Reducing the need for large labeled dataset in the learning to learn framework

Arman Afrasiyabi
Computer Vision and Systems Laboratoy (CVSL)
Electrical Engineering and Computer Engineering Department

Contents

Chapter 1

Introduction

1.1 Motivation

$\mathcal{Z}^n$

Recent machine learning approaches like deep learning are continuing to perform outstanding performances in different areas of research such as computer vision and natural language processing. These algorithms typically require many examples (in the order of hundreds or thousands) to reach an acceptable accuracy. This requirement is not aligned with the learning process in the human level because the human can learn new concepts in more productive ways using only one or few examples (Lake, Salakhutdinov, and Tenenbaum, 2015). Here, the important question is: *how can we decrease the number of required examples while reaching an acceptable learning performance?* To answer this question, researchers opened a new direction in machine learning to propose the algorithms that are trainable using only few labeled instances. This is very important because the labeling cost and burden of the current machine learning algorithms are very high and make them hard for real-world application.

More recently, *learning to learn* or *meta-learning* is one of the dominant framework for most of the proposed algorithms for few labeled data. We positioned ourselves in the meta-learning trend by aims at making better learning with small labeled datasets. Proposing a model with the ability to learn from only one or few examples like human seems to be a promising goal for the ongoing research in the meta-learning domain. Different terminologies like low-shot (Wang et al., 2018), few-shot (Snell, Swersky, and Zemel, 2017) or one-shot (Santoro et al., 2016) are used in literature for learning in like human. To keep consistent terminology, k-shot learning is used in this proposal, where k refers to the number of few labeled examples.

1.2 Meta-learning for k-shot

The initial definition of the meta-learning term backs to the late 1980s and 1990s which was used in different senses at the beginning. Schmidhuber, 1987 and Schmidhuber, Zhao, and Wiering, 1997 used the meta-learning term in incremental self-training framework. Generally, meta-learning refers to two level of learning (*"learning-to-learn"*) for rapid learning from little of samples (Thrun, 1998). In recent years, meta-learning is becoming a main framework to learn from a few annotated data.

However, it is important to note that k-shot learning happens in the second level (or meta level) of the learning to *learn*. Let illustrate this by omniglot dataset proposed by (Lake, Salakhutdinov, and Tenenbaum, 2015). This dataset contains 1623 different types of handwritten characters from different languages, and there are only 20 examples per category (type of character). Suppose that we have access to

1028 characters as annotated categories, and we don't know the label of remaining 595 classes. In this example, the goal is to build a model using 1028 base character/classes such that the model can recognize characters from the remaining classes with only few examples. Please note that this is different from supervised learning when we separate the dataset based on examples, not classes. To build such a model Santoro et al. (2016) defined *episodic* training for the first level (or base) of meta-learning. In each episode, we randomly select n number of classes out of all M number of them at the base level. Then, we choose k random examples for each of the n classes. This setting is called *n-way k*-shot setting which is typically followed by the other researchers in k-shot learning literature.

1.3 Related Subjects

Transfer Learning:
Transfer learning, as a close related topic for k-shot learning, aims at using the knowledge of labeled data in the source domain to help the learning in the target domain. Unlike k shot learning, transfer learning does not have the assumption of few labeled data. Most commonly, researchers use two approaches for transfer learning: develop a model approach and pre-trained model approach (Chen and Liu, 2016). In the model approach, we first select a source domain which contains lots of data and has a relationship with the target domain. Next, the chosen source domain is used to build a model. Finally, the whole or some parts of the fitted model are used as the starting point of the target domain. This part might need input-output adaptation for the target domain. In the pre-trained model approach, we select a model from existing pre-trained models. Then this pre-trained model is used as the starting point for the target domain. Like the build model approach, some adaptation or redefine can be made to fit for the target domain fitting purpose.

Domain Adaptation:
Domain adaptation is also one of the close research topic to k-shot learning. In both transfer learning and k shot learning, the input distribution $p(X)$ is supposed to be same while the label distribution $p(Y|X)$ is different between source and target domains. On the other hand, p(X) changes between source and target domains while $p(Y|X)$ remains same. In other words, we have a domain shift between the source and target distributions, while the set of categories remains same in both of the domains. Some of the reason for the domain shift in DA are illumination, pose, and image quality (Wang and Deng, 2018). However, the term domain adaptation and transfer learning can interchangeable used in different studies. For example some NLP researchers refers domain adaptation as transfer learning (Chen and Liu, 2016).

Zero-data Learning:
Unlike few-shot learning which we have at least a few labeled examples per classes, some of the classes do not have examples in zero-shot learning. Larochelle, Erhan, and Bengio (2008) studied *zero-data* learning to predict novel classes of digits which training set did not contain them. The goal was to learn a classifier $f : X \rightarrow Y$, which can predict new values of Y omitted in the training set (Palatucci et al., 2009). We will discuss zero-shot learning in more detail in Chapter 2 in section 2.4.4.

1.4 Proposal Outlines

The proposal is broken down into sub-objectives results in the following:

From k-shot to k-plus shot learning
Deep learning algorithms are shown to have problems in the case of k-shot learning, because these algorithms require a lot of data for training. The necessity for large labeled data contradicts with the human learning process. The question is: *how can we make a better imitation of the human learning process*? One obvious answer might be the brain's imagination which is missed from current neural network algorithms. For example, we can learn a new animal species using few visual shots of them. We believe, one of the reasons could be imagining the variations of seen images in different patterns. Therefore, we interested to propose a new data generator model for the problem of k-shot learning with the ability to generate different patterns of the seen examples. This idea of data generation is investigated in some studies (like Wang et al. (2018)). We refer these methods as k-plus shot learning where additional generated instances are added to the existing input images to extend the number of examples in k shot learning. In these studies, the authors proposed data augmentation using the generative adversarial network (GANs) to help deep learning algorithms, and they do not consider the variation and realistic data generation.

However, GANs have mode collapse problem, which the generator has a bias toward producing some classes more than the others. Therefore, if we use GAN in one-shot learning and do data augmentation with GAN, then our learning algorithm will have bad performance without solving mode collapse problem. This limitation motivated us to work on realistic data generation to be sure that the generated points are useful. We proposed conditional variational autoencoder GANs (cVAE-GANs) to produce a more variations and realistic examples for data augmentation in meta-learning setup. We introduced a technique for the condition part in cVAE-GAN, which is our first contribution.

Interactive fast adaptation
As the second direction of research, we aim to extend the k-shot learning algorithms to cases where an annotator is available for labeling, but data labeling is very hard and expensive (e.g., bioinformatics). In this situation, the main question is: *"how to decrease the cost of labeling by reducing the number of needed examples?"*. To answer this question, we motivated to work on the idea of active-learning in a meta-learning setting which is called learning active *learning* (LAL).

In this direction, we are planning to propose a LAL based on memory augmented networks. Previously, memory contained networks like LSTM has been proposed for LAL, but we took one step further and proposed memory augmented networks (like neural Turing machines). Studies like Santoro et al. (2016) showed that memory augmented networks are quick in capturing the informative information from the input data. Additionally, an active learner could be fast on detecting the most valuable examples for labeling. Therefore, we are interested to propose a new memory augmented networks which benefit from the best of two worlds.

Reducing annotation cost of unseen categories
For the third direction of research, we are planning to extend k-shot learning approach to a real-world problem. Our goal is to reduce the labeling cost in real problems such as object detection or visual place recognition. We will decide to work either on visual place recognition or object detection problems. However, we will

address the problem of large dataset by proposing a *k*-shot learner to one of these problems.

Chapter 2

Extending k-shot to k-plus shot

2.1 Introduction

Comparing to the most of the current machine learning algorithms, human recognition is fast. Machine learning algorithms (e.g. deep learning) need many examples to recognize a category. This motivated machine learning researchers to investigated and propose some algorithms which are able to learn from few example. The ability to learn from only single or few examples is referred to k-shot learning which aims to be fast in the learning process. Here, k is the number of examples; for example, if we have only one example ($k = 1$), then we would called this as one-shot learning problem. Several studies attempted to propose different forms of criterion for learning common features between seen and new categories. Contrastive loss (Hadsell, Chopra, and LeCun, 2006) and triple loss (Fink, 2005) were designed to push the examples of same class to the close region in the learning space while pulling them apart from other classes. However, Thrun (1996) and Thrun (1998) proposed meta-learning ("learning to learn"), which is the main framework for k-shot learning and capturing interest more recently. Additionally, adding informative information has shown to increase the generalization on new categories. For example, while one class of approaches benefit from using unlabeled data in semi-supervised setup (Triantafillou et al., 2018), the other approaches take advantage of generative models to learn transformable priors between classes (Hariharan and Girshick, 2017; Wang et al., 2018).

2.1.1 Motivation

We as human can learn new visual concepts using single or few shots, possibly because we can visualize and create various versions of the concepts in our mind. Even when we are reading a text, we can make the image of the words in our mind. Most of the classic machine learning techniques do not have these abilities.

Recently, data augmentation using generative models is one of interested research directions tackling the problem of k-shot learning. The goal of this research path is to increase the size of support set to more than k examples per class a.k.a. k-plus learning. Specifically, two interesting studies has been done by (Hariharan and Girshick, 2017; Wang et al., 2018) in which the author designed generative adversarial networks (GANs) to generate new examples. However, in the best of our knowledge, no studies has considered the realistic and variational data generation in meta-learning framework.

In the following sections, we will first cover the fundamentals of generative models in section 2.2. Next, we will see baselines in k shot learning and an attempt for data augmentation for k shot learning in section 2.3. Next, we will cover a proposed method and preliminary results in section 2.4. Then, we will cover our proposed

ideas for realistic and variational data augmentation based on cVAE-GAN. Finally, we will conclude this chapter by highlighting our main contribution.

2.2 Generative Models

Generative methods are a class of algorithms with the aim of learning the underlying data distribution. Nowadays, there is a tremendous interest in using generative methods as data augmentation. In this research path, researchers first synthesizing new examples using generative models, and add these new examples to the existing k examples. We call this extended dataset as *k-plus* shot learning. The generated data by these methods could help us to move from k shot to *k-plus* shot learning, which is very important in the case of learning to learn from both data augmentation and regularization prospective. In the following subsections, we will cover two important class of generative networks which can be extended to learning to *learn* problem.

2.2.1 Variational Autoencoder (VAE)

Variational autoencoder (VAE) (Kingma and Welling, 2013) is one unsupervised generative models to learn the underlying distribution $p(x)$ of the input data x. The learning of $p(x)$ is done by mapping from the data set's unknown distribution to one of the tractable like a Guassian distribution. VAE is based on variational inference. Suppose that we have a graphical model with two nodes shown in subfigure **A** in 2.1, which refers to computing the posterior probability of a hidden unit z given an observation x: $p(z|x) = p(x|z)p(z)/p(x) = p(x,z)/p(x)$. However, computing the marginal distribution $p(x)$ in n dimension would be $p(x) = \int_1 \int_2 ... \int_n p(x|z)p(z)dz$, and it is intractable in many cases. In literature, two main approaches proposed for estimating these integrals: Monte-Carlo method and variational inference. In the case of varitional inference, $p(z|x)$ can be estimated by $q(z)$, where $q(.)$ is a tractable like a Gaussian distribution. Kingma and Welling (2013) tried to design a neural networks that can learn the parameters of $q(.)$ in order to make it as closer as $p(z|x)$ possible. In this study, they proposed VAE which minimize a criterion called KL divergence, a similarity measure between two distributions: $KL(p||q) \approx H(q) - H(p)$. Here, $H(p) = -\sum p(x)logp(x)$ is the entropy, and it measures the expectation of the information. More precisely, $KL(p||q)$ is called the KL divergence of p and q w.r.t p. Therefore, $H(q)$ is replaced with $\sum p(x)logq(x)$. In other words,

$$\begin{aligned} KL(p||q) &= -\sum p(x)logq(x) + \sum p(x)logp(x) \\ &= -\sum p(x)\log\frac{q(x)}{p(x)} \end{aligned} \tag{2.1}$$

Therefore, min $KL(q(z)||p(z|x))$ means minimizing the KL divergence between $q(z)$ and $p(z|x)$.

$$\begin{aligned} KL\big(q(z)||p(z|x)\big) &= -\sum q(z)\log\frac{\frac{p(x,z)}{p(x)}}{q(z)} \\ &= -\sum q(z)\log\frac{p(x,z)}{q(z)}\frac{1}{p(x)} \\ &= -\sum q(z)\Big[\log\frac{p(x,z)}{q(z)} - \log p(x)\Big] \\ &= -\sum q(z)\log\frac{p(x,z)}{q(z)} + \sum_z q(z)\log\, p(x) \\ &= -\sum q(z)\log\frac{p(x,z)}{q(z)} + \log p(x)\sum_z q(z) \\ &= -\sum q(z)\log\frac{p(x,z)}{q(z)} + \log p(x) \\ \log p(x) &= KL\big(q(z)||p(z|x)\big) + \sum q(z)\log\frac{p(x,z)}{q(z)} \end{aligned} \tag{2.2}$$

where, the second term $\sum q(z)\log\frac{p(x,z)}{q(z)}$ is called *variational lower bound*. It is called lower bound because KL divergence is always greater or equal to zero. Therefore, we can say that log $p(z)$ always is less or equal to $\sum q(z)\log\frac{p(x,z)}{q(z)}$. In VAE, we want to maximize this lower bound while minimizing the KL-divergence. The lower bound can also be written in the following form:

$$\begin{aligned} \sum q(z)\log\frac{p(x,z)}{q(z)} &= \sum q(z)\log\frac{p(x|z)p(x)}{q(z)} \\ &= \sum q(z)\Big[\log p(x|z) + \log\frac{p(z)}{q(z)}\Big] \\ &= \sum q(z)\log p(x|z) + \sum q(z)\log\,\frac{p(z)}{q(z)} \\ &= E_q\log p(x|z) - KL(q(z)||p(z)) \end{aligned} \tag{2.3}$$

where, $E_q\log p(x|z)$ is the expectation of $\log p(x|zq)$ w.r.t q. This part ($E_q\log$) is conceptually **reconstruction error**. By assuming $KL(q(z)||p(z)) = KL(q(z|x)||p(z))$, we can formulate the lower bound as an autoencoder neural network. The schematic overview of a VAE is shown in section **B** of figure 2.1. Like autoencoders, the network contains encoding and decoding parts. In encoding phase, the network is learned to model $q(z|x)$, and it maps the z to the input space by estimating $p(z|x)$. In other words, we want to match the distribution of input dataset in the embedding space (z) with a tractable distribution.

