在人工智能发展的路上,存在一个目前仍然保持开放的问题,就是需要开发出来一种可以增量学习的系统,随着时间的增长他可以学习到越来越多的概念.本文我们提出了一个新的训练策略,称为iCaRL.该训练册策略可以使得模型能够以类增的方式来进行学习,即每次学习的时候都只有一些类,而在后续的训练过程中会有新的类.iCaRL可以异步的学习到强大的分类器和数据的表征.也正是因为这种异步特性,先前所有的方法都是在固定的数据表示上学习的,因此我们的方法和别的深度学习的方法无法直接比较.我们最终在Cifar100和ImageNet2012上训练了我们的模型,证明了我们的模型随着时间的的过去,可以逐渐的学习到更多的类

iCaRL: Incremental Classifier and Representation Learning

CVPR2017

Sylvestre-Alvise Rebuffi
University of Oxford/IST Austria

Alexander Kolesnikov, Georg Sperl, Christoph H. Lampert
IST Austria

idea:看看每个task之后embedding的分布的变化(需要一些降维算法)

缺点:没有研究size的影响,Cifar100中k=2000,可是一共就50000个数据.每个类大概400多个样本,所以前面几个任务差不多类似于联合学习了看后面实验部分的质疑

Abstract

A major open problem on the road to artificial intelligence is the development of incrementally learning systems that learn about more and more concepts over time from a stream of data. In this work, we introduce a new training strategy, iCaRL, that allows learning in such a class-incremental way: only the training data for a small number of classes has to be present at the same time and new classes can be added progressively.

iCaRL learns strong classifiers and a data representation simultaneously. This distinguishes it from earlier works that were fundamentally limited to fixed data representations and therefore incompatible with deep learning architectures. We show by experiments on CIFAR-100 and ImageNet ILSVRC 2012 data that iCaRL can learn many classes incrementally over a long period of time where other strategies quickly fail.

自然中的视觉信息在本质上连续不断的,并且多种类别信息是混合的,在混合的类中可能包含了先前的知识.例如小孩在参观动物园的时候会学习到很多新的动物,同时也不会忘记自己家里的宠物.然而绝大多数是人工物体识别系统都是在batch设置下训练,这一设置要求所有的类在训练前都是已知的,并且所有类的训练数据都可以在同一时间以任意的顺序访问

1. Introduction

Natural vision systems are inherently incremental: new visual information is continuously incorporated while existing knowledge is preserved. For example, a child visiting the zoo will learn about many new animals without forgetting the pet it has at home. In contrast, most artificial object recognition systems can only be trained in a batch setting, where all object classes are known in advance and they the training data of all classes can be accessed at the same time and in arbitrary order.

随着计算机视觉领域不断地向人工智能方向发展,越来越多的训练方案被提出来处理真实世界物体分类问题中的大规模和灵活多变的特性.

As the field of computer vision moves closer towards artificial intelligence it becomes apparent that more flexible strategies are required to handle the large-scale and dynamic properties of real-world object categorization situations. At the very least, a visual object classification system should be able to incrementally learn about new classes, when training data for them becomes available. We call this scenario *class-incremental learning*.

Formally, we demand the following three properties of an algorithm to qualify as class-incremental:

i) it should be trainable from a stream of data in which examples of different classes occur at different times,

真实世界中的物体分类问题中的一个最基本的问题就是当新的类的example被观测到之后,视觉物体分类系统应该能够学习到这个新的类.我们称这种场景为类增学习

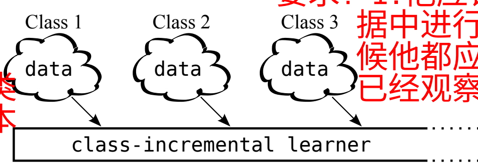


Figure 1: Class-incremental learning: an algorithm learns continuously from a sequential data stream in which new classes occur. At any time, the learner is able to perform multi-class classification for all classes observed so far.

对于一个能够进行类增学习的算法,我们认为它需要满足如下三个要求: 1.他应该能够从流式的数据中进行学习 2.在任何时候他都应该能够提供对所有已经观察到的类的分类结果 3.计算开销和内存消耗应该在合理的范围内,或者说随着见过的类越来越多,计算开销增长的比较缓慢

ii) it should at any time provide a competitive multi-class classifier for the classes observed so far,

iii) its computational requirements and memory footprint should remain bounded, or at least grow very slowly, with respect to the number of classes seen so far.

前两个评判标准表达了类增学习的本质,而第三个标准则是避免了一些没有价值的算法,例如把所有见过的数据全部储存下来

The first two criteria express the essence of class-incremental learning. The third criterion prevents trivial algorithms, such as storing all training examples and retraining an ordinary multi-class classifier whenever new data becomes available.

有趣的是,经过图像分类任务在过去的十年间有了很大的发展,但是目前还没有一个令人满意的类增学习方法.绝大多数方法不满足一或者二.

Interestingly, despite the vast progress that image classification has made over the last decades, there is not a single satisfactory class-incremental learning algorithm these days. Most existing multi-class techniques simply violate *i)* or *ii)* as they can only handle a fixed number of classes and/or need all training data to be available at the same time. Naively, one could try to overcome this by training classifiers from class-incremental data streams, *e.g.* using stochastic gradient descent optimization. This, however, will cause the classification accuracy to quickly deteriorate, an effect known in the literature as *catastrophic forgetting* or *catastrophic interference* [22]. The few existing techniques that do fulfill the above properties are principally limited to situations with a fixed data representation. They cannot be extended to deep architectures that learn classifiers and feature representations at the same time and are therefore not competitive anymore in terms of classification accuracy. More related work is discussed in Section 3.

