A Brief Survey on Data Augmentation and Few-shot Learning

Fei Jie
hfut_jf@aliyun.com

March 18, 2019

Abstract

It is common knowledge that the more data an ML algorithm has access to, the more effective it can be. Since the high cost of data labeling, often few supervised information is available. To reduce the required amount of data, various strategies and models were proposed to solve or alleviate this problem. This article tries to explore possible ways to improve models' performance in low-data regime, gain insights into development of this field, and relate to different methods. I will first introduce some concepts and problems. Then, two representative models are detailed to illustrate the general idea of mainstream methods.

1 Introduction

Low-data regime will result in ML models' overfitting in training dataset, which furthermore degrades the ability of generalization to test dataset. Thus, some tricks have been proposed to alleviate this problem, such as regularization, dropout[9], batch normalization[10], and layer normalization[2]. However in low data regimes, even these techniques fall short, since the the flexibility of the network is so high.

We first introduce some terms or concepts related to our topic.

Data Augmentation [12] is routinely used in classification problem. Often it is non-trivial to encode known **invariances** in a model. It can be easier to encode those invariances in the data instead by generating additional data items through transformations from existing data items. For example, the labels of handwritten characters should be invariant to small shifts in location, small rotations or shears, changes in intensity, changes in stroke thickness, changes in size etc. Almost all cases of data augmentation are from a prior known invariance. Traditional data augmentation operations include: *random translations, rotations, flips, addition of Gaussian noise,* etc. Note that aforementioned transformations **do not affect the class**.

Few-shot Learning is tasks that study the ability to learn from few examples [6]. When the classes covered by training instances and the classes we aim to classify are disjoint, this paradigm are called **zero-shot learning** [22]. If we only observe a single example of each possible before making a prediction about a test instance, we call it **one-shot learning** [11], which was firstly proposed in [4]. Here, we give a formal problem setup on several different learning settings (few-shot learning, semi-supervised learning and active learning) from the perspective of image classification.

We consider input-output pairs $(\mathcal{T}_i, Y_i)_i$ drawn iid from a distribution P of partially-labeled image collections

$$\mathcal{T} = \left\{\{(x_1, l_1), \ldots, (x_s, l_s)\}, \{\tilde{x}_1, \ldots, \tilde{x}_r\}, \{\bar{x}_1, \ldots, \bar{x}_t\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}(\mathbb{R}^N)\right\},$$

and $Y = (y_1, \ldots, y_t) \in \{1, K\}^t$, for arbitrary values of s, r, t and K. Where s is the number of labeled samples, r is the number of unlabeled samples (r > 0 for the semi-supervised and active learning scenarios) and t is the number of samples to classify. K is the number of classes. $\mathcal{P}_l(\mathbb{R}^N)$ denotes a class-specific image distribution over $\mathbb{R}^N$. In our context, the targets Y_i are associated with images categories of designated images $\hat{x}_1, \ldots, \hat{x}_t \in \mathcal{T}_i$ with no observed label. Given a training set $\{(\mathcal{T}_i, Y_i)_i\}_{i \leq L}$, we consider the standard supervised learning objective

$$\min_{\Theta} \frac{1}{L} \sum_{i \leq L} \ell(\Phi(\mathcal{T}_i; \Theta), Y_i) + \mathcal{R}(\Theta),$$

using the model $\Phi(\mathcal{T}; \Theta) = p(Y|\mathcal{T})$ and $\mathcal{R}$ is a standard regularization objective.

Few-shot Learning When $r = 0, t = 1$ and $s = qK$, there is a single image in the collection with unknown label. If moreover each label appears exactly q times, this setting is referred as the **q-shot, K-way learning**.

Semi-supervised Learning When $r > 0$ and $t = 1$, the input collection contains auxiliary images $\tilde{x}_1, \ldots, \tilde{x}_r$ that model can use to improve the prediction accuracy, by leveraging the fact that these samples are drawn from common distributions as those determining the outputs.