2.2.2 Generative Adversarial Networks (GANs)

In adversarial learning, we have two networks: a classifier called discriminator and a generator. The generator aims to output an adversarial example given a random sample. Then, the discriminator tries to distinguish between adversarial examples (produced by generator) and real examples of the dataset. Suppose that we have samples from the real world like MNIST as p, and we have some other fake class

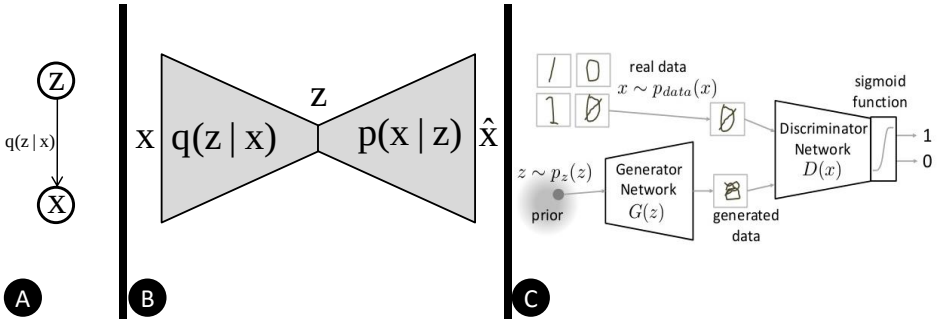


FIGURE 2.1: Generative models. A) variational inference, B) variational autoencoder, and C) generative adverserial network.

q from a distribution (Gaussian). Having p and q, we want to play an adversarial game. The goal of the generator (G) is to fool the classifier/discriminator (D), and the purpose of the discriminator is to classify correctly fake and real images. Goodfellow et al., 2014 proposed GAN and used following "minimax" function for training:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}\big[\log D(x)\big] + \mathbb{E}_{z\sim p(z)}\big[\log(1 - D(G(z)))\big] \tag{2.4}$$

where, $V(.)$ is the objective function over the generator (G) and discriminator (D) functions. Figure 2.1 (C) shows an illustration of a GAN architecture.

2.2.3 Conditional Generative Models

In the previous section, we reviewed VEA and GAN, which are unsupervised generative models. However, conditional generative models are used to specify a class which we wanted to generate examples from that class. To convert unsupervised generative models to supervised, researcher condition the generator with the class label information. For example, in the case of VAE, we can condition the generated example by concatenating the one-hot vector of the class label vector with the z vector at the middle of the VAE.

2.3 Distance based approaches for k-shot learning

In this section, we will cover two famous studies for k-shot learning which are inspired by a metric learning and neural networks algorithms. First, we are going to cover Matching Nets (MN), inspired by recent advances in attention and memory that enable rapid learning (Vinyals et al., 2016). Second, we will cover Prototypical Networks (PN) (Snell, Swersky, and Zemel, 2017) to learn a metric space such that the classification can be performed by computing distance to the prototype representations in the embedding space.

2.3.1 Matching Network

Matching Net (MN) is a well known k-shot learning technique based on memory contained networks LSTM and shown to have fast adaptation from few examples (Vinyals et al., 2016). Generally, we want to estimate the posterior probability $P(y_j|S,\hat{x})$

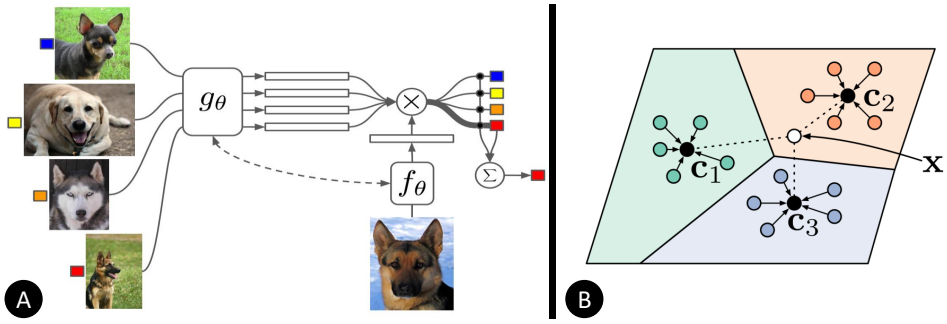


FIGURE 2.2: Distance based approaches. A) Matching Nets (Vinyals et al., 2016). B) Prototypical network (Snell, Swersky, and Zemel, 2017).

in k shot learning where $\hat{x}$ is the test example, and $S = \{(x_i, y_i)\}_{i=1}^{N}$ is the support set. The idea behind MN is to model this posterior probability as an attention mechanism. The author proposed matching network for one-shot learning problem in N-way case. Here, N is the number of classes in both base and meta levels. More specifically, the author used a set of examples by considering them in a sequence based on set-to-set matching (Vinyals, Bengio, and Kudlur, 2015). Then, the probability of $\hat{x}$ belonging to the jth class $p(\hat{y}_j|\hat{x}, S)$ is computed by:

$$\hat{y}_j = \sum_{i=1}^{N} a(\hat{x}, x_i) y_i \tag{2.5}$$

where, $a(.,.)$ is an attention mechanism in the form of softmax over the cosine distance d between the embedding of $\hat{x}$ and all other examples in S:

$$a(\hat{x}, x_i) = \frac{\exp^{d(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} \exp^{d(f(\hat{x}), g(x_j))}} \tag{2.6}$$

with f and g being appropriate neural network to embed $\hat{x}$ and x_i. In this study, the author used a bidirectional Long-Short Term Memory (LSTM) to encode $\hat{x}$ by considering the support set S in sequence form:

$$f(\hat{x}, S) = \text{attLSTM}(f'(\hat{x}), g(S), N) \tag{2.7}$$

where, attLSTM(.) is attention based LSTM, f' is a neural network function which embeds the input query $\hat{x}$. N is the fixed number of unrolling steps in LSTM.

2.3.2 Prototypical Network

As discussed above, matching network was originally designed for 1-shot learning where we have N examples of image-label pairs $S = \{(x_i, y_i)\}_{i=1}^{N}$. $x_i \in \mathbb{R}^D$ is D dimensional feature vector, and $y_i \in \{1, ..., k\}$ is the corresponding labels.

Snell, Swersky, and Zemel, 2017 proposed prototypical network, for k shot learning problem. Therefore, they used k number examples per N classes. Prototypical network maps all k examples in each of the N classes from the input space to single

prototype μ_c in the output space:

$$\mu_c = \frac{1}{k} \sum_{(x_i, y_i) \in S} f_\theta(x_i), \tag{2.8}$$

where, k is the number of support examples (shots) that are used in each iteration of the training, and $f_\theta(.)$ is the mapping function from the input to the embedding space (see Figure 2.2 section **B**). Like matching network (Vinyals et al., 2016), $f_\theta(.)$ can be a convolution neural networks. Here, $\mu*_c$ represents the mean of the all examples belonging to the class c in the embedding space for assigning class probabilities. Specifically, given μ_c, the probability of a new input example (query) x_q to be assigned to the class c is computed as by following softmax layer:

$$p_\theta(y_c|x_q) = \frac{exp\big(-d(f_\theta(x_q), \mu_c)\big)}{\sum_{c'} exp\big(-d(f_\theta(x_q), \mu'_c)\big)} \tag{2.9}$$

where, d is distance metric. The following negative log-probability $p_\theta(y_c|x_q)$ is minimized for training purpose:

$$\begin{aligned} \mathcal{L}_a(\theta) &= -log\big(p_\theta(y_c|x_q)\big) \\ &= \frac{1}{N}\left[\big(d(f_\theta(x_q), \mu_c)\big) + log \sum_{c'} \big(exp\big(-d\big(f_\theta(x_q), \mu_{c'}\big)\big)\right]. \end{aligned} \tag{2.10}$$

where, N is the number of all examples in one sequence or episode. For example, if we have 5-shot problem in 5 ways, the N would be 25. During the classification, the distance between the new example and all prototypes specifies the class of the example. In this phase, the idea is very similar to non-parametric methods like nearest neighbor method, and the label of unseen examples would be the label of the closest prototype in the output space.

2.3.3 Imaginary data as data-augmentation

The main advantage of matching network over the prototypical network is the ability of capturing the contextual embedding. However, the use of attention LSTM might not be appropriate to attend rare classes in meta-learning. Therefore, Wang et al., 2018 combined the prototypical network with the matching network. In this study, the author proposed a meta-learner along with a designed data generator network, called "hallucinator", to produce and augment new examples to the existing few examples. In other words, the hallucinator maps real examples to imaginary or hallucinated examples. The model is optimized as *end-to-end* model and resulted up to 6 point classification accuracy increases in one-shot learning problem. The model tries to sample additional hallucinated examples, using the existing training set X_{train} for a specific class. Inspired by Goodfellow et al., 2014 and Kingma and Welling, 2013, the hallucinator network takes a seed example $x \in X_{train}$ and a noise vector z as input and outputs a hallucinated examples. More specifically, the hallucinator receives n number of examples set X_{train}, and produces n_s number of additional examples X^{sample}_{train}. Then, the generated samples are combined with the existing training example, $X^{aug}_{train} = X_{train} \cup X^{sample}_{train}$, to increase the size of training set for meta-learning purpose. For k-shot learning purpose, a combination of the

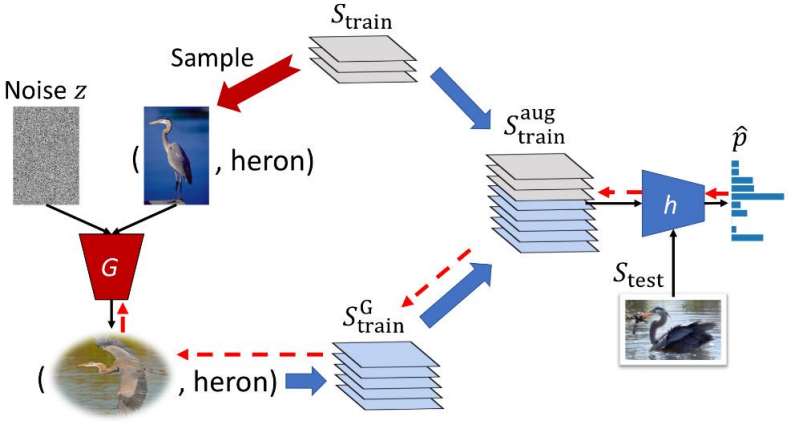


FIGURE 2.3: Low-shot learning from imaginary data.

prototypical network is added at the top of matching network and called **prototypical matching network (PMN)**. The purpose of adding prototypical network to the matching network is to overcome the problem of matching network's bias against rare classes (they are swamped by examples of common classes). The dashed arrows in Figure 2.3 shows the back propagation path from the output of the model toward the input data and generated examples.

2.4 Preliminary work

As discussed in the previous section, (Wang et al., 2018) used generative adversarial networks (GAN) for data augmentation and transferring k-shot learning to k-plus shot learning. The authors only used multilayer perceptron architecture as generative models. They also did not investigate the effect of diversity in the generated samples. In other words, the model generates the augmented samples without considering the realistic and variations of the generated samples. Additionally, one of the main limitations of GANs are *mode collapse*, where the generator collapses which produces limited varieties of samples. This part of the Ph.D. project will be dedicated at reducing the mode collapse problem and producing more diverse and realistic examples (not "hallucinated" examples like in Wang et al., 2018). Additionally, we interested to add diversity in the generated examples which can effectively reduce the over-fitting problem in deep neural networks. Our inspiration was based on recent advances in *image-to-image* translation. We worked on conditional VAE-GAN which has two parts of variational autoencoders and generative adversarial networks like Zhu et al., 2017.

