One-shot Learning Using Sparse Model for Resource-Constrained Systems

Seulki Lee
Department of Computer Science
UNC Chapel Hill
seulki@cs.unc.edu

ABSTRACT

We introduce a one-shot learning algorithm for resource-constrained systems based on a sparse model. One-shot learning is a learning model trying to build a knowledge system by learning only one or a few examples. In order to achieve the goal of one-shot learning, the conventional sparse model is revised in our algorithm. With the sparse model revised for one-shot learning, it becomes able to build a reliable knowledge by leveraging a dictionary as a prior knowledge. It helps one-shot learning algorithm create more general and bias-free knowledge by letting it overcome the weakness of a small number of training examples. In order to make one-shot learning run on resource-constrained systems, a new learning algorithm of a sparse model which only requires one dictionary is devised. In this way, the traditional sparse model which requires heavy computation for a number of dictionaries becomes available to such systems. Our implementation of human video activity classification for resource-constrained system shows that one-shot learning on resource-constrained systems is feasible with the proposed algorithm.

1. INTRODUCTION

Neural Networks (NN) and its extended models such as Deep Neural Networks (DNN) have shown great learning ability. However, they require a large number of training data and computational power to achieve a decent performance.

For example, in order to train a NN for image classification, the number of training examples per class should be larger than the number of neurons in the order of magnitude in practice. As the number of neurons and depth of layers increases, the more training examples become necessary. If the number of training data is not enough, the NN will suffer the problem of overfitting and the performance of generalization will also decrease.

Besides the need for a large number of training examples, training (learning) of NN is usually performed by high-performance GPUs since they have a large number of neurons and parameters to be computed.

Unfortunately, the two major requirements of NN mentioned above are usually not available in resource-constrained systems such as embedded or IoT (the Internet of things) systems. Thus, NN has not been directly applied to such systems so far despite their powerful learning ability.

However, an ability to learn is becoming a crucial part of such systems. Thus, there have been numerous approaches to fit NN to such devices. Quantization of trained model is one of the popular approaches [1]. Even though it is able to fit a large model into a resource-constrained device, it just borrows a NN model that is pre-trained on different machines. Thus, the device is only able to make inference from the borrowed model but unable to learn new knowledge by itself.

Hence, an alternative learning model that fits into such systems which are not based on the traditional NN model needs to be researched in order for such systems to have an ability of learning by itself.

In order to do that, we propose *one-shot learning algorithm based on sparse model* for resource-constrained systems which replaces the traditional NN model. The main goal of this algorithm is to give such devices an ability of learning that is able to build its own knowledge by learning only one or few training examples. It does not require heavy parallel-computation that is usually supported by GPUs in NN model.

One-shot learning tries to mimic human's learning process that requires only one or few examples to learn. For example, a person can recognize a new object after looking at only a few images of it. It is possible because the person is able to extract a necessary information or feature from the object and abstract them based on the person's own prior knowledge. On contrary, NN requires a large number of images to learn it.

One-shot learning is a perfect concept for resource-constrained systems since 1) they cannot perform heavy computation and 2) they also cannot access and handle a large number of training data due to its limited resource.

With one-shot learning, fast and accurate learning requiring only a small number of training examples is able to happen inside a resource-constrained device. Such a device trains its own model without receiving help from any others. It only uses its own computation unit and does not require any additional computing hardware such as GPUs or TPU (Tensor Processor Unit).

We choose *sparse model* as a base for our one-shot learning algorithm since richness of sparse model provides useful prior knowledge that a small number of one-shot learning data is unable to. Specifically, we use a dictionary of a sparse model as a prior knowledge that helps it build a reliable knowledge system. With a dictionary, one-shot learning algorithm becomes able to learn knowledge from a single or few example.

Also, compact representation of data is possible with a sparse model since it represents data in a sparse manner using a dictionary. This is a huge benefit to resource-constrained systems.

Another reason for choosing sparse model is that it is proven effective in many areas such as Computer Vision and Image Processing. In particular, it shows one of the best performances for image de-noising and activity recognition.

We set the following design goals for our one-shot learning algorithm and address them one by one in the following sections.

- Take only one or few training examples per class and make a reliable learning performance.
- Build its own knowledge system without receiving any help from other systems.
- Run the proposed algorithm on a resource-constrained system.

2. RELATED WORK

The idea of one-shot learning has been explored since the advent of a learning algorithm. However, it recently receives the spotlight again and many researchers are trying to achieve true one-shot learning. Several approaches that have been proposed so far are listed as in the following.

A pre-trained deep neural network is used as a base of one-shot learning [2]. Recently, Google DeepMind [3] employed ideas from metric learning based on deep neural features and from recent advances that augment neural networks with external memories. Their framework learns a network that maps a small labeled support set and an unlabelled example to its label, obviating the need for fine-tuning to adapt to new class types. tasks. Another work based on neural network is [4]. It explored a method for learning siamese neural networks which employ a unique structure to naturally rank similarity between inputs. Once a network has been tuned, it can capitalize on powerful discriminative features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions.

Meanwhile, one-shot learning method based on the statistical and probabilistic model is also made. A variational Bayesian framework [5] was proposed to represent object categories by probabilistic models. Prior knowledge is represented as a probability density function on the parameters of models. The posterior model for an object category is obtained by updating the prior in the light of one or more observations. [6] proposed methodology that is a variant of principal component regression (PCR). They show that classical PCR estimators may be inconsistent in the specified setting, unless they are multiplied by a scalar $c > 1$; that is, unless the classical estimator is expanded.

Generally, learning models are conducted on machines with high computational power and they are not suitable for resource-constrained systems such as embedded or mobile devices. However, some mobile devices are able to run a neural network such as convolutional neural network [7] by decreasing the size of the network. Also, by using quantization technique which reduces the precision or size of weight parameters of the neural network, it becomes able to run a pre-trained neural network [8, 9] on an embedded device.