Active Learning In the active learning setting, the learner has the ability to request labels from the sub-collection $\{\tilde{x}_1, \dot{,}\tilde{x}_r\}$.

Some other terms related to this articles are Generative Adversarial Networks (GANs) [7], Transfer Learning, Meta Learning. I will introduce them when they are to be used.

Except aforementioned tricks to overcome the lack of data and achieve few-shot learning, two intuitive solutions for this problems are **data augmentation** and **meta learning** [18]. Data augmentation is a intuitive strategy to increase the amount of available data and thus also useful for few-shot learning. In contrast to data-augmentation methods, meta-learning is a task-level learning method [20]. Meta-learning aims to accumulate experience from learning multiple tasks [13, 15, 5], while base-learning focuses on modeling the data distribution of a single task.

2 Data Augmentation

Data augmentation is another way we can reduce overfitting on models, where we increase the amount of training data using information only in our training data. Based on difference between deployment of data augmentation, it can be categorized into three classes:

Traditional Data Augmentation A very generic and accepted current practice for augmenting image data is to perform geometric and color augmentations, such as reflecting the image, cropping and translating the image, and changing the color palette of the image. All of the transformation are affine transformation of the original image that take the form:

$$y = Wx + b$$

Generative Adversarial Networks GANs has been a powerful technique to perform unsupervised generation of new images for training. They have also proven extremely effective in many data generation tasks. By using a min-max strategy, one neural net successively generates better counterfeit samples from the original data distribution in order to fool the other net. The other net is then trained to better distinguish the counterfeits. GANs have been used for style transfer such as transferring images in one setting to another setting (CycleGAN). These generated images could be used to train a car to drive in night or in the rain using only data collected on sunny days for instance. Further- more, GANs have been effective even with relatively small sets of data [8] by performing transfer learning techniques. We can use GANs or its variations to generate extra data. Note that in this setting the data augmentation model is independent of the classifiers [14, 1], i.e., augmented data is generated before training the classifier.

A variation of GAN for data augmentation can be illustrated as figure 2 [1].

DAGAN's generator differs from the standard GANs'. The generative model learnt by a Generative Adversarial Network (GAN) takes the form

$$\mathbf{z} = \tilde{N}(\mathbf{0}, \mathbf{I})$$
$$\mathbf{v} = f(\mathbf{z})$$

where f is implemented via a neural network. Here, $\mathbf{v}$ are the vectors being generated (that, in distribution, should match the data D), and $\mathbf{z}$ are the latent Gaussian variables that provide the variation in what is generated. The generator of DAGAN model takes the form

$$\mathbf{r} = g(\mathbf{x})$$
$$\mathbf{z} = \tilde{N}(\mathbf{0}, \mathbf{I})$$
$$\mathbf{v} = f(\mathbf{z}, \mathbf{r})$$

where the neural network f now takes the representation $\mathbf{r}$ and the random $\mathbf{z}$ as inputs.

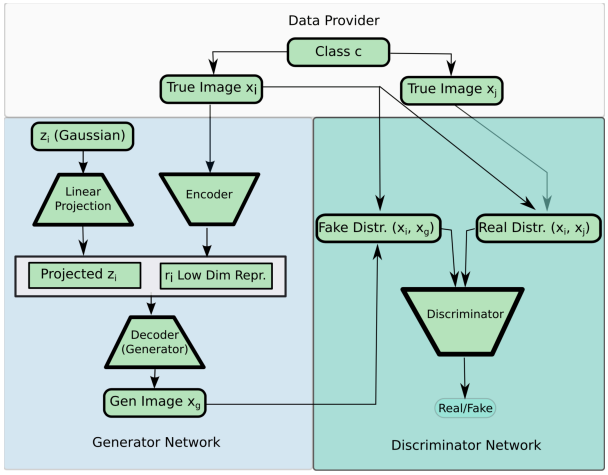

Figure 1: DAGAN Architecture. Left: the generator network is composed of an encoder taking an input image (from class c), projecting it down to a lower dimensional manifold (bottleneck). A random vector (zi) is transformed and concatenated with the bottleneck vector; these are both passed to the decoder network which generates an augmentation image. Right: the adversarial discriminator network is trained to discriminate between the samples from the real distribution (other real images from the same class) and the fake distribution (images generative from the generator network). Adversarial training leads the network to generate new images from an old one that appear to be within the same class (whatever that class is), but look different enough to be a different sample.