2.4.1 Difficulties of generative models in meta-learning

Data augmentation by generative models is a non-trivial task in the meta-learning setting. As we discussed in the previous chapter, the base (or first) level classes are different with the classes in meta (second) level of learning to *learn*; therefore, generating an example in meta-level using only the trained model in base level is a hard problem. For example, suppose the base level classes are *cat*, *dot*, and *bird*, and let the meta level classes be *car*, *tree*, and *fish*. In this case, after training our generative model with base level classes and examples, we can not expect the trained generative model to produce a discriminate example in meta level. Before going to

our proposed solution, we should note that the discussed generative models (GANs and VAE) are unsupervised methods. Therefore, we must turn them to supervised, which is done by augmenting condition vector to the model. Most of the researchers use the class information as the conditional vector. However, this is not possible in the meta-learning case because there is no intersection between the classes in meta level with the base level. As a solution to this problem, we proposed conditioning on some information belonging to each image itself rather than class information. Inspired by recent advances in the image-to-image domain like Zhu et al., 2017, we conditioned each image to the combination of Sobel filter and some other information to produce informative samples for transforming k shot learning to k-puls shot learning. In the next section, we will talk about conditional variational autoencoder GAN (cVAE-GAN) which is added to a few-shot learning network for data augmentation.

2.4.2 Conditional Variational Autoencoder GAN (cVAE-GAN)

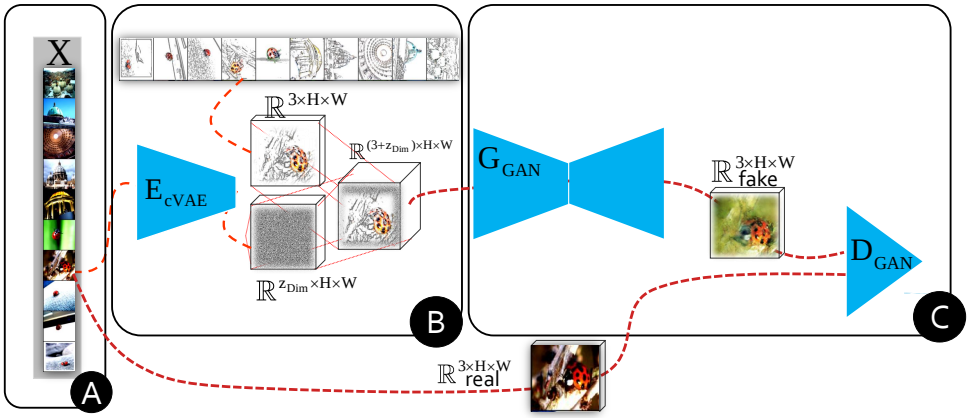


FIGURE 2.4: conditional Variational Autoencoder GAN. A) input images, B) cVAE, and C) GAN.

Our proposed model for data generation (in meta learning case) has two conditional end-to-end network: I) conditional variation autoencoder (cVAE), and II) generative adversarial network(GAN).

Part I) Conditional variational autoencoder (cVAE)

The first part of the overall model is cVAE which is shown in part B in figure 2.4. Here, an encoder $E(.)$ first map the input image(x_i) to the latent space $E(x_i)$. Meanwhile, we propose a new function for condition image of that input image. For this purpose, we concatenate $E(x_i)$ tot a linear combination of sable filter ($sobel(x_i)$) and the output of a gradient visualization method called guided backpropagation (GBP). Therefore, the final conditioned part of the generative modle would be:

$$x_i^c = [E(x_i)|(sobel(x_i) + GBP(x_i))] \tag{2.11}$$

where, $[. \mid .]$ is the concatenation operation, and $sobel(.)$ is the soble function, GPB is a gradient visualization called guided back-propagation (GBP). The aim of $sobel(x_i)$ is to force the generative model (GAN) to produce a variation of the input image. However, we found that this idea is naive in some cases. To illustrate this, suppose that the input would be an image of a ladybug. We expect that our network to do not generate a black ladybug (without no red pattern) which is almost impossible in real world. This could be a problem because a the color pattern of ladybug specifies

its difference with other black circle shaped insects. The other example could be and orange fruit which color information is necessary detection. As a solution to this problem, we used the projection of the gradient of the pretrained network on the input image. More specifically, we used guided backpropagation (GBP) (Springenberg et al., 2014) using a pre-trainied VGG16 like network. Please note we obtain the pre-trained model from scratch in the same meta-learning setup which is used for k-shot learning.

Please note that the latent vector of VAE's encoder, $E(x_i)$, is expanded before concatenation with x_i^c (part A in figure 2.4). For VAE, we used KL divergence between $f_E(x_i)$ and and a random Gaussian distribution in the following form:

$$\mathcal{L}_{KL}(E) = \mathbb{E}_{x_i \sim p(x_i)}\big[KL\big(E(x_i)||\mathcal{N}(0,1)\big)\big] \tag{2.12}$$

Part II) Generative adversarial network (GAN)
In the GAN part, the encoded input image x_i and the condition image x_i^c (at the end of part I) are given to a generative G to synthesize the input image x_i in different patterns. In this case, the loss of cVAE-GAN is defined as follow;

$$\mathcal{L}_{GAN}^{VAE} = \mathbb{E}_{x_i, x_i^c \sim p(x_i, x_i^c)}\big[log(D(x_i, x_i^c))\big] + \mathbb{E}_{x_i, x_i^c \sim p(x_i, x_i^c), z \sim E(x_i)}\big[log(1 - D(x_i^c, G(x_i^c, z)))\big] \tag{2.13}$$

where, x_i is the input image to VAE, and x_i^c is the conditioned image and input of the generator G at GAN. D is the discriminator of GAN, and $z = E(x_i)$ is the latent representation of input x_i the encoder of VAE. Putting both of the losses in 2.13 and 2.12 the overall objective function will be:

$$G^*, E^* = \arg\min_{\text{E, G}} \max_{\text{D}} \mathcal{L}_{GAN}^{VAE}(G, D, E) + \lambda \mathcal{L}_{KL}(E) \tag{2.14}$$

where, λ is a trade-off coefficient.

2.4.3 Extending k-shot to k-plus shot learning

In the previous section, we discussed our proposed cVAE-GAN for synthesizing variants of the input images. Our main contribution was the use of cVAE-GAN as data generator for meta-learning by conditioning an some information of the example itself. Currently, we are working on two different setups for combining the proposed generative model (discussed in the previous section) with a k-shot learner model in learning to learn framework. In the following sections, we will try to overview both of our current studies.

We followed the same setting with (Snell, Swersky, and Zemel, 2017), (Finn, Abbeel, and Levine, 2017) and (Vinyals et al., 2016) for building the learning to *learn* framework. At each training episode, N number of classes are randomly selected out of all M classes. Then, we build support set $\mathcal{S} = \{xs_1, ..., xs_{kN}\}$ by sampling k examples randomly from each of N classes. Additionally, we sample query set $Q = \{xq_1, ..., xq_k\}$. Therefore, episode at i^{th} iteration consists of support set S_i and Q_i. In research terminology this is called *N-way k-shot*.

I) Consecutive cVAE-GAN for k-plus shot learning
As the first research direction, we consecutively add a k shot learning network to the end of cVAE-GAN. Figure 2.5 illustrates the schematic overview of our proposed generative model to extend k shot learning to k-plus shot learning. In the abstract

view of the proposed model, we have two section: a generative model and one k shot model. First, the episode is given to the cVAE-GAN generative model which produces variations of S the input. The result of this section is generated variant versions of the input examples S^G. As discussed in previous section cVAE-GAN contains an encoder $E(.)$ for VAE, a generator $G(.)$, and a discriminator $D(.)$. We used the following architectures for each of the functions:

- $E(.)$ contains 3 residual blocks. Each block contains 2 convolution layer, 2 batch normalization, 2 LeakyReLU, and a max-average pooling layer. Additionally, $E(.)$ contains 2 fully connected (FC) layers to map the output of the last residual layer to a vector of size 256. These FC layers learns the parameters of the Gaussian distributions in VAE.
- $G(.)$ is a Unet Ronneberger, Fischer, and Brox, 2015 like architecture contains 7 convolution as the encoder and 7 transpose convolution as the decoder. Each of the blocks.
- $D(.)$ contains five convolution layers blocks

Finally, we combined the original support set S and the generated ones S^G to build augmented dataset: $S^{aug} = S^G \cup S$. We used the prototypical network of (Snell, Swersky, and Zemel, 2017) as k-shot learning model which consists of 4 layer of convolution block with max-pooling, batch-normalization, and ReLU activation function at each block.

Table 2.1 shows our experimental results on *Mini*ImageNet dataset. This dataset composed of 100 classes, and 600 examples of size $3 \times 84 \times 84$. Based on categories, we dived the dataset into 64 training, 16 validation, and 20 test classes like (Snell, Swersky, and Zemel, 2017), (Finn, Abbeel, and Levine, 2017) and (Vinyals et al., 2016).

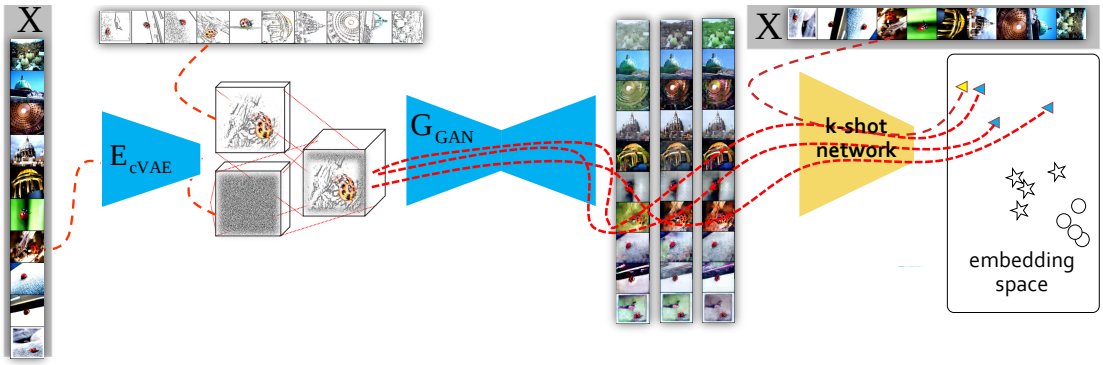


FIGURE 2.5: Consecutive cVAE-GAN model k-plus shot learning.

II) Lateral cVAE-GAN for k-plus shot learning

The first discussed consecutive method has the mismatching problem between the distribution of real examples and generated ones. More specifically, our generator cVAE-GAN synthesizes examples which are not necessarily in the same distribution of the data. Therefore, the k-shot model is supposed to solve two problems: first bringing the generated and real examples from the same class to the same region in the space, and second doing prototypical learning. We address this problem by adding the k-shot network at the middle of cVAE-GAN. We aimed to do data-augmentation within the embedding space instead of image-space.

TABLE 2.1: Average classification accuracies on *mini*ImageNet in 5 ways, 5 Shot (with 5 query) setting

Method	Accuracy
MN (Vinyals et al., 2016)	55.3%
MAML (Finn, Abbeel, and Levine, 2017)	63.1%(±0.92%)
PN (Snell, Swersky, and Zemel, 2017)	63.20%(±0.04%)
Ours(Consecutive)	62.4%(±0.03%)
Ours(Lateral)	**65.7** % (±0.04%)

Figure 2.6 illustrates the schematic view of our proposed lateral method. An example x_i first passes from the encoder $E(x_i)$ of variational autoencoder (VAE), then the latent representation of x_i is concatenated with conditional image x_i^c, where x_i^c is the concatenation $sobel(x_i) + GBP(x_i)$ and $E(x_i)$. Next, the x_i^c is given to the Unet like network (Ronneberger, Fischer, and Brox, 2015) generator (G), which has its own encoder-decoder architecture. However, the encoder of G outputs $\mathbf{z}_i^G 1$ vector. Meanwhile, x_i is given to a k-shot network which maps it to $\mathbf{z}_i$ vector. At the middle of G, both of $\mathbf{z}_i$ and $\mathbf{z}_i^{G1}$ are concatenated to built one version of x_i for the k-shot learning. For another version of x_i, we back to the VAE and randomly sample from a normal distribution to generate second version of $\mathbf{z}_i^{G2}$. Again, we concatenate $\mathbf{z}_i^{G2}$ with $\mathbf{z}_i$ to make the second version of In fact, $\mathbf{z}_i$ is fix for all of the generated latent vectors to only generate points around $\mathbf{z}_i$.