Even though they are able to run a pre-trained model, they still need to borrow already-trained models that are trained on powerful computing machines with a large number of data. Thus, such embedded or mobile devices cannot learn or train by itself. They only able to utilize a pre-trained model.

The relationship between one-shot learning and sparse model was explored before. [10] proposed a model of fast learning that exploits the properties of sparse representations. It tried to build a hardware model based on the fact that humans rapidly and reliably learn many kinds of regularities and generalizations.

3. RESEARCH CHALLENGES

One-shot learning based on sparse model poses some research challenges that should be explored.

First, a clear definition and goal of one-shot learning have to be defined. Ideally, one-shot learning aims to take only one or few training examples to learn.

For example, it should be able to classify an apple and orange by learning only a few images of them. This simple learning process brings some advantages over the traditional NN such as no need of large training data, easiness to train, lightweightness, and swiftness.

Those advantages are achieved by carefully choosing training examples. They severely determine the performance of one-shot learning. What examples should be taken and learned? What type of training examples or learning can be said to be one-shot learning? How many are training examples needed?

Right choice and construction of prior knowledge is another critical issue. Since a small number of examples can hardly provide a reliable statistic of a learning object, it needs help from prior knowledge. Thefore, the choice of prior knowledge seriously affects the result and performance of one-shot learning.

Second, the challenge on how to learn training examples and how to relate them to sparse model has to be researched. It raises broads questions such as why we choose sparse model and how it is combined into a unified one-shot learning algorithm. In order to do that, the characteristics of sparse model that can be leveraged for one-shot learning should be first understood. In more detail, we aim to achieve the richness of sparse model and the lightness of one-shot learning at the same time.

Even though we consider a dictionary of sparse model as our prior knowledge, there is no clear relationship between one-shot learning and sparse model. In order to make it clear and fit sparse model into a united one-shot learning algorithm, the traditional sparse model should be revised or modified.

After being successfully revised, the number of training examples that is able to build a reliable sparse model needs to be discovered. We also need to find the related parameters of the model such as a minimum size of a dictionary.

Third, our one-shot learning algorithm should be applied to resource-constrained systems. How can the algorithm fit into a system with a limited computing resource such as memory and computational power?

Specifically, a large size of the dictionary should be stored in a limited memory space of such devices. It might be done by reducing the size of a dictionary or improving the density of information that it is able to hold.

Also, heavy computations that need to be performed for sparse model such as sparse coding, dictionary learning should be able to run on them by reducing a computational complexity of the algorithm. To overcome these limitations, sparse model including one-shot learning algorithm needs to be revised or modified even more to fit it into resource-constrained systems.

Even though we might be able to reduce the number of training examples by successfully applying one-shot learning algorithm to resource-constrained systems, sparse model can be still computationally expensive to them. Then, what is the lower bound of required resource for one-shot learning? The minimum requirement of computing resource will determine which system is able to run one-shot learning algorithm and which is not.

Lastly, actual applications should be implemented based on our algorithm. We need to find useful examples that are feasible to resource-constrained systems. An application requiring to learn a huge number of categories with only a few available training examples per category can be a potential target of one-shot learning.

Several potential applications would be the following. 1) A mobile phone facing toward in a car. It only takes a photo or records a video when some user-defined contexts or activities happen (car accident, broken road sign or road signal) by learning only a few examples of them. 2) Home security cameras can alert when some dangerous situation happens or a suspicious person is detected. Both are also pre-defined by a user and learned with only a few examples.

The research challenges we investigate in this paper are the followings:

- Definition and goal of one-shot learning: What is one-shot learning and how it should learn? How many and what kind of training examples to be learned?
- Sparse model revised: How the traditional sparse model should be utilized for one-shot learning? How is it revised or modified for it?
- Applicability of one-shot learning to resource-constrained systems: Given a limited computation ability and memory space, how to make one-shot learning algorithm run in such environment?
- Implementation: As an example of one-shot learning, a human video activity recognition system is implemented for a resource-constrained system.

4. ONE-SHOT LEARNING

In this section, the definition and goal of one-shot learning especially for classification problem are defined and its connection to sparse model is introduced. Also, the design philosophy of one-shot learning for resource-constrained systems are discussed.

4.1 Definition

We define one-shot learning as a learning algorithm or system that is able to learn only a few examples which are much smaller than the number of examples that the traditional learning models such as NN require and show an equivalent or similar performance as them.

It should be able to learn or train by using only small number examples. From those few examples, it also should be able to create or build its own knowledge that can be utilized for generalization.

Specifically for classification, one-shot learning takes $n \ll m$ number of training examples for one class, where n is the required number of training examples per class and m is the required number of training examples per class for other learning algorithms such as NN.

For example, Omniglot character dataset which is the encyclopedia of writing systems and languages can be learned with one-shot learning approach. It is one of the best datasets for one-shot learning since it has a relatively small number of training examples per class compared to the number of the class itself.

An ideal one-shot learning algorithm only requires one example per class which is $n = 1$. However, at present, it is almost impossible to achieve a reasonable performance by only learning one example since it can provide only a limited and biased information that is unable to represent and cover all the other example of the same class.

Therefore, many one-shot learning algorithms try to solve this problem by taking more than one training example. Usually, they take tens of examples per class which is $n \neq 1 \ll m$. Note that n is still much less than m.

Unfortunately, this approach does not address the problem entirely. In order to overcome the problem, a one-shot learning algorithm requires another piece called prior knowledge.

4.2 Prior Knowledge

One-shot learning needs a help from some other knowledge in order to overcome its shallow understanding provided by a small number of training examples.

As mentioned above, an assistant called prior knowledge is able to fill the gap. Here, prior knowledge is defined as any existing knowledge that can provide basic knowledge or understanding of target data or class that one-shot learning examples cannot construct or understand due to its limited or biased information.

