FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCE

SCALABLE LEARNING WITH PROBABILISTIC PCA

By

BOSHI WANG

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2023

Boshi Wang defended this dissertation on April 10, 2023.
The members of the supervisory committee were:

Adrian Barbu
Professor of Directing Dissertation

Xiuwen Liu
University Representative

Wei Wu
Committee Member

Chao Huang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

This is for my girlfriend Yunwei Wang and my parents

ACKNOWLEDGMENTS

I would first like to thank my advisor, Dr. Adrian Barbu, for his patient guidance on my research. This dissertation would not be completed without his support and advice. I also would like to thank my committee members: Dr. Xiuwen Liu, Dr. Wei Wu, Dr. Chao Huang for their valuable advice and support. Thank you all for helping me complete this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following short list of symbols is used throughout the document.

I	the image intensity
τ	Incremental Learning Problem
T	A Incremental Task
f	Feature Extractor
r	Classifier
Ω	The space of Input Images
$\mathbf{x}$	A extracted feature
$\boldsymbol{\mu}$	The mean vector of a PPCA neuron
$\boldsymbol{\Sigma}$	The covariance matrix of a PPCA neuron
$\mathbf{L}$	The first q eigenvector of SVD decomposition
$\mathbf{D}$	The diagonal matrix with first q eigenvalue
q	The dimension of the linear subspace
K	The total number of image classes
S	The total number of super classes
K_s	A partition of image classes

LIST OF ABBREVIATIONS

PPCA	Probabilistic Principal Components
LWF	Learning without forgetting
iCarl	Classifier and Representation Learning
RBF	Radial basis function

ABSTRACT

Deep neural networks have drawn much attention due to their success in various vision tasks. Class Incremental Leaning is a paradigm where instances from new object classes are added sequentially. Traditional training schemes cause a problem called catastrophic forgetting where the updated model forgets the old classes and focuses only on the new classes. In this dissertation, we propose a framework called incremental PPCA for class incremental learning. Incremental PPCA uses a self-supervised pre-trained feature extractor to obtain meaningful features and trains Probabilistic PCA models on the extracted features for each class separately. The Mahalanobis distance is used to obtain the classification result, and an equivalent equation is derived to make the approach computationally affordable. Experiments on standard and large datasets show that the proposed approach outperforms existing state-of-the-art incremental learning methods by a large margin. The fact that the model is trained on each class separately makes it applicable to training on very large datasets such as the whole ImageNet with more than 10,000 classes. Traditional image classifiers adopt neural network-based structures that treat all the image classes equally. While human can exploit semantic information between image classes and learn with much less samples. Inspired by our understanding of the human hierarchical cognition models, we propose a framework called Hierarchical PPCA for image classification. The framework uses probabilistic PCA models as basic classification units and adopts a modified k-means clustering to group the image classes into a smaller number of super-classes. During classification, Hierarchical PPCA assigns a sample to a small number of most likely super-classes, and restricts the image classification to the image classes corresponding to these super-classes. Experiments on three standard datasets ImageNet-100, ImageNet-1k and ImageNet-10k indicate the hierarchical classifier can achieve the same or superior accuracy with a 4-16 times speedup compared to a standard classifier.

CHAPTER 1

INTRODUCTION

Class incremental learning is a type of machine learning technique that allows a model to continuously learn and adapt to new classes or categories without having to retrain the entire model. This approach is useful when new classes are being added to the dataset over time, or when the number of classes is so large that training on all of them at once is not feasible. Class incremental learning can be defined as the process of adding new classes to a model without losing the knowledge of previously learned classes. In other words, the model is able to learn new classes while retaining the information it has learned about previous classes. Instead of training models from scratch every time new data becomes available, incremental learning allows the model to learn from new data while retaining the knowledge it has gained from previous data.

The traditional fine-tuning methods encounter a problem called catastrophic forgetting where models trained with the new classes have a drastic drop in classification performance on old classes. Learning without forgetting (LWF) (Li and Hoiem, 2017) proposed to use the distillation loss to generate consistent features. Classifier and Representation Learning (iCaRL) (Rebuffi et al., 2017) use task-specific parameters to preserve the knowledge of old classes.

In this dissertation we propose a simple framework called Hierarchical PPCA for incremental class learning. We adopt the pre-trained feature extractor and classifiers constructed by Probabilistic Principal Components (PPCA) (Tipping and Bishop, 1999) to mitigate catastrophic forgetting to a large extent. The Incremental PPCA method is introduced in the first part of the dissertation. The Incremental PPCA surpasses the previous state of art methods on class-incremental learning tasks.

Hierarchical classification is a type of machine learning approach that involves organizing classes or categories in a hierarchical tree structure. This allows for more complex relationships between classes to be modeled and can lead to more accurate and interpretable classification results. For example, a classification task involving different types of animals could be organized into a tree structure, with "mammals" as the parent category and "cats" and "dogs" as child categories. This allows the model to take into account the fact that cats and dogs are more closely related to

each other than to other mammals. By organizing classes into a tree structure, hierarchical classification can model parent-child relationships between classes. Traditional hierarchical clustering methods such as agglomerative clustering are not applicable to large-scale datasets due to their time complexity of $O(n^3)$.

In the second part of the dissertation, we propose a classification framework called Hierarchical PPCA. It models the classifier with a 2-level taxonomy. A k-means clustering method is adopted to exploit the semantic information of image classes. The Hierarchical PPCA obtains equal or superior classification performance with fewer computation resources on 3 large-scale datasets.

CHAPTER 2

SCALABLE LEARNING WITH INCREMENTAL PROBABILISTIC PCA

2.1 Introduction

Incremental learning is a learning paradigm that allows a model to be updated to handle new tasks (e.g. new classes in classification) by being trained on new data without using the whole dataset. This paradigm facilitates training models on large-scale datasets with less computational cost and memory requirements. A conventional approach to incremental learning is to fine-tune a pre-trained model on the new data. The fine-tuning approach suffers a serious problem called catastrophic forgetting, which means that the model trained on the new classes will have a drastic performance drop on the old classes.

Various approaches have been proposed to overcome catastrophic forgetting. Learning without forgetting (Lwf) (Li and Hoiem, 2017) introduces the distillation loss that encourages the feature extractor to generate consistent features for the old classes. This pioneering design became an essential component in subsequent approaches. Rebuffi et al. (2017) keeps exemplars of the old classes to preserve the knowledge of these classes. Despite all the efforts to mitigate the effect of catastrophic forgetting, the overall performance remains significantly inferior to those obtained by joint training on the entire dataset.

In this dissertation, we explore a new incremental learning strategy for multi-class image classification, using a standard pre-trained feature extractor and a novel method to classify the extracted features. Assuming that the feature extractor can obtain meaningful features that don't need retraining, there is still a major problem that needs to be addressed for conventional neural networks. The problem is that the conventional classifiers are based on standard projection-based neurons that have high responses on half of the feature space. For this reason, it is quite likely that instances from a new class will have high responses on an old class neuron, and this issue has to be mitigated by retraining that neuron.

The RBF neuron (Broomhead and Lowe, 1988) in principle alleviates this problem because an RBF neuron $f(\mathbf{x}) = g(\|\mathbf{x} - \boldsymbol{\mu}\|)$ will only have a high response for data near the neuron's center

$\boldsymbol{\mu}$. However, experiments will show that the RBF neuron is too simple to properly handle the complexities of real data in the feature space.

For this reason, we introduce a new type of neuron that is a refinement of the RBF neuron and is based on the Probabilistic PCA model. This neuron will have high responses close to a low dimensional subspace near a neuron center $\boldsymbol{\mu}$. Experiments will reveal that this representation is capable to handle the complexities of real data, obtaining state-of-the-art incremental learning results on standard datasets. It even allows us to obtain a reasonable classification result on ImageNet-10k, the subset of ImageNet (Deng et al., 2009) containing all the classes that have at least 450 training observations, for a total of 10,450 classes and 11 million observations.

Our Incremental PPCA innovates with:

1. We use a pretrained model to generate consistent features in the whole training process;
2. We introduce a PPCA-based classifier to model the complexity of classes instead of standard neurons;
3. We obtain an efficient computation equation for the PPCA score to make it applicable to classification on large datasets.

2.2 Related Work

There are two main types of approaches to class incremental learning: regularization-based approaches and bias correction approaches.

2.2.1 Regularization Based Approaches

To mitigate catastrophic forgetting, regularization methods apply the distillation loss as a regularization term along with the classification loss. This technique aims to encourage the feature extractor to generate consistent features. The distillation loss, introduced by Hinton et al. (2015), was originally used to encourage the outputs of a student network to approximate the outputs of the teacher network. In class incremental learning, the distillation loss is used as a penalty for changes in the features of old classes.

One pioneering work is Learning without Forgetting (Li and Hoiem, 2017) (LwF), which proposed to use the distillation loss to generate stable feature representations. When the model is introduced with a new task, LwF adds task-specific parameters to the model for the new task. Despite the fact that LwF was originally proposed for task incremental learning, the distillation loss

and task-specific parameters have become essential components in many class incremental learning approaches.

Compared to LwF, the Incremental Classifier and Representation Learning (iCaRL) (Rebuffi et al., 2017) method goes a step further to preserve the knowledge of old classes. iCarl also takes advantage of the distillation loss and task-specific parameters. Moreover, iCarl proposes to use a limited set of exemplars to store representative samples. iCaRL also builds a dynamic mechanism to update exemplars after each training stage.