Table 2.1 shows our preliminary results of the discussed lateral architecture on *mini*ImageNet. For k-shot learning, we used the prototypical network of (Snell, Swersky, and Zemel, 2017) with the same architecture of 4 convolution blocks. Each block has a convolution layer with 64 number of 3×3 filters, and one max-pooling, batch normalization, and ReLU as non-linearity. We trained cVAE-GAN by the mentioned architecture in the previous section.

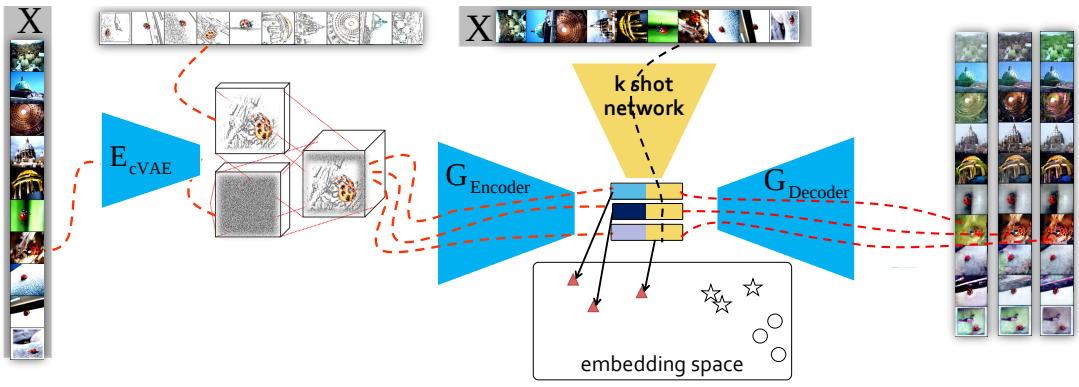


FIGURE 2.6: Lateral cVAE-GAN model for k-plus shot learning.

2.4.4 Zero-Shot learning

We are also planning to extend our proposed method to zero-shot scenario discussed in 1.3. The goal of zero-shot learning, a particular form of transfer learning, is to find the relationship in ($\mathbf{x}_i$, $\mathbf{y}_i$) pair. Here, $\mathbf{x}_i$ is an observation from $\mathbf{y}_i$ category. In fact, finding the relationship is needed for the cases where we only have full descriptions of classes, but the examples of some classes are missed in testing time. We are going

to test our proposed method on some of the zero-shot learning benchmarks like the Caltech-UCSD Birds (CUB) 200-2011 dataset (Welinder et al., 2010). This dataset contains the images of 200 different bird species with their description of attributes such as birds color.

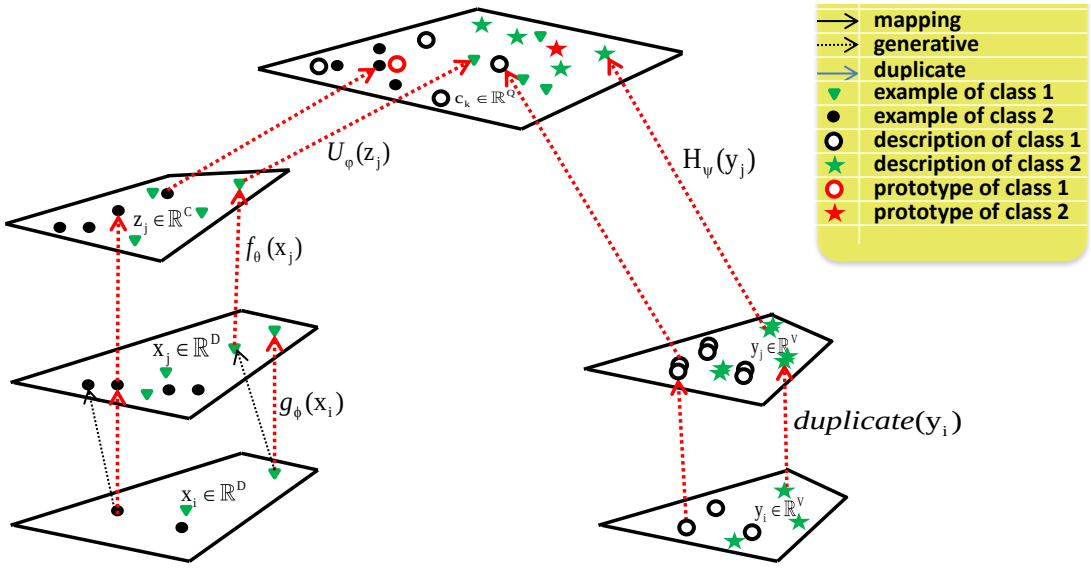


FIGURE 2.7: Zero Shot Learning based on using our proposed generative model $g_\phi^{cVAE-GAN}(x_i)$ and prototypical network at top.

We would create episodes containing random selection of N classes from all M training categories. Then, k number of samples will been randomly chosen for each class to have the support set: $S = \{(x_1, y_1), ..., (x_N k, y_N k)\}$, where $y_i \in \{1, ..., N\}$ is the description vector of x_i. Figure 2.7 shows an episode of two classes $N = 2$ with two example per class $k = 2$. After defining the episodes, we will, first, use our proposed generative model, $g_\phi^{cVAE-GAN}(x_i)$, to extend the k examples to k-plus examples.

$$S^+ = (X^{aug}, Y^{aug}) = \left(g_\phi^{cVAE-GAN}(X^{aug}), duplicate(Y^{aug})\right) \cup S \tag{2.15}$$

Please note the label and description of the generated points will be same with the labels of original images.

Then, a representation learning function, f_θ, like ResNet He et al., 2015 will be used extract the embedding features (Z^{aug}) of images in S ($\forall x_i \in X^{aug};\ x_i \in \mathbb{R}^D$). As the description of the classes (Y), we are going to use the V-dimensional continuous attribute vector provided in the dataset. Here, the use of attributes (like birds color pattern, neck size, types and etc.) would help us to have more representative label vectors instead of using only one-hot vector. Next, we will design two functions $j_{\Phi_1}(Z^{aug})$ and $j_{\Phi_2}(Y^{aug})$ (e.g., MLP) to map both Z^{aug} and Y^{aug} to the same size feature space $U = f_{\Phi_1}(Z^{aug}, f_{\Phi_2}(Y^{aug})$.

Finally, we will use prototypical network of (Snell, Swersky, and Zemel, 2017) or the proposed nearest neighbour in section **??** to map the image features (c_i) to prototype (or nearest) description's embedding (μ_i)

$$p_\theta((y = i)|c_i) \propto exp(-d(j_{\Phi 1}(c), \mu_i)) \tag{2.16}$$

where, d is the distance metrics, and $c \in \mathbb{R}^Q$ is a sample in U space, and $\mu_i \in \mathbb{R}^Q$ is the proto or nearest neighbour of i^{th} examples(s) and computed using embedded

attribute vectors. During the meta-level, we will test our model using a continuous attributes vector as class meta-level. This vector contains the characteristic information about the shape and patterns on feather.

2.5 Proposal

2.5.1 Empirical Extensions

Our preliminary work consists of testing the proposed method in a small network and one dataset. Typically, researchers are following one of two different experimental tracks. In the first track, studies contain experiments in Omniglot(1623 classes; 20 example per classes) and *mini*ImageNet (100 classes; 600 examples per class) benchmarks. In these studies, researchers use a four-layered convolutions block (convolution, Relu, and pooling) in pre-defined setup. For example, they use images size of $84 \times 84 \times 3$ as the input without any data augmentation like rotation. These studies are mostly demonstrated in machine learning conferences like NIPS and ICLR. In the second track, researchers like (Lowry et al., 2016) and (Hariharan and Girshick, 2017) completed their experiments on ImageNet dataset containing 1000 classes (Russakovsky et al., 2015). Most of the studies in this track are publishing their results in computer vision community like CVPR, and they use very deep network like deep residual neural networks for representation learning.

Because we are proposing realistic data augmentation for the first time in k shot setting which is similar to the studies in computer vision track like (Wang et al., 2018), we want to extend our method to ImageNet benchmark which is proposed in (Hariharan and Girshick, 2017). We will use a very deep network like residual neural network for data representation in both base and meta-level. Our primary challenge would be performing experiments on such a big scale because we will be faced with implementation challenges to training the model on meta-learning fashion. Perhaps, out main problem would be fitting the feature vectors of k examples of n base classes.

2.5.2 Adding Decoder

In typical k-shot distance metric space learning, most of the models embed x_i input example using an encoder function f_θ to the corresponding feature $z_i = f_\theta(x_i)$. We do this for all of the examples in an episode containing N classes and k examples per class.

In metric learning methods like matching network (Vinyals et al., 2016), we minimize the loss at the embedding ($\mathcal{L}_e$). The goal of this loss is to reduce within-class distance while maximizing the distance between support set and the query example after forwarding pass of the support set and computing $f_\theta(x_i)$. We believe that adding a **decoder** similar to auto-encoders could be beneficial in metric space learning. This is because reconstructing the input images at the output of decoder could help for the clustering of the examples in the embedding space.

In the decoding part, our model reproduces x_i by reconstruction function g_ϕ as $\tilde{x}_i = g_\phi(z_i)$. The reconstructed $\tilde{x}_i$ has the same size with the input x_i. Finally, the following mean-squared-loss is minimized for reconstruction purpose: $\mathcal{L}_d(\phi) = \sum_{i=1}^{k} ||\tilde{x}_i - x_i||_2^2$. Putting both of the encoder and decoder losses together, the overall loss $\mathcal{L}$ would computed in the following form,

$$\mathcal{L} = \mathcal{L}_e(\theta) + \lambda \mathcal{L}_d(\phi) \tag{2.17}$$

2.6 Summary of the contributions

Our contributions are summarized as follows:

1. **Extending cVAE-GAN to meta-learning framework**:
   We propose a conditional generative model (cVAE-GAN) based on some information (sobel and gradient) about the input image. More specifically, we proposed a linear combination of *sobel* operation and gradient information to generate variational versions of the input examples.

2. ***k*-plus learning with realistic and variational data augmentation**: In this chapter, we are proposing an extension of cVAE-GAN (as a generative model) to data synthesizing and augmentation for *k*-shot meta-learning. To the best of our knowledge, we are proposing an original work on generating realistic and variational examples as data augmentation using cVAE-GAN.

3. **Zero-shot learning**: We are proposing to extend our preliminary work on the conditional generative model to zero-shot learning.

4. **Adding Decoder**: We are proposing to take advantage of the discriminative characteristic of the latent space in an encoder-decoder for distance-based learning to *learn*. Our main difference with some of the related studies is coupling the encoder-decoder architecture with generative models.

Chapter 3

Interactive Fast Adaptation

3.1 Introduction

In this chapter, the idea of learning active *learning* (LAL) in the case of a few labeled data will be covered. Active learning is a sub-field of machine learning with the aim of reducing the amount of required data annotation when a data annotator or Oracle is available. Additionally, we will cover the concept of memory augmented networks with the purpose of fast adaptation in machine learning. Memory augmented neural networks (inspired by Neural Turing Machines (NTM) (Graves, Wayne, and Danihelka, 2014)) have the advantage of rapid adaptation in meta-learning compared to traditional gradient-based networks (Santoro et al., 2016). Both memory augmented networks and active learning motivated us to create a research path based on the interactive (active) memory augmented network.

3.1.1 Motivation for active learning

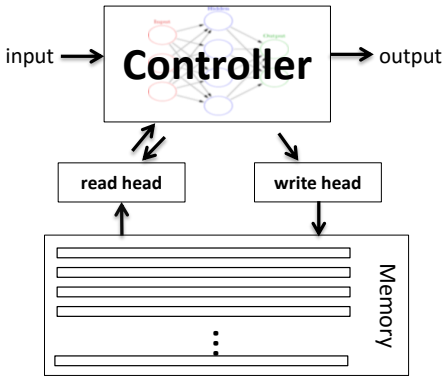
In most of the real world problems, we have a large amount of unlabeled data. In some of such cases the cost of data annotation can be very high, like cancer classification (Liu, 2004). Therefore, it is much appreciated to build a model to decide the more valued data for labeling. The idea of reducing the amount and cost of data labeling is the primary motivation in *active learning*, a specific case of semi-supervised learning. Unlike semi-supervised learning, an oracle is available in active learning to annotate the unlabeled examples selected by the learning model. Therefore, in addition to building a classifier, the primary goal of an active learner is to construct a selection strategy for choosing the most useful data to annotate, then deciding how to use the description of the selected data (after receiving from Oracle) to build a robust model. This data is selected such that the overall performance of the prediction model will be maximized. In active learning literature, several studies have proposed as data selection strategies for selecting and asking the most valuable unlabeled example(s) from the oracle. For example, most of the researchers tried to choose the most uncertain examples such that their description information could reduce the overall uncertainty about the other unlabeled data (Settles, 2014). However, based on the nature of unlabeled examples, different sampling scenarios can be considered. For instance, *online-based* setting happens where a stream of data is coming continuously (Woodward and Finn, 2017), while *pool-based* setting is suitable when we have available set of unlabeled data (Bachman, Sordoni, and Trischler, 2017).

