Deep Learning for Computer Vision

Zero-shot and Few-shot Learning

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Review

Question

- Last lecture, we saw methods that use videos as input for generating videos. Can we generate a video from a single image?

Review

Question

- Last lecture, we saw methods that use videos as input for generating videos. Can we generate a video from a single image? **See this article and this paper; it uses single-shot learning, which is what this lecture is about**

Learning with Limited Supervision

Problem

- Deep learning models $\implies$ heavy reliance on labeled data during training
- Not directly suited to learn from few samples
- Regularization techniques reduce overfitting in low data regimes, but do not solve the problem

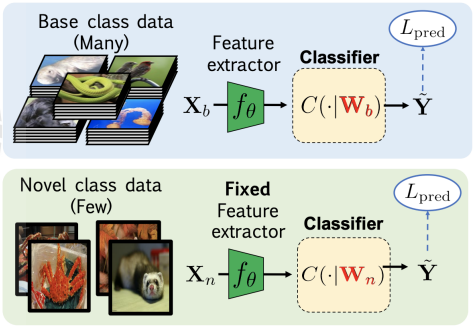
Learning with Limited Supervision

Problem

- Deep learning models $\implies$ heavy reliance on labeled data during training
- Not directly suited to learn from few samples
- Regularization techniques reduce overfitting in low data regimes, but do not solve the problem

Solution

- Train models capable of rapidly generalizing to new tasks with only a few samples
- Enable models to perform under practical scenarios where data annotation is infeasible or new classes are dynamically included with time



Credit: Chen et al, A Closer Look at Few-Shot Classification, ICLR 2019

Problem Setting

Let x denote an image/feature (produced by a pre-trained network), y denote label

Few-Shot Learning (FSL)

- Training data $D_{train} = (x_i, y_i)_{i=1}^{I}$, where few training samples for certain classes
- Specifically, **N-way-K-shot FSL** problem: D_{train} contains only few examples, K, from N of the overall number of classes (other classes called **base classes**)

Problem Setting

Let x denote an image/feature (produced by a pre-trained network), y denote label

Few-Shot Learning (FSL)

- Training data $D_{train} = (x_i, y_i)_{i=1}^{I}$, where few training samples for certain classes
- Specifically, **N-way-K-shot FSL** problem: D_{train} contains only few examples, K, from N of the overall number of classes (other classes called **base classes**)

Zero-Shot Learning (ZSL)

- Training data $S = \{(x, y, a(y))|x \in X^s, y \in Y^s, a(y) \in A\}$, where X^s is set of image/features from seen classes, Y^s is set of seen class labels, $a(y)$ is semantic embedding for class y
- Test set $U = \{(x, y, a(y))|x \in X^u, y \in Y^u, a(y) \in A\}$, where X^u is set of unseen class image/features, Y^u is set of unseen class labels, $Y^u \cap Y^s = \emptyset$

Problem Setting

Based on classes that a model sees in test phase, FSL/ZSL problem generally categorized into two settings:

Conventional FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u$
- Image/feature x to be recognized at test time belongs only to unseen/few-shot classes

Problem Setting

Based on classes that a model sees in test phase, FSL/ZSL problem generally categorized into two settings:

Conventional FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u$
- Image/feature x to be recognized at test time belongs only to unseen/few-shot classes

Generalized FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u \cup Y^u$
- Image/feature x to be recognized at test time may belong to seen/base or unseen/few-shot classes
- Practically more useful and challenging than conventional setting, since assumption that images at test time come only from unseen/few-shot classes need not hold

Recall: Supervised Learning

Empirical Risk Minimization

Let $p(x, y)$ be ground-truth joint probability distribution of input x and output y

Recall: Supervised Learning

Empirical Risk Minimization

Let $p(x, y)$ be ground-truth joint probability distribution of input x and output y

- Given hypothesis h, we want to minimize its expected risk R, loss measured w.r.t. $p(x, y)$, i.e.

$$R(h) = \int l(h(x), y) d(p(x, y)) = E[l(h(x), y)]$$

Recall: Supervised Learning

Empirical Risk Minimization

Let $p(x, y)$ be ground-truth joint probability distribution of input x and output y

- Given hypothesis h, we want to minimize its expected risk R, loss measured w.r.t. $p(x, y)$, i.e.

$$R(h) = \int l(h(x), y) d(p(x, y)) = E[l(h(x), y)]$$

- As $p(x, y)$ is unknown, empirical risk $R_I(h)$ is used as proxy for $R(h)$, leading to empirical risk minimization:

$$R_I(h) = \frac{1}{I} \sum_{i=1}^{I} l(h(x_i, y_i))$$

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = \arg\min_h R(h)$ be function that minimizes expected risk;
  $h^* = \arg\min_{h \in H} R(h)$ be function in H that minimizes expected risk
  $h_I = \arg\min_{h \in H} R_I(h)$ be function in H that minimizes empirical risk

Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = \arg\min_h R(h)$ be function that minimizes expected risk;
  $h^* = \arg\min_{h \in H} R(h)$ be function in H that minimizes expected risk
  $h_I = \arg\min_{h \in H} R_I(h)$ be function in H that minimizes empirical risk
- **Error Decomposition:**

$$E[R(h_I) - R(\hat{h})] = \underbrace{E[R(h^*) - R(\hat{h})]}_{\mathcal{E}_{app}(H)} + \underbrace{E[R(h_I) - R(h^*)]}_{\mathcal{E}_{est}(H,I)}$$

Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = \arg\min_h R(h)$ be function that minimizes expected risk;
  $h^* = \arg\min_{h \in H} R(h)$ be function in H that minimizes expected risk
  $h_I = \arg\min_{h \in H} R_I(h)$ be function in H that minimizes empirical risk
- **Error Decomposition:**

$$E[R(h_I) - R(\hat{h})] = \underbrace{E[R(h^*) - R(\hat{h})]}_{\mathcal{E}_{app}(H)} + \underbrace{E[R(h_I) - R(h^*)]}_{\mathcal{E}_{est}(H,I)}$$