Learning without memorizing (LwM) (Dhar et al., 2019) utilizes an attention-based method proposed by Zagoruyko and Komodakis (2016). In order to prevent catastrophic forgetting, the attention used by the network trained on the previous tasks should not change while training the new tasks. With this restriction, features of a certain class are expected to change less when the model is trained with new classes. Different from the previous approaches, LwM takes the gradient flow information into account. By combining the distillation loss (LD) and attention distillation loss (LAD), the attention map transfers the knowledge without requiring data from the base classes during training.

To avoid catastrophic forgetting, the proposed Incremental PPCA method adopts a different approach. Instead of using the distillation loss, Incremental PPCA uses a self-supervised feature extractor pretrained on a large dataset, which is frozen during the whole training process. This simple method makes sure that the feature extractor can generate consistent features across all learning tasks.

2.2.2 Bias Correction Approaches

Bias-correction methods aim to address the problem of task recency bias. The bias is caused by the fact that the model will see more examples from the new classes than the old classes.

Hou et al. (2019) reveal that the imbalance between old classes and new classes is the main challenge causing catastrophic forgetting. In their Learning a Unified Classifier Incrementally via Rebalancing (UCIR) method, they propose to replace the standard softmax layer with a cosine normalization layer.

Wu et al. (2019) discovered that the last fully connected layer of a CNN has a strong bias towards the new classes. They proposed a method called Bias Correction (BiC), which adds an additional layer to correct the task bias of the model. BiC divides the training process into two stages. The training data is split into a training set for the first stage and a validation set for the second

stage. The validation set is used to help estimate the bias in the FC layer. The earlier mentioned approach iCarl (Rebuffi et al., 2017) doesn't use a neural network-based classifier. Instead iCaRL uses the Nearest Mean Exemplar for classification. For each class, there will be a mean feature that is computed by averaging the feature representation of the exemplar images. The classification result is determined by the euclidean distance with the mean of the features of each class. This approach is therefore similar to the RBF neurons (which have responses based on learned centers $\boldsymbol{\mu}_i$) that are evaluated in our experiments. Our experiments reveal that a Euclidean distance-based approach is too simple to capture the complexities of real data, and the proposed PPCA approach based on Mahalanobis distance is better at capturing the intrinsic data variability and has good generalization.

To avoid the effect of catastrophic forgetting, the regularization-based methods focus on preserving the knowledge of old classes while bias correction methods focus on improving the classifier. In these approaches, the memory buffer to store the representative examples, the distillation loss-based regularization, and a learning system to balance old and new classes have become three essential components for class incremental learning.

2.2.3 Discussion

Different from previous approaches, the proposed method applies a pretrained feature extractor to obtain a consistent feature representation. Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1999) models are used to approximate the complexity of the extracted features for each class. The classification scores of the extracted feature vector of an image are related to the log-likelihoods of the PPCA distribution of the classes. The Gaussian distribution representing each image class remains the same when new classes are added, therefore catastrophic forgetting does not occur in this case.

2.3 Proposed Method

In this section, we describe the proposed Incremental PPCA method and explain how it facilitates Incremental Learning.

Section 2.3.1 gives a formal definition of Incremental Learning. Section 2.3.2 delivers an overview of the Incremental PPCA architecture. Section 2.3.3 explains how the representative features are extracted, Section 2.3.4 states the underlying assumption about the data and obtains the

Probabilistic PCA model. Section 2.3.5 details the training and Section 2.3.6 explains how the model can be sped-up for computation efficiency.

2.3.1 Problem Definition

More formally, an incremental learning problem τ consists of a sequence of T tasks:

$$\tau = [(C^1, D^1), (C^2, D^2), ...(C^T, D^T)] \tag{2.1}$$

where each task t is represented by a set of classes $C^t = \{c_1^t, c_2^t, ...c_{K_t}^t\}$ and training data D^t.

We consider the class incremental classification problems in which the classes C^t are disjoint for each task. Let $D^t = \{(I_1, y_1), (I_2, y_2), ...(I_{n_t}, y_{n_t})\}$, where I_k are the input images and $y_k \in C^t$ are the corresponding ground truth labels.

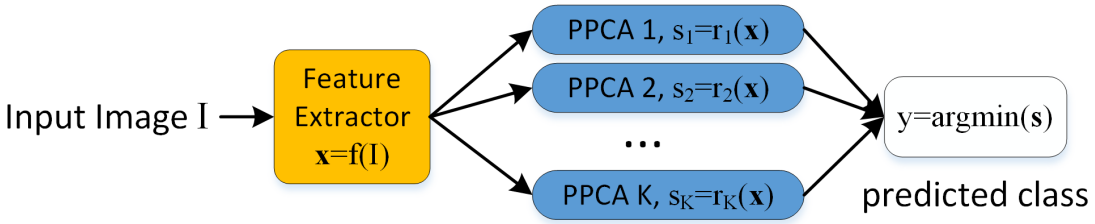


Figure 2.1: Diagram of the proposed Incremental PPCA method.

2.3.2 Incremental PPCA Architecture

As shown in Figure 2.1, the Incremental PPCA method can be written as $y = \text{argmin}_k\, r_k(f(\mathbf{I}))$, consisting of two main components: a feature extractor f and a classifier r. We interpret the convolutional part of a CNN model as a feature extractor $f : \Omega \to \mathbb{R}^d$, where Ω is the space of input images. Incremental PPCA takes as a feature extractor a frozen convolutional section of a model pretrained on a large-scale dataset. The underlying design and choice of feature extraction are detailed in Section 2.3.3.

Instead of using fully connected layers for classification, the classifier is designed based on the idea of Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1999). In each training stage, Incremental PPCA encodes the information about each class in a Gaussian distribution with parameters $\boldsymbol{\theta}_k$ for class k. In the inference stage, the classification results are decided by the log-likelihood scores of each image to each PPCA class model. In practice, a more computationally efficient classification equation is used to avoid large computational expenses

involved in multiplying with a full covariance matrix. The underlying assumption and theoretical framework are detailed in Section 2.3.4.

2.3.3 Feature Extraction

As a major obstacle for Incremental Learning, catastrophic forgetting happens because the information about the new classes overwhelms the information about the old classes stored in the model. Hou et al. (2019) reveals that the imbalance between old classes and new classes is the main challenge causing catastrophic forgetting. A common underlying solution is to encourage the feature extractor to generate consistent features in the process of training with new classes. LwF (Li and Hoiem, 2017) uses the distillation loss as a regularization term. This pioneer loss design became a popular approach to avoid catastrophic forgetting. iCarl(Rebuffi et al., 2017) goes one step further by introducing exemplars to store the information about the old classes. Yet even with these efforts, the performance of existing incremental learning methods falls behind by a large margin compared to the joint training of old and new classes. In practice, we choose two different pre-trained models as feature extractor. The choice and implementation of the feature extractors are detailed in Section 3.1.3.

To overcome catastrophic forgetting, the feature extractor needs to generate consistent features for the old classes when training with new classes. The choice of feature extraction follows a simple and intuitive solution: if the feature extractor is frozen in the training stage, the features of the old classes will remain consistent.

Incremental PPCA adopts as a feature extractor a CNN pretrained on a large dataset in a self-supervised manner. This feature extractor remains frozen at all times during the class incremental training process. The features are generated as $\mathbf{x} = f(I)$ where $\mathbf{x} \in \mathbb{R}^d$ is the feature extracted as a d dimensional vector from the image $I \in \Omega$ using the feature extractor $f : \Omega \to \mathbb{R}^d$.

The underlying assumption supporting this design is that with proper training, the feature extractor is able to generate features that are invariant enough to different transformations such as rotation, translation, and scaling, yet contain enough information about objects without training on a specific dataset.

The experimental results indicate that an approach based on feature extraction can obtain a competitive test accuracy. Compared to previous approaches, training the proposed Incremental PPCA is computationally efficient because the feature extractor is frozen and the model for each class is trained separately on the observations from that class and does not need retraining when

more classes are added. In experiments, we will also explore the influence of other essential factors such as image preprocessing and model complexity.

2.3.4 Probabilistic PCA Classification

The basic classification unit is constructed by Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1999). Each image class is encoded as a Gaussian Distribution by PPCA. PPCA has the capability to model instances for one class using a low dimensional representation that is localized near a center $\boldsymbol{\mu}$.

In the framework of PPCA, a latent variable model seeks to relate a d-dimensional observation vector $\mathbf{x}$ to a corresponding q-dimensional vector of latent (or unobserved) variables $\mathbf{t}$.

$$\mathbf{x} = \mathbf{W}\mathbf{t} + \boldsymbol{\mu} + \boldsymbol{\epsilon}. \tag{2.2}$$

The $d \times q$ matrix $\mathbf{W}$ relates the two sets of variables. $\boldsymbol{\mu} \in \mathbb{R}^d$ represents the nonzero mean of observations. The latent vector $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I}_q)$ contains i.i.d. Gaussians with unit variance, while $\boldsymbol{\epsilon}$ represents Gaussian noise, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_d)$, so the probability distribution of $\mathbf{x}$ given latent variable $\mathbf{t}$ is

$$\mathbf{x}|\mathbf{t} \sim \mathcal{N}(\mathbf{W}\mathbf{t} + \boldsymbol{\mu}, \sigma^2\mathbf{I}_d). \tag{2.3}$$

By integration of the latent variable $\mathbf{t}$, the marginal distribution of $\mathbf{x}$ is

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),\ \boldsymbol{\Sigma} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}_d. \tag{2.4}$$

Using Equation (2.4), we can characterize each class as a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Thus the classifiers of Incremental PPCA use the likelihood

$$p(\mathbf{x}|y=k) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)), \tag{2.5}$$

which in log terms, without the common factor $(2\pi)^{d/2}$, is simplified to the class k score:

$$s_k(\mathbf{x}) = -2\log p(\mathbf{x}|y=k) = \log|\boldsymbol{\Sigma}_k| + (\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k), \tag{2.6}$$

where a smaller value is better.