3.1.2 Motivation for memory augmentation networks

The memory contained neural networks like Long Short-Term Memory (LSTM) are fast for capturing the necessary representation from few labeled data. (Hochreiter, Younger, and Conwell, 2001) showed that meta-learned LSTMs could rapidly extend to never-before-seen quadratic functions from a low number of samples. One of the fundamental ideas in k-shot learning is finding the contextual representations of data. For example, (Vinyals et al., 2016), and (Wang et al., 2018) and (Hariharan and Girshick, 2017) showed that LSTM can successfully find the contextual pattern in meta-learning. However, the size of LSTM's memory cannot extend to the classification of examples in many situations (Cai et al., 2018). Additionally, the LSTM's memory is not addressable and retrievable when we need information.

Memory-augmented neural networks like neural Turing machines (NTM) (Graves, Wayne, and Danihelka, 2014) tries to solve the limitations of LSTM by proposing addressable memories. As an example of these networks, NTMs are fully differentiable neural networks with a large and addressable memory for keeping extensive, accessible information in the memory compared to LSTM. For a given input x_i, the memory encoding and retrieving are done by placing the vector presentations of x_i into or taken out of memory in every time-step. Additionally, memory-augmented neural networks are known for their fast encoding and retrieve abilities which is shown by (Santoro et al., 2016) and Cai et al., 2018. This ability makes them an appropriate alternative for conventional neural networks like LSTM which require a lot of examples for their training.

FIGURE 3.1: Neural Turing Machine architecture.
(Graves, Wayne, and Danihelka, 2014)
The overall idea is to map the external input like an image to external output such as class label by using a memory.



At the heart of these memory augmented networks, there is a controller, which can be any neural network algorithm, to manage the read and write heads. Figure 3.1 illustrates the schematic overview of a memory augmented network which contains an external memory to keep most relevant information and read-write heads for retrieving and storing the information.

Inspired by NTMs and (Hochreiter, Younger, and Conwell, 2001), Santoro et al., 2016 proposed a version of NTMs called memory-augmented neural network (MANN) for fast adaptation in the case of meta-learning (for more information pleasee see section 3.3). MANN interacts with external memory using read and write heads similar to NTMs.

3.1.3 Applications

In this chapter, we will first discuss active meta-learning. Then, we will cover a proposed method based on both active meta-learning and memory-augmented networks. Next, we will see the concept of memory augmented network. Finally, we will propose to combine these two ideas which could have huge range of applications. Let see some applications of such proposed idea.

- **Bioinformatic data classification**
  Annotation of medical data is very expensive due to the specialist requirement in bioinformatic data classification, and active learning are shown to reduce the labeling cost in these situations (Liu, 2004) and (Hoi et al., 2006).

- **Malware Detection**
  Active learning is shown to increase the performance of a classifier which has the goal of detecting unknown worms in the case of high noise datasets (Moskovitch et al., 2008).

- **Autonomous driving**
  Active learning can also reduce the supervision burden too in autonomous driving (Laskey et al., 2016) and (Zhang and Cho, 2016).

.

3.2 Active meta-learning

As discussed in previous section, active learning requires a selection strategy for finding the most important unlabeled data for asking from an oracle. In classic active learning methods, researchers usually proposed selection strategies based on theoretical measures or heuristics. Recently, with the advances in meta-learning, some studies such as (Bachman, Sordoni, and Trischler, 2017), (Woodward and Finn, 2017) and (Contardo, Denoyer, and Artières, 2017) opened a new pathway for active learning in meta-learning framework which is also called *learning active-learning* (LAL). In one of the first attempt of LAL studies, Woodward and Finn, 2017 extend the memory contained LSTM network to active learning form, and they used reinforcement learning for training purpose.

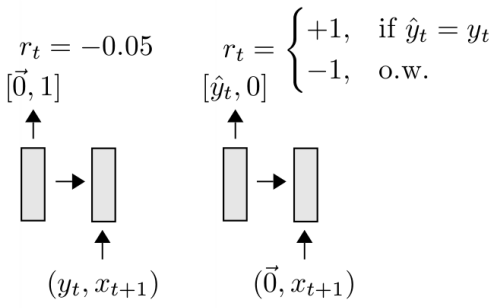
Similarly, (Bachman, Sordoni, and Trischler, 2017) proposed a LAL algorithm as selection model which is trainable in an end-to-end meta-learning fashion, using reinforcement learning. The proposed idea outperforms both of the traditional hand-designed selection models and task-agnostic heuristics. In fact, the proposed model in (Bachman, Sordoni, and Trischler, 2017) was an extension of Matching Network (MN) (Vinyals et al., 2016) to a pool-based active learning setting, a pool of unlabeled data. In the following sections, we will briefly cover two methods for understanding the methodology.

3.2.1 Online-based active meta learning

In (Woodward and Finn, 2017), the authors proposed an online-based active meta-learning with two main contributions. First, they set a LSTM as a reinforcement learning problem to select the most important question to ask from an oracle. Second, they introduced an active meta-learning style for quick learning.

Let illustrate the proposed method by Figure 3.2 (Woodward and Finn, 2017). First, the agent outputs an one-hot vector as its action a_t. The last element of the

FIGURE 3.2: Schematic view of the propose active meta learning in (Woodward and Finn, 2017)



(b) Request label (c) Predict label

vector specifies whether the unlabeled example will go for labeling by the oracle (1), or not (0). Based on the actions of agent, three different rewards (R) are considered. In the label requesting scenario, the agent will receive -0.05 reward. In this case, after receiving the label from the oracle, it will update the parameters of the learner using the label information of the requested example. In the prediction scenario, (when no label is requested), the agent or the learner will do prediction without sending label request. Here, the agent will receive $+1$ reward if the prediction is correct, and -1 otherwise.

The training model was based on Q-learning, and the Q-function was as a single-layered LSTM with 200 hidden units. The following mean squared loss function over the Bellman equation is used to train the model.

$$\mathcal{L}(\Theta) := \sum_t [Q_\theta(x_t, a_t) - (r_t + \gamma \max_{a_t+1} Q(x_{t+1}, a_{t+1}))]^2 \tag{3.1}$$

where, θ is the parameters of Q-function, r_t is the reward received by taking action a_t after observing x_t, and γ is the discount factor for future rewards. The input of the LSTM is concatenated one-hot vector y_t and the flattened input x_t. Note that if we are in label request scenario, the one hot vector would be all zero except the last value.

3.2.2 Pool-based active meta learning

In (Bachman, Sordoni, and Trischler, 2017), the author considered a pool-based scenario where the model has access to a pool of labeled and unlabeled data. The algorithm has three modules: 1) a data representation learner module, 2) a selection module to choose the most valued unlabeled example for asking from an oracle, and 3) a prediction module.

In this study, the reward is defined as log-likelihood of predictions on evaluation set:

$$R(E, S_t, h_t) = \sum_{(\hat{x},\hat{y})\in E} logp(\hat{y}|\hat{x}, h_t, S_t) \tag{3.2}$$

where, S_t is a pool of labeled and unlabeled data after asking t examples from an oracle, and $E = \{(\hat{x}, \hat{y})\}_1^n$ is the query or evaluation set. At time t, the model requests the label of a selected item x from the sub-set of unlabeled data at the S_t. Based on

Algorithm 1 Pool based active learning (Bachman, Sordoni, and Trischler, 2017)

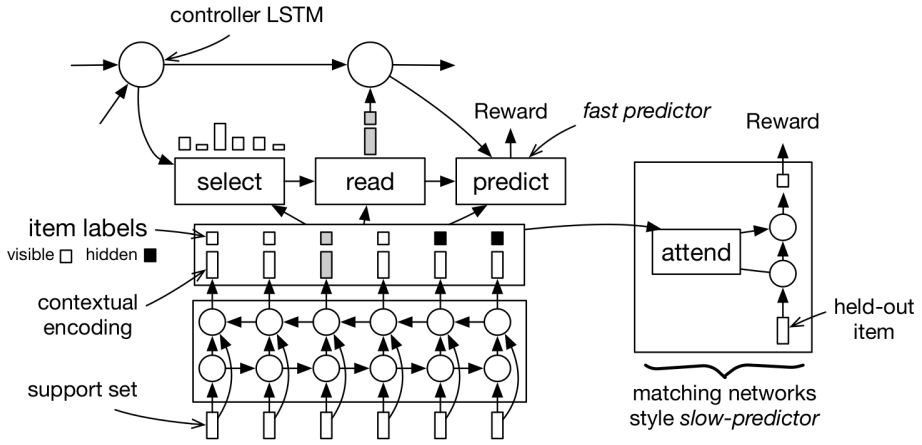
S = $\{(x, y)\}$, $S_0^u = \{(x, .)\}$, $S_0^k = 0$, $E = \{(\hat{x}, \hat{y})\}$ ▷ S is set of all examples, S_0^u is the set of unlabeled data in S. S_0^k is the set of labeled data in S, E is the query items.
# encode all of examples (S) with context-sensitive encoder
# encode all of query items (E) with context-free encoder
for t = 1 ... T **do**
    $i \leftarrow SELECT(S_{t-1}^u, S_{t-1}^k, h_{t-1})$ ▷ select next instance
    $h_t \leftarrow UPDATE(h_{t-1}, x_i, y_i)$ ▷ update the controller
    $S_t^k \leftarrow S_k^{t-1} \cup (x_i, y_i), S_t^k \leftarrow S_u^{t-1}/(x_i, .)$ ▷ update labeled and unlabeled sets
    $L_t^S \leftarrow \text{FAST} - \text{Pred}(S, S_t^u, S_t^k, h^t)$ ▷ perform fast prediction (save train-loss)
$L_T^E \leftarrow \text{SLOW} - \text{Pred}(E, S_T^u, S_T^k, h_T)$

$R(E, S_t, h_t)$, the model updates the control state from h_{t-1} to h_t by maximizing the following objective function:

$$\underset{\theta}{maximize} \; \mathbb{E}_{(S,E)\sim\mathcal{D}} \left[\mathbb{E}_{\pi(S,E)} \left[\sum_{t-1}^{T} R(E, S_t, h_t) \right] \right] \tag{3.3}$$

where, T is the maximum number of label queries to perform, (S, E) is the episode sampled from data distribution $\mathcal{D}$, and $\pi(S, T)$ indicates unrolling the model's active learning policy π. Algorithm 1 is formalized the idea, and Figure 3.3 shows the overall architecture which consists of the following components:

FIGURE 3.3: Schematic view of the propose active meta learning in (Bachman, Sordoni, and Trischler, 2017)



- **Context-free encoder** module takes item $x_i \in S$ to embedding space $x_i^{'}$ using a CNN arcitecture.
- **Context-sensitive encoder** module takes $x_i^{'}$ to the second level of embedding space $x_i^{''}$ for all S using matching network (Vinyals et al., 2016).
- **Read** module concatenates the embedding $x_i^{''}$ and label y_i for selection module. This module also linearly passes the concatenated item to the controller after linear transformation.

- **Controller** module, then, receives the input r_t from reading module, and perform an LSTM update $h_t = \text{LSTM}(h_{t-1}, r_t)$.

- **Selection** module is the most important contribution of this study. Let S_t be the set or pool of labeled and unlabeled data after asking t examples from an oracle. Let S_t^u be the sub-set of examples of support set whose labels are unknown till time t, and let S_t^k denote the sub-set of S_t whose labels are known (have been asked from oracle). Let h_t denote the state of the model (e.g. LSTM) at time t within a task.

  The aim of selection module is to fit a distribution P_t^u over the unlabeled items $x_i^u \in S_t^u$. Here, P_t^u is computed by a gated linear combinations of the *similarity features*. This module samples the index of an item from S_t^u to label and feed it to the reading module.

  For each item, we first compute the controller-item similarity features by the following gated function: $b_t^i = x_i^{''} \odot W_b h_t$, where W_b is a trainable matrix, and $\odot$ indicates element-wise multiplication. They also compute [max | mean | min] cosine similarity for all labeled and unlabeled items. The computed cosine similarities of controller-item features and item-item similarity features are concatenated to get d_t^i vector. Next, a gated function $g = \sigma(W_b h_t)$ is computed for each $x_i^u \in S_t^u$ in the following form:

$$p_t^i = (g_t \odot d_t^i)^T w_p \tag{3.4}$$

  where, W_b is a trainable matrix, and w_p is a trainable vector. Finally, we compute P_t^u using a softmax (for normalizing) over logits p_t^i, and the item with highest score is selected for asking label.

- **Fast prediction** module makes an attention-based prediction for each unlabeled item using cosine similarities between unlabeled data and labeled data.

- **Slow prediction** module is the modified Matching Network wich is covered in previous chapter. This module gives the final prediction for the remaining unlabeled examples which were not chosen by selection module.