  $\mathcal{E}_{app}$ (**approximation error)** measures how closely functions in H can approximate optimal hypothesis $\hat{h}$
  $\mathcal{E}_{est}$ (**estimation error**) measures effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

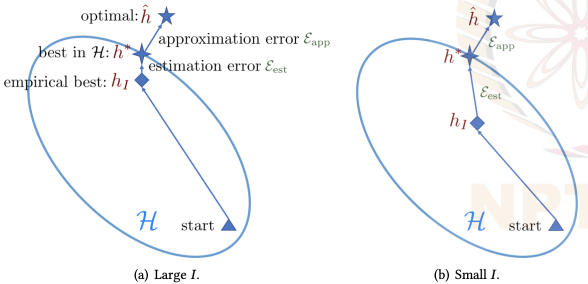
Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Problems with Few/Zero-shot Setting

- $\mathcal{E}_{est} \implies$ can be reduced with large training dataset
- Sufficient labeled train data with (i.e. large I) $\implies$ Empirical risk minimizer $R(h_I)$ gives good approximation to best possible $R(h^*)$

Problems with Few/Zero-shot Setting

- $\mathcal{E}_{est} \implies$ can be reduced with large training dataset
- Sufficient labeled train data with (i.e. large I) $\implies$ Empirical risk minimizer $R(h_I)$ gives good approximation to best possible $R(h^*)$



(a) Large I. (b) Small I.

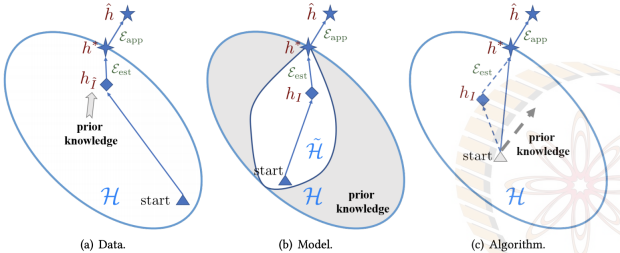
Comparison of learning with sufficient and few training samples

Few-shot Learning:

- Number of labeled examples I is small
  - $R_I(h) \implies$ far from being good approximation of expected risk $R(h)$
  - Resultant empirical risk minimizer h_I overfits

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Addressing Few/Zero-shot Learning



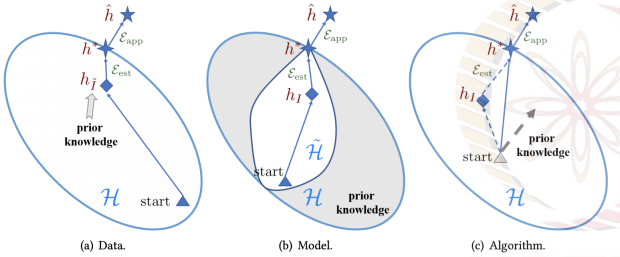
(a) Data. (b) Model. (c) Algorithm.

Different perspectives of addressing few-shot learning

Data

- Learn to augment training set $\Longrightarrow$ increase number of samples to $\tilde{I} >> I$
- More accurate empirical risk minimizer $h_{\tilde{I}}$ can be obtained

Addressing Few/Zero-shot Learning



(a) Data. (b) Model. (c) Algorithm.

Different perspectives of addressing few-shot learning

Data

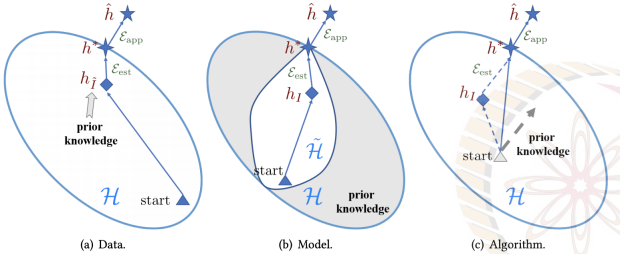
- Learn to augment training set $\implies$ increase number of samples to $\tilde{I} >> I$
- More accurate empirical risk minimizer $h_{\tilde{I}}$ can be obtained

Model

- Constrain complexity of H $\implies$ much smaller hypothesis space $\tilde{H}$
- Then, D_{train} may be sufficient to learn a reliable h_I

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Addressing Few/Zero-shot Learning



(a) Data. (b) Model. (c) Algorithm.

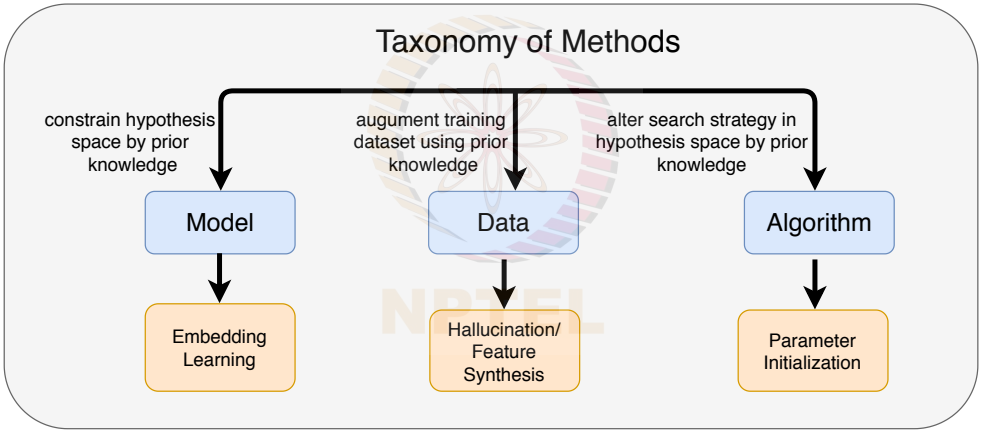
Different perspectives of addressing few-shot learning

