[ongoing] **NeurIPS 2019 Reproducibility Challenge**

Selected the paper Fast AutoAugment (FAA) [1]
- Fast AutoAugment learns augmentation policies using a more efficient search strategy based on density matching.
- Fast AutoAugment speeds up the search time by orders of magnitude while maintaining the comparable performances of original AutoAugment [2] paper

Excerpt from our proposed work - "*We are interested in evaluating FAA's effectiveness in out-of-distribution settings, that is, whether the policies learned are significantly more effective than randomized data augmentation strategies when the models are deployed in unseen test data.*"

[Report] Fast AutoAugment

Introduction

Deep learning has received increasing attention from researchers in machine learning community and has been successfully applied to various real-world applications.

Deep learning has a very strong dependence on massive training data compared to traditional machine learning method, because it needs a large amount of data to understand the latent pattern of data. Sufficient training data is one of the major deciding factors in feasibility of a deep learning. The collection of data is a time consuming and expensive task which makes it difficult to build large scale dataset. Data augmentation is one of the solutions to tackle the problem of lack of data to some extent. The "**Fast Autoaugmentation**" proposes the method of searching a policy for data augmentation as a density matching problem between a pair of train dataset. Let D be a probability distribution on X ×Y and assume dataset D is sampled from this distribution. For a given classification model M(·|θ) : X → Y that is parameterized by θ, the expected accuracy and the expected loss of M(·|θ) on dataset D are denoted by R(θ|D) and L(θ|D), respectively

We perform an ablation study to gauge the transferability of learned augmentation policy on one dataset to other dataset following the paradigm of transfer learning.
Transfer learning relaxes the hypothesis that the training data must be independent and identically distributed (i.i.d) with the test data, which motivates us to use the transfer learning against the problems of insufficient training data. In transfer learning, the training data and test data are not required to be i.i.d., and model in the target domain are not needed to be trained from scratch, which can significantly reduce the amount of training data required.

Experimentation

https://github.com/Ankit-Dhankhar/Fast-Auto-Augmentation
We performed all experiments on a single GeForce RTX 2080Ti GPU of 11 GB memory. System had 256 GB RAM, 16 cores where CPU is 2 x Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz.

Literature Review | References
[1] *Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, Sungwoong Kim. Fast AutoAugment. NeurIPS 2019*
[2] *Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. CVPR 2019*

We ran experiments on smaller dataset like MNIST, Fashion MNIST, Kuzushiji MNIST and Caltech 101 dataset with **Imagenet augmentation policy**. And we found that Imagenet policy generalize to other datasets as well. We performed ablation study to verify the results.

Dataset	Top-1 Accuracy without augmentation*	Top-1 accuracy with fast auto augmentation policy
Caltech	76.07%	79.72%
MNIST	99.55%	99.57%
Kuzishiji MNIST	98.79%	99.05%
Fashion MNIST	93.54%	95.75%

*Without augmentation refers to Cutout augmentation technique

Cutout augmentation technique is used in both methods without augmentation and fast autoaugmentation technique.

We used 40 layer deep Wide-resnet model for MNIST, Kuzushiji MNIST and Fashion MNIST dataset. And we used 200 layer deep Resnet model for Caltech 101 dataset.

For Caltech 101 dataset we used following hyper-parameters:
We used train test split of 90:10.
Batch Size : 16
Number of epochs model trained : 270 epochs
We used SGD optimizer with Nestrov momentum and weight decay(L2 penalty) of 0.0001
We used a learning rate of 0.1 to train the model which is decayed by a factor of 10 after epoch 90,180 and 240.

For MNIST dataset we used following hyperparameters:
Batch size : 512
Number of epochs for training model : 300 epochs
We used SGD optimizer with Nestrov momentum and weight decay(L2 penalty) of 0.0002

Scope of Improvement

This paper proposes search for data augmentation policy as search on small/proxy task which is transferable larger dataset. This formulation takes a makes a strong assumption that proxy task provides a predictive information of the larger task. They didn't compare their result with random grid search for finding policy.

This method find policy for a specific dataset, though our experiments proved them to be effective for other datasets as well. Another scope of improvement is that they try to search a optimal augmentation policy across variety of dataset which can be finetuned with the small amount of training data for other dataset in a meta-learning fashion.

<u>Intermediary Updates</u>:

- *(24 Nov)*
  CALTECH-101 isn't running on WideResnet 40, shifted to ResNet200;.
  FAA exhibits consistent yet small improvements over basic-augment training.

- *(18 Nov)*
  <u>Followed unofficial codebase</u> (scratch coded), reverified results of FAA
  Training on both MNIST and KMNIST takes ~3X longer to converge
  Lot of instability/oscillation in training, stabilizes in the late epochs

- *(12 Nov)*
  In FAA, Hyperparameters closely resemble values from AutoAugment [2] paper.
  I won't be exploring the direction of Generative models for augmentation.
  I read some of the 26 different augmentation techniques used in the paper
  I am looking deeper into the 3 main ideas of the paper from online resources:
  ..Density matching search
  ..Exploit-Explore Bayesian Optimization
  ..Tree-Parzen Estimator
  From an initial analysis, it looks like this method may not always translate well for bigger datasets/models but more tests beyond ImageNet are needed.