In practice, we will use a simpler score (the Mahalanobis distance)

$$r_k(\mathbf{x}) = (\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \geq 0, \tag{2.7}$$

which differs from $s_k(\mathbf{x})$ from Eq. (2.6) by the log determinant term $\log|\boldsymbol{\Sigma}_k|$. We observed that slightly better results are obtained this way.

2.3.5 Incremental PPCA Training

Training the PPCA model for class k means finding the $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k = \mathbf{W}_k\mathbf{W}_k^T + \sigma^2\mathbf{I}_d$. This is done by standard PCA using the class k observations $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$. The mean $\boldsymbol{\mu}_k$ is:

$$\boldsymbol{\mu}_k = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i,$$

and the PPCA covariance matrix is

$$\boldsymbol{\Sigma}_k = \mathbf{L}\mathbf{D}\mathbf{L}^T + \lambda\mathbf{I}_d, \tag{2.8}$$

where $\lambda > 0$ is a small number (e.g. $\lambda = 0.01$ in our experiments) and $\mathbf{L}$ consists of the first $q < d$ columns of $\mathbf{V}$, where

$$\mathbf{V}\mathbf{D}\mathbf{V}^T = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \tag{2.9}$$

is the SVD decomposition of the sample covariance matrix.

The parameter q represents the dimension of the linear subspace that models the variability of the class k observations. We will use the same value of q for all classes and experiments will show how the choice of q influences the model performance.

2.3.6 Efficient Computation

When d is large (e.g. $d = 1000$), computing the score for each observation involves multiplication with a large $d \times d$ matrix, which can be expensive.

Fortunately, denoting by $\mathbf{d} \in \mathbb{R}^q$ the vector containing the first q elements of the diagonal matrix $\mathbf{D}$ from Eq. (2.9), the score (2.7) can be computed faster using the following

Theorem 1. *The score* (2.7) *can also be written as:*

$$r(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}\|^2/\lambda - \|\mathbf{u}(\mathbf{x})\|^2/\lambda, \tag{2.10}$$

where $\mathbf{u}(\mathbf{x}) = \text{diag}(\frac{\sqrt{\mathbf{d}}}{\sqrt{\mathbf{d}+\lambda\mathbf{1}_q}})\mathbf{L}^T(\mathbf{x} - \boldsymbol{\mu})$*, and the determinant is:*

$$\log|\boldsymbol{\Sigma}| = (d-q)\log\lambda + \sum_{i=1}^{q}\log(d_i + \lambda). \tag{2.11}$$

Here $\text{diag}(\mathbf{v})$ constructs a square matrix with diagonal elements $\mathbf{v}$, and $\sqrt{\mathbf{v}}$ for a vector $\mathbf{v}$ is computed element-wise. Observe that computing $r(\mathbf{x})$ using Eq. (2.10) could be 10-100 times faster than (2.7) since it only involves multiplication with the $q \times d$ skinny matrix $\mathbf{L}^T$ where q is usually $10 - 100$ times smaller than d.

Proof. We have

$$\mathbf{V}^T\mathbf{\Sigma}\mathbf{V} = \mathbf{V}^T(\mathbf{L}\mathbf{D}\mathbf{L}^T + \lambda\mathbf{I}_d)\mathbf{V} = \text{diag}(\mathbf{d}, 0, ..., 0) + \lambda\mathbf{I}_d = \text{diag}(\mathbf{d} + \lambda, \lambda, ..., \lambda)$$

because $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}_d$ and $\mathbf{L}^T\mathbf{V} = (\mathbf{I}_q, \mathbf{0})$.

Thus:

$$\mathbf{V}^T\mathbf{\Sigma}^{-1}\mathbf{V} = \text{diag}(\frac{1}{\mathbf{d} + \lambda}, 1/\lambda, ..., 1/\lambda) = \mathbf{I}_d/\lambda + \text{diag}(\frac{1}{\mathbf{d} + \lambda} - \frac{1}{\lambda}, 0, ..., 0)$$

so

$$\mathbf{V}^T\mathbf{\Sigma}^{-1}\mathbf{V} = \mathbf{I}_d/\lambda - \text{diag}(\frac{\mathbf{d}}{\lambda(\mathbf{d} + \lambda)}, 0, ..., 0)$$

and

$$\mathbf{\Sigma}^{-1} = \mathbf{V}(\mathbf{I}_d/\lambda + \text{diag}(\frac{\mathbf{d}}{\lambda(\mathbf{d} + \lambda)}, 0, ..., 0))\mathbf{V}^T = \mathbf{I}_d/\lambda - \mathbf{L}\,\text{diag}(\frac{\mathbf{d}}{\lambda(\mathbf{d} + \lambda)})\mathbf{L}^T.$$

The score is now:

$$r(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) = \frac{1}{\lambda}(\mathbf{x} - \boldsymbol{\mu})^T(\mathbf{x} - \boldsymbol{\mu}) - (\mathbf{x} - \boldsymbol{\mu})^T\mathbf{L}\,\text{diag}(\frac{\mathbf{d}}{\lambda(\mathbf{d} + \lambda)})\mathbf{L}^T(\mathbf{x} - \boldsymbol{\mu}),$$

from which Eq. (2.10) follows. □

2.3.7 Computation Complexity

In this section, we are interested in the computation complexity of the Incremental PPCA method for training and testing in terms of the number N of observations, dimension d of the feature vector, number K of classes, and number q of principal vectors. Since the computation complexity of the feature extractor is $O(N)$, it will be ignored.

For training, accumulating the K covariances (e.g. using running averages) takes $O(Nd^2)$ time. Computing the PCA for the K classes takes Kd^3, thus the total training time is $O(Nd^2 + Kd^3)$, so it is linear in the number N of training observations and the number of classes K, and cubic in the dimension d of the feature representation. In practice, d is on the order $d \sim 1000 - 4000$, so the PCA computation is fast.

CHAPTER 3

EXPERIMENTAL RESULTS AND ABLATION STUDY FOR INCREMENTAL PPCA

3.1 Experiments

In this section, we use a standard protocol for evaluating incremental methods and compare the proposed Incremental PPCA classification accuracy to the state-of-the-art methods. We also explain how the Incremental PPCA is implemented in detail.

3.1.1 Datasets

We apply Incremental PPCA to four image datasets. Three of the datasets have been extensively used in the incremental learning literature: CIFAR-100 (Krizhevsky, 2009), ImageNet-100, and ImageNet-1k (ILSVRC 2016) (Deng et al., 2009). ImageNet-100 is a subset of ImageNet-1k with only 100 classes, randomly sampled from the original 1000. The choice of these datasets facilitates the comparison with existing incremental methods from the literature. The fourth dataset is Imagenet-10k, the subset of the whole ImageNet (Deng et al., 2009) that contains all 10,450 classes that have at least 450 training observations.

3.1.2 Evaluation Protocol

We evaluate the proposed method using the protocol introduced by Hou et al. (2019). We start with a model trained on a random subset containing half the classes (ie., 50 for CIFAR-100 and ImageNet-100, 500 for ImageNet-1k, 5225 for ImageNet-10k). Then the remaining classes (e.g. 50 for CIFAR-100 and ImageNet-100) are incrementally added. They are equally divided among all steps, for example, we could have 5 steps of adding 10 classes each time for CIFAR-100. In this case, there are 6 learning tasks in total. The trained model is evaluated after each step on the test sets of all classes that were trained so far. To facilitate the comparison, the process is repeated 5 times and the final step accuracy is averaged into a unique score called average accuracy (Rebuffi et al., 2017).

3.1.3 Incremental PPCA Implementation

As a feature extractor, we choose a model that has been trained in a self-supervised way on a different dataset than those evaluated. CLIP (Contrastive Language-Image Pre-Training)(Radford et al., 2021) is a CNN trained on 400 million image-text pairs obtained from the web from 500,000 text queries. The image CNN part of the CLIP model adopts the popular attention mechanism as the last layer before the classification layer. We used a pre-trained modified ResNet-50 classifier called `RN50x4` from the CLIP GitHub+ package (OpenAI, 2022).

SWSL is a student model whose teacher is trained on 940 million images with 1500 hashtags matching the 1000 ImageNet synsets. The student is trained on a subset of 64 million images selected by the teacher. The SWSL model we use is a pre-trained standard ResNet-50, called `resnet50_swsl` from the Facebook Research model repository on GitHub (Facebook, 2022).

The input images are resized to 288×288 except for CIFAR-100 where they were resized to 144×144. The features were extracted before the classification layer using the classifier's attention pooling layer, except for CIFAR-100 where average pooling was used instead. The dimension of the extracted features was $d = 640$ except for CIFAR-100 where it was $d = 2560$.