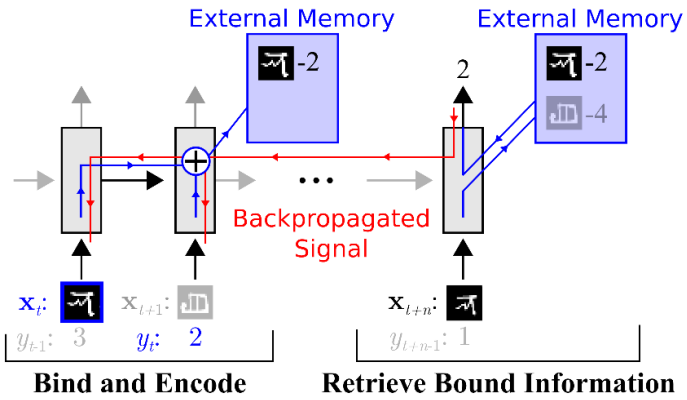
The model has two prediction functions: *fast* predictor and *slow* predictor as shown in figure 3.3. The goal of fast predictor is to make an attention-based prediction for each unlabeled example x_t^u using labeled data x_t^k. The fast prediction returns its *reward* (R) based on cosine similarity between x_t^u and x_t^k.

3.3 Memory-augmented model for quick adaptation

The main problem of meta-learning using recurrent neural networks like LSTM in matching network (Vinyals et al., 2016) is an unstructured way of the storing and retrieving of the information. To address this problem, Santoro et al., 2016 proposed memory augmented network for k-shot learning. The first contribution was that the information stored in a structured way such that we can retrieve when they needed. In this case, we should have element-wise addressability in the sense that relevant pieces of information can be accessed selectively. As the second contribution, the number of parameters will independent of the memory size. In fact, both

of these characteristics (which were missed in LSTM) are provided by **Neural Turing Machines (NTMs)** (Graves, Wayne, and Danihelka, 2014). Thus, NTMs might use its high speed of memory storage and retrieve for rapid updates and increase the prediction accuracy in *k*-shot learning problem. In (Santoro et al., 2016), the author adapted NTM for one shot learning and called it memory augmented neural network (MANN). In the following section we will discussed the components of MANN (shown in figure 3.4).

FIGURE 3.4: Schematic overview of MANN proposed by Santoro et al., 2016



Reading from memory matrix (M_t)

At the core of a NTM, there is a neural network algorithm (like LSTM or MLP) called *controller*, which manages the memory interactions. The controller takes the input x_t at time t and produces a key vector k_t for that time. The model either stores k_t in the one of the row of a memory matrix M_t using write-weight, or it uses k_t to retrieve the necessary information from the memory.

Let see how the controller retrieve information from the memory (M_t) using read-weight vector. During the reading of a particular raw, i, of the memory, M_t is addressed using cosine similarity,

$$d(k_t, M_t(i)) = \frac{k_t.M_t(i)}{||k_t||.||M_t(i)||} \tag{3.5}$$

Then, the read-head uses a softmax over $d(k_t, M_t(i))$ to produce a read-weight vector for the i^{th} row of the memory ($\mathbf{w(i)}_t^r$) in the following way:

$$\mathbf{w}_t^r(i) \leftarrow \frac{exp(d(k_t, M_t(i))}{\sum_j exp(d(k_t, M_t(j)))} \tag{3.6}$$

Finally, the model reads memory using following retrieve vector ($\mathbf{r}_t$):

$$\mathbf{r}_t \leftarrow \sum_i \mathbf{w}_t^r(i)\mathbf{M}_t(i) \tag{3.7}$$

In fact, $\mathbf{r}_t$ contains all of the retrieved information which is necessary in finally output. Therefore, the controller uses $\mathbf{r}_t$ as the input to preform classification or regression.

Writing on the memory (M_t)
The NTM proposed in (Graves, Wayne, and Danihelka, 2014) contain two memory address mechanisms: content and location-based mechanisms. Location-based addressing was used to write the sequence-dependent data. However, we can omit these mechanisms in the case of learning from tasks which their embedding are not sequenced base like k-shot classification from images. Consequently, a new access mechanism called Least Recently Used Access (LRUA) module is proposed by (Santoro et al., 2016). LRUA is a pure content-based memory writer and writes on either:

1. the least used memory location: preserving recently encoded information
2. the most recently used memory location: update of the memory with newer and possibly more relevant information

To distinguish between (1) and (2), the author proposed to compute the interpolation between the previous read weights and weights scaled according to usage weights $\mathbf{w}_t^u$ and updated in the following form:

$$\mathbf{w}_t^u \leftarrow \lambda \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w \tag{3.8}$$

where, λ is a decay parameter, and a parameterized sigmoid gate called *write weights* is used to train $\mathbf{w}_t^w$, and this is used to compute a convex combination of the previous read weights and last least-used weights:

$$\mathbf{w}_t^w \leftarrow \sigma(\alpha)\mathbf{w}_{t-1}^r + (1 - \sigma(\alpha)\mathbf{w}_{t-1}^{lu}) \tag{3.9}$$

where, $\sigma(.)$ is the sigmoid function, and $\mathbf{w}_{t-1}^{lu}$ is the *least-used* weights for a given time-step and can be computed using $\mathbf{w}_t^u$. Let $m(v, n)$ be the n^{th} smallest element in vector v. Therefore, $w_{t-1}^{lu}(i) = 1$ if $w_t^u(i) \leq m(\mathbf{w}_t^u, n)$, and 0 otherwise. Here, n is set to be the number of reads on memory. Finally, writing on the memory happens in the following way:

$$M_t(i) \leftarrow M_{t-1}(i) + w_t^w k_t, \forall i \tag{3.10}$$

3.3.1 Extensions

Learning to Remember rare Events (Kaiser et al., 2017)
Kaiser et al., 2017 proposed a large-scale life-long memory module for deep learning. Except for a fast nearest-neighbor which is efficient for large-scale memory, all parts of the model was fully differentiable in the proposed model. In this study, the model trained in end-to-end in a life-long way. The author added the proposed memory to a convolution neural network for supervised classification. In their experiments, the memory is tested by CNN augmentation for lifelong one-shot classification.

A Simple Neural Attentive Meta-Learner (SNAIL) (Mishra et al., 2018)
According to (Mishra et al., 2018), the goal of meta-learning is the generalization between tasks instead of data points. Episode or task $\mathcal{T}_i$ defines by x_t and outputs y_t, and a loss function $\mathcal{L}_i(x_i, a_i)$. In this case, x_t comes from a transition distribution $P_i(x_t|x_{t-1}, a_{t-1})$, and an episode length H_i, and y_t is produced by a meta-learner models which has the distribution $\pi(y_i|x_1, ..., x_t; \theta)$ by minimizing the expected loss with respect to θ.

$$\min_\theta \mathop{\mathbb{E}}_{\mathcal{T}_i \sim \mathcal{T}} \left[\sum_{t=0}^{H_i} \mathcal{L}_i(x_t, a_t) \right] \tag{3.11}$$

The main idea of SNAIL was to proposing a meta-learner which can be universally applicable to both supervised and reinforcement learning. Additionally, the author tried to overcome the problem of MANN by Santoro et al., 2016 which used RNNs as meta-learner. The issue of MANN is that traditional RNN architectures propagate information by keeping the information in their hidden state during forwarding propagation; this temporally-linear dependency bottlenecks their capacity to perform better results over the hand designed methods.

3.4 Proposal

In the previous sections, we discussed the idea of *learning active learning* (LAL), and memory augmented neural network in the meta-learning framework. However, the key question is: *how we can build a bridge between these ideas and benefit from the best of two worlds?*

3.4.1 Interactive quick adaptation

In all of the discussed LAL algorithms, the active learners were based on LSTM to make a prediction and selection. However, Santoro et al. (2016) showed memory augmented networks could adapt quickly compared to LSTM in the case of meta-learning. To the best of our knowledge, there is no attempt to use the idea of augmented neural networks for LAL. The advantage of the discussed methods motivated us to investigate the idea of augmented neural networks to benefit from their quick adaptation capacities.

Learning active *learning* (LAL) by memory augmented neural networks
As a LAL scenario, we are planning to work on pool of unlabeled examples within each episode for building a selection module. At time t, we will consider an episode or task $\mathcal{T}_t = \{S_t, Q_t\}$ as the union of support set $S_t = \{(x_1, y_1), (x_2, y_2), ..., (x_{kN}, y_{kN})\}$ and the query set $Q_t = \{(x_1, null), ..., (x_n, null)\}$. Here, k is the number of examples selected from N classes, and $n \geq 1$ is the number of examples in query (evaluation) set.

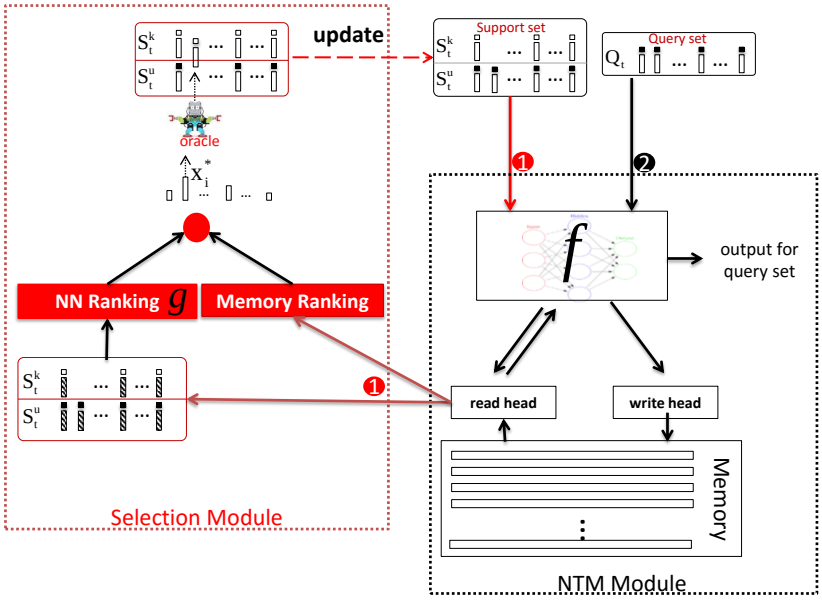
However, our main contribution would be designing a memory augmented model for active learning purpose. In other words, we want to design a *memory* module ($\mathcal{M}$) and a *selection* module. The memory module ($\mathcal{M}$) would have two purposes: 1) helping a controller (e.g. MLP, LSTM or etc.) to complete the final prediction of the model, and 2) helping the selection module to find the best-unlabeled example for asking from the Oracle. On the other hand, the selection module aims to find the most valued example. The schematic overview of this idea is shown in figure 3.5.

Our designed selection module sequentially learn to select the most critical example. Let $S_t^u \in S_t$ be the subset of the support set that we pretend to don't know their labels, and let $S_t^k \in S_t$ be the subset of examples (in support set) whose labels have been requested from the oracle before time t. First, our proposed method sequentially selects the most valuable examples from S_t^u.

Selection Module
Figure 3.4.1 represents our proposed idea. In this section, we follow the red arrows and illustrate the red box in the figure which are showing the selection procedure. At time t, our proposed selection module fits a distribution over the examples in S_t^u, and select the most discriminative example x_t^u from S_t^u. To reach this aim, we are

proposing a combination of two ranking function which gives scores on the importance of the samples for asking from an oracle. Our intuition is based on two criteria: 1) we want to select an example with a large distance to the memory slots, and 2) we want to consider high selection score for the samples which are far from their "classmates". Let discuss them in detail.

FIGURE 3.5: Schematic overview of the proposed memory augmented LAL with two module: (1) selection module (colored by red), and (2) memory augmented network (colored by black).



- **1) Memory-sample ranking**
  In this case, first, S_t^u is given to the controller, and it returns a key $k_{t,i}$ for each of $x_{t,i}^u \in S_t^u$. Then, we can compute the similarity vector between $k_{t,i}$ and memory slots. Here, we compute the cosine similarity vector $d_{t,i}$ between the memory slot M_t and the $k_{t,i}$:

$$d_{t,i} = d(k_{t,i}, M_t) = \frac{k_{t,i}.M_t}{||k_{t,i}||.||M_t||} \tag{3.12}$$

  Then, we normalize the distances using a softmax layer

$$R_t^i = \frac{exp(k_{t,i})}{\sum_j exp(k_{t,j})}, \tag{3.13}$$

  to obtain R_t^i, which is the rank of of ith example in the S_t^u.

- **Nearest neighbour ranking**
  First, we give S_t^u and S_t^k to the controller, and it returns a set of keys $\mathcal{K}_t^u = f(S_t^u)$ and $\mathcal{K}_t^k = f(S_t^k)$ for each of the input sets. Then, the memory augmented network retrieves the necessary information from all the examples. For $k_{t,i} \in \mathcal{K}_t^k \cup \{\mathcal{K}_t^u\}$ the retrieved vector is computed by:

$$\mathbf{r}_{t,i} \leftarrow \sum_i \mathbf{w}_t^r(i)\mathcal{M}_t(i), \tag{3.14}$$

and $\mathbf{w}_t^r(i)$ is the read weights and defined as:

$$\mathbf{w}_t^r(i) \leftarrow \frac{exp(d(k_{t,i}, \mathcal{M}_t(i))}{\sum_j exp(d(k_{t,i}, \mathcal{M}_t(j)))} \tag{3.15}$$

where, d is a distance metric like cosine distance.