如果对原模型或者算法不做任何修改,让模型直接在流式数据上SGD的话.就会导致灾难性遗忘问题.目前的确有一些算法来解决这个问题,但是无法被运用到深度网络上,因此对于提升分类准确性来说没有竞争力

In this work, we introduce *iCaRL (incremental classifier and representation learning)*, a practical strategy for simultaneously learning classifiers and a feature representation in the class-incremental setting. Based on a careful analysis of

本文我们提出了iCaRL,它是一种能够在类增设置下具有可以实际运用的能异步学习的分类器和特征表示

在对现有的方法分析的基础上,我们提出了三个主要的组件,这三个组件结合在一起使得iCaRL能够达到前面要求的三个的标准.在Section2中我们详细介绍了我们的方法,Section3中我们和先前方法进行了比较,在Section4中我们介绍了在Cifar100和ImageNet上的实验.最后在Section5中我们对存在的限制和未来可能进行的工作进行了讨论

Algorithm 1 iCaRL CLASSIFY

input x // image to be classified
require $\mathcal{P} = (P_1, \dots, P_t)$ // class exemplar sets
require $\varphi : \mathcal{X} \to \mathbb{R}^d$ // feature map
  for $y = 1, \dots, t$ **do**
    $\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$ // mean-of-exemplars
  end for
  $y^* \leftarrow \operatorname{argmin}_{y=1,\dots,t} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label y^*

预测代码,解决了类增最后一层长度的问题

这里w是向量,使用w只是用于训练,一个类一个w

Algorithm 2 iCaRL INCREMENTALTRAIN

input $X^s, \dots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets
  $\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \dots, X^t; \mathcal{P}, \Theta)$
  $m \leftarrow K/t$ // number of exemplars per class
  for $y = 1, \dots, s-1$ **do**
    $P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
  end for
  for $y = s, \dots, t$ **do**
    $P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
  end for
  $\mathcal{P} \leftarrow (P_1, \dots, P_t)$ // new exemplar sets

使用iCaRL来实现类增式分类包含三步: 1.使用NME规则来进行分类 2.基于herding对exemplar分配优先级 3.使用知识蒸馏和排练来实现表示学习

the shortcomings of existing approaches, we introduce three main components that in combination allow iCaRL to fulfill all criteria put forth above. These three components are:

- classification by a *nearest-mean-of-exemplars* rule,
- *prioritized exemplar selection* based on herding,
- representation learning using *knowledge distillation* and *prototype rehearsal*.

We explain the details of these steps in Section 2, and subsequently put them into the context of previous work in Section 3. In Section 4 we report on experiments on the CIFAR and ImageNet datasets that show that iCaRL is able to class-incrementally learn over a long periods of time, where other methods quickly fail. Finally, we conclude in Section 5 with a discussion of remaining limitations and future work.

本节我们描述了iCaRL的主要组件,并且解释了为什么这些组件的结合能够实现真正的类增学习

2. Method

In this section we describe iCaRL's main components and explain how their combination allows true class-incremental learning. Section 2.1 explains the underlying architecture and gives a high-level overview of the training and classification steps. Sections 2.2 to 2.4 then provides the algorithmic details and explains the design choices.

2.1节给出了类增学习的形式化描述,并且给出了iCaRL的高度概况的描述.2.2到2.4节给出了算法的细节,并且解释了为什么这样设计

iCaRL以类增的形式从流式数据中心异步的学习分类器和特征表示.用X^i表示属于第i类的数据,x^i_k表示第i类数据中的第k个example

对于分类任务来说,iCaRL依赖于从流式数据中动态选择的examplar图像来进行分类.对于每一个观察到的类,iCaRL都会维持一个Examplar Set,并且保证所有的examplar set中最多只有K个图像

2.1. Class-Incremental Classifier Learning

iCaRL learns classifiers and a feature representation simultaneously from on a data stream in class-incremental form, *i.e.* sample sets $X^1, X^2, \dots$, where all examples of a set $X^y = \{x_1^y, \dots, x_{n_y}^y\}$ are of class $y \in \mathbb{N}$.

Classification. For classification, iCaRL relies on sets, $P_1, \dots, P_t$, of *exemplar images* that it selects dynamically out of the data stream. There is one such exemplar set for each observed class so far, and iCaRL ensures that the total number of exemplar images never exceeds a fixed parameter K. Algorithm 1 describes the mean-of-exemplars classifier that is used to classify images into the set of classes observed so far, see Section 2.2 for a detailed explanation.

iCaRL每次使用类增学习策略处理一批数据.每次有新的类别的数据到来时候,iCaRL就会调用更新程序

Training. For training, iCaRL processes batches of classes at a time using an incremental learning strategy. Every time data for new classes is available iCaRL calls an update routine (Algorithm 2, see Sections 2.3 and 2.4). The routine adjusts iCaRL's *internal knowledge* (the network parameters and exemplars) based on the additional information available in the *new observations* (the current training data). This is also how iCaRL learns about the existence of new classes.

iCaRL中把网络视为一个特征提取器,即从X到R^d的一个函数.最后一层会使用一个sigmoid函数来输出所有观测到的类

Architecture. Under the hood, iCaRL makes use of a convolutional neural network (CNN) [19][1]. We interpret the network as a *trainable feature extractor*, $\varphi : \mathcal{X} \to \mathbb{R}^d$, followed by a single classification layer with as many sigmoid output nodes as classes observed so far [3]. All feature vectors are L^2-normalized, and the results of any operation on feature vectors, *e.g.* averages, are also re-normalized, which we do not write explicitly to avoid a cluttered notation.

此外,所有特征向量都经过了L2正则化,并且经过处理之后的特征向量重新进行了正则化,在符号中我们没有表示出来,是为了避免过于复杂的符号

We denote the parameters of the network by Θ, split into a fixed number of parameters for the feature extraction part and a variable number of weight vectors. We denote the latter by $w_1, \dots, w_t \in \mathbb{R}^d$, where here and in the following sections we use the convention that t denotes the number of classes that have been observed so far. The resulting network outputs are, for any class $y \in \{1, \dots, t\}$,

$$g_y(x) = \frac{1}{1 + \exp(-a_y(x))} \quad \text{with} \quad a_y(x) = w_y^\top \varphi(x). \quad (1)$$

我们用\theta表示网络的参数,网络的参数又可以分为两个部分,第一个部分是固定的特征提取部分,第二个是可变的权重向量.对于每一个类来说,网络的输出如下

Note that even though one can interpret these outputs as probabilities, iCaRL uses the network only for representation learning, not for the actual classification step.