Learning the Augmentation This approach attempts to learn augmentation through a pre-pended neural net. At training time, the neural net takes data from the training set and outputs fake (augmented) data. Then the augmented data is fed into the second classifying network along with the original training data. The training loss is then backpropagated to train the augmenting layers of the networks as well as the classification layers of the network. In test time, instances from the validation or test set is ran through only the classification network. The motivation is to identify the best augmentations for a given dataset [14]. Note that the architecture of learning the augmentation is similar with DVN.

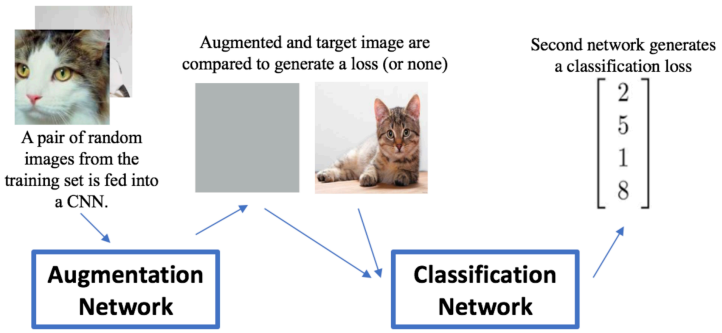

Figure 2: Architecture of Learning the Augmentation.

3 Meta Learning

Another solution for few-shot learning is meta learning paradigm. We can divide methods in this paradigm into three categories. 1) Metric learning methods [17, 19, 21] learn a similarity space in which learning

is particularly efficient for few-shot exam- ples. 2) Memory network methods [13, 16] learn to store "experience" when learning seen tasks and then gener- alize that to unseen tasks. 3) Gradient descent methods [5, 15] have a specific meta-learner that learns to adapt a specific base-learner (to few-shot examples) through different tasks.

Metric learning I will use Prototypical Networks [17] as a instance to illustrate the idea of metric learning. The Prototypical Networks is based on the idea that *there exists an embedding in which points cluster around a single prototype representation for each class.* To do this, we learn a non-linear mapping of the input into an embedding space using a neural network and take a class's prototype to be the mean of its support set in the embedding space.

Given a small support set of N labeled examples $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where each $x_i \in \mathbb{R}^D$ is the D-dimensional feature vector of an example and $y_i \in \{1, \ldots, K\}$ is the corresponding label. S_k denotes the set of examples labeled with class k.

Prototypical Networks compute an M-dimensional representation $c_k \in \mathbb{R}^M$, or *prototype*, of each class through an embedding function $f_\phi : \mathbb{R}^D \to \mathbb{R}^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class:

$$c_k = \frac{1}{S_k} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i) \tag{1}$$

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, Prototypical Networks produce a distribution over classes for a query point x bases on a softmax over distances to the prototypes in the embedding space

$$p_\phi(y = k|x) = \frac{\exp(-d(f_\phi(x), c_k))}{\sum_{k'} \exp(-d(f_\phi(x), c_{k'}))} \tag{2}$$

Learning proceeds by minimizing the negative log-probability $J(\phi) = -logp_\phi(y = k|x)$ of the true class k via SGD. Training episodes are formed by randomly selecting a subset of classes from the training set, then choosing a subset of examples within each class to act as the support set and a subset of the reminder to serve as query points.

Algorithm 1 Training episode loss computation for Prototypical Networks.