Algorithm

- Search for parameters θ for best hypothesis h^* in H
- Prior knowledge alters search strategy by providing a good initialization (gray triangle in Fig (c))

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Taxonomy of Methods



Embedding Learning Methods

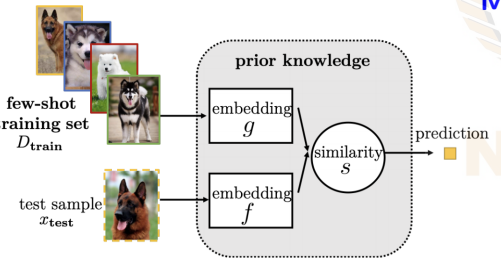
Intuition:

- Address few-shot learning by "learning to compare"
- If model can determine similarity of two images (and perhaps corresponding semantics of classes), it can classify unseen input in relation to labeled instance seen during training

Embedding Learning Methods

Intuition:

- Address few-shot learning by "learning to compare"
- If model can determine similarity of two images (and perhaps corresponding semantics of classes), it can classify unseen input in relation to labeled instance seen during training



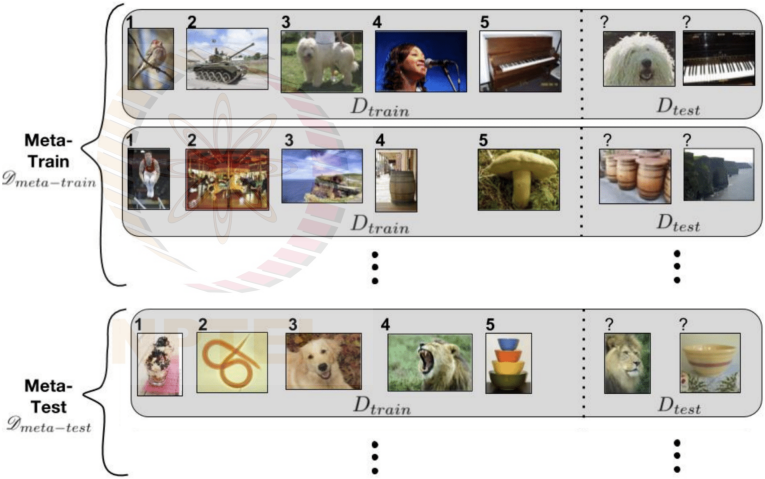
Method:

- Learn separate embedding functions for training samples D_{train} and test samples D_{test}
- Train sophisticated comparison models end-to-end via **meta-learning**
- At test time, predict based on comparing distance between x_{test} feature and training set features from each class

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Problem Setup: Meta-Learning

- **N-way-K-shot:** N different classes in D_{train} with K samples per class
- $D_{meta-train} = (D_{train}, D_{test})$ set $\rightarrow$ one task/episode
- Ensure $D_{meta-train}$ and $D_{meta-test}$ have disjoint/different classes



Credit: Hugo Larochelle

Problem Setup: Meta-Learning

Learning Algorithm A

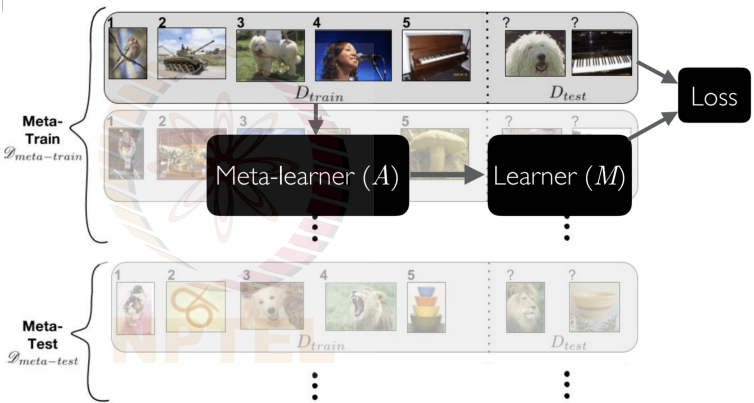
- **Input**: Training set $D_{train} = (x_i, y_i)_{i=1}^{I}$
- **Output**: Parameter θ model M (the **learner**)
- **Objective**: Good performance on $D_{test} = (x'_i, y'_i)$

Meta-Learning Algorithm

- **Input**: Meta-training set $D_{meta-train} = (D_{train}^{(n)}, D_{test}^{(n)})_{n=1}^{N}$ of tasks/episodes
- **Output**: Parameter Θ algorithm A(the **meta-learner**)
- **Objective**: Good performance on meta-test set $D_{meta-test} = (D_{train}^{\prime(n)}, D_{train}^{\prime(n)})_{n=1}^{N}$

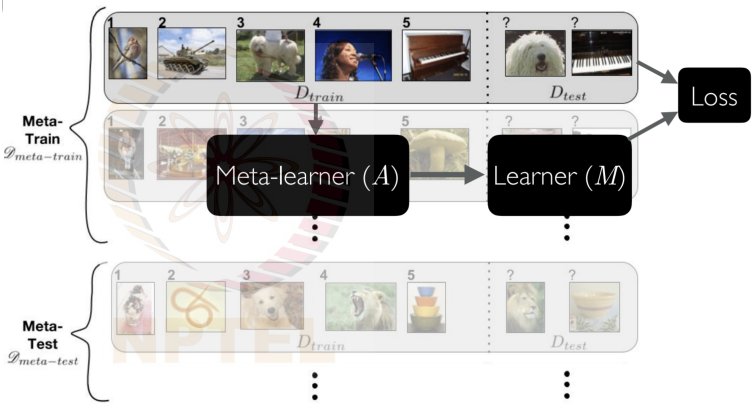
Training Setup: Meta-Learning

- **Training:** Repeat meta-loop for each task/mini-batch of tasks in $D_{meta-train}$ as shown
- **Inference:** On tasks/episodes in $D_{meta-test}$