Resource usage. Due to its incremental nature, iCaRL does not need a priori information about which and how many classes will occur, and it can –in theory– run for an unlimited amount of time. At any time during its runtime its memory requirement will be the size of the feature extraction parameters, the storage of K exemplar images and as many weight vectors as classes that have been observed. This knowledge allows us to assign resources depending on

[1] In principle, the iCaRL strategy is largely architecture agnostic and could be use on top of other feature or metric learning strategies. Here, we discuss it only in the context of CNNs to avoid an overly general notation.

因为iCaLR是逐渐类增的,因此iCaLR并不需要提前知道到底一共有多少个类,事实上其可以无限的运行下去.算法在运行期间其需要的内存包含三部分: 网络的参数,存储的K个examplar图像和t个观察到的权重向量

我们这样做的好处就在于如果所有的类别都是已知的，那么我们就可以提前为权重向量分配好内存，然后用剩下的所有内存存储examplar图像.如果是未知的话，那么我们可以在增加分配权重的时候去减少examplar图像的数量，从而保持内存的稳定

算法三是训练代码，注意损失函数分两部分，对于新的类的样本使用交叉熵，对于旧的类使用蒸馏损失

虽然我们说iCaLR可以无穷的学习下去，但是每类图像的examplar set中都至少要有一张图像，因此我们最终还是只能够学习有限的类别

the application scenario. If an upper bound on the number of classes is known, one can simply pre-allocate space for as many weight vectors as required and use all remaining available memory to store exemplars. Without an upper limit, one would actually grow the number of weight vectors over time, and decrease the size of the exemplar set accordingly. Clearly, at least one exemplar image and weight vector is required for each classes to be learned, so ultimately, only a finite number of classes can be learned, unless one allows for the possibility to add more resources over the runtime of the algorithm. Note that iCaRL can handle an increase of resources on-the-fly without retraining: it will simply not discard any exemplars unless it is forced to do so by memory limitations.

2.2. Nearest-Mean-of-Exemplars Classification

iCaLR用NME来进行分类.结果其实就是所有的Examplar算一个均值然后求一个最近的类作为预测

iCaRL uses a *nearest-mean-of-exemplars* classification strategy. To predict a label, y^*, for a new image, x, it computes a prototype vector for each class observed so far, $\mu_1, \dots, \mu_t$, where $\mu_y = \frac{1}{|P_y|}\sum_{p \in P_y} \varphi(p)$ is the average feature vector of all exemplars for a class y. It also computes the feature vector of the image that should be classified and assigns the class label with most similar prototype:

之所以使用最近距离来作为分类的方法而不是使用传统分类的对结果的softmax之后结果的argmax因为传统的分类网络输出的类都是已知的，因此对一个输入的example输出是一个固定长度的向量.

$$y^* = \operatorname*{argmin}_{y=1,\dots,t} \|\varphi(x) - \mu_y\|. \qquad (2)$$

Background. The nearest-mean-of-exemplars classification rule overcomes two major problems of the incremental learning setting, as can be seen by contrasting it against other possibilities for multi-class classification.

The usual classification rule for a neural network would be $y^* = \operatorname{argmax}_{y=1,\dots,t} g_y(x)$, where $g_y(x)$ is the network output as defined in (1) or alternatively with a softmax output layer. Because $\operatorname{argmax}_y g_y(x) = \operatorname{argmax}_y w_y^\top \varphi(x)$, the network's prediction rule is equivalent to the use of a linear classifier with non-linear feature map φ and weight vectors $w_1, \dots, w_t$. In the class-incremental setting, it is problematic that the weight vectors w_y are *decoupled* from the feature extraction routine φ: whenever φ changes, all $w_1, \dots, w_t$ must be updated as well. Otherwise, the network outputs will change uncontrollably, which is observable as catastrophic forgetting. In contrast, the nearest-mean-of-exemplars rule (2) does not have decoupled weight vectors. The class-prototypes automatically change whenever the feature representation changes, making the classifier robust against changes of the feature representation.

但是因为在类增的设置下，新增类之后输出的向量的长度就不一样所以权重向量和特征提取器是解耦的因此如果

The choice of the average vector as prototype is inspired by the *nearest-class-mean* classifier [24] for incremental learning with a fixed feature representation. In the class-incremental setting, we cannot make use of the true class mean, since all training data would have to be stored in order to recompute this quantity after a representation change. Instead, we use the average over a flexible number of exemplars that are chosen in a way to provide a good approximation to the class mean.

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

对于每一个类，计算该类每一个example的q^y，然后当做label

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) + \sum_{y=1}^{s-1} \boxed{q_i^y \log g_y(x_i) + (1-q_i^y)\log(1-g_y(x_i))} \Big]$$

这个式子用Pytorch里的BCEloss有实现

that consists of *classification* and *distillation* terms.

Note that, because we work with normalized feature vectors, Equation (2) can be written equivalently as $y^* = \operatorname{argmax}_y \mu_y^\top \varphi(x)$. Therefore, we can also interpret the classification step as classification with a weight vector, but one that is not decoupled from the data representation but changes consistently with it.

2.3. Representation Learning

当新的数据来了之后，iCaRL首先会用新的数据和保存的Examplar set来构建一个增强的训练集.接下来会对所有的可用数据进行分类.并且原先的类的结果会被保存下来.接下来模型会最小化分类损失和蒸馏损失

Whenever iCaRL obtains data, $X^s, \dots, X^t$, for new classes, $s, \dots, t$, it updates its feature extraction routine and the exemplar set. Algorithm 3 lists the steps for incrementally improving the feature representation. First, iCaRL constructs an augmented training set consisting of the currently available training examples together with the stored exemplars. Next, the current network is evaluated for each example and the resulting network outputs for all previous classes are stored (not for the new classes, since the network has not been trained for these, yet). Finally, the network parameters are updated by minimizing a loss function that for each new image encourages the network to output the correct class indicator for new classes *(classification loss)*, and for old classes, to reproduce the scores stored in the previous step *(distillation loss)*.

