagation을 진행

$$X \text{ from Normal}(\mu,\sigma^2) \rightarrow Z \text{ from Normal}(0,1)$$

$$g_{\mu,\sigma}(Z) = \mu + \sigma Z$$

- 즉 optimization 대상인 $\varphi(\mu, \sigma)$가 더 이상 distribution에 dependent하지 않게 되어 stochastic 성질과 무관

- 이를 discrete한 distribution에 적용한 것이 Gumbel max trick



Proposed Approach

Gumbel max trick & softmax relaxation

- Re-parametrization trick과 동일하게 stochastic node를 분리하여 back propagation으로 학습

- Policy network의 output log α_i는 freeze or fine-tune의 logit값

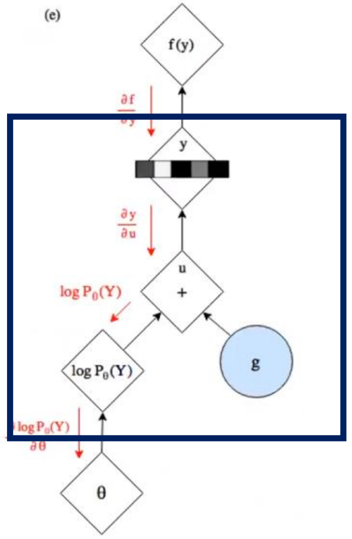
- argmax operation is non-differentiable → Gumbel softmax distribution

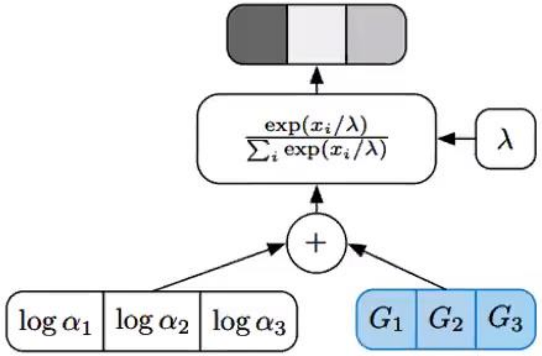
$$X = \arg\max_i [\log \alpha_i + G_i].$$

$$\downarrow$$

$$Y_i = \frac{\exp((\log \alpha_i + G_i)/\tau)}{\sum_{j=1}^{z} \exp((\log \alpha_j + G_j)/\tau)} \quad \text{for } i = 1, .., z$$

- 즉 이를 통해 policy network의 output인 logit 값을 gumbel distribution을 따르게 함





(b) Concrete(α, λ)

Experiments

Experimental setup

Dataset	Training	Evaluation	Classes
CUBS	5,994	5,794	200
Stanford Cars	8,144	8,041	196
Flowers	2,040	6,149	102
Sketch	16,000	4,000	250
WikiArt	42,129	10,628	195

- Datasets은 총 5개를 사용 (CUBS, Stanford Cars, Flowers, ImageNet: Sketches, WikiArt)

- Baseline은 다음과 같음

  1) Standard fine-tuning : 모든 파라미터 재학습

  2) Feature extractor : add one classifier layer

  3) Stochastic fine-tuning : randomly sample 50% of the blocks of the pre-trained network to fine-tune

  4) Fine-tuning last-k block: k = 1,2,3

  5) Fine-tuning ResNet 101: Spottune의 경우 ResNet 50을 사용

  6) Random policy : always fine-tunes the last three layers and random decision for other layers

  7) L^2-SP : ICML 2018, recently proposed SOTA regularization method for fine-tuning

Experiments

Results

Model	CUBS	Stanford Cars	Flowers	WikiArt	Sketches
Feature Extractor	74.07%	70.81%	85.67%	61.60%	75.50%
Standard Fine-tuning	81.86%	89.74%	93.67%	75.60%	79.58%
Stochastic Fine-tuning	81.03%	88.94%	92.95%	73.06%	78.30%
Fine-tuning last-3	81.54%	88.21%	89.03%	72.68 %	77.72%
Fine-tuning last-2	80.34%	85.36%	91.81%	70.82%	78.37%
Fine-tuning last-1	78.68%	81.73%	89.99%	68.96%	77.20%
Random Policy	81.63 %	88.57%	93.44%	73.82%	78.30%
Fine-tuning ResNet-101	82.13%	90.32%	94.21%	**76.52%**	78.92%
L^2-SP	83.69%	91.08%	95.21%	75.38%	79.60%
Progressive Neural Nets	83.08 %	91.59%	95.55%	75.41%	79.71%
SpotTune (running fine-tuned blocks)	82.36%	92.04%	93.49%	67.27%	78.88%
SpotTune (Global-k)	83.48%	90.51%	**96.60%**	75.63%	80.02%
SpotTune	**84.03 %**	**92.40%**	96.34%	75.77%	**80.20%**

Table 2: Results of *SpotTune* and baselines on CUBS, Stanford Cars, Flowers, WikiArt and Sketches.

Reference

Gumbel distribution trick

- http://jaejunyoo.blogspot.com/2018/09/
- https://hulk89.github.io/machine%20learning/2017/11/20/reparametrization-trick/
- https://data-newbie.tistory.com/263

Q&A