With help of prior knowledge which originates from either the learning process of one-shot learning itself or other learning systems such as trained NN or other knowledge systems, it creates its own unique knowledge that can be applied to one-shot learning examples such their features or abstraction of them.

As mentioned above, prior knowledge can be constructed by our own one-shot learning system or can be borrowed from a totally different system. Specifically, in our model, we use sparse model as a prior knowledge. In particular, we focus on a dictionary of sparse model and leverage it as prior knowledge for our one-shot learning algorithm.

Sparse model is chosen as a base for one-shot learning algorithm for the following reasons:

- Richness of sparse model especially dictionary is expected to fill the gap between shallow knowledge of one-shot learning example and statistical correctness such as variation.

- Compact representation of data is possible with sparse model since it represents data with a sparse coding based on a dictionary. This is a huge benefit to resource-constrained systems.

- It has been proven effective in many areas such as Computer Vision and Image Processing. Specifically, it shows one of the best performances for image-related tasks.

Since the conventional sparse model which is widely used is not designed for one-shot learning, it needs to be modified for our purpose. The next section describes in more detail on how it is changed and the consequential variation is applied to one-shot learning.

4.3 Design goals for Resource-Constrained System

Before exploring sparse model in detail, we need to clarify design goals of one-shot learning for resource-constrained systems.

One-shot learning is an ideal model for resource-constrained systems. No need for a large number of training examples and its relatively simple learning process makes it an alternative learning model for them.

However, such systems' limited resource such as computational power and memory space still prevents one-shot learning based on sparse model which will be described in the following sections from being directly applied to them.

Specifically, the following constraints need to be considered for resource-constrained systems.

- Small memory space: A number of dictionaries that requires a large memory space or a large number of training examples cannot be handled and saved.

- Computational limitation: Computationally expensive operations that need to be performed for sparse model such as dictionary learning or sparse coding can be hardly or slowly computed.

- Miscellaneous constraints: Other constraints caused by a limited resource such as a limitation on the type of data that can be learned, real-time requirement or s/w (or h/w) limitation of implementation.

By keeping in mind these constraints, the following sections describe revised sparse model for a general system as well as resource-constrained systems in detail.

5. SPARSE MODEL REVISED

In this section, we revise the traditional sparse model to fit it into one-shot learning algorithm. First, the traditional sparse model is introduced briefly. Then, the revised model which includes sparse coding, one dictionary, reconstruction is described in detail. Lastly, the entire learning and inference algorithm is presented based on them.

5.1 Introduction to Sparse Model

In sparse model, a vector $\mathbf{x}$ is represented by $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$ where $\mathbf{D}$ is a $m \times p$ matrix ($m \ll p$) and $\mathbf{x} \in \mathbb{R}^m$, $\boldsymbol{\alpha} \in \mathbb{R}^m$. $\mathbf{D}$ is called the dictionary or the design matrix. The problem is to estimate the signal $\boldsymbol{\alpha}$, subject to the constraint that it is sparse. The underlying motivation for sparse problems is that even though the signal is in high-dimensional (p) space, it can actually be obtained in some lower-dimensional subspace ($m < p$) due to it being sparse ($k < m$), where k denotes the sparsity of $\boldsymbol{\alpha}$. Sparsity implies that only a few (k) components of $\boldsymbol{\alpha}$ are non-zero and the rest are zero. This implies that $\mathbf{x}$ can be decomposed as a linear combination of only a few $m \times 1$ vectors in $\mathbf{D}$, called atoms. The column-span of $\mathbf{D}$ is over-complete ($m \ll p$). Such vectors are sometimes called the basis of $\mathbf{x}$, even though being over-complete means they are not a basis. In addition, unlike other dimensionality reducing decomposition techniques such as Principal Component Analysis, the basis vectors are not required to be orthogonal.

The sparse decomposition problem is represented as,

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\boldsymbol{\alpha}\|_0 \quad \text{s.t. } \mathbf{x} = \mathbf{D}\boldsymbol{\alpha} \tag{1}$$

where $\|\boldsymbol{\alpha}\|_0 = \#\{i : \boldsymbol{\alpha_i} \neq 0,\, i = 1, \ldots, p\}$ is a pseudo-norm, l_0, which counts the number of non-zero components of $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]^T$. This problem is NP-Hard with a reduction to NP-complete subset selection problems in combinatorial optimization. A convex relaxation of the problem can instead be obtained by taking the ℓ_1 norm instead of the ℓ_0 norm, where $\|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^{p} |\alpha_i|$. The ℓ_1 norm induces sparsity under certain conditions involving the mutual coherence of $\mathbf{D}$. The ℓ_1 problem is called basis pursuit.

We use dictionary $\mathbf{D}$ as our prior knowledge. Our goal is to find out $\boldsymbol{\alpha}$ that reproduces $\mathbf{x}$ based on $\mathbf{D}$ ($\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$). Here, the vector $\mathbf{x}$ represents a training example. At the same time the dictionary $\mathbf{D}$ is also learned by itself using $\mathbf{x}$. Thus, the problem of sparse model for one-shot learning is reduced as in the following.

$$\min_{\mathbf{D},\mathbf{A}} \sum_{j=1}^{p} \|\mathbf{D}\boldsymbol{\alpha_j} - \mathbf{x_j}\|_2^2 \quad \text{s.t.} \forall j, \|\boldsymbol{\alpha_j}\|_p^p \leq k \tag{2}$$

where $\mathbf{A}$ denotes matrix of $\boldsymbol{\alpha}$. In order to compute the optimal values of $\mathbf{D}$ and $\boldsymbol{\alpha}$ in Equation (2), K-SVD algorithm [11] is used. It fixes one of the value $\mathbf{D}$, $\boldsymbol{\alpha}$ then computes the optimal value for the other one. It alternates the target value for optimization repeatedly. For example, if $\boldsymbol{\alpha}$ is first fixed, the optimal values of $\mathbf{D}$ is computed. Then, $\mathbf{D}$ is fixed and $\boldsymbol{\alpha}$ is updated next. This process continues repeatedly until convergence.