Background. The representation learning step resembles ordinary network finetuning: starting from previously learned network weights it minimizes a loss function over a training set. As a consequence, standard end-to-end learning methods can be used, such as backpropagation with mini-batches, but also recent improvements, such as *dropout* [38], *adaptive stepsize selection* [14] or *batch nor-*

iCaLR的学习过程和一般的网络学习过程是一样的，在上次学习结束之后，这次学习的目的就是最小化训练集上的损失函数.因此标准的端到端的学习方法是可以直接用于iCaLR中的训练的.例如mini-batch反向传播，dropout，ass和bn

算法四就是先算新的类的中心,然后选取样本使得中心点尽可能的保持不变

Algorithm 4 iCaRL ConstructExemplarSet

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n}\sum_{x\in X}\varphi(x)$ // current class mean
  for $k = 1, \ldots, m$ **do**
    $p_k \leftarrow \underset{x\in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1}\varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \ldots, p_m)$
output exemplar set P

Algorithm 5 iCaRL ReduceExemplarSet

input m // target number of exemplars
input $P = (p_1, \ldots, p_{|P|})$ // current exemplar set
  $P \leftarrow (p_1, \ldots, p_m)$ // *i.e.* keep only first m
output exemplar set P

相比于单纯的finetune,iCaLR中进行了两处修改以避免灾难性遗忘.首先是iCaLR使用了增强的训练集.增强的训练集不仅包含了新的数据,还包含了过去类别的数据

通过加强训练集,算法确保至少先前类别的数据分布能够进入这一轮的训练

malization [13], as well as potential future improvements.

There are two modifications to plain finetuning that aim at preventing or at least mitigating catastrophic forgetting. First, the training set is augmented. It consists not only of the new training examples but also of the stored exemplars. By this it is ensured that at least some information about the data distribution of all previous classes enters the training process. Note that for this step it is important that the exemplars are stored as images, not in a feature representation that would become outdated over time. Second, the loss function is augmented as well. Besides the standard classification loss, which encourages improvements of the feature representation that allow classifying the newly observed classes well, it also contains the distillation loss, which ensures that the discriminative information learned previously is not lost during the new learning step.

除了增强的数据集以外,另外一个改进就是使用了增强的损失函数.我们使用了蒸馏损失.我们的损失函数中正常的损失函数这一项使得模型能够学习到对新类的分类,而蒸馏损失这一项则确保先前学到的知识在新的训练过程中得到保留

2.4. Exemplar Management

Whenever iCaRL encounters new classes it adjusts its exemplar set. All classes are treated equally in this, *i.e.*, when t classes have been observed so far and K is the total number of exemplars that can be stored, iCaRL will use $m = K/t$ exemplars (up to rounding) for each class. By this it is ensured that the available memory budget of K exemplars is always used to full extent, but never exceeded.

Two routines are responsible for exemplar management: one to select exemplars for new classes and one to reduce the sizes of the exemplar sets of previous classes. Algorithm 4 describes the exemplar selection step. Exemplars $p_1, \ldots, p_m$ are selected and stored iteratively until the target number, m, is met. In each step of the iteration, one more example of the current training set is added to the exemplar set, namely the one that causes the average feature vector over all exemplars to best approximate the average feature vector over all training examples. Thus, the exemplar "set"

对于Examplar的管理,主要包括两部分:1.构建新的Examplar Set.2.更新已有的Examplar Set.

新的Examplar的构建算法见算法4.每次迭代都会向examplar set中添加一个或者一些examplar,而添加的examplar则是使examplar set的平均特征向量和训练样本的平均特征向量最接近的examplar

从Examplar Set中删除样本很简单.就是取前面的几个就行

is really a prioritized list. The order of its elements matters, with exemplars earlier in the list being more important. The procedure for removing exemplars is specified in Algorithm 5. It is particularly simple: to reduce the number of exemplars from any m' to m, one discards the exemplars $p_{m+1}, \ldots, p_{m'}$, keeping only the examples $p_1, \ldots, p_m$.

Background. The exemplar management routines are designed with two objectives in mind: the initial exemplar set should approximate the class mean vector well, and it should be possible to remove exemplars at any time during the algorithm's runtime without violating this property.

设计Examplar管理的两个算法的目的就是:1.初始的Examplar Set应该和全部样本中心一直,2.在算法运行阶段要能够在任何时候移除一些Examplar

The latter property is challenging because the actual class mean vector is not available to the algorithm anymore when the removal procedure is called. Therefore, we adopt a data-independent removal strategy, removing elements in fixed order starting at the end, and we make it the responsibility of the exemplar set construction routine to make sure that the desired approximation properties are fulfilled even after the removal procedure is called at later times. The prioritized construction is the logical consequence of this condition: it ensures that the average feature vector over any subset of exemplars, starting at the first one, is a good approximation of the mean vector. The same prioritized construction is used in *herding* [39] to create a representative set of samples from a distribution. There it was also shown that the iterative selection requires fewer samples to achieve a high approximation quality than, *e.g.*, random subsampling. In contrast, other potential methods for exemplar selection, such as [7, 26], were designed with other objectives and are not guaranteed to provide a good approximation quality for any number of prototypes.

Overall, iCaRL's steps for exemplar selection and reduction fit exactly to the incremental learning setting: the selection step is required for each class only once, when it is first observed and its training data is available. At later times, only the reduction step is called, which does not need access to any earlier training data.

3. Related work

iCaRL builds on the insights of multiple earlier attempts to address class-incremental learning. In this section, we describe the most important ones, structuring them on the one hand into learning techniques with *fixed data representations* and on the other hand into techniques that also learn the data representation, both from the *classical connectionist* era as well as recent *deep learning* approaches.

Learning with a fixed data representation. When the data representation is fixed, the main challenge for class-incremental learning is to design a classifier architecture that can accommodate new classes at any time during the training process without requiring access to all training data seen so far.

Mensink *et al*. [23] observed that the *nearest class mean (NCM)* classifier has this property. NCM represents each class as a prototype vector that is the average feature vector of all examples observed for the class so far. This vector can be computed incrementally from a data stream, so there is no need to store all training examples. A new example is classified by assigning it the class label that has a prototype most similar to the example's feature vector, with respect to a metric that can also be learned from data. Despite (or because of) its simplicity, NCM has been shown to work well and be more robust than standard parametric classifiers in an incremental learning setting [23, 24, 31].