All experiments were implemented using the PyTorch framework. After all the features were extracted, a separate PPCA model was trained on the training data from each class, as described in Section 2.3.5. Observe that these models do not change no matter what or how many other classes are added. Training the PPCA models takes about 20s per class on a NVIDIA GeForce RTX3080m GPU laptop, thus about 5.6h for 1000 classes.

3.1.4 Results

We evaluate the Incremental PPCA mainly by its classification performance on the validation set of the four datasets. The classification performance is compared with four state-of-the-art models. The models deliver quite remarkable performance, as shown in Table 3.1. Some of their main components are designed with an underlying philosophy similar to Incremental PPCA. As a baseline, BIC (Wu et al., 2019) stated that traditional Neural Networks have a strong bias toward the new classes. BIC adopts a linear model as the classifier. iCarl (Rebuffi et al., 2017) and UCIR (Hou et al., 2019) both use the Nearest-Mean-Exemplar (NME) for classification. UCIR also uses a second inference method based on probabilities (UCIR-CNN).

Table 3.1: Average incremental accuracy for Incremental PPCA vs state-of-the-art. The Incremental PPCA results are averages of 5 independent runs.

	CIFAR-100		ImageNet-100		Imagenet-1K		Imagenet-10k	
	5 steps	10 steps	5 steps	10 steps	5 steps	10 steps	5 steps	10 steps
iCarl(Rebuffi et al., 2017)	57.17	52.57	65.04	59.53	51.36	46.72	-	-
BIC(Wu et al., 2019)	56.86	53.21	68.97	65.14	45.72	44.31	-	-
UCIR(NME)(Hou et al., 2019)	63.12	60.12	68.43	66.16	61.56	59.92	-	-
UCIR(CNN)(Hou et al., 2019)	63.42	60.18	70.47	68.09	64.34	61.28	-	-
PODNet(Douillard et al., 2020)	64.83	63.19	75.82	73.14	66.95	64.13	-	-
PPCA-CLIP	69.71	69.71	82.02	82.02	71.25	71.25	35.42	35.42
PPCA-SWSL	77.07	77.07	86.78	86.78	76.89	76.89	34.39	34.39
RBF-CLIP	47.60	47.60	77.58	77.58	64.84	64.84	28.83	28.83
RBF-SWSL	68.64	68.64	84.14	84.14	73.414	73.414	25.81	25.81

All the comparable models discard the neuron-based structure and explore new classification structures. So the comparison with these models can be an opportunity to evaluate the power of the PPCA-based classifier.

We also compare with a simpler classifier based on the nearest Euclidean distance to the observation means $\boldsymbol{\mu}_k$ instead of the Mahalanobis distance (2.7):

$$r_k(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}_k\|^2. \tag{3.1}$$

This classifier is shown as RBF-CLIP and RBF-SWSL in Table 3.1.

CIFAR-100. The CIFAR-100 dataset contains images of size 32×32 pixels. The CLIP feature extractor is trained with images of size 288×288, so it will not work best with such small images even when resized to 288×288. To gain the best inference performance, experiments were designed to find the best input size for CIFAR-100. The experiments, discussed in Section 3.2.1, reveal that the best input size is 144×144 when the feature extractor is from CLIP, the best input size is 128×128 when the feature extractor is SWSL. From Table 3.1, we can see the Incremental PPCA with CLIP obtains an average accuracy of 69.71%, outperforming the state of the art by almost 5%. SWSL obtains an average accuracy of 77.07%, surpassing the previous state-of-the-art PODNet by 12.34%.

The RBF-CLIP experiment shows that when only the class means $\boldsymbol{\mu}_k$ are used, obtaining RBF-style neurons, the average accuracy drops drastically to 47.6% with CLIP. The average accuracy drops to 68.64% with SWSL. This result indicates that the PPCA model is capable of capturing meaningful variation in the data, much better than the Euclidean distance to the class means.

ImageNet-100. ImageNet-100 contains 100 randomly selected classes from the ILSVRC2016 dataset. The images have a comparable size to the size the CLIP feature extractor has been trained on. From Table 3.1, when the feature extractor is CLIP, the average accuracy on the ImageNet-100 validation set has reached 82.02%, surpassing the next best method (PODNet) by 6.2%. When the features extractor is SWSL, the average accuracy is 86.78%, surpassing PODNet by 10.96%.

Considering that the CLIP feature extractor was not trained on ImageNet-1k or ImageNet-100, the performance of Incremental PPCA indicates that using a pretrained feature extractor can obtain meaningful features for classification. Again, the RBF-CLIP falls behind PPCA-CLIP by about 5%.

ImageNet-1k. ImageNet-1k is a more challenging dataset, containing 1000 classes. The images are also comparable in size with the size the feature extractor has been trained with. From Table 3.1, Incremental PPCA with CLIP as feature extractor has reached 71.25% average accuracy. The performance has surpassed the state of art PODNet by 4.3% and RBF-CLIP by more than 6%. Incremental PPCA with SWSL as feature extractor has reached 76.89% average accuracy. The performance has surpassed the state of art PODNet by 12.76% and RBF-CLIP by more than 3.45%.

ImageNet-10k. ImageNet-10k is the most challenging dataset, containing 10,450 classes and 11 million training images. None of the other methods report results on it, probably because it is very large. From Table 3.1, Incremental PPCA with CLIP as feature extractor obtains 35.42% average accuracy with 20 principal components. This is much better than random guessing, which would have an accuracy of 0.01% in this case. At the same time, the RBF-CLIP that just uses the class means $\boldsymbol{\mu}_k$ and the Euclidean distance obtains a 28.83% accuracy, a 6.59% difference.

3.1.5 Discussion

The results from Table 3.1 indicate that the proposed PPCA-CLIP method can outperform the state-of-the-art incremental class learning methods by at least 4% on all three datasets where a comparison can be obtained. Moreover, it can obtain reasonable classification results on ImageNet-10k, a dataset with 10,450 classes and more than 11 million images.

Furthermore, the RBF-CLIP classifier that just uses the class means and Euclidean distance for classification falls behind by about 5% in accuracy. This shows that the PPCA model is capable of capturing a meaningful representation of the data that goes beyond just radial basis functions and Euclidean distance.

Table 3.2: Average accuracy of PPCA-CLIP on CIFAR-100 for different input sizes and numbers of principal components.

Input size	0PC	5PC	10PC	20PC	50PC	100PC	200PC
32×32(original)	7.05	21.04	21.04	25.7	27.37	28.60	28.58
36×36 (avg pooling)	7.47	22.26	24.83	26.66	28.9	29.55	29.51
72×72 (avg pooling)	40.32	53.43	55.04	55.04	58.89	59.56	60.07
144×144 (avg pooling)	47.60	61.57	64.34	66.58	68.50	69.56	69.71
288×288 (avg pooling)	30.97	51.09	56.35	60.48	63.94	65.05	65.71
288×288 (with attention)	40.88	55.29	58.54	61.44	62.78	62.14	61.26

Table 3.3: Average accuracy of PPCA-SWSL on CIFAR-100 for different input sizes and numbers of principal components.

Input size	5PC	10PC	20PC	50PC	100PC
32×32(original)	41.62	44.14	45.4	47.17	48.41
64×64	61.93	63.4	64.72	66.08	66.63
128×128	73.53	75.02	75.97	76.89	77.07
256×256	69.32	71.68	73.51	74.98	74.92
224×224 (default)	69.03	71.17	72.8	73.72	74.43

3.2 Ablation Study

In this section, we evaluate the contribution of different hyper-parameters to the proposed Incremental PPCA method's accuracy. There are two essential components of the proposed method: feature extraction and the PPCA classifier. The size of the feature extractor's input influences the quality of the extracted features. In Section 3.2.1, we will see how the input size changes the accuracy of the Incremental PPCA method for CIFAR-100. The number of principal components (PCs) decides how much complexity is encoded for each class. Its effect will be analyzed in Section 3.2.2.

3.2.1 The Input Image Size

As shown in Table 3.2 and Table 3.3, the PPCA accuracy varies with the size of the feature extractor input. The CLIP feature extractor is trained with high resolution (288×288) images, while the images of the CIFAR-100 dataset are 32×32. If these images are resized to 288×288, they will look very blurred. The experiment aims to explore how the input size improves the performance on low-resolution images. It might as well be possible that the 288×288 input might not be the best setting for low-resolution images for the high-resolution feature extractor.

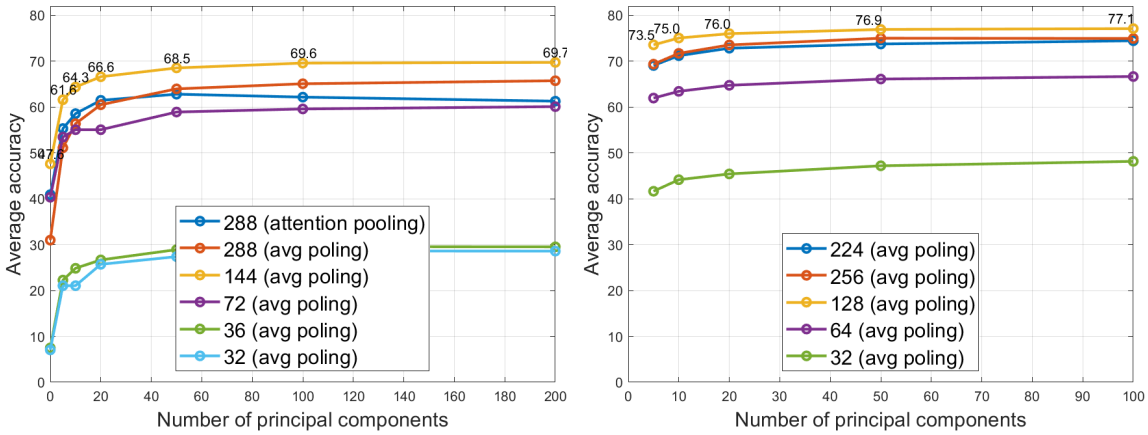


Figure 3.1: Incremental PPCA with different input sizes on CIFAR-100.