5.2 Sparse Coding

If we assume that dictionary $\mathbf{D}$ is given, the problem of sparse coding for one training example $\mathbf{x}$ is given by:

$$\boldsymbol{\alpha} = \min_{\boldsymbol{\alpha}} \|\mathbf{D}\boldsymbol{\alpha} - \mathbf{x}\|_2^2 \quad \text{s.t. } \|\boldsymbol{\alpha}\|_p^p \leq k \tag{3}$$

which is derived from Equation 2. $\boldsymbol{\alpha}$ which is he sparse representation of $\mathbf{x}$ can be obtained by various algorithms. Basis Pursuit [12] and Matching Pursuit are the most popular solution for it. For our algorithm, Orthogonal Matching Pursuit [13] is chosen to obtain $\boldsymbol{\alpha}$ and $\mathbf{x}$. Equation (3) is performed for every training example $\mathbf{x}$ one by one. Then their summation minimizes the error.

Once $\boldsymbol{\alpha}$ is obtained, an original training example $\mathbf{x}$ is approximated by $\hat{\mathbf{x}} = \mathbf{D}\boldsymbol{\alpha}$ and its error is calculated as $\boldsymbol{\epsilon} = \mathbf{x} - \hat{\mathbf{x}}$

5.3 One Universal Dictionary

A dictionary $\mathbf{D}$ in Equation 2 is also learned (updated) as well as $\boldsymbol{\alpha}$ which is computed using sparse coding in Equation 3.

A dictionary can be updated at once if all training examples are available. However, one-shot learning algorithm assumes that it is only able to access a small number of examples and they can be accessed one by one at a time. Therefore, a dictionary needs to be updated every time a new example becomes available. That is called online dictionary learning [14]. Specifically for our algorithm, mini-batch version of online dictionary learning is used based on Coordinate descent algorithm (block-coordinate algorithm).

Next, unlike an ordinary sparse model, we create and update only one dictionary $\mathbf{D}$ since a resource-constrained system has a limited amount of memory.

Usually, a learning algorithm using sparse model has a number of dictionaries. Each dictionary corresponds to one class and the number of dictionary increases as the number of classes increase. With multiple dictionaries, each example for one class is reconstructed by finding $\boldsymbol{\alpha}$ using $\mathbf{D_n}$, where n is the total number of classes.

Although it is able to classify a new example with high accuracy by using multiple dictionaries, it requires a large memory space because the size of one dictionary is large. This might increase the accuracy of classification, multiple dictionaries with huge size cannot be stored in a limited space especially for resource-constrained systems. Moreover, it requires a huge computational resource since there are a number of dictionaries to be managed and updated.

Instead, we use only one universal dictionary for our one-shot learning algorithm. It does not construct each dictionary for different classes. Thus, only one dictionary is updated every time a new example comes in an online manner regardless of its class. One dictionary requires a fixed amount of memory space even though the number of classes increases. Also, it can be learned faster than multiple dictionaries.

It might degrade the accuracy since only one dictionary is available for all classes. However, one dictionary with online learning enables a resource-constrained system to perform one-shot learning even though their limited memory space and computational power.

However, it requires an additional memory space to keep the current state of online learning. The additional space required for online learning is $p \times p + m \times p$ since the two matrix $\mathbf{A} = [\mathbf{a_1}, \dots, \mathbf{a_p}] \in \mathbb{R}^{p \times p} = \sum_{i=1}^{t} \boldsymbol{\alpha_i} \boldsymbol{\alpha_i^T}$ and $\mathbf{B} = [\mathbf{b_1}, \dots, \mathbf{b_p}] \in \mathbb{R}^{m \times p} = \sum_{i=1}^{t} \boldsymbol{x_i} \boldsymbol{\alpha_i^T}$ are used to keep the current state of learning [14]. Although the size of matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$ is larger than that of a dictionary $\mathbf{D} \in \mathbb{R}^{m \times p}$, it can be reduced to size of p since it is a diagonal matrix.

By keeping only one dictionary $\mathbf{D}$ and multiple sparsity representations $\mathbf{A_s}$ which is the column matrix of $\boldsymbol{\alpha}$ for one class, we can save a large memory space since the size of $\mathbf{A_s}$ is much smaller than the size of a whole dictionary $\mathbf{D}$ ($(m \times 1) \times n \ll m \times p$, where n is the number of sparsity for one class).

5.4 Reconstruction

Given a dictionary $\mathbf{D}$, a vector signal $\mathbf{x}$ which is a training example can be reconstructed by performing sparse coding in Equation 3.

First, a dictionary $\mathbf{D}$ is initialized by randomizing or borrowing an existing dictionary from another system.

For example, if the first vector $\mathbf{x_1}$ comes and the dictionary $\mathbf{D}$ is updated online, then the first vector $\mathbf{x_1}$ becomes able to be reconstructed by performing sparse coding.

$$\mathbf{\hat{x}_i^j} = \mathbf{D^j} \boldsymbol{\alpha_i^j} \tag{4}$$

Here, $\mathbf{\hat{x}}$ is the reconstructed vector of the original vector $\mathbf{x}$. Also, superscript j indicates the total number of examples that are used to update dictionary and subscript i denotes the i vector.