NCM's main shortcoming is that it cannot easily be extended to the situation in which a nonlinear data representation should be learned together with the classifiers, as this prevents the class mean vectors from being computable in an incremental way. For iCaRL we adopt from NCM the idea of prototype-based classification. However, the prototypes we use are not the average features vectors over all examples but only over a specifically chosen subset, which allows us to keep a small memory footprint and perform all necessary updates with constant computational effort.

Alternative approaches fulfill the class-incremental learning criteria i)–iii), that we introduced in Section 1, only partially: Kuzborskij *et al*. [17] showed that a loss of accuracy can be avoided when adding new classes to an existing linear multi-class classifier, as long as the classifiers can be retrained from at least a small amount of data for all classes. Chen *et al*. [4, 5] and Divvala *et al*. [6] introduced systems that autonomously retrieve images from web resources and identifies relations between them, but they does not incrementally learn object classifiers. Royer and Lampert [33] adapt classifiers to a time-varying data stream but their method cannot handle newly appearing classes, while Pentina *et al*. [29] show that learning multiple tasks sequentially can beneficial, but for choosing the order the data for all tasks has to be available at the same time.

Li and Wechsler [20], Scheirer *et al*. [37], as well as Bendale and Boult [2] aimed at the related but distinct problem of *Open Set Recognition* in which test examples might come from other classes than the training examples seen so far. Polikar *et al*. [28, 30] introduced an ensemble based approach that can handle an increasing number of classes but needs training data for all classes to occur repeatedly. Zero-shot learning, as proposed by Lampert *et al*. [18], can classify examples of previously unseen classes, but it does not include a training step for those.

Representation learning. The recent success of (deep) neural networks can in large parts be attributed to their ability to learn not only classifiers but also suitable data representations [3, 21, 25, 36], at least in the standard batch setting. First attempts to learn data representations in an incremental fashion can already be found in the classic neural network literature, *e.g*. [1, 8, 9, 32]. In particular, in the late 1980s McCloskey *et al*. [22] described the problem of *catastrophic forgetting*, *i.e*. the phenomenon that training a neural network with new data causes it to overwrite (and thereby forget) what it has learned on previous data. However, these classical works were mainly in the context of connectionist memory networks, not classifiers, and the networks used were small and shallow by today's standards. Generally, the existing algorithms and architectural changes are unable to prevent catastrophic forgetting, see, for example, Moe-Helgesen *et al*.'s survey [27] for classical and Goodfellow *et al*.'s [10] for modern architectures, except in specific settings, such as Kirkpatrick *et al*.'s [15].

A major achievement of the early connectionist works, however, is that they identified the two main strategies of how catastrophic forgetting can be addressed: 1) by *freezing* parts of the network weights while at the same time *growing* the network in order to preserve the ability to learn, 2) by *rehearsal*, *i.e*. continuously stimulating the network not only with the most recent, but also with earlier data.

Recent works on incremental learning of neural networks have mainly followed the freeze/grow strategy, which however requires allocating more and more resources to the network over time and therefore violates principle iii) of our definition of class-incremental learning. For example, Xiao *et al*. [40] learn a tree-structured model that grows incrementally as more classes are observed. In the context of multi-task reinforcement learning, Rusu *et al*. [35] propose growing the networks by extending all layer horizontally.

For iCaRL, we adopt the principle of *rehearsal*: to update the model parameters for learning a representation, we use not only the training data for the currently available classes, but also the exemplars from earlier classes, which are available anyway as they are required for the prototype-based classification rule. Additionally, iCaRL also uses *distillation* to prevent that information in the network deteriorates too much over time. while Hinton *et al*. [12] originally proposed distillation to transfer information between different neural networks, in iCaRL, we use it within a single network between different time points. The same principle was recently proposed by Li and Hoiem [21] under the name of *Learning without Forgetting (LwF)* to incrementally train a single network for learning multiple tasks, *e.g*. multiple object recognition datasets. The main difference to the class-incremental multi-class situation lies in the prediction step: a multi-class learner has to pick one classifier that predicts correctly any of the observed classes. A multi-task (multi-dataset) leaner can make use of multiple classifiers, each being evaluated only on the data from its own dataset.

4. Experiments

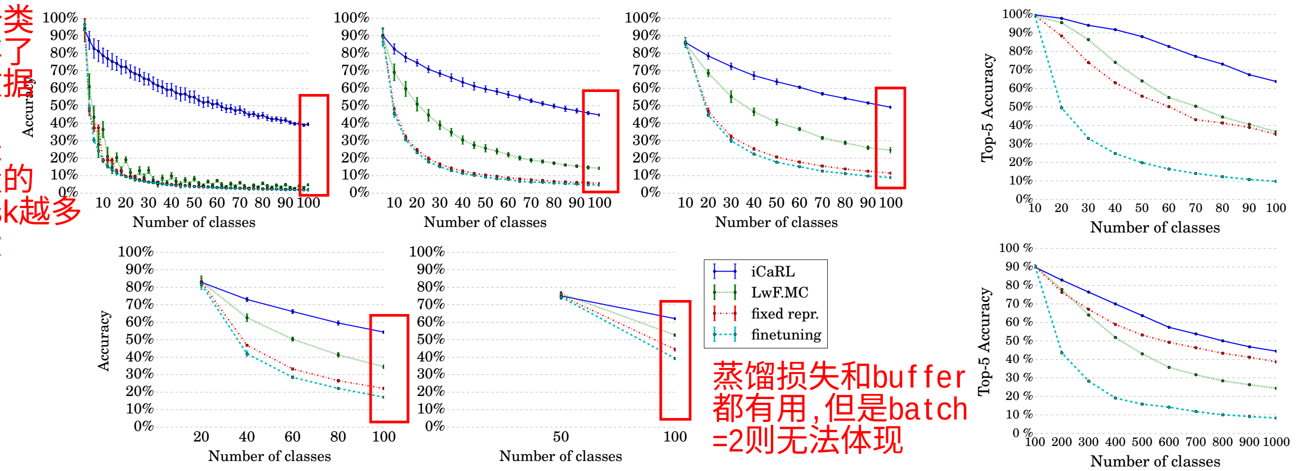
In this section we propose a protocol for evaluating incremental learning methods and compare iCaRL's classifi-