Then, the retrieved vectors for all of the examples in S_t^u and S_t^k are used to train a function g, which is a distance based neural network. The goal of g is to learn a space such that we will be able to do sample ranking according to the categories of nearest neighbours.

Please note that we must design our network according to the meta-learning framework. One possibility could be a prototypical network (PN) (Snell, Swersky, and Zemel, 2017). However, PN works with k-shot learning when we have a uniform number of examples in the support set. In our case, the number of examples per classes in S_t^k could be unbalanced. Therefore, we are proposing a network called nearest neighbor network. The goal of this network is to push all retrieved vectors $\mathbf{r}_{t,i}^u$ corresponding to S_t^u towards their nearest neighbor in the same classes in $\mathbf{r}_{t,i}^k$. We designed the following cost function:

$$\mathcal{L}_\theta = \exp\big(d(g_\theta(\mathbf{r}_{t,i}^u), a_c)\big) + log \sum_{c'} \big(exp\big(- d\big(g_\theta(\mathbf{r}_{t,i}^u), a_{c'}\big)\big) \tag{3.16}$$

where, d is a distance metric, $\mathbf{r}_{t,i}^u$ is the ith instance from $\mathcal{K}_t^u$, a_c is the nearest neighbour of $\mathbf{r}_{t,i}^u$ in the set of known instances $\mathcal{K}_t^k$ from the same class, and a_c' is the nearest neighbour from all other classes.

After computing the loss, we rank the examples according to their within class and between class distances. For instance, the $\mathbf{r}_{t,i}^u$ would get a higher rank to select, if it is far from its nearest classmate and close to the other class examples. Finally, we will use a softmax layer for normalizing the rankings of all unknown retrieved vectors.

After both of the rankings, we can compute the overall selection scores over S_t^u using simple addition or a gating function. Finally, the item with the highest score is selected for label request from the oracle.

3.4.2 Optimization

Due to the complexity of the model, we think the optimization of the discussed memory augmented LAL method is a non-trivial task. Generally, memory augmented networks are fully differentiable, and researchers use gradient descent for optimization. However, we added a selection module to a memory augmented network. To the best of our knowledge, there is no attempt for optimizing such a complex network. In this section, we cover our proposed optimization technique which can be used for training our discussed memory augmented LAL network.

Background

(Finn, Abbeel, and Levine, 2017) proposed model-agnostic meta-learning (MAML) algorithm presented in Algorithm 2 which is compatible with any model trained with gradient descent and applicable to different learning problems like classification, regression and reinforcement learning. The idea is to take two level of gradient descent for base and meta layers respectively.

Algorithm 2 MAML's algorithm proposed in (Finn, Abbeel, and Levine, 2017)

while not convergence **do** ▷ We have the answer if r is 0
  Randomly select N number of tasks $\mathcal{T}_N$ from all tasks
  for $\mathcal{T}_i \in \mathcal{T}_s$ **do**
    $D_{base} \leftarrow$ {S, E} sample k labeled and n unlabeled examples from N classes
    $G_{base} \leftarrow \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$ ▷ Compute gradient of $\mathcal{L}$ at base level using D_{base}
    $\theta'_i \leftarrow \theta - \alpha G_{base}$ ▷ Update Parameters
    $D^t_{meta} \leftarrow$ {S, E} sample examples for meta training
  endfor
  $G_{meta} \leftarrow \nabla_\theta \sum_{\mathcal{T}_i \sim \mathcal{T}_S} \mathcal{L}_{T_i}(f_{\theta'_i})$ ▷ Compute meta-gradient using D_{meta}
  $\theta \leftarrow \theta - \beta G_{meta}$ ▷ Update meta-parameters
endwhile

Our proposed method for optimization

Let f be the memory augmented network in the proposed active learner model discussed in 3.4.1, and $\mathcal{T}_t = \{S, E\}$ be a task as the union of support set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_{kN}, y_{kN})\}$ and the evaluation set $E = \{(x_1, null), ..., (x_n, null)\}$. Where, k is the number of examples selected from N classes, and $n \geq 1$ is the number of examples in query (evaluation) set. Based on the define loss function, we are planning to propose a two level gradient base (MAML like) algorithm for our discussed memory augmented LAL model. Our proposed algorithm is shown in algorithm 3. To the best of our knowledge, this is the first time that we are proposing MAML for a memory augmented network.

Algorithm 3 Adapted MAML's algorithm for memory augmented learning active *learning* (changes are in red color line)

while not convergence **do**
  Randomly select N number of tasks $\mathcal{T}_N$ from all tasks
  for $\mathcal{T}_t \in \mathcal{T}_N$ **do**
    $D_{base} \leftarrow$ {S, E} sample k labeled and n unlabeled examples from N classes
    for $i = 1$ to $|S^u_t|$ **do**
      $x^u_{t,i} \leftarrow SELECT(S^u_{t-1}, S^k_{t-1}, M_{t-1})$ ▷ Select next instance
      $S^k_t \leftarrow S^{t-1}_k \cup x^u_{t,i}, S^k_t \leftarrow S^{t-1}_u / x^u_{t,i}$ ▷ Update labeled and unlabeled sets
      $G_{base} \leftarrow \nabla_\theta \mathcal{L}_{\mathcal{T}_t}(f_\theta)$ ▷ Compute gradient of $\mathcal{L}$ at base level usingS^{t-1}_k
      $\theta'_t \leftarrow \theta - \alpha G_{base}$ ▷ Update Parameters
      $D^t_{meta} \leftarrow$ {S, E} sample examples for meta training
    endfor
  endfor
  $G_{meta} \leftarrow \nabla_\theta \sum_{\mathcal{T}_t \sim \mathcal{T}_S} \mathcal{L}_{T_t}(f_{\theta'_t})$ ▷ Compute meta-gradient using D^t_{meta}
  $\theta \leftarrow \theta - \beta G_{meta}$ ▷ Update meta-parameters
endwhile

3.5 Summary of the contributions

Our contributions are summarized as follows:

1. **Memory augmented learning active *learning* (LAL) for fast adaptation**:
   To the best of our knowledge, this is the first time of expanding NTMs machine for learning active *learning* in meta-learning framework. Following the discussed research path, we aim to provide an alternative method to classical and theoretical methods for active learning which are not successful compared to the neural network based methods.

2. **Investigating two level gradient**:
   Based on our knowledge, we are the first to proposing two level of gradient descent in memory augmented network.

Chapter 4

Reducing Annotation Amount in Visual Learning

4.1 Introduction

As the third research direction, we are planning to extend our proposed method in Chapter 3 to a real-world application. This direction would be at the end of my graduate studies; therefore, we are suggesting to follow one of the last updates in object detection or recognition (like place recognition) from temporal data. More specifically, we are going to extend *interactive fast adaptation* (discussed in Chapter 3) to either object detection or visual place recognition. Our primary purpose is to reduce the cost of labeling by learning an active *learning* method in one of the object detection or visual object recognition problems.

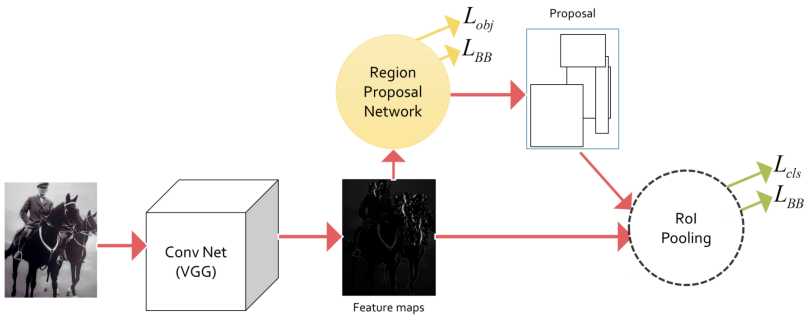
Object detection is one of the core problems in computer vision, and its goal is to find the pre-defined categories like cats, dogs, fish, etc. The task is to draw a box around the known object class when we saw an image of that class. Object detection is different from supervise classification because an image can contain a different number of objects. Therefore, the problem of object detection is a challenging one. However, the current object detection algorithms are hard to extend to a real-world problem in the cases when we do not have a dataset with a few labeled data. In such cases, we could propose to use learning active *learning* methods discussed in Chapter 3 for reducing the amount of annotated data needed for building a robust object detector model.

Visual place recognition is also a challenging problem where the agent has to decide whether or not an input image was seen before. Generally, visual place recognition consists of two fundamental subproblems. First, the agent must create an internal representation of the input images. Second, the agent must be able to decide whether the representation of that current input images have seen before or not, and if so, which one. (Lowry et al., 2016). The visual place recognition is a very challenging problem in robotics because of the high range of changes in the appearance of the real world. In other words, we can consider the place recognition as matching the representation of the current input image with the existing representations of the places. We believe that the idea is very similar to the discussed k-shot learning methods like matching network in Chapter 2.

4.2 Object detection

Girshick et al., 2014 proposed first region based convolutional neural network (R-CNN) for the object detection problem from temporal data. Although R-CNN showed an outstanding performance increase over the past state-of-the-art methods at that time, the algorithm was very slow. The testing time was about 49 second per frame, and it was not applicable in real world temporal situations. This motivated Girshick, 2015 to propose a new version of the algorithm called *fast* R-CNN to decrease the testing time from 49 to 2.3 seconds. In fast R-CNN, first, a selective search extracts region proposals. Then, a region of interest (ROI) pooling layer returns a fixed-length feature vector from the feature maps. Finally, a branch of fully connected layers encodes a softmax probability for object recognition, and the other branch outputs 4 real values (as the coordinates of the bounding box) for each of the object in the input frame.

FIGURE 4.1: Faster R-CNN architecture. The loss are shown as arrow connected to RPN and RoI polling layer.



In the following year, Ren et al., 2015 proposed a new algorithm called *faster* R-CNN which was composed of two modules: a deep fully convolutional network and a detector. The idea was to design a region proposal network instead of a slow selective search in fast R-CNN. While the first module, deep fully convolutional network, proposes the regions of interest by a network called Region Proposal Network (RPN), the second module which is called Fast R-CNN (Girshick, 2015) uses the proposed regions for object detection. RPN could decrease the testing time to 0.2 seconds per image. The RPN is a fully convolutional network which takes an image in any size and outputs a set of rectangular object proposals. We assume that both RPN and Fast R-CNN to have the same set of convolutional layers. The convolutional layers can be any CNN architecture like VGG-16.

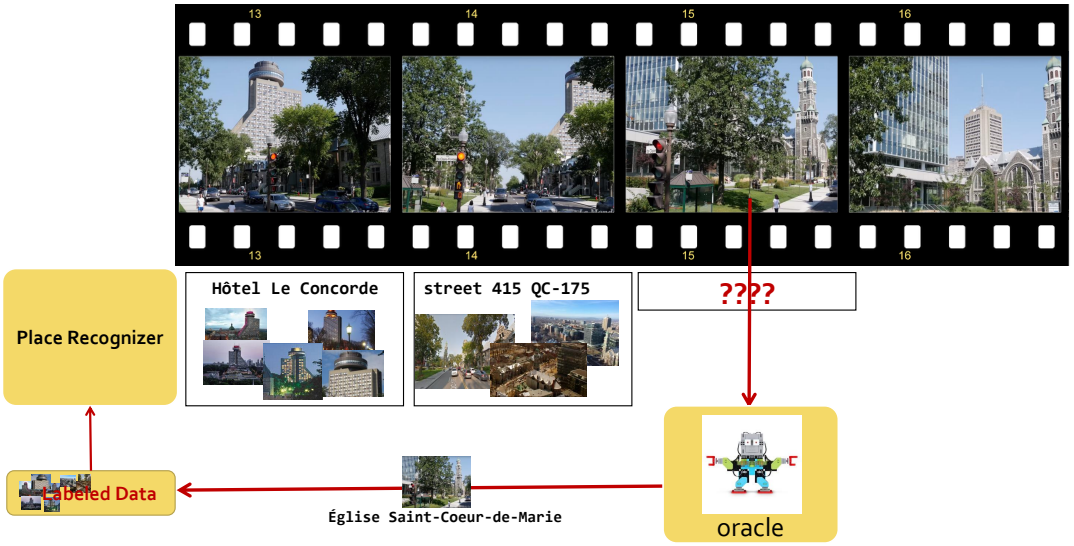
Redmon et al., 2016 proposed a method called *You Only Look Once* (YOLO) which was even faster than all discussed region based methods. The method's speed was 45 frame per second. YOLO divides the input image into the fixed-size grids. The assumption was each grid cell can propose a fixed number of boundary boxes to predict only one object. After the first version of YOLO, Redmon and Farhadi, 2017 proposed second version of YOLO called YOLOv2 which contains few tricks for training improvement and performance increase. Additionally, Redmon and Farhadi, 2017 proposed an even more robust and powerful version called *YOLO*9000 at the same study.