Since the dictionary $\mathbf{D^j}$ is updated to $\mathbf{D^{j+1}}$ every time a new example $\mathbf{x_{i+1}}$ comes, $\mathbf{\hat{x}_i^j}$ in Equation (4) can be no longer reconstructed. Thus, new reconstruction $\mathbf{\hat{x}_i^{j+1}}$ has to be also updated by calculating Equation (3) as in the following.

$$\boldsymbol{\alpha_i^{j+1}} = \min_{\boldsymbol{\alpha_i^{j+1}}} \|\mathbf{D^{j+1}} \boldsymbol{\alpha_i^{j+1}} - \mathbf{D_j} \boldsymbol{\alpha_i^j}\|_2^2 \quad \text{s.t. } \|\boldsymbol{\alpha}\|_p^p \leq k \tag{5}$$

Note that $\mathbf{\hat{x}_i^j} = \mathbf{D^j} \boldsymbol{\alpha_i^j}$ is used instead of $\mathbf{x}$ in Equation (5). Therefore, every time a new example comes, every sparsity variable $\boldsymbol{\alpha_i^{j+1}}$ has to be updated by using old dictionary $\mathbf{D^j}$ and old sparsity variable $\boldsymbol{\alpha_i^j}$ itself.

However, the number of classes increases, the reconstruction error also increases. Since a reconstructed value $\mathbf{\hat{x}_i^j} = \mathbf{D^j} \boldsymbol{\alpha_i^j}$ is saved instead of the actual value of vector $\mathbf{x_i}$, the reconstruction error $\boldsymbol{\epsilon_i^j} = \mathbf{x_i} - \mathbf{\hat{x}_i^j}$ becomes larger as the number of $\mathbf{x_i}$ increases since

$$\begin{aligned}
\boldsymbol{\epsilon_i^1} &= \mathbf{x_i} - \mathbf{\hat{x}_i^1} = \mathbf{x_i} - \mathbf{D^1} \boldsymbol{\alpha_i^1} \\
\boldsymbol{\epsilon_i^2} &= \mathbf{\hat{x}_i^1} - \mathbf{\hat{x}_i^2} = \mathbf{D^1} \alpha_{\mathrm{i}}^1 - \mathbf{D^2} \boldsymbol{\alpha_i^2} \\
&\vdots \\
\boldsymbol{\epsilon_i^j} &= \mathbf{\hat{x}_i^{j-1}} - \mathbf{\hat{x}_i^j} = \mathbf{D^{j-1}} \boldsymbol{\alpha_i^{j-1}} - \mathbf{D^j} \boldsymbol{\alpha_i^j}
\end{aligned} \tag{6}$$

If all equations in (6) are summed up, the error between vector $\mathbf{x_i}$ and reconstructed vector obtained from dictionary $\mathbf{D^j}$ becomes $\mathbf{x_i} - \mathbf{\hat{x}_i^j} = \mathbf{x_i} - \mathbf{D^j} \boldsymbol{\alpha_i^j} = \boldsymbol{\epsilon_i^1} + \boldsymbol{\epsilon_i^2} + \cdots + \boldsymbol{\epsilon_i^j}$. Therefore, the number of training examples increase, the reconstruction error also increases.

5.5 Learning and Classification Algorithm

Now that all the necessary computations including sparse coding, dictionary learning, and reconstruction are ready, one-shot learning algorithm can be constructed from them. It is depicted as shown in Algorithm 1.

Algorithm 1 One-shot Learning using Revised Sparse Model

```
take a new matrix X_i
if D = ∅ then
    initialize D ∈ R^{m×p} by randomizing X_i
    initialize A ∈ R^{p×p}, B ∈ R^{m×p} as 0
else
    load D, A, B
    D_old ← D
    ϵ ← MaxValue
    t = 1
    while t < MaxIter or ϵ < ErrorThreshold do
        A ← A + AsAs^T where A_s is matrix of α
        B ← B + XAs^T where A_s is matrix of α
        D, A, B ← online learning(D, A, B) [14]
        A_{s,i} ← sparse coding(D, A_{s,i})
        ϵ = X_i − DA_{s,i}
        t ← t + 1
    for j = 1 to i − 1 do
        X̂_j = D_old A_{s,j}
        X ← X ∪ X̂_j
    A_s ← sparse coding(X, D)
    D, A_s ← dictionary learning(X, D, A_s)
```

Also, classification algorithm is depicted in the below. More detail description of it is provided in the next implementation section.

5.6 Memory Space and Computational Complexity

The performance of the revised sparse model needs to be compared with the traditional sparse model.

In terms of memory space, one dictionary updated by online learning requires $2m \times p + p + m \times p + nm = (3m+1)p + nm$ memory space while multiple dictionary

Algorithm 2 Classification

take a new matrix $\mathbf{X}$

$\mathbf{A_s} \leftarrow$ sparse coding$(\mathbf{X}, \mathbf{D})$

for j=1 to j=i **do**

  p_j = HI($\mathbf{A_s}$, $\mathbf{A_{s,i}}$) (histogram intersection)

  $p_g \leftarrow$ Gaussian pdf$(n(\mathbf{A_s}), n(\mathbf{A_{s,i}}))$

  $p_j \leftarrow p \times p_g$

classify $\mathbf{X}$ as class $\arg\max_m p_m$

requires $n(m \times p) = nmp$ memory space. Therefore, one dictionary needs less memory space than multiple dictionaries if the number of class is $n \geq 4$ since $(3m+1)p + nm \leq nmp$ for $n \geq 4$ for the same dictionary size $(m \times p)$.

In terms of computational complexity analysis, our one dictionary approach requires total i number of online dictionary update every time a new training example $\mathbf{x_i}$ comes. Meanwhile, the traditional sparse model with the multiple-dictionary approach requires to constructing only one dictionary $\mathbf{D_i}$ when a new training example $\mathbf{x_i}$ comes. Thus, the revised sparse model requires i times more computations than the traditional one for learning.

However, the revised model requires less computation for classification. It just needs to perform one sparse coding using one dictionary $\mathbf{D}$ to obtain $\boldsymbol{\alpha}$ and compare it with a i number of $\boldsymbol{\alpha_i}$ to find the closest match. On the other hand, the traditional model needs to perform a i number of sparse coding to obtain $\boldsymbol{\alpha_i}$. Thus, for classification, the revised model requires i times less computation.