本节我们提出了衡量类增学习算法性能的评判标准，并且和其他的方法进行了比较.此外我们也通过消融实验探究了iCaLR每个组件的能力

这个图报告了多分类准确率,是重复了10次得到的所有准确率的平均和方差.每次分别学习2各类,5个类,10个类,20个类和50个类
每次学习完了之后报告在所有学过的类上的准确率

最后每个类都只保存了1/10的数据

还有就是task数量的影响,task越多遗忘越大



蒸馏损失和buffer都有用,但是batch =2则无法体现

(a) Multi-class accuracy (averages and standard deviations over 10 repeats) on iCIFAR-100 with 2 (top left), 5 (top middle), 10 (top right), 20 (bottom left) or 50 (bottom right) classes per batch.

(b) Top-5 accuracy on iILSVRC-small (top) and iILSVRC-full (bottom).

Figure 2: Experimental results of class-incremental training on iCIFAR-100 and iILSVRC: reported are multi-class accuracies across all classes observed up to a certain time point. iCaRL clearly outperforms the other methods in this setting. Fixing the data representation after having trained on the first batch (*fixed repr.*) performs worse than distillation-based LwF.MC, except for *iILSVRC-full*. Finetuning the network without preventing catastrophic forgetting (*finetuning*) achieves the worst results. For comparison, the same network trained with all data available achieves 68.6% multi-class accuracy.

因为目前位置并没有统一的类增学习的评价准则,因此本文我们提出了下面的评价方法:对于一个多分类数据集,不同的类以固定的随机顺序进行训练然后根据类的顺序进行训练,在每个类学习结束之后,在测试集上测试得到的分类器.注意,只会在已经学习过的类上进行测试.虽然测试集中的数据被多次使用但是因为模型没有学习测试数据,因此不存在数据泄露问题

cation accuracy to that of alternative methods (Section 4.1). We also report on further experiments that shed light on iCaRL's working mechanisms by isolating the effect of individual components (Section 4.2).

Benchmark protocol. So far, no agreed upon benchmark protocol for evaluation class-incremental learning methods exist. Therefore, we propose the following evaluation procedure: for a given multi-class classification dataset, the classes are arranged in a fixed random order. Each method is then trained in a class-incremental way on the available training data. After each batch of classes, the resulting classifier is evaluated on the test part data of the dataset, considering only those classes that have already been trained. Note that, even though the test data is used more than once, no overfitting can occur, as the testing results are not revealed to the algorithms. The result of the evaluation are curves of the classification accuracies after each batch of classes. If a single number is preferable, we report the average of these accuracies, called *average incremental accuracy*.

For the task of image classification we introduce two instantiations of the above protocol. 1) *iCIFAR-100 benchmark*: we use the CIFAR-100 [16] data and train all 100 classes in batches of 2, 5, 10, 20 or 50 classes at a time. The evaluation measure is the standard multi-class accuracy on the test set. As the dataset is of manageable size, we run this benchmark ten times with different class orders and reports averages and standard deviations of the results. 2) *iILSVRC benchmark*: we use the ImageNet ILSVRC 2012 [34] dataset in two settings: using only a subset of 100 classes, which are trained in batches of 10 (*iILSVRC-small*) or using all 1000 classes, processed in batches of 100 (*iILSVRC-full*). The evaluation measure is the *top-5* accuracy on the *val* part of the dataset.

在每一个类学习完之后会报告平均的分类准确度

iCaRL implementation. For iCIFAR-100 we rely on the *theano* package and train a 32-layers ResNet [11], allowing iCaRL to store up to $K = 2000$ exemplars. Each training step consists of 70 epochs. The learning rate starts at 2.0 and is divided by 5 after 49 and 63 epochs (7/10 and 9/10 of all epochs). For iILSVRC the maximal number of exemplars is $K = 20000$ and we use the *tensorflow* framework to train an 18-layers ResNet [11] for 60 epochs per class batch. The learning rate starts at 2.0 and is divided by 5 after 20, 30, 40 and 50 epochs (1/3, 1/2, 2/3 and 5/6 of all epochs). For all methods we train the network using standard backpropagation with minibatches of size 128 and a weight decay parameter of 0.00001. Our source code and further data are available at http://www.github.com/srebuffi/iCaRL.

每个task学习70个epoch,在49和63个epoch把学习了除以5

使用128的batch size,wd设置为1e-5

4.1. Results

Our main set of experiments studies the classification accuracy of different methods under class-incremental conditions. Besides iCaRL we implemented and tested three alternative class-incremental methods. *Finetuning* learns an ordinary multi-class network without taking any measures to prevent catastrophic forgetting. It can also be interpreted as learning a multi-class classifier for new incoming classes by finetuning the previously learned multiclass classifica-

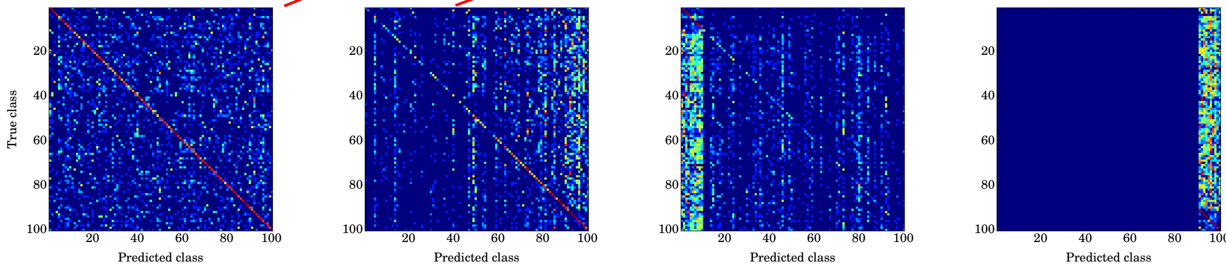
除了iCaLR以外,我们还是实现了Finetuning,Fixed Representation,仅蒸馏损失的iCaLR(LwF)