Input $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subsets of D containing all elements (x_i, y_i) such that $y_i = k$.
Output The loss J for a randomly generated training episodes.
$V \leftarrow$ RandomSample($\{1, \ldots, K\}, N_C$) ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RandomSample($\mathcal{D}_{V_k}, N_S$) ▷ Select support examples
  $Q_k \leftarrow$ RadnomSample($\mathcal{D}_{V_k \setminus S_k}, N_Q$) ▷ Select query examples
  $c_k \leftarrow \frac{1}{N_C} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for (x, y) in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(x), c_k) + \log \sum_{k'} \exp(-d(f_\phi(x), c_{k'})) \right]$ ▷ Update loss
  end for
end for

Prototypical Networks as Mixture Density Estimation For a particular class of distance functions, Bregman divergences [3], the Prototypical Networks is equivalent to performing mixture density estimation on the support set with an exponential family density. Examples of Bregman divergences include squared Euclidean distance $\|z - z'\|_2^2$.

Prototype computation can be viewed in terms of hard clustering on the support set, with one cluster per class and each support point assigned to its corresponding class cluster. It has been shown for Bregman divergences that the cluster representative achieving minimal distance to its assigned points is the cluster mean. Thus the prototype computation in Equation (1) yields optimal cluster representatives given the support set labels when a Bregman divergence is used.

Moreover, any regular exponential family distribution $p_\psi(z|\theta)$ with parameters θ and cumulant function ψ can be written in terms of a uniquely determined regular Bregman divergence

$$p_\psi(z|\theta) = \exp(z^T\theta - \psi(\theta) - g_\psi(z)) = \exp(-d_\varphi(z, \mu(\theta)) - g_\varphi(z)) \tag{3}$$

Consider now a regular exponential family mixture model with parameters $\Gamma = \{\theta_k, \pi_k\}_{k=1}^K$

$$p(z|\Gamma) = \sum_{k=1}^K \pi_k p_\psi(z|\theta_k) = \sum_{k=1}^K \exp(-d_\varphi(z, \mu(\theta_k)) - g_\varphi(z)) \tag{4}$$

Given Γ, inference of the cluster assignment y for an unlabeled point z becomes

$$p(y = k|z) = \frac{\pi_k \exp(-d_\varphi(z, \mu(\theta_k)))}{\sum_{k'} \pi_{k'} \exp(-d_\varphi(z, \mu(\theta_k)))} \tag{5}$$

For an equally-weighted mixture model with one cluster per class, cluster assignment inference (5) is equivalent to query class prediction (2) with $f_\phi(x) = z$ and $c_k\mu(\theta_k)$. In this case, Prototypical Networks are effectively performing mixture density estimation with an exponential family distribution determined by d_φ. The choice of distance therefore specifies modeling assumptions about the class-conditional data distribution in the embedding space.

References

[1] Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[2] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. Clustering with bregman divergences. *Journal of machine learning research 6*, Oct (2005), 1705–1749.

[4] Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence 28*, 4 (2006), 594–611.

[5] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 1126–1135.

[6] Garcia, V., and Bruna, J. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043* (2017).

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[8] Gurumurthy, S., Kiran Sarvadevabhatla, R., and Venkatesh Babu, R. Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 166–174.

[9] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[10] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[11] Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop* (2015), vol. 2.

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[13] Munkhdalai, T., and Yu, H. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 2554–2563.

[14] Perez, L., and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

[15] Ravi, S., and Larochelle, H. Optimization as a model for few-shot learning.

[16] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International conference on machine learning* (2016), pp. 1842–1850.

[17] Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems* (2017), pp. 4077–4087.

[18] Sun, Q., Liu, Y., Chua, T.-S., and Schiele, B. Meta-transfer learning for few-shot learning. *arXiv preprint arXiv:1812.02391* (2018).

[19] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1199–1208.

[20] Thrun, S., and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

[21] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems* (2016), pp. 3630–3638.

[22] Wang, W., Zheng, V. W., Yu, H., and Miao, C. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol. 10*, 2 (Jan. 2019), 13:1–13:37.