Training Setup: Meta-Learning

- **Training:** Repeat meta-loop for each task/mini-batch of tasks in $D_{meta-train}$ as shown
- **Inference:** On tasks/episodes in $D_{meta-test}$



Problem setup matches training and inference to enable generalization to new classes at test-time

Credit: Hugo Larochelle

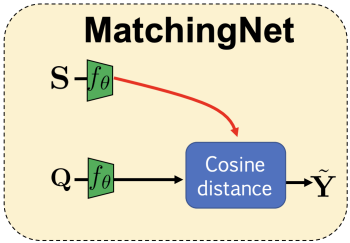
Matching Networks[1]

- **Parametric models:** Slowly learn model parameters from training examples;
  Require large datasets to avoid overfitting
- **Non-parametric models:** Allow novel examples to be rapidly assimilated;
  Robust to **catastrophic forgetting**

[1] Vinyals et al, Matching Networks for One Shot Learning, NeurIPS 2016

Matching Networks[1]

- **Parametric models:** Slowly learn model parameters from training examples; Require large datasets to avoid overfitting
- **Non-parametric models:** Allow novel examples to be rapidly assimilated; Robust to **catastrophic forgetting**

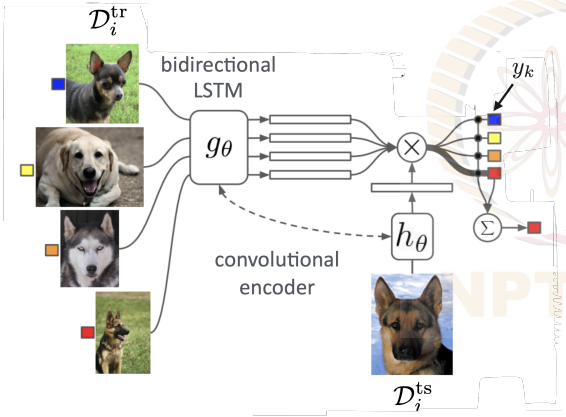


Idea: Combine best of both worlds

- **Training Phase:** Learn cosine similarity-based embedding models (parametric meta-learners)
- **Test Phase:** Use Nearest Neighbors (non-parametric) in learned embedding space

[1] Vinyals et al, Matching Networks for One Shot Learning, NeurIPS 2016

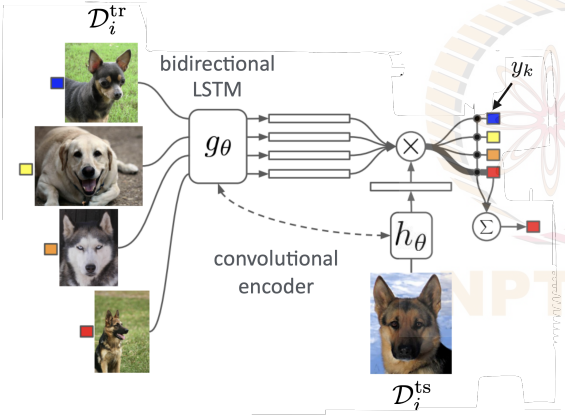
Matching Networks



- $\hat{y}_{ts} = \sum_{i=1}^{k} a(\hat{x}, x_i) * y_i$

where $a(\hat{x}, x_i)$ denotes the attention mechanism over examples

Matching Networks



- $\hat{y}_{ts} = \sum_{i=1}^{k} a(\hat{x}, x_i) * y_i$
  where $a(\hat{x}, x_i)$ denotes the attention mechanism over examples
- Simplest form of attention mechanism $\implies$ softmax over cosine distances $c(.,.)$

$$a(\hat{x}, x_i) = \frac{e^{c(h(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(h(\hat{x}), g(x_j))}}$$

Relation Networks[2]

Idea:

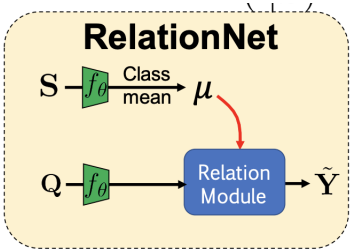
- Train data-driven learnable non-linear metric end-to-end instead of manually picking one
- Enables model to extract complex non-linear relationship among data samples
  $\implies$ generalize better to novel classes
- Easily extensible to tackle more difficult zero-shot setting

[2] Sung et al, Feature Generating Networks for Zero-Shot Learning, CVPR 2018

Relation Networks[2]

Idea:

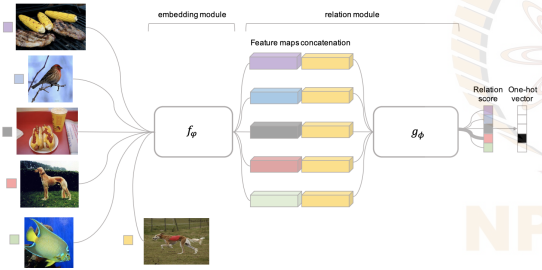
- Train data-driven learnable non-linear metric end-to-end instead of manually picking one
- Enables model to extract complex non-linear relationship among data samples
  $\implies$ generalize better to novel classes
- Easily extensible to tackle more difficult zero-shot setting



- **Training Phase:** Meta-learn both embedding module (for feature representation) and relation module (learned transferable deep metric) end-to-end
- **Test Phase:** Use relation scores in embedding space to classify new samples

[2]Sung et al, Feature Generating Networks for Zero-Shot Learning, CVPR 2018

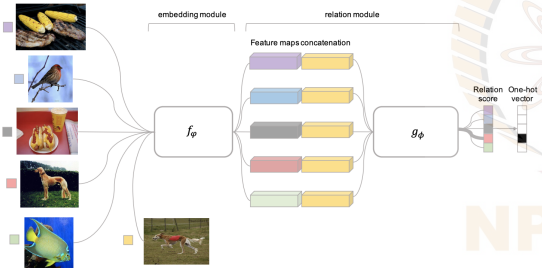
Relation Networks



One-shot Learning:

- Samples $x_j \in Q$ and $x_i \in S$ are fed into embedding module f_φ
- Feature maps $f_\varphi(x_j)$ and $f_\varphi(x_i)$ combined with concatenation operator $C(f_\varphi(x_j), f_\varphi(x_i))$ to extract relations

Relation Networks

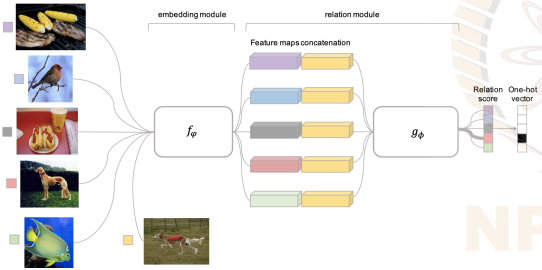


One-shot Learning:

- Samples $x_j \in Q$ and $x_i \in S$ are fed into embedding module f_φ
- Feature maps $f_\varphi(x_j)$ and $f_\varphi(x_i)$ combined with concatenation operator $C(f_\varphi(x_j), f_\varphi(x_i))$ to extract relations
- Combined feature map fed into relation module g_ϕ, $\implies$ produces a scalar $\in (0, 1)$

$$r_{ij} = g_\phi(C(f_\varphi(x_j), f_\varphi(x_i)))$$

Relation Networks



One-shot Learning:

- Samples $x_j \in Q$ and $x_i \in S$ are fed into embedding module f_φ
- Feature maps $f_\varphi(x_j)$ and $f_\varphi(x_i)$ combined with concatenation operator $C(f_\varphi(x_j), f_\varphi(x_i))$ to extract relations
- Combined feature map fed into relation module g_ϕ, $\implies$ produces a scalar $\in (0, 1)$

$$r_{ij} = g_\phi(C(f_\varphi(x_j), f_\varphi(x_i)))$$

- **Objective:**

$$\varphi, \phi \leftarrow \arg\min_{\varphi,\phi} \sum_{i=1}^{m} \sum_{i=1}^{n} (r_{ij} - 1(y_i == y_j))^2$$

Relation Networks: Application to Few/Zero-shot Learning

Few-shot Learning (K shot; $k > 1$)

- Use element-wise sum over embedding module outputs of samples from each training class (class feature map)
- Combine pooled class-level feature maps with query image feature map
  Number of relation scores for one query is always C (both one-shot or few-shot setting)

Relation Networks: Application to Few/Zero-shot Learning

Few-shot Learning (K shot; $k > 1$)

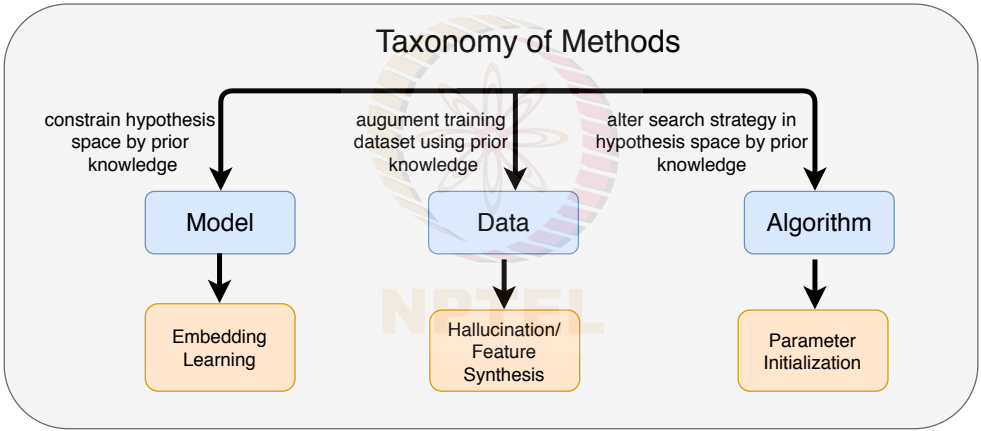
- Use element-wise sum over embedding module outputs of samples from each training class (class feature map)
- Combine pooled class-level feature maps with query image feature map
  Number of relation scores for one query is always C (both one-shot or few-shot setting)

Zero-shot Learning:

- Support set contains semantic class embeddings $\implies$ vector v_c; instead of one-shot image for each class
- In addition to f_{φ_1} (for query images), introduce embedding module f_{φ_2} to handle semantic attributes

$$r_{ij} = g_\phi(C(f_{\varphi_2}(v_c), f_{\varphi_1}(x_j)))$$

Taxonomy of Methods



Hallucination/Feature Synthesis Methods

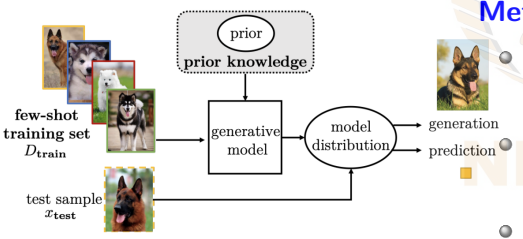
Intuition:

- Directly deal with data deficiency by "learning to augment"
- Learn a generative model to hallucinate new novel class data for data augmentation
- Reduce few/zero-shot problem to a standard supervised learning problem

Hallucination/Feature Synthesis Methods

Intuition:

- Directly deal with data deficiency by "learning to augment"
- Learn a generative model to hallucinate new novel class data for data augmentation
- Reduce few/zero-shot problem to a standard supervised learning problem