Finetuning的意思就是当新的task来了之后当做新的分类任务来直接学习.Fixed representation在第一个task学习结束之后就不在改变学习到的权重,只改变分类层的权重向量,即对于后续任务只会学习新的weight vector

错的很均匀

会把很多的类预测成新学的类

会把很多类预测称一开始的类



直接gg

(a) iCaRL (b) LwF.MC (c) fixed representation (d) finetuning

Figure 3: Confusion matrices of different method on iCIFAR-100 (with entries transformed by $\log(1+x)$ for better visibility). iCaRL's predictions are distributed close to uniformly over all classes, whereas LwF.MC tends to predict classes from recent batches more frequently. The classifier with fixed representation has a bias towards classes from the first batch, while the network trained by finetuning predicts exclusively classes labels from the last batch.

tion network. *Fixed representation* also learns a multi-class classification network, but in a way that prevents catastrophic forgetting. It freezes the feature representation after the first batch of classes has been processed and the weights of the classification layer after the corresponding classes have been processed. For subsequent batches of classes, only the weights vectors of new classes are trained. Finally, we also compare to a network classifier that attempts at preventing catastrophic forgetting by using the distillation loss during learning, like iCaRL does, but that does not use an exemplar set. For classification, it uses the network output values themselves. This is essentially the *Learning without Forgetting* approach, but applied to multi-class classification we, so denote it by LwF.MC.

Figure 2 shows the results. One can see that iCaRL clearly outperforms the other methods, and the more so the more incremental the setting is (*i.e.* the fewer classes can be processed at the same time). Among the other methods, *distillation*-based network training (LwF.MC) is always second best, except for *iILSVRC-full*, where it is better to fix the representation after the first batch of 100 classes. *Finetuning* always achieves the worst results, confirming that catastrophic forgetting is indeed a major problem for in class-incremental learning.

Figure 3 provides further insight into the behavior of the different methods. Is shows the confusion matrices of the 100-class classifier on iCIFAR-100 after training using batches of 10 classes at a time (larger versions can be found in the supplemental material). One can see very characteristic patterns: iCaRL's confusion matrix looks homogeneous over all classes, both in terms of the diagonal entries (*i.e.* correct predictions) as well as off-diagonal entries (*i.e.* mistakes). This shows that iCaRL has no intrinsic bias towards or against classes that it encounters early or late during learning. In particular, it does not suffer from catastrophic forgetting.

In contrast to this, the confusion matrices for the other classes show inhomogeneous patterns: *distillation*-based training (LwF.MC) has many more non-zero entries towards the right, *i.e.* for recently learned classes. Even more extreme is the effect for *finetuning*, where all predicted class labels come from the last batch of classes that the network has been trained with. The finetuned network simply *forgot* that earlier classes even exist. The *fixed representation* shows the opposite pattern: it prefers to output classes from the first batch of classes it was trained on (which were used to obtained the data representation). Confusion matrices for iILSVRC show the same patterns, they can be found in the supplemental material.

4.2. Differential Analysis

后面接着看,先看到这里

To provide further insight into the working mechanism of iCaRL, we performed additional experiments on iCIFAR-100, in which we isolate individual aspects of the methods.

First, we analyze why exactly iCaRL improves over plain finetuning-based training, from which it differs in three aspects: by the use of the mean-of-exemplars classification rule, by the use of exemplars during the representation learning, and by the use of the distillation loss. We therefore created three hybrid setups: the first *(hybrid1)* learns a representation in the same way as iCaRL, but uses the network's outputs directly for classification, not the mean-of-exemplar classifier. The second *(hybrid2)* uses the exemplars for classification, but does not use the distillation loss during training. The third *(hybrid3)* uses neither the distillation loss nor exemplars for classification, but it makes use of the exemplars during representation learning. For comparison, we also include LwF.MC again, which uses distillation, but no exemplars at all.

Table 1a summarizes the results as the average of the classification accuracies over all steps of the incremental training. One can see that the hybrid setups mostly achieve results in between iCaRL and LwF.MC, showing that indeed all of iCaRL's new components contribute substan-

Table 1: Average multi-class accuracy on iCIFAR-100 for different modifications of iCaRL.

(a) Switching off different components of iCaRL (*hybrid1*, *hybrid2*, *hybrid3*, see text for details) leads to results mostly inbetween iCaRL and LwF.MC, showing that all of iCaRL's new components contribute to its performance.

batch size	iCaRL	*hybrid1*	*hybrid2*	*hybrid3*	LwF.MC
2 classes	57.0	36.6	57.6	57.0	11.7
5 classes	61.2	50.9	57.9	56.7	32.6
10 classes	64.1	59.3	59.9	58.1	44.4
20 classes	67.2	65.6	63.2	60.5	54.4
50 classes	68.6	68.2	65.3	61.5	64.5

(b) Replacing iCaRL's mean-of-exemplars by a nearest-class-mean classifier (NCM) has only a small positive effect on the classification accuracy, showing that iCaRL's strategy for selecting exemplars is effective.

batch size	iCaRL	NCM
2 classes	57.0	59.3
5 classes	61.2	62.1
10 classes	64.1	64.5
20 classes	67.2	67.5
50 classes	68.6	68.7

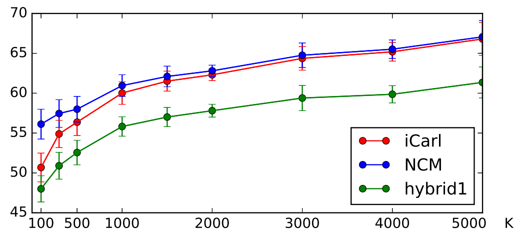


Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K.

tially to its good performance. In particular, the comparison of iCaRL with *hybrid1* shows that the mean-of-exemplar classifiers is particularly advantageous for smaller batch sizes, *i.e.* when more updates of the representation are performed. Comparing iCaRL and *hybrid2* one sees that for very small class batch sizes, distillation can even hurt classification accuracy compared to just using prototypes. For larger batch sizes and fewer updates, the use of the distillation loss is clearly advantageous. Finally, comparing the result of *hybrid3* with LwF.MC clearly shows the effectiveness of exemplars in preventing catastrophic forgetting.