6. IMPLEMENTATION

In this section, we implement a real application of one-shot learning for resource-constrained systems. One-shot learning is a great solution for such systems since it does not impose a heavy burden on the systems that other learning models such deep learning do.

6.1 Human Video Activity Recognition

As an example of one-shot learning algorithm using sparse model, we implement human activity recognition in video. Activities such as walking, running, waving or bending are classified with only a few examples per each activity class.

We implement it with a limited resource such as small memory space and not heavy computational power. As described in the previous section, only one dictionary is used for the all activity classes and they are classified based on their sparsities $\boldsymbol{\alpha}$.

A video including a human activity with the resolution of 180×144 is taken as an input data (training example). The dataset used for implementation is described in more detail in the next evaluation section.

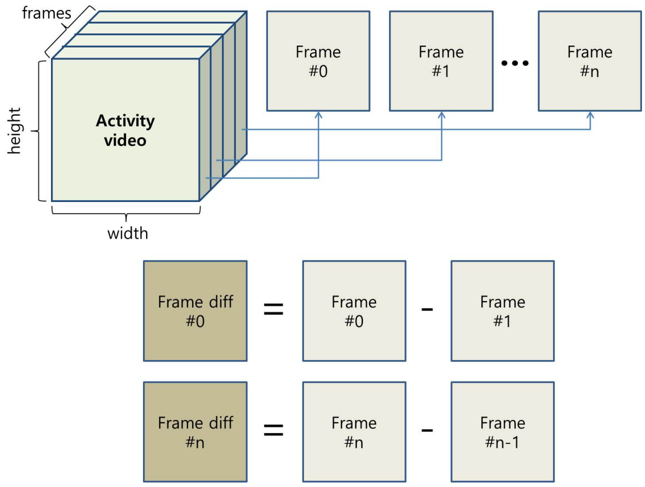


Figure 1: Time-series frames of video are separated into individual frames. Then, the frame-difference is obtained by subtracting two consecutive frames.

Then, the input video is decomposed into a number of image frames. The difference between frames is calculated by subtracting the previous frame from the next frame as shown in Figure 1. Only frame difference that exceeds a certain level of energy is taken.

Then, the frame difference is divided into multiple sub-patches with size $64 = 8 \times 8$ without overlap. Example patches are shown in Figure 5. 8×8 2D patches are converted to a vector $\mathbf{p_k}$ with length $64 = 8 \times 8$.

Generally, creating patches using overlap usually enriches a dictionary and results in better performance. However, in a limited resource environment, it is computationally too heavy and memory space is not enough. Thus, we extract patches from one frame difference without overlapping.

The dictionary has a fixed size of 64×1024 and sparsity level k is 1.

6.2 Learning

As described in the previous section, a dictionary is updated every time a new training activity comes. A set of sub-patch vectors $\mathbf{p_k}$ (maximum 405 patches) form an input matrix $\mathbf{X}$ with size $64 \times k$.

Thus, every time a new activity video comes, maximum 405 number of patches (64×1 vector per each) are created. Then, they construct the final input matrix $\mathbf{X}$ and it is learned through online learning in order to update the dictionary $\mathbf{D}$. The process of constructing $\mathbf{x}$ is shown in Figure 2.

We do not update the dictionary $\mathbf{D}$ for one patch at a time. Instead, a mini-batch update is performed for every 50 patches. By doing this, we improve the convergence speed of our algorithm by learning more than one

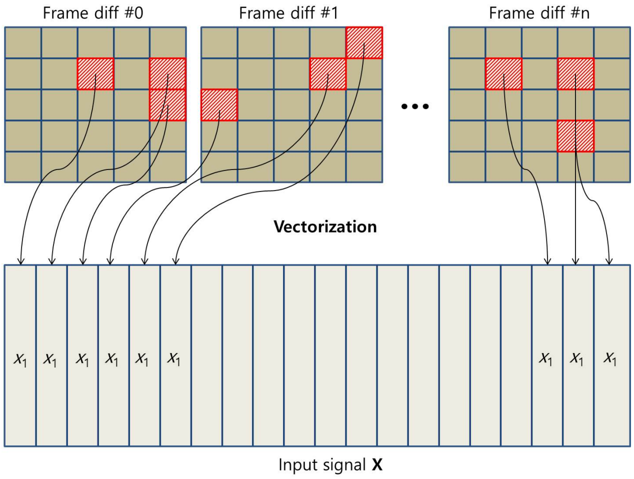


Figure 2: Patches that only has a larger amount of energy than a threshold are extracted each frame-difference matrix. They are used to build an input matrix X which is shown as a column matrix in the below.

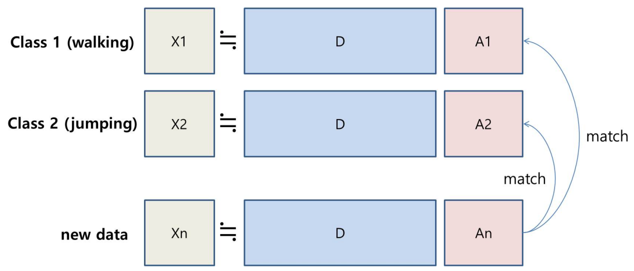


Figure 3: Sparsity matching process for classification. Sparsity matrix $\mathbf{A_{s,n}}$ obtained from a new video is compared to each $\mathbf{A_{s,i}}$ which represents an activity of class i to find the best match.

patch at each iteration instead of a single one, which is a classical heuristic in stochastic gradient descent algorithms.

The online dictionary learning is iterated maximum 10 times. Or, it stops learning if the reconstruction error $\boldsymbol{\epsilon} = \mathbf{X} - \mathbf{DA_s}$ does not decrease anymore.