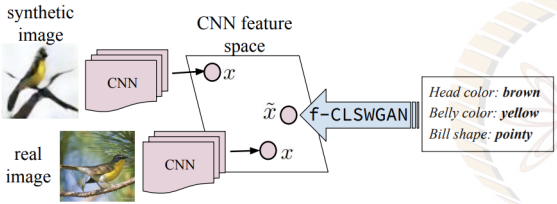


Method:

- Learn a generator conditioned on meta-information using data in base classes
- Generate novel class features conditioned on unseen class meta-information
- Train a classifier on base class samples and generated novel class samples

Credit:Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

Feature Generating Networks (f-CLSWGAN)[3]

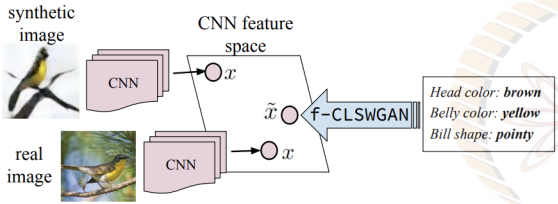


Method:

- Given train set S of seen classes, learn a conditional generator $G : Z \times C \rightarrow X$, which takes random Gaussian noise $z \in Z \subset R^{dz}$ and class embedding $c(y) \in C$, and outputs image feature $\tilde{x} \in X$
- To ensure $\tilde{x}$ are well-suited to train a discriminative classifier, minimize classification loss over generated features $\tilde{x}$

[3]Xian et al, Feature Generating Networks for Zero-Shot Learning, CVPR 2018

Feature Generating Networks (f-CLSWGAN)[3]



Method:

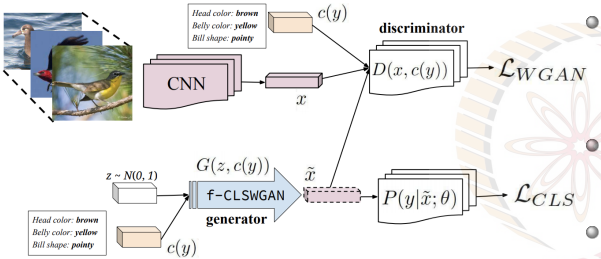
- Given train set S of seen classes, learn a conditional generator $G : Z \times C \rightarrow X$, which takes random Gaussian noise $z \in Z \subset R^{dz}$ and class embedding $c(y) \in C$, and outputs image feature $\tilde{x} \in X$
- To ensure $\tilde{x}$ are well-suited to train a discriminative classifier, minimize classification loss over generated features $\tilde{x}$

Extension to Few-shot Learning:

- For FSL, along with seen classes data set S, the training data also includes few labeled samples for each unseen class as well

[3]Xian et al, Feature Generating Networks for Zero-Shot Learning, CVPR 2018

Feature Generating Networks (f-CLSWGAN)



Loss Formulation:

- **GAN Loss:**

$$\mathcal{L}_{WGAN} = E[D(x, c(y))] - E[D(\tilde{x}, c(y))] - \lambda E[(||\nabla_{\hat{x}} D(\hat{x}, c(y))||_2 - 1)^2]$$

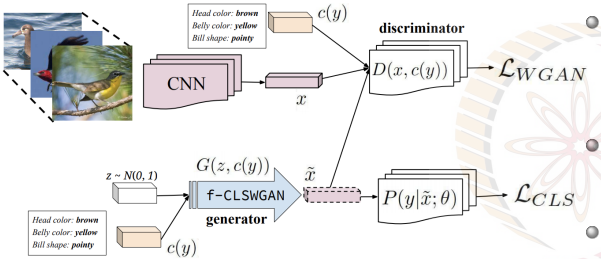
- **Classification Loss:**

$$\mathcal{L}_{CLS} = -E_{\tilde{x} \sim p_{\tilde{x}}}[\log P(y|\tilde{x};\theta)]$$

- **Final loss:**

$$L_{total} = \min_G \max_D \mathcal{L}_{WGAN} + \beta \mathcal{L}_{CLS}$$

Feature Generating Networks (f-CLSWGAN)



Loss Formulation:

- **GAN Loss:**

$$\mathcal{L}_{WGAN} = E[D(x, c(y))] - E[D(\tilde{x}, c(y))] - \lambda E[(||\nabla_{\hat{x}} D(\hat{x}, c(y))||_2 - 1)^2]$$

- **Classification Loss:**

$$\mathcal{L}_{CLS} = -E_{\tilde{x} \sim p_{\tilde{x}}}[\log P(y|\tilde{x}; \theta)]$$

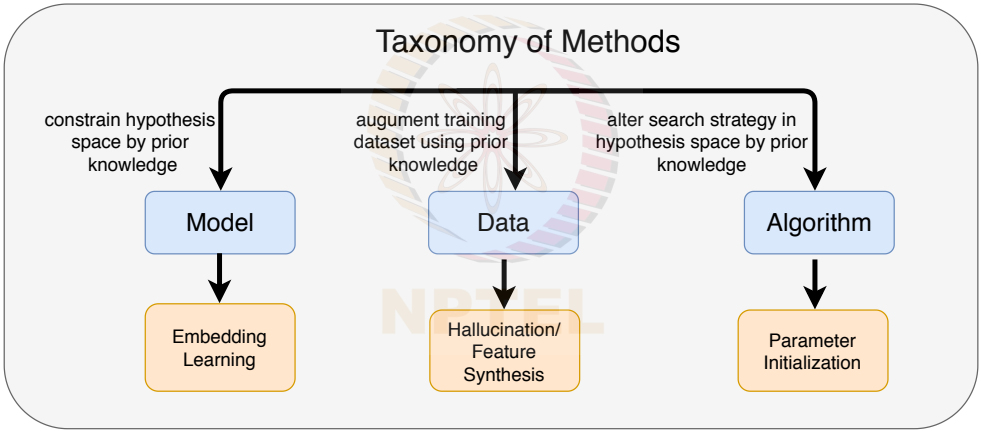
- **Final loss:**

$$L_{total} = \min_G \max_D \mathcal{L}_{WGAN} + \beta \mathcal{L}_{CLS}$$

Training Classifier:

- Use pre-trained generator to generate samples novel/unseen class samples conditioned on class embeddings
- Train softmax classifier on train set and generated unseen class image features

Taxonomy of Methods



Parameter Initialization Methods

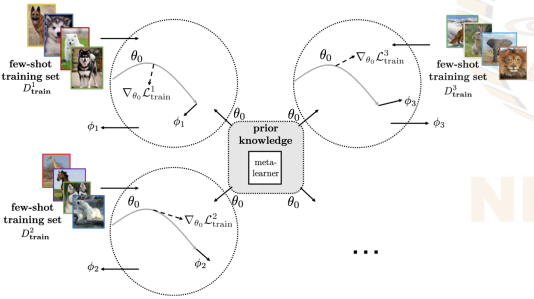
Intuition:

- Tackle the few-shot learning problem by "learning to fine-tune"
- Learn parameters that transfer via few-gradient steps (fine-tuning) to novel tasks

Parameter Initialization Methods

Intuition:

- Tackle the few-shot learning problem by "learning to fine-tune"
- Learn parameters that transfer via few-gradient steps (fine-tuning) to novel tasks

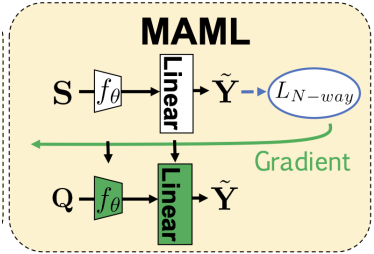


Method:

- For each task/episode $(D^i_{train}, D^i_{test})$, update task-specific parameters Φ_i to minimize $L(\theta, D^i_{train})$
- Update meta parameter θ to minimize $\sum L(\Phi_i, D^i_{test})$
- At test time, use few gradient steps to adapt classify novel classes

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20

MAML[4]



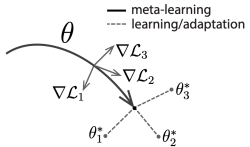
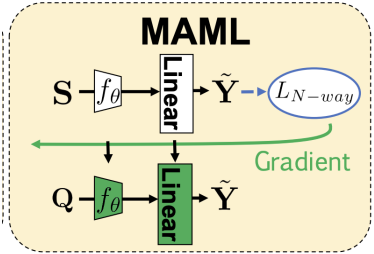


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Key Idea: Acquire task-specific parameters (Φ_i) through optimization

[4]Finn et al, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, ICML 2017

MAML[4]



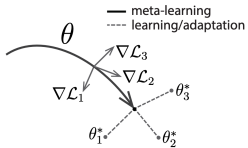


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

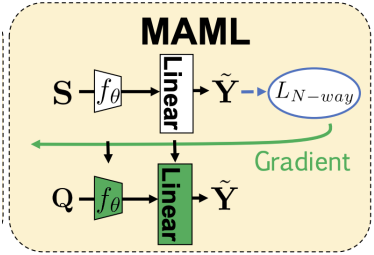
Key Idea: Acquire task-specific parameters (Φ_i) through optimization

Formulation

- Learn prior such that model can adapt to new tasks
  One form of prior knowledge: **Parameter Initialization**

[4] Finn et al, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, ICML 2017

MAML[4]



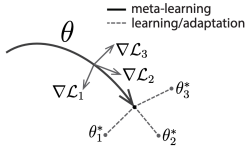


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

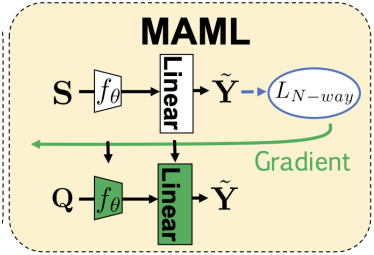
Key Idea: Acquire task-specific parameters (Φ_i) through optimization

Formulation

- Learn prior such that model can adapt to new tasks
  One form of prior knowledge: **Parameter Initialization**
- **Task-specific Update:**
  $\theta_i' = \theta - \alpha \nabla_\theta L(\theta, D^i_{train})$ (Single or multiple SGD updates)

[4] Finn et al, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, ICML 2017

MAML[4]



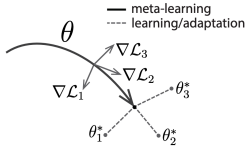


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Key Idea: Acquire task-specific parameters (Φ_i) through optimization

Formulation

- Learn prior such that model can adapt to new tasks
  One form of prior knowledge: **Parameter Initialization**
- **Task-specific Update:**
  $\theta_i' = \theta - \alpha\nabla_\theta L(\theta, D^i_{train})$ (Single or multiple SGD updates)
- **Meta Fine-tuning:**
  $\theta = \theta - \beta\nabla_\theta \sum_i L(\theta_i', D^i_{test})$
  $\theta = \theta - \beta\nabla_\theta \sum_i L(\theta - \alpha\nabla_\theta L(\theta, D^i_{train}), D^i_{test})$
  (Second-order derivatives)

[4] Finn et al, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, ICML 2017

MAML: Few-shot Algorithm

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

Meta-Learning for Generalized Zero-Shot Learning[5]

Idea:

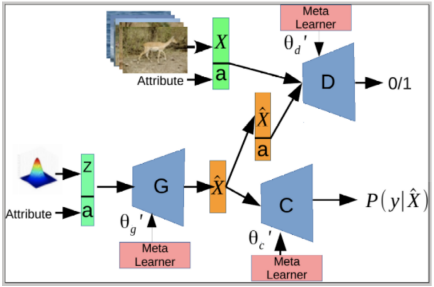
- Learn a GAN conditioned on class attributes and train using meta-learning framework (MAML) to facilitate generalization to novel classes
- Mimic ZSL setup during training $\implies$ for each task/episode $T_i = T_{tr}, T_{val}$, classes of T_{tr} and T_{val} are disjoint