Although the discussed object detection methods has shown acceptable performances benchmarks with large labeled data, it is very hard to extend to the real world problems with few annotated data. Therefore, one possible research path could be converting these methods to k shot learning active learning.

4.3 Visual Place Recognition

Before flourishing deep learning algorithms, algorithms such as FAB-MAP Cummins and Newman, 2008, which matches the appearance of an input image to the past representations of the places using bag-of-words, was the state-of-the-art in visual place recognition. After 2012, researchers tried to use deep learning algorithms as feature extractor for visual place recognition. (Chen et al., 2014) tried the first attempt for place recognition based on CNN, and achieve a 75% increase in recall at 100% over 70 km benchmark place recognition. The author proposed an approach with two key components: feature extraction and spatiotemporal filter of places. The idea was to match the extracted features of an input image with the features of pre-seen images. In anohter attempt for visual place recognition, Chen et al., 2017 proposed a CNN-based image features. The proposed model can create regional representations of salient regions using convolution layer activation. The model has four following steps. First, the author used a pre-trained model to extracts the local descriptors from convolutional activation map according to the receptive field of convolutional layers. Here, convolutional filters at low layer are used to extract each salient region. Then, the convolutional filters at higher layer identify salient regions. Finally, the model matches the input image with the pre-seen images after weighting each of the salient regions.

FIGURE 4.2: Active visual place recognizer



4.4 Proposal: Learning visual learning

As we discussed in the previous sections, deep learning algorithms are showing good performances in object recognition and detection on the existing benchmarks. One of the big challenges with training the existing algorithms is the amount of annotated data. Most of the current algorithms require large labeled data for training. However, gathering the datasets like PASCAL VOC (Everingham et al., 2015) and ImageNet (Russakovsky et al., 2015) can be very expensive. To reduce the necessity

for huge labeled data and to train deep learning algorithms with less data, we are planning to propose *k*-shot learning like the discussed one in Chapter 2 for visual place recognition. More specifically, we are planning to propose a *k*-shot learning algorithm which has the ability of data augmentation like generative models. However, extending *k*-shot learning to a real world problem would not be a trivial task. The reason is that *k*-shot learning algorithms like ours are still far from of recognition in real-time real-world application. Therefore, we are proposing a mixture of *k*-shot and active learning methods for visual place recognition. First, we want to propose the the idea of *k*-shot learning for visual place recognition. Then, we are proposing to design a selection module to select the most valuable instances for annotation, when the *k* shot learner is unable or not confident about their labels.

Here, our main contribution and aim is to propose an idea for decreasing the amount of annotated data while an oracle is available. However, the oracle is not necessarily a human, and it can be GPS, position of camera and etc. The schematic overview of the idea is shown in Figure 4.2. The model will have a place recognizer which tries to recognize and match the current input image with the existing few annotated examples in the dataset. Additionally, the model is targeted to find new categories (places) and select them for labeling by asking from an oracle.

4.5 Summary of the contributions

Due to the fast changing in research domains of machine learning, we propose to work on one of the research directions listed at bellow. In each of the directions, our possible contribution is summarized as follows:

- **Direction I)** *k*-shot object recognition:
  We are proposing extend our proposed idea *k*-shot learning active learning for object recognition purpose.

- **Direction II)** *k*-shot based active learning for visual place recognition:
  If we choose to work on place recognizer, the model will do *k*-shot learning on frames of a temporal data. When the model is not confident, the selection module will select the input image of places for asking from an oracle.

Chapter 5

Conclusion and Schedule

5.1 Conclusion

In this proposal, we investigate some solutions for slow adaptation problem in most of the classic machine learning approaches. Generally, these approaches are slow in "learning" new tasks, and they required a lot of annotated examples in the learning period. However, we know that the human adaptation to new tasks is swift. Here, the big question is: *can we decrease the size of training dataset and adapt quickly out of few examples*? Recently, several methods have been proposed for fast learning process. Typically, researchers consider limited k (where $k \leq 5$) number of examples per class in their studies, and we call this research direction as k-shot learning. After a literature review on the proposed ideas, we are introducing new algorithms for some of the existing limitations in the current k-shot learning studies.

We proposed three new ideas for the discussed research directions in the framework of "learning to learn" or meta-learning which aims to fast adaptation like the human. Our first contribution is realistic and variational data augmentation. There are some attempts on unrealistic data augmentation using generative adversarial networks (GAN), but we proposed a new realistic data generation using conditional variational autoencoder GAN (cVAE-GAN). Our main contribution is to adapt cVAE-GAN to the meta-learning framework by a new conditioning method. Our preliminary results show that realistic data augmentation helps for fast adaptation.

As the second contribution, we proposed memory augmented networks for learning active *learning* (LAL). As the name LAL refers, our idea is still in learning to *learn* framework. Memory augmented networks are proposed to be fast in capturing the essential features of input datasets. Some studies investigated memory contained networks like LSTM to the problem of k-shot learning. However, we took one step further and proposed memory augmented networks like neural Turing machines in LAL framework. Our idea is to decrease the amount of labeled data when an oracle (e.g., a human) is accessible for data labeling. We proposed a new selection mechanism based on memory augmented networks for selecting the most valuable unlabeled data for sending the request for the annotator. This idea is essential when the cost of labeling is very high. For example, the idea can have a huge contribution in bio-informatics, where the cost of labeling is high.

Finally, we propose two possible extension of k-shot learning for a real-world problem. Like the previous sections, our goal is to decrease the demand of large datasets for training deep learning methods which are showing outstanding performances in object detection and recognition.

5.2 Schedule

The schedule and time-line of my Ph.D. plan with the goal of defending at summer 2021 is presented in Figure 5.1.

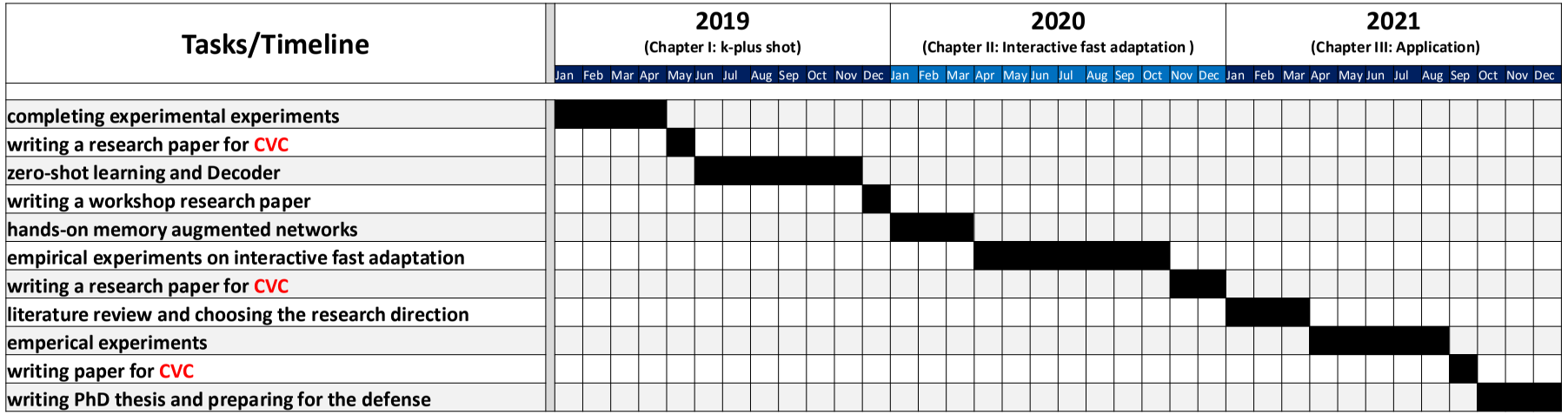


FIGURE 5.1: PhD time-line. CVC stands for conmputer vision conferences like CVPR, ICCV, ECCV.

Bibliography

Bachman, Philip, Alessandro Sordoni, and Adam Trischler (2017). "Learning algorithms for active learning". In:

Cai, Qi et al. (2018). "Memory Matching Networks for One-Shot Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4080–4088.

Chen, Zetao et al. (2014). "Convolutional neural network-based place recognition". In: *arXiv preprint arXiv:1411.1509*.

Chen, Zetao et al. (2017). "Only look once, mining distinctive landmarks from convnet for visual place recognition". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 9–16.

Chen, Zhiyuan and Bing Liu (2016). "Lifelong machine learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10.3, pp. 1–145.

Contardo, Gabriella, Ludovic Denoyer, and Thierry Artières (2017). "A Meta-Learning Approach to One-Step Active Learning". In:

Cummins, Mark and Paul Newman (2008). "FAB-MAP: Probabilistic localization and mapping in the space of appearance". In: *The International Journal of Robotics Research* 27.6, pp. 647–665.

Everingham, Mark et al. (2015). "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111.1, pp. 98–136.

Fink, Michael (2005). "Object classification from a single example utilizing class relevance metrics". In: *Advances in neural information processing systems*, pp. 449–456.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *arXiv preprint arXiv:1703.03400*.

Girshick, Ross (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, Ross et al. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). "Neural turing machines". In:

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping". In: *null*. IEEE, pp. 1735–1742.

Hariharan, Bharath and Ross B Girshick (2017). "Low-Shot Visual Recognition by Shrinking and Hallucinating Features." In: pp. 3037–3046.

He, K et al. (2015). *Deep residual learning for image recognition. CoRR, vol. abs/1512.03385*.

Hochreiter, Sepp, A Steven Younger, and Peter R Conwell (2001). "Learning to learn using gradient descent". In: *International Conference on Artificial Neural Networks*. Springer, pp. 87–94.

Hoi, Steven CH et al. (2006). "Batch mode active learning and its application to medical image classification". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 417–424.

Kaiser, Łukasz et al. (2017). "Learning to remember rare events". In: *arXiv preprint arXiv:1703.03129*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In:

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266, pp. 1332–1338.

Larochelle, Hugo, Dumitru Erhan, and Yoshua Bengio (2008). "Zero-data learning of new tasks." In: *AAAI*. Vol. 1. 2, p. 3.

Laskey, Michael et al. (2016). "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 462–469.

Liu, Ying (2004). "Active learning with support vector machine applied to gene expression data for cancer classification". In: vol. 44. 6. ACS Publications, pp. 1936–1941.

Lowry, Stephanie et al. (2016). "Visual place recognition: A survey". In: *IEEE Transactions on Robotics* 32.1, pp. 1–19.

Mishra, Nikhil et al. (2018). "A simple neural attentive meta-learner". In:

Moskovitch, Robert et al. (2008). "Active learning to improve the detection of unknown computer worms activity." In: *FUSION*, pp. 1–8.

Palatucci, Mark et al. (2009). "Zero-shot learning with semantic output codes". In: *Advances in neural information processing systems*, pp. 1410–1418.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *arXiv preprint*.

Redmon, Joseph et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ren, Shaoqing et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Santoro, Adam et al. (2016). "One-shot learning with memory-augmented neural networks". In: *arXiv preprint arXiv:1605.06065*.

Schmidhuber, Jürgen (1987). "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook". PhD thesis. Technische Universität München.

Schmidhuber, Jürgen, Jieyu Zhao, and Marco Wiering (1997). "Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement". In: *Machine Learning* 28.1, pp. 105–130.

Settles, Burr (2014). "Active learning literature survey. 2010". In: *Computer Sciences Technical Report* 1648.

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: pp. 4077–4087.

Springenberg, Jost Tobias et al. (2014). "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806*.

Thrun, Sebastian (1996). "Is learning the n-th thing any easier than learning the first?" In: *Advances in neural information processing systems*, pp. 640–646.

— (1998). "Lifelong learning algorithms". In: *Learning to learn*. Springer, pp. 181–209.

Triantafillou, Eleni et al. (2018). "Meta-Learning for Semi-Supervised Few-Shot Classification". In:

Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur (2015). "Order matters: Sequence to sequence for sets". In:

Vinyals, Oriol et al. (2016). "Matching networks for one shot learning". In: *Advances in Neural Information Processing Systems*, pp. 3630–3638.

Wang, Mei and Weihong Deng (2018). "Deep Visual Domain Adaptation: A Survey". In: *Neurocomputing*.

Wang, Yu-Xiong et al. (2018). "Low-Shot Learning from Imaginary Data". In: *arXiv preprint arXiv:1801.05401*.

Welinder, Peter et al. (2010). "Caltech-UCSD birds 200". In:

Woodward, Mark and Chelsea Finn (2017). "Active one-shot learning". In: *arXiv preprint arXiv:1702.06559*.

Zhang, Jiakai and Kyunghyun Cho (2016). "Query-efficient imitation learning for end-to-end autonomous driving". In: *arXiv preprint arXiv:1605.06450*.

Zhu, Jun-Yan et al. (2017). "Toward multimodal image-to-image translation". In: *Advances in Neural Information Processing Systems*, pp. 465–476.