In a second set of experiments we study how much accuracy is lost by using the means-of-exemplars as classification prototypes instead of the nearest-class-mean *(NCM)* rule. For the latter, we use the unmodified iCaRL to learn a representation, but we classify images with NCM, where the class-means are recomputed after each representation update using the current feature extractor. Note that this requires storing all training data, so it would not qualify as a class-incremental method. The results in Table 1b show only minor differences between iCaRL and NCM, confirming that iCaRL reliably identifies representative exemplars.

Figure 4 illustrates the effect of different memory budgets, comparing iCaRL with the *hybrid1* classifier of Table 1a and the NCM classifier of Table 1b. Both use the same data representation as iCaRL but differ in their classification rules. All method benefit from a larger memory budget, showing that iCaRL's representation learning step indeed benefits from more prototypes. Given enough prototypes (here at least 1000), iCaRL's mean-of-exemplars classifier performs similarly to the NCM classifier, while classifying by the network outputs is not competitive.

本文中我们提出了iCaLR,一种用于类增学习的策略.他可以让模型异步的学习特征表示和分类器.

5. Conclusion

iCaLR中三个主要的组件是:1.MNE分类器,2.基于herding的examplar选择算法,3.使用蒸馏损失来避免以往的表示学习方法

We introduced iCaRL, a strategy for class-incremental learning that learns classifiers and a feature representation simultaneously. iCaRL's three main components are: 1) a *nearest-mean-of-exemplars* classifier that is robust against changes in the data representation while needing to store only a small number of exemplars per class, 2) a *herding*-based step for prioritized exemplar selection, and 3) a representation learning step that uses the exemplars in combination with *distillation* to avoid catastrophic forgetting. Experiments on CIFAR-100 and ImageNet ILSVRC 2012 data show that iCaRL is able to learn incrementally over a long period of time where other methods fail quickly.

最终在Cifar100和ImageNet上的实验表明iCaLR可以在长时间内避免遗忘

The main reason for iCaRL's strong classification results are its use of exemplar images. While it is intuitive that being able to rely on stored exemplars in addition to the network parameters could be beneficial, we nevertheless find it an important observation how pronounced this effect is in the class-incremental setting. We therefore hypothesize that also other architectures should be able to benefit from using a combination of network parameters and exemplars, especially given the fact that many thousands of images can be stored (in compressed form) with memory requirements comparable to the sizes of current deep networks.

iCaLR能够在CL下拥有强劲表现的一个原因就是他存储了Examplar Set.虽然直觉上存储过去的数据肯定会对CL有帮助,但是我们从实验上有力的说明了过去的数据是多么的重要.因为在iCaLR上Examplar Set确实有所帮助,因此我们认为其他的网络结构应该也会有所获益

Despite the promising results, class-incremental classification is far from solved. In particular, iCaRL's performance is still lower than what systems achieve when trained in a batch setting, *i.e.* with all training examples of all classes available at the same time. In future work we plan to analyze the reasons for this in more detail with the goal of closing the remaining performance gap. We also plan to study related scenarios in which the classifier cannot store any of the training data in raw form, *e.g.* for privacy reasons.

Acknowledgments. This work was in parts funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 308036: "Life-long learning of visual scene understanding" (L3ViSU). The Tesla K40 cards used for this research were donated by the NVIDIA Corporation.

虽然iCaLR具有令人振奋的结果,但是相比与一个batch的设置,iCaLR学习的性能还是不行,因此未来的工作一方面将关注于如何弥补这一差距,另外一方面将会关注于在一个场景下无法保存原始训练数据(例如隐私问题)时如何进行CL

References

[1] B. Ans and S. Rousset. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences*, 320(12), 1997. 5

[2] A. Bendale and T. Boult. Towards open world recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(8), 2013. 2, 5

[4] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *International Conference on Computer Vision (ICCV)*, 2013. 5

[5] X. Chen, A. Shrivastava, and A. Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[6] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[7] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(11):2765–2781, 2013. 4

[8] R. M. French. Catastrophic interference in connectionist networks: Can it be predicted, can it be prevented? In *Conference on Neural Information Processing Systems (NIPS)*, 1993. 5

[9] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4), 1999. 5

[10] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 5

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 6

[12] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Workshop on Deep Learning*, 2014. 5

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learing (ICML)*, 2015. 4

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2017. 5

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[17] I. Kuzborskij, F. Orabona, and B. Caputo. From n to $n+1$: Multiclass transfer incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 5

[18] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2013. 5

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998. 2

[20] F. Li and H. Wechsler. Open set face recognition using transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 27(11), 2005. 5

[21] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[22] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989. 1, 5

[23] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision (ECCV)*, 2012. 5

[24] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(11), 2013. 3, 5

[25] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[26] I. Misra, A. Shrivastava, and M. Hebert. Data-driven exemplar model selection. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 339–346, 2014. 4

[27] O.-M. Moe-Helgesen and H. Stranden. Catastophic forgetting in neural networks. Technical report, Norwegian University of Science and Technology (NTNU), 2005. 5

[28] M. D. Muhlbaier, A. Topalis, and R. Polikar. Learn^{++}.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks (T-NN)*, 20(1), 2009. 5

[29] A. Pentina, V. Sharmanska, and C. H. Lampert. Curriculum learning of multiple tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5

[30] R. Polikar, L. Upda, S. S. Upda, and V. Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4), 2001. 5

[31] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool. Incremental learning of NCM forests for large-scale image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[32] A. V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. 5

[33] A. Royer and C. H. Lampert. Classifier adaptation at prediction time. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015. 6

[35] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 5

[36] S. Saxena and J. Verbeek. Convolutional neural fabrics. In *Conference on Neural Information Processing Systems (NIPS)*, 2016. 5

[37] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36, 2013. 5

[38] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 2014. 3

[39] M. Welling. Herding dynamical weights to learn. In *International Conference on Machine Learing (ICML)*, 2009. 4

[40] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *International Conference on Multimedia (ACM MM)*, 2014. 5