Therefore, the 2D video stream in time series is converted to a matrix of concatenated column patches to construct $\mathbf{X}, \mathbf{D}$. It is considered as space-time data.

By constructing D, the sparsity representation matrices $\boldsymbol{A_{s,i}}$ for each class of activity are obtained by using sparse coding method described in the previous section. They are not required for learning but classification task is performed based on them later.

6.3 Classification

In order to classify activities from given videos, the same decomposition that is performed for learning is conducted again.

After patches of input video are created from the given video, we reconstruct the matrix $\mathbf{X}$ and obtain a sparsity matrix $\mathbf{A_s}$ by using sparse coding based on the current dictionary $\mathbf{D}$ ($\mathbf{X} = \mathbf{DA_s}$).

Then, $\mathbf{A_s}$ is compared to a set of $\mathbf{A_{s,i}}$, where $\mathbf{A_{s,i}}$ represents a sparsity matrix that reconstructs video activity of class i. If $\mathbf{A_s}$ best matches to one of the $\mathbf{A_{s,i}}$, the given video is classified as same class as class i.

In order to find the best match, we use a modified version of histogram intersection method and Gaussian distribution.

For histogram intersection method, we make it simple by creating a new sparsity vector $\mathbf{h_r}$ by summing each row of sparsity matrix $\mathbf{A_s}$ and put them into the vector $\mathbf{h_r}$ as shown in the following equation.

$$\mathbf{h_r} = \sum_{j=1}^{k} \mathbf{A_s}_{r,j} \tag{7}$$

Here, k is the number of patches of the given video.

Then, histogram intersection HI is given by:

$$\mathrm{HI}(\mathbf{h_1}, \mathbf{h_2}) = \frac{\sum_{j=1}^{r} \min(\mathbf{h_{1,j}}, \mathbf{h_{2,j}})}{\sum_{j=1}^{r} \mathbf{h_{2,j}}} \tag{8}$$

By comparing $\mathbf{h}$ of the given video to a set of $\mathbf{h_i}$ one by one and finding the largest matching score, the activity of the video is easily classified without imposing any computational burden.

We concatenate all the patches in different time frame into one matrix $\mathbf{X}$ and then try to match it to each class. That is because 1) activities we are trying to classify are performed repeatedly during a certain amount of time, and 2) we do not need to synchronize the starting time of an activity if we put all patches to one matrix and try to match it.

Also, we compare the number of patches between two activities to improve classification accuracy. In order to do that Gaussian distribution with a mean and variance is used to obtain the probability of how they are likely to be in the same class. The final classification algorithm is depicted in Algorithm 2.

7. EVALUATION

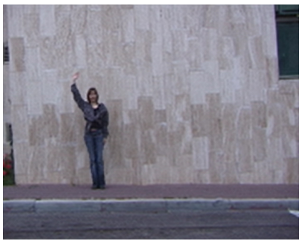
In this section, we evaluate the one-shot learning algorithm for human activity recognition system described in section the previous section.

7.1 Target Dataset

For evaluation, we took human activity video data from the study of Weizmann Institute of Science [15]. Original dataset's number of action classes are ten including walking, running, jumping, gallop sideways, bending, one-hand waving, two-hands waving, jumping in place, jumping jack, skipping. The total 90 videos with

(a) Activity1: walking

(b) Activity2: running



(c) Activity3: waving

(d) Activity5: bending

Figure 4: Snapshots of four kinds of human video activity. Walking (a), running (b), hand waving (c), and bending (d) with resolution 180×144.

the resolution of 180×144 were filmed in a homogeneous outdoor background using a static camera.

The goal of our system is to recognize activities given a human activity video by learning only one training example per class.

Thus, it is a multi-class recognition problem. It needs to classify the four activities correctly by learning only one example per class according to our one-shot learning philosophy. Therefore, a dataset for one class is divided into two sub-dataset – training and test dataset.

Since our target platform is a resource-constrained system, it learns only four activities and classifies them. The four classes of activities it tries to learn are walking, running, waving, and bending. Figure 4 shows snapshots of each activity.

7.2 Patches and Dictionary

As described in the previous section, every video activity training example is decomposed into frames. Then, one frame (image) is subtracted from its next frame in order to obtain a frame difference. After obtaining it, the frame difference is subdivided into sub-patches with size 8. Figure 5 shows the top 64 patches that have the largest energy for each activity.

Then, a universal dictionary is created based on patches using a dictionary learning algorithm described in the previous section. Every time it takes a new example (frame difference patch) for one class, the existing dictionary is updated since only one dictionary is used for classification of all the classes. The universal dictionary that changes when it takes a new example is depicted in Figure 6. The change of dictionary which is conducted by learning a new example (patch) is observable in the figure every time a new activity is added to the dictionary. Also, the background color of dictionary turns into gray from black when the dictionary is updated.

7.3 Classification Accuracy

Now that the universal dictionary is created by learning each activity, a new video can be recognized in a one-shot learning manner.

The entire dataset has 9 videos per class and one of them is used for learning. Thus, the rest 8 videos per class are used for testing the accuracy of classification. Figure 7 shows the classification result of the system.

We make some observations from the test result. First, the classification accuracy of running and waving shows better performance than other two activities even though it learns the same number of example for each class – one example per class.