[5] Verma et al, Meta-Learning for Generalized Zero-Shot Learning, AAAI 2020

Meta-Learning for Generalized Zero-Shot Learning[5]

Idea:

- Learn a GAN conditioned on class attributes and train using meta-learning framework (MAML) to facilitate generalization to novel classes
- Mimic ZSL setup during training $\implies$ for each task/episode $T_i = T_{tr}, T_{val}$, classes of T_{tr} and T_{val} are disjoint

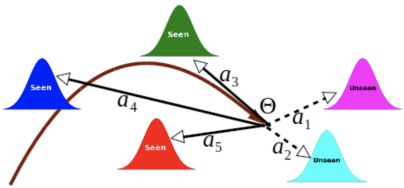


Method:

- Consider a GAN coupled with classifier module (to classify examples generated by generator) and three meta-learners (Generator G, Discriminator D, and Classifier C) as shown
- Let θ_d, θ_g and θ_c be parameters of D, G and C respectively, and $\theta_{gc} = [\theta_g, \theta_c]$

[5] Verma et al, Meta-Learning for Generalized Zero-Shot Learning, AAAI 2020

Objective

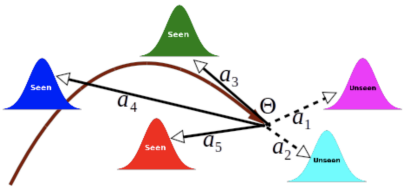


Objective

$$\max_{\theta_d} \mathbb{E}[D(\mathbf{x}, \mathbf{a_c}|\theta_d)] - \mathbb{E}_{\mathbf{a_c}, \hat{\mathbf{x}} \sim P_{\theta_g}}[D(\hat{\mathbf{x}}, \mathbf{a_c}|\theta_d)]$$

$$\min_{\theta_{gc}} -\mathbb{E}_{\mathbf{a_c}, \mathbf{z} \sim \mathcal{N}(0,I)}[D(G(\mathbf{a_c}, \mathbf{z}|\theta_g), \mathbf{a_c}|\theta_d)] + C(y|\hat{\mathbf{x}}, \theta_c)$$

Objective



Objective

$$\max_{\theta_d} \mathbb{E}[D(\mathbf{x}, \mathbf{a_c}|\theta_d)] - \mathbb{E}_{\mathbf{a_c}, \hat{\mathbf{x}} \sim P_{\theta_g}}[D(\hat{\mathbf{x}}, \mathbf{a_c}|\theta_d)]$$

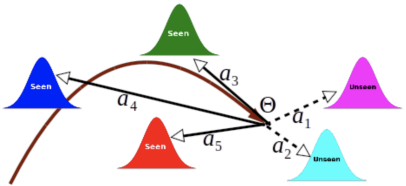
$$\min_{\theta_{gc}} -\mathbb{E}_{\mathbf{a_c}, \mathbf{z} \sim \mathcal{N}(0,I)}[D(G(\mathbf{a_c}, \mathbf{z}|\theta_g), \mathbf{a_c}|\theta_d)] + C(y|\hat{\mathbf{x}}, \theta_c)$$

Updates

$$\theta'_d = \theta_d + \eta_1 \nabla_{\theta_d} l_{\in}^{D}(\theta_d)$$

$$\theta'_{gc} = \theta_{gc} - \eta_2 \nabla_{\theta_{gc}} l_{\in}^{GC}(\theta_{gc})$$

Objective



Meta-Update

$$\max_{\theta_d} \sum_{\in \sim p(\mathcal{T})} l^D(\theta'_d) = \max_{\theta_d} \sum_{\in \sim p(\mathcal{T})} l^D(\theta_d + \eta_1 \nabla_\theta l^D(\theta_d))$$

$$\theta_d \leftarrow \theta_d + \beta_1 \nabla_{\theta_d} \sum_{\in \sim p(\mathcal{T})} l^D(\theta'_d)$$

$$\min_{\theta_{gc}} \sum_{\in \sim p(\mathcal{T})} l^{GC}(\theta'_{gc})$$

$$\theta_{gc} \leftarrow \theta_{gc} - \beta_2 \nabla_{\theta_{gc}} \sum_{\in \sim p(\mathcal{T})} l^{GC}(\theta'_{gc})$$

Generation and ZSL Classification

Generation

- After training, we can generate unseen class examples given class-attribute vectors as:

$$\hat{\mathbf{x}} = G_{\theta_g}(\mathbf{z}, \mathbf{a}) : \quad \mathbf{a} \in \mathbb{R}^d$$

where $\mathbf{a}$ denotes class-attributes of unseen classes and $\mathbf{z} \sim \mathcal{N}(0, I)$ and $\mathbf{z} \in \mathbb{R}^k$

Generation and ZSL Classification

Generation

- After training, we can generate unseen class examples given class-attribute vectors as:

$$\hat{\mathbf{x}} = G_{\theta_g}(\mathbf{z}, \mathbf{a}) : \quad \mathbf{a} \in \mathbb{R}^d$$

where $\mathbf{a}$ denotes class-attributes of unseen classes and $\mathbf{z} \sim \mathcal{N}(0, I)$ and $\mathbf{z} \in \mathbb{R}^k$

Classification

- Once unseen class samples are generated $\implies$ we train any classifier (e.g., SVM or softmax classifier) with these samples as labeled training data

Homework

Readings

- Lilian Weng, Meta-Learning: Learning to Learn Fast
- (YouTube video) Few-shot Learning with Meta-Learning: Progress Made and Challenges Ahead
- Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning, Tutorial at ICML 2019
- Zero-shot Learning: An Introduction