Table 3.4: Average accuracy of PPCA-CLIP on ImageNet-100, ImageNet-1k and ImageNet-10k for different numbers of principal components.

Input size	0PC	5PC	10PC	20PC	50PC	100PC	200PC
ImageNet-100 with CLIP	77.58	80.92	81.26	82.02	81.24	79.92	78.12
ImageNet-100 with SWSL	84.14	86.22	86.78	86.58	86.72	86.76	86.78
ImageNet-1k CLIP	64.84	69.16	70.73	71.25	69.93	68.11	65.79
ImageNet-1k with SWSL	73.41	76.46	76.81	76.89	76.76	76.57	76.04
ImageNet-10k CLIP	28.83	33.49	34.85	35.42	34.80	33.28	31.01
ImageNet-10k SWSL	25.81	31.75	33.39	34.39	34.75	34.40	33.56

The evaluation of input size is based on the experiment with 5 steps of 10 classes on CIFAR-100. Table 3.2 and Table 3.3 shows the contribution of the input size to the method's accuracy. If the image is input without resizing, the method's accuracy is very poor. When the feature extractor is SWSL, the original preprocessing that resizes the images to 224×224 doesn't produce the best performance. When the 32×32 images are resized to 128×128, the average accuracy reaches a maximum of 77.07%.

When the feature extractor is CLIP, the original preprocessing that resizes the images to 288×288 doesn't produce the best performance either. When the 32×32 images are resized to 144×144, the average accuracy reaches a maximum of 69.71%. The results are also shown as a graph in Figure 3.1.

3.2.2 The Number of Principal Components

The number of principal components q is an essential hyperparameter that influences the method's accuracy. Theoretically, q represents the dimension of the (linear) manifold fitting the observations in each class.

We design an experiment to explore the relationship between the number of principal components and classification accuracy. Again, the evaluation is based on the experiment with 5 steps of 10 classes on CIFAR-100, ImageNet-100, ImageNet-1k, and ImageNet-10k. We use 0 principal components as a baseline in which case no variations are encoded. Even though the SWSL is fine-tuned on ImageNet-1k, the results with SWSL can also be representative for the ResNet-based models. Figure 3.1 reveals that the average accuracy saturates as the number of PCs is increased

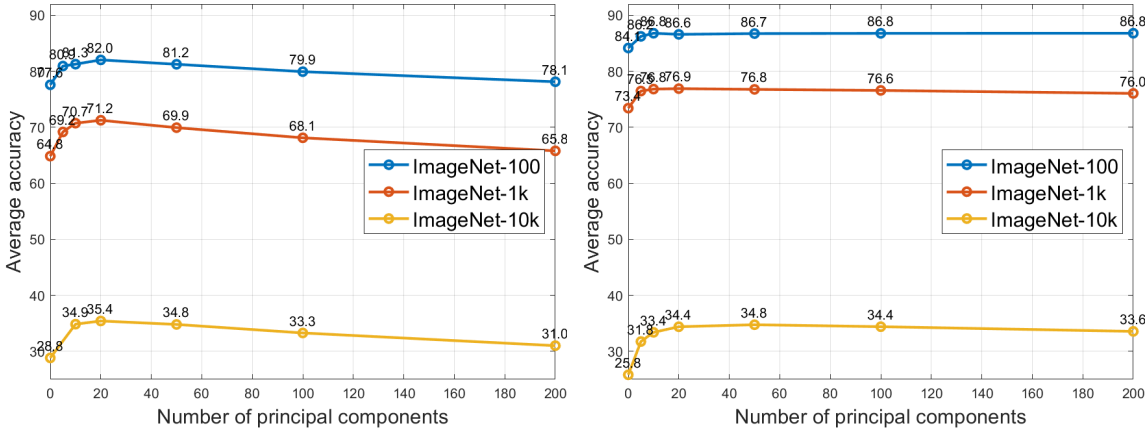


Figure 3.2: Incremental PPCA with different number of principal components on the ImageNet datasets evaluated.

for CIFAR-100, the same phenomenon for all input sizes evaluated.

Table 3.4 and Figure 3.2 show that the average accuracy for the ImageNet datasets increases with the number of principal components until the number of PCs reaches 20, after which the average accuracy drops gradually.

These experiments reveal that an appropriate number of principal components can retain a balance between overfitting and lack of variation to reach an optimal classification accuracy.

3.3 Conclusion

We presented a novel and simple method for large-scale class incremental learning with pretrained feature extractors and a classifier based on probabilistic PCA models. The pretrained feature extractor guarantees that the generated features are consistent during the whole training process. The PPCA-based classifier is able to encode the complexity of each image class using a low dimensional representation and small computational expenses. This approach can prevent catastrophic forgetting to a large extent.

The proposed method outperforms existing state-of-the-art methods in incremental class learning on three standard datasets used in the literature. Furthermore, it obtains reasonable classification results on ImageNet-10k, the subset of ImageNet containing all classes with at least 450 training images, for a total of more than 11 million images.

The performance of Incremental PPCA reveals that generic pretrained models have the ability to extract meaningful features from images and gain promising performance without being trained on labeled data.

Humans are capable of classifying millions of classes of objects and of learning a new object class from only 1-2 examples. However, humans are the subjects of millions of years of evolution, which can be seen as a kind of supervised learning since the fittest specimens (e.g. fit for understanding images) are better at reproduction and evolution. Thus the human brain is pre-wired by evolution to be able to accomplish these large scale classification tasks. The pretrained feature extractor that we use in the Incremental PPCA can be seen as the analog of the pre-wired brain in humans and other mammals.

CHAPTER 4

HIERARCHICAL CLASSIFICATION FOR LARGE SCALE LEARNING

4.1 Introduction

Deep learning models emerged to surpass human performance on multiple vision tasks (He et al., 2015; Zhang et al., 2021). Artificial neural networks have layers of biologically inspired neurons that are learned by gradient descent (Bishop, 1994; Werbos, 1990). Despite their success, the deep learning models are regarded as a black box because the process in which the individual neurons generate outputs is not interpretable (Samek et al., 2017; Ribeiro et al., 2016). The deep learning methods do not construct a hierarchy of concepts explicitly and therefore require training on large labeled datasets from the same distribution as the desired task (LeCun et al., 2015).

Jeon (2014) stated that human cognitive architecture is composed of substructures as in hierarchical processing. Bergman et al. (2003) has proposed that rule-based hierarchical classification is biologically inherited by humans or other non-human mammals like baboons. Humans can establish hierarchical semantic relations between categories, which are learned by much fewer samples than deep learning models. Models like symbolic systems attempted to simulate the high-level reasoning processes of humans (classification logic and temporal logic). Garcez et al. (2008) proposed a framework known as a neural symbolic system artificial neural networks (ANNs). It provides the framework for parallel computation and robust learning while logic units provide interpretability. However, most symbolic systems often require problem-specific manual tuning features (Gardenfors, 2004) and are not able to learn features from raw input data (Minsky, 1991).

Inspired by our understanding of human hierarchical cognition, we propose a hierarchical framework for object classification called Hierarchical PPCA. We use a 2-level taxonomy to model the semantic hierarchy of image classes. Image classes are conceptualized as Gaussian distributions with Probabilistic Principal Component Analysis (PPCA) models. Instead of standard neurons, the classifier is constructed using PPCA neurons. We assume image classes with similar semantic information will cluster together, each semantic cluster is characterized by its centroid which we called a super-class. With this assumption, we modified the k-means clustering algorithm to ex-

plore underlying semantic relations and build the class hierarchy. In the classification, an image of a cat will first be assigned to a super-class like 'animals' and then classified among the image classes associated with the 'animal' super-class. This hierarchical scheme facilitates sparse neuron firing and takes much fewer computation resources than the traditional structure. If the dataset contains K classes, the hierarchical model can classify images in $O(\sqrt{K})$ time instead of $O(K)$ as in standard classification. Experiments have shown that Hierarchical PPCA can applied to large-scale datasets with superior accuracy and efficiency.

The main contributions of Hierarchical PPCA are:

1. It introduces a hierarchical classification model for learning from datasets with a large number of classes, without hierarchical annotation. The image classes are modeled as Gaussian distributions based on Probabilistic PCA (PPCA). The model reduces the classification time for observation from $O(K)$ to $O(\sqrt{K})$, and potentially to $O(\log K)$ when using more hierarchy levels, where K is the number of classes.

2. It presents an efficient training procedure based on a generalization of k-means clustering that clusters image classes instead of features. This design can handle unbalanced data where the number of observations in each class can differ widely.

3. It conducts experiments on large-scale datasets that show that indeed the hierarchical approach can speed-up classification without any loss in accuracy.