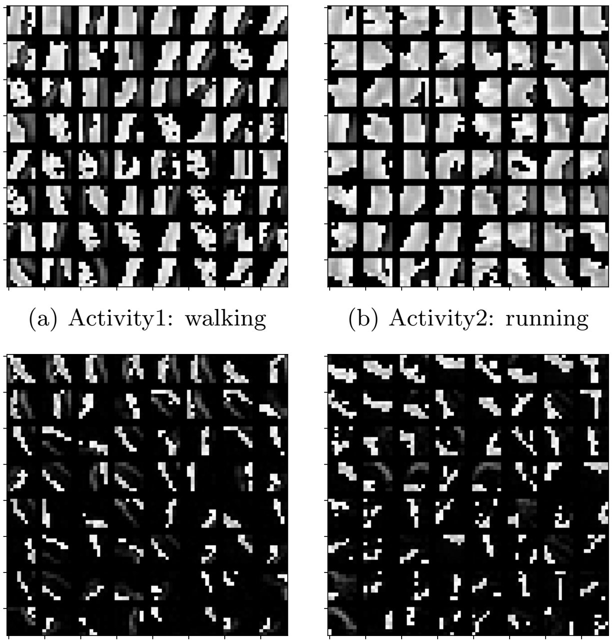
Next, the classification accuracy of walking and bending activity is relatively lower than other two activities. Walking activity show 55% of accuracy and bending activity shows 50% of accuracy.

For walking activity, most of the false classifications fall into running activity. One plausible explanation is that running activity has a larger set of patches that intersects with walking activity. The other way of false classification – running activity classified as the walking activity does not happen often since walking activity has not enough patches that are able to cover running patches.

Bending activity shows the lowest accuracy (50%). It has the smallest number of patches (76 patches) that have enough amount of energy. In light of a larger number of patches generated from other activities (average 400 patches), the chance of bending activity to be correctly classified might be influenced by the sheer volume of patches. Even though it is taken to be accounted in Algorithm 2 by applying Gaussian distance, it does not work well for this case.

8. CONCLUSION

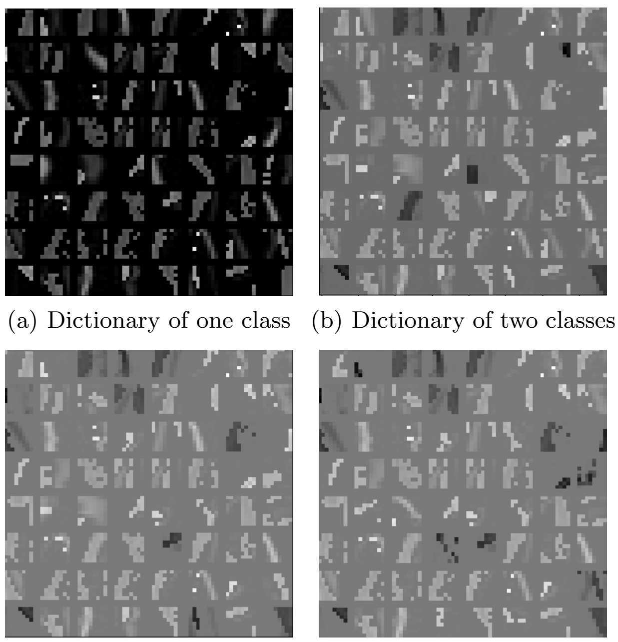
We introduced an one-shot learning algorithm for resource-constrained system based on sparse model. One-shot learning is a learning model trying to build a knowledge system by learning only one or a few examples. In order to achieve the goal of one-shot learning, the conventional sparse model is revised in our algorithm. With the sparse model revised for one-shot learning, it becomes able to build a reliable knowledge by leveraging a dictionary as a prior knowledge. It helps one-shot learning algorithm create more general and bias-free knowledge by letting it overcome the weakness of a small number of training examples. In order to make one-shot learning run on resource-constrained system, a



(a) Activity1: walking (b) Activity2: running

(c) Activity3: waving (d) Activity4: bending

Figure 5: The top 64 frame-difference patches of each activity. They are used to create and update an universal dictionary.



(a) Dictionary of one class (b) Dictionary of two classes

(c) Dictionary of three classes (d) Dictionary of four classes

Figure 6: Universal dictionary evolves every time it learns a new example (patch) for one class. (a) Dictionary after learning walking activity. (b) After walking and running. (c) After walking, running, and waving. (d) After walking, running, waving, and bending.

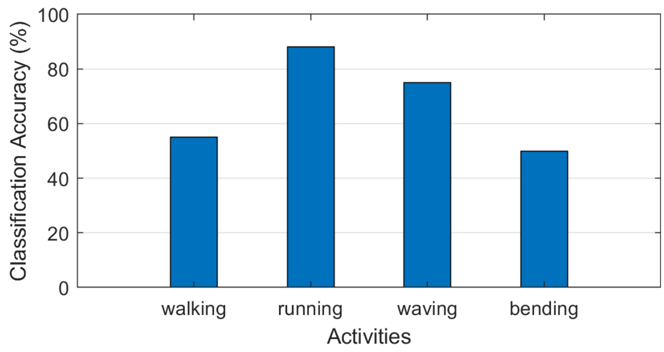


Figure 7: Test result of video activity classification.

new learning algorithm of sparse model which only requires one dictionary is devised. In this way, the traditional sparse model which requires heavy computation for a number of dictionary becomes available to such systems. Our implementation of human video activity classification for resource-constrained system shows that one-shot learning on resource-constrained system is feasible with the proposed algorithm.

9. REFERENCES

[1] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

[2] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[3] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[4] Di Wu, Fan Zhu, and Ling Shao. One shot learning gesture recognition from rgbd images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 7–12. IEEE, 2012.

[5] Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE, 2003.

[6] Lee H Dicker and Dean P Foster. One-shot learning and big data with n= 2. In *Advances in Neural Information Processing Systems*, pages 270–278, 2013.

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias

Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[8] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[9] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[10] Kenneth Yip and Gerald Jay Sussman. Sparse representations for fast, one-shot learning. 1997.

[11] Michal Aharon, Michael Elad, and Alfred Bruckstein. *rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[12] Chen Shaobing and D Donoho. Basis pursuit. In *28th Asilomar conf. Signals, Systems Computers*, 1994.

[13] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[14] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[15] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *The Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1395–1402, 2005.