4.2 Related Work

4.2.1 Hierarchical Clustering

Hierarchical clustering (HCA) is an unsupervised method of cluster analysis that seeks to build a hierarchy of clusters (Day and Edelsbrunner, 1984). Much early work on hierarchical clustering was in the field of biological taxonomy.

Agglomerative clustering (Gowda and Krishna, 1978) is a dominant HCA method that works in a bottom-up manner. The main idea behind agglomerative clustering is to find the closest pair of clusters and merge them together. This process is repeated until all the data points are merged into a single cluster. The distance between two clusters is usually measured using a distance metric, such as the Euclidean distance or cosine similarity. The most common method for merging two clusters is the single linkage method, which merges the two clusters that are closest to each other. One of the main advantages of agglomerative clustering is its simplicity and ease of implementation. It is also useful for identifying hierarchical structures within the data, as it produces a dendrogram

that shows how the clusters are nested within each other. This can be useful for visualizing the relationships between different groups of data. However, agglomerative clustering can be computationally expensive and may not scale well to large datasets. Agglomerative clustering for n data points has a time complexity of $O(n^3)$ and requires $O(n^2)$ memory, which makes it unpractical for even medium datasets.

Another variant of HCA is divisive clustering (Samek et al., 2017), which works in a top-down manner. The algorithm starts at the top with all observations in one cluster. The cluster is split using a flat clustering algorithm. This procedure is applied recursively until each observation is in its own singleton cluster. Top-down clustering is conceptually more complex than bottom-up clustering. Divisive clustering with exhaustive search is $O(2^n)$, but it is common to use faster heuristics to choose the splits, such as k-means. However, divisive clustering is more efficient if it does not generate a complete hierarchy all the way down to individual observations. Hierarchical clustering has the distinct advantage that any valid measure of distance can be used. However, due to its limitations on time complexity, it might not applicable to large-scale datasets.

4.2.2 Hierarchical Models for Vision Tasks

Some efforts aim to explore the hierarchical semantic nature for vision tasks (Tousch et al., 2012). Kumar and Zheng (2017) propose a model that learns the visual similarities between various clothing categories and predicts a tree of categories. They propose a novel object detection method that can handle newer categories without the need of obtaining new labeled data and retraining the network. Jia et al. (2013) propose a hierarchical framework capable of learning visual abstraction from a small number of images. Zweig and Weinshall (2007) utilize the hierarchical nature for class recognition. It combines image classifiers from different hierarchical levels into a single classifier to improve classification accuracy. Marszalek and Schmid (2007) utilize the semantics of image labels to integrate prior information about inter-class relationships into visual concept learning. They built a semantic hierarchy of discriminative classifiers for object detection. Srivastava and Salakhutdinov (2013) propose a method that benefits from CNN and a hierarchical prior knowledge when the training set is small. They have shown that the label tree prior can be used to transfer knowledge between classes and boost performance with insufficiently many training examples are available.

This dissertation proposes a new framework applicable to the classification of large-scale datasets with thousands or even millions of classes. By utilizing PPCA to construct classification units,

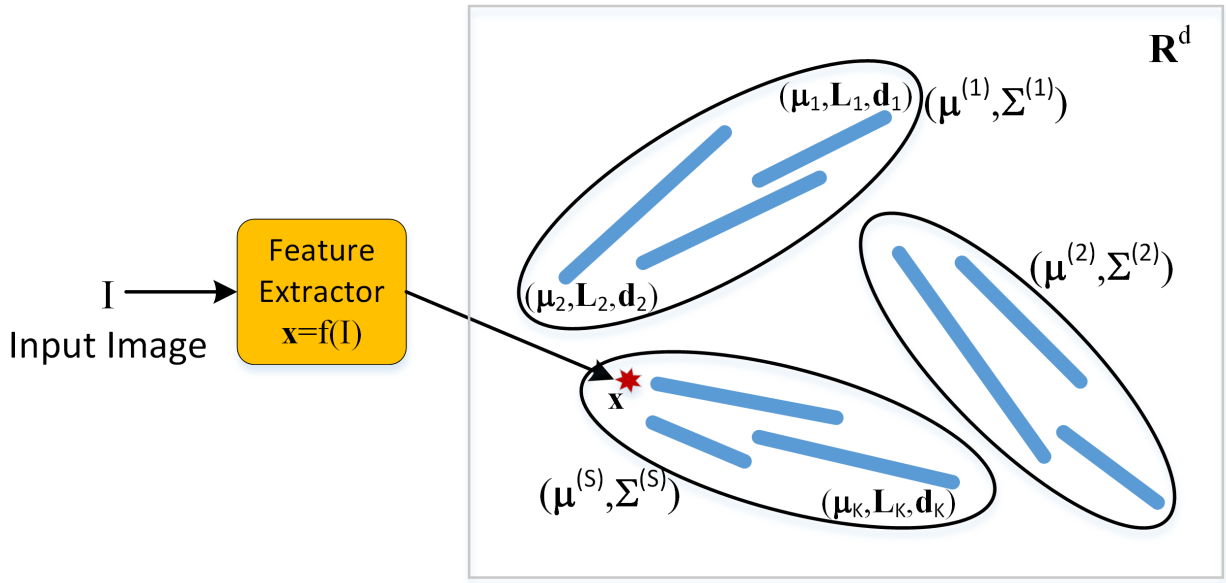


Figure 4.1: Diagram of the hierarchical classification with Gaussian superclasses and Probabilistic PCA classes.

Hierarchical PPCA is capable of controlling and localizing the information encoded for each image class. A special k-means clustering algorithm is introduced to build a hierarchy between image classes and a small number of super-classes to speed up classification. Experiments show that Hierarchical PPCA can obtain superior accuracy with less computational expense.

4.3 Proposed Approach

Inspired by the human hierarchical representation of objects, this dissertation proposes a hierarchical classification framework that exploits the semantic information of image classes.

As shown in Figure 4.2.2, the classification of an input image I is represented as $y = \text{argmin}_k\, r_k(f(\mathbf{I}))$, based on a feature extractor f and a number of class models $r_k(\mathbf{x})$.

4.3.1 Feature Extraction

The feature extractor $\mathbf{f} : \Omega \rightarrow \mathbb{R}^d$ aims to generate informative features from the input images, where Ω is the space of input images. In practice, Hierarchical PPCA adopts a CNN pretrained on a large dataset as a feature extractor. The intuition of this choice is that with proper training, the feature extractor is able to generate features that are invariant enough to different transformations such as rotation, translation, and scaling, yet contain enough information about objects without

training on a specific dataset. The implementation of the feature extractor is detailed in Section 5.1.2.

4.3.2 Hierarchical Classifier

Newman et al. (2010) stated that human cognitive architecture is built up of a hierarchy of multiple system levels. Humans, even babies, use the hierarchical cognition system to conduct categorization without training with huge numbers of images. Inspired by this finding about human hierarchical cognition, the classifier is modeled as a 2-level taxonomy. It aims to organize the information with two levels of abstraction.

In the proposed Hierarchical PPCA model, each image class is conceptualized as a Gaussian distribution by Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1999). As the basic classification unit, each PPCA neuron represents a class. The theoretical assumptions of PPCA are detailed in Section2.3.4 and Section2.3.5. The first level of the classifier consists of PPCA neurons representing super-classes. The second level of the classifier consists of PPCA neurons encoding information about image classes. The essential assumption is that image classes with similar semantic information cluster together. A cluster is characterized by its centroid which we call a super-class. A super-class maintains the least distance to the image classes belonging to the same cluster. If there are K image classes to be classified (e.g. $K = 1000$ for ImageNet), they are divided into S disjoint sets $K_1, ..., K_S$ such that:

$$\cup_{s=1}^{S} K_s = \{1, ..., K\}$$

In the framework of Hierarchical PPCA, the first level models (the super-classes) are represented as $(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, with $k \in \{1, ..., S\}$, where S is the number of super-classes. For a super-class $s \in \{1, ..., S\}$, the corresponding image classes are represented as $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with $k \in K_s$.

4.3.3 Hierarchical classification

Given an observation $\mathbf{x} \in \mathbb{R}^d$, the first level classifier aims to find the most likely super-class s for the observation, a process called super-classification. After the sample $\mathbf{x}$ is assigned to a super-class s, the second level of the classifier will find the most likely image class among image classes $k \in K_s$ associated with super-class s. This process is called image classification. Since both level models are Gaussians, we will use the Mahalanobis distance

$$r(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \geq 0, \tag{4.1}$$

to measure how well an observation $\mathbf{x} \in \mathbb{R}^d$ fits the Gaussian model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where a smaller score is better. In practice, classification involving high dimensional features, we derived a computation efficient method which is detailed at Section 2.3.6.

However, any super-classification failure would result in the failure of the whole classifier. To avoid the impact of such failures, instead of considering a single most likely super-class, the hierarchical classifier will consider a number T of the most likely superclasses, as described in Algorithm 1. In experiments, T was taken in the range of $T \in \{1, 2, 3, 4\}$.

Algorithm 1 Hierarchical Classification

Input: Observation $\mathbf{x} \in \mathbb{R}^d$, super-class models $(\boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}), s \in \{1, ..., S\}$,image class models $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k \in \{1, ..., K\}$.

Output: Predicted class label $\hat{k} \in \{1, ..., K\}$.

1: Compute the super-class scores $v_s = r(\mathbf{x}; \boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}), s \in \{1, ..., S\}$
2: Find the indices $J \subset \{1, ..., S\}, |J| = T$ of the T lowest scores $v_j, j \in J$
3: Compute the index set $U = \cup_{j \in J} K_j$
4: Obtain the predicted class label

$$\hat{k} = \underset{k \in U}{\operatorname{argmin}}\, r(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

4.3.4 Computation Complexity

The computation demand of a model can be measured by the average number of neurons used in the classification. For an incoming sample feature, a flat classifier requires the computation of the scores for all K image classes to generate the classification output. In contrast, Hierarchical PPCA requires the score computation for all S super-classes and the score computation of the image classes corresponding to the top T the super-classes. Let A to denote the average number of image classes associated with one super-class. The computation cost for one classification in Hierarchical PPCA is therefore $S + TA$ neurons (score computations).

We use two measures to evaluate the computational cost of Hierarchical PPCA compared with the flat classification. The density is defined as

$$\text{density} = \frac{S + TA}{K}. \tag{4.2}$$

The inverse of the density measures the speed-up of Hierarchical PPCA compared to flat classification

$$\text{speed-up} = K/(S + TA). \tag{4.3}$$

Assuming the image classes are uniformly distributed for clusters then the density is

$$\text{density} \sim \frac{S + KT/S}{K} = \frac{S}{K} + \frac{T}{S}. \tag{4.4}$$

From an efficiency perspective, the density reaches a minimum when $S = \sqrt{KT}$, where the speed-up reaches the maximum of $O(\sqrt{K/T})$. The experimental results in Section 5.2 match with our theoretical derivation.

4.4 Training the Hierarchical Classifier

One important assumption is that image classes with similar semantic information form clusters in the feature space. With this assumption, we adopt a variant of the k-means algorithm to explore the semantic structure of image classes and train the hierarchical classifier. The difference from the standard k-means is that instead of clustering observations, the proposed algorithm clusters image classes encoded as Gaussian distributions. The advantage of this approach is that it only needs to cluster a small number of elements (e.g. 1000 Gaussian distributions for ImageNet) instead of millions of observations. Moreover, this kind of clustering is robust to data imbalance in the classes. The training algorithm is summarized in Algorithm 2.

Algorithm 2 Hierarchical PPCA Training

Input: Training observations $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{Z}$, number S of super-classes, number q of PCs for the image class models, number r of PCs for the super-class models.

Output: Super-class models $(\boldsymbol{\mu}^{(s)}, \mathbf{L}^{(s)}, \mathbf{d}^{(s)}), s \in \{1, ..., S\}$, image class models $(\boldsymbol{\mu}_k, \mathbf{L}_k, \mathbf{d}_k), k \in \{1, ..., K\}$.

Train the image class PPCA models $(\boldsymbol{\mu}_k, \mathbf{L}_k, \mathbf{d}_k), k \in \{1, ..., K\}$ (Section 4.4.1).
Initialize S super-class models using k- means++ (Section 4.4.2).
while not converged **do**
  Assign the image classes to the super-classes using Eq. (4.11)
  Update the super-class models based on the image classes in the same clusters (Section 4.4.4).
end while

The training procedure is a generalization of k-means clustering. The clustering subjects are image classes encoded as Gaussian distributions instead of vectors. We innovate with the following modification for the new clustering subjects:

- The distance measure between the image classes, which is used for the k-means++ initialization (Arthur and Vassilvitskii, 2006b), is replaced from a Mahalanobis distance to the Bhattacharyya distance between Gaussians (Section 4.4.2).
- The distance measure between an image class and a Gaussian super-class model is the KL divergence, described in Section 4.4.3.
- The Gaussian super-class models are parameterized to minimize the sum of the KL-divergences of the image class models, as described in Section 4.4.4.

These steps will be presented in the next sections, together with the training of the image class PPCA models.

4.4.1 Training the Image Classifiers

The second level of the Hierarchical PPCA model contains PPCA models representing information for the image classes. The PPCA neurons are trained with SVD decomposition separately for each class as follows. If the feature vectors of all training observations of class k are gathered as X_k, the mean $\boldsymbol{\mu}_k$ is:

$$\boldsymbol{\mu}_k = \frac{1}{|X_k|} \sum_{x \in X_k} \mathbf{x}$$

and the full covariance matrix is

$$\mathbf{C}_k = \frac{1}{|X_k| - 1} \sum_{\mathbf{x} \in X_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T. \tag{4.5}$$

Then the PPCA covariance matrix is obtained from the sample covariance matrix $\mathbf{C}_k$, via the SVD decomposition

$$\mathbf{V}\mathbf{S}\mathbf{V}^T = \mathbf{C}_k, \tag{4.6}$$

as

$$\boldsymbol{\Sigma}_k = \mathbf{L}\mathbf{D}\mathbf{L}^T + \lambda \mathbf{I}_d, \tag{4.7}$$

where $\lambda > 0$ is a small number (e.g. $\lambda = 0.01$ in our experiments) and $\mathbf{L}$ consists of the first $q < d$ columns of $\mathbf{V}$ and $\mathbf{D}$ is the $q \times q$ diagonal matrix containing the first q rows and columns of $\mathbf{S}$.

The same approach is used to obtain super-class PPCA models from the super-class covariance matrices.

4.4.2 Initialization by k-Means++

The performance of k-means relies significantly on its initialization. Arthur and Vassilvitskii (2006a) showed that the worst-case running time of the k-means algorithm is super-polynomial in the input size. We adopt the k-means++ (Arthur and Vassilvitskii, 2006b) initialization method to accelerate the convergence speed and improve clustering performance. The process of k-means++ is detailed in Algorithm 3.

Algorithm 3 k-means++ centroid initialization

Input: Image classes represented as $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $k \in \{1, ..., K\}$, number S of clusters

Output. Initialized super-class models $C_s = (\boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}), s \in \{1, ..., S\}$

Compute the matrix of distances between all pairs of image classes.
Randomly pick the first centroid C_1
for $i = 2$ to S **do**
  Update the distance between all image classes and the newly selected centroid C_{i-1} in the pairwise distance matrix
  Generate the discrete distribution based on the distance of the image classes to their closest centroids
  Sample the image class $\mathbf{x}_j$ from the discrete distribution generated
  Initialize the super-class model $C_i = (\mathbf{x}_j, \mathbf{I}_d)$
end for

The elements that are clustered are the image classes represented by Gaussian distributions. We adopt the Bhattacharyya distance to measure the pairwise distance between two distributions. For two distributions P, Q, the Bhattacharyya distance is defined as

$$D_B(P, Q) = -\ln BC(P, Q),\ BC(P, Q) = \int \sqrt{P(x)Q(x)}dx \tag{4.8}$$

For Gaussian distributions, $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, the Bhattacharyya distance is

$$D_{ij} = \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\ln \frac{|\boldsymbol{\Sigma}|}{\sqrt{|\boldsymbol{\Sigma}_i||\boldsymbol{\Sigma}_j|}}, \tag{4.9}$$

where $\boldsymbol{\Sigma} = \frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2}$.

4.4.3 Assignment of Image Classes

In the training process, the image classes are assigned to the nearest cluster. A distance measure is needed to measure the distance from image classes to the clusters, which are characterized by

super-class Gaussian distributions. We adopt the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951), denoted by $D_{KL}(P \parallel Q)$ to measure the distance between image classes and super-classes. The KL divergence is a type of statistical distance: it measures how one probability distribution P is different from a second, reference probability distribution Q.

$$D_{KL}(P \parallel Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} dx. \tag{4.10}$$

For our application, the image classes are chosen as P and the super-classes as reference distributions Q. In this dissertation, all classes are encoded as Gaussian distributions, the KL divergence between a super-class $Q = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and an image class $P = (\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is (Duchi, 2007)

$$D_{KL}(P \parallel Q) = \frac{1}{2}\{\log \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_i|} - d + \mathrm{Tr}[\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_i]\} + (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu})/2 \tag{4.11}$$

4.4.4 Super-class Model Update

By our assumption of image classes, the super-class model is the centroid of the corresponding cluster, i.e. the Gaussian distribution that has the smallest sum of the distances to the image classes within the same cluster.

The image classes within a cluster C are normal distributions $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i \in C$. The corresponding super-class model is a normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ so that the sum of the distances from the image classes to the super-class

$$D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i \in C} D_{KL}(N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \parallel N(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \tag{4.12}$$

is minimized, where $D_{KL}(P \parallel Q)$ is defined in Eq. (4.11).

The following theorem gives a closed-form solution of the minimization.

Theorem 2. *The Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ that minimizes $D(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ from Eq. (4.12) has parameters:*

$$\boldsymbol{\mu} = \frac{1}{|C|} \sum_{i \in C} \boldsymbol{\mu}_i, \tag{4.13}$$

and

$$\boldsymbol{\Sigma} = \frac{1}{|C|} \sum_{i \in C} [(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \boldsymbol{\Sigma}_i]. \tag{4.14}$$

Proof. From (Duchi, 2007), the KL divergence between normal distributions $P = N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $Q = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is

$$D_{KL}(P||Q) = \frac{1}{2}[\log \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_i|} - d + (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}) + Tr\{\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_i\}], \tag{4.15}$$

so the sum is

$$D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2} \sum_{i \in C} \{ \log \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_i|} - d + (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) + Tr(\boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_i) \}. \tag{4.16}$$

Setting the partial derivative of $D(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\mu}$

$$\frac{\partial}{\partial \boldsymbol{\mu}} D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i \in C} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) \tag{4.17}$$

to zero we obtain $\boldsymbol{\mu} = \frac{1}{|C|} \sum_{i \in C} \boldsymbol{\mu}_i$.

Because $(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$ can be written as $Tr[(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})]$, the third term of the distance can be written as

$$\sum_{i \in C} Tr[(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu})] = \sum_{i \in C} Tr[(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}] \tag{4.18}$$

So the distance can be written as

$$\begin{aligned} D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{1}{2} \sum_{i \in C} \left(\log \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_i|} + Tr[(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}] + Tr\{\boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_i\} \right) \\ &= \frac{1}{2} \sum_{i \in C} \log \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_i|} + \frac{1}{2} Tr\{ \sum_{i \in C} [(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \boldsymbol{\Sigma}_i] \boldsymbol{\Sigma}^{-1} \} \end{aligned} \tag{4.19}$$

Therefore the partial derivative of $D(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}$ is

$$2 \frac{\partial}{\partial \boldsymbol{\Sigma}} D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |C| \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} \sum_{i \in C} [(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \boldsymbol{\Sigma}_i] \boldsymbol{\Sigma}^{-1} \tag{4.20}$$

Setting $\frac{\partial}{\partial \boldsymbol{\Sigma}} D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = 0$ and multiplying to the left and to the right by $\boldsymbol{\Sigma}$, we obtain

$$\boldsymbol{\Sigma} = \frac{1}{|C|} \sum_{i \in C} [(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \boldsymbol{\Sigma}_i]. \tag{4.21}$$

□

After the covariance matrix is obtained by Eq. (4.14), the super-class PPCA model $(\boldsymbol{\mu}, \mathbf{L}, \mathbf{d})$ is obtained by writing $\mathbf{V} \mathbf{S} \mathbf{V}^T = \boldsymbol{\Sigma}$ by SVD, and obtaining $\mathbf{L}$ as the first q columns of $\mathbf{V}$ and $\mathbf{d}$ as the first q diagonal elements of $\mathbf{S}$.

CHAPTER 5

EXPERIMENTAL RESULTS AND ABLATION STUDY FOR HIERARCHICAL PPCA

5.1 Experiments

Experiments are conducted to compare the accuracy and computation efficiency of Hierarchical PPCA with flat classification: the 1-level classifier containing the same PPCA image models but where the classification $y = \text{argmin}_k\, r_k(f(\mathbf{I}))$ finds the minimum score for an image $\mathbf{I} \in \Omega$ among all classes. Experiments are designed to explore how some of the main parameters such as the number of super-classes, the number of principal components, and the parameter T from Algorithm 1 influence accuracy and computation efficiency.

5.1.1 Datasets

Experiments are conducted on three standard datasets: ImageNet-100, ImageNet-1k (ILSVRC 2016) Deng et al. (2009) and ImageNet-10k. ImageNet-100 is a subset of ImageNet-1k with only 100 classes, randomly sampled from the original 1000 classes. Imagenet-10k is the subset of the whole ImageNet (Deng et al., 2009) that contains all 10,450 classes with at least 450 training observations. They are standard image datasets that are adopted to prove the robustness and facilitate comparison. All results are reported as averages of 4 independent runs for better reproducibility.

5.1.2 Feature Extractor

The feature extractor $\mathbf{f} : \Omega \to \mathbb{R}^d$ used in this work is the ResNet50x4 from CLIP (Contrastive Language-Image Pre-Training)(Radford et al., 2021). CLIP models are pairs of image embedding and language embedding models, trained on 400 million pairs of images and their corresponding captions. It is trained on a wide variety of images with a wide variety of natural language supervision that's abundantly available on the internet. By design, the network can be instructed in natural language to perform a great variety of classification benchmarks, without directly optimizing for the benchmark's performance. As such, it was not trained or fine-tuned on the three evaluation datasets.

The choice of the feature extractor is important for the validity of the assumption made for hierarchical PPCA, which is that semantically similar classes cluster together.

5.1.3 Evaluation Measures

Efficiency Measure. There are two efficiency measures: density and speed-up. The density evaluates how much less computation is required for classification using Hierarchical PPCA compared to flat classification. The theoretical definition and analysis were presented in Section 4.3.4. In practice, density is defined as

$$\text{density} = \frac{S + A}{K} \tag{5.1}$$

where A is the average number of image classes used, S is the number of super-classes and K is the total number of classes. The speed-up is the inverse of density and measures the computational efficiency of Hierarchical PPCA. The theoretical analysis indicates the density reaches a minimum and the speed-up a maximum when $S = \sqrt{KT}$,

Super-class accuracy. The super-classes are constructed without human annotation using k-means clustering by exploring the semantic relations between the image classes. Classifying an observation to the wrong super-class would result in a failure in the overall classification of this observation. The super-class accuracy measures the top-T classification accuracy of the level 1 (super-class) classifier.

5.1.4 Raw Data Clustering

In Hierarchical PPCA, the superclasses are learned by clustering the image classes, which are parameterized as Gaussian distributions. Another hierarchical method that will be evaluated is raw data clustering, which is similar to Hierarchical PPCA, but the super-class models are obtained from clustering a subsample of observations from the image classes instead of clustering the Gaussian models of image classes.

In this experiment, 20 images are randomly sampled from each image class, obtaining a sample of $20K$ images, where K is the number of image classes. The images are clustered with standard k-means clustering.

5.1.5 Main Results

The main comparison of the Hierarchical PPCA, Hierarchical PPCA with raw data clustering, and flat PPCA in terms of test accuracy and computational efficiency are shown in Table 5.1.

Table 5.1: Evaluation in terms of test accuracy and speedup vs. flat classification of the proposed Hierarchical PPCA Algorithm 1 with $T = 1$ and $T = 4$. Also shown are the raw data clustering test accuracy and speedup results.

	ImageNet-100		Imagenet-1k		Imagenet-10k	
	Accuracy	Speed up	Accuracy	Speed up	Accuracy	Speed up
Flat Classification	0.820	1.0	0.713	1.0	0.354	1.0
Hierarchical PPCA $T = 1$	0.802	4.4	0.602	10.5	0.298	43.5
Hierarchical PPCA $T = 4$	0.836	1.9	0.719	4.1	0.360	16.8
Raw data clustering $T = 1$	0.723	4.4	0.593	13.4	0.305	22.0
Raw data clustering $T = 4$	0.835	1.8	0.704	5.9	0.365	13.9

Table 5.2: Comparison between the accuracy of Hierarchical PPCA when clustering Gaussians vs clustering raw data.

	ImageNet-100			Imagenet-1k			Imagenet-10k		
T	Flat	Raw data	Gaussians	Flat	Raw data	Gaussians	Flat	Raw data	Gaussians
1	0.820	0.723	0.802	0.713	0.593	0.602	0.354	0.305	0.298
2	-	0.811	0.832	-	0.667	0.684	-	0.346	0.338
3	-	0.823	0.836	-	0.693	0.709	-	0.359	0.353
4	-	0.835	0.836	-	0.704	0.719	-	0.365	0.360

For all three datasets, when $T = 4$, the Hierarchical PPCA surpasses the flat classifier both in terms of accuracy and computational efficiency.

Table 5.2 indicates that clustering Gaussians is better than clustering raw data. On ImageNet-100, the learning from raw data starts to surpass the flat accuracy when $T = 3$. On Imagenet-1k, the performance of clustering from raw data is slightly inferior to Hierarchical PPCA and the flat classifier.

5.2 Ablation Study

5.2.1 Number of Super-Classes

The super-classes represent the general concepts in the Hierarchical PPCA approach, which are used to speed up classification. The number of super-classes S is an important factor for computation efficiency and overall accuracy.

Table 5.3 shows an evaluation of the test accuracy and speed-up for different numbers of super-classes for the three datasets.

ImageNet-100 contains 100 image classes. The experiment adopts three values for S, namely 5, 10, 15. The density does not decrease monotonically with increasing S. The speed-up reaches a

Table 5.3: Evaluation of speed-up and accuracy for different numbers S of super-classes, when $T = 1$, $q = r = 20$.

S	Density	Speed-up	Accuracy	Super Accuracy
ImageNet-100				
Flat	1.0	1.0	0.820	-
5	0.316	3.2	0.811	0.946
10	0.229	4.4	0.802	0.921
20	0.269	3.7	0.809	0.920
ImageNet-1k				
Flat	1.0	1.0	0.713	-
10	0.179	5.6	0.636	0.833
20	0.108	9.3	0.620	0.789
33	0.095	10.5	0.602	0.753
40	0.096	10.5	0.604	0.756
50	0.091	11.0	0.597	0.736
66	0.096	10.4	0.586	0.710
100	0.122	8.2	0.577	0.687
ImageNet-10k				
Flat	1	1	0.354	-
50	0.030	33.9	0.307	0.710
100	0.023	43.5	0.298	0.665
200	0.026	39.1	0.295	0.615
300	0.033	30.2	0.296	0.596

maximum of 4.36 when $S = 10$. The super-class test accuracy for the three values of S is larger than the flat accuracy, indicating unsupervised generated superclasses are interpretable for classification. The super accuracy decreases monotonically with the increase in S. The result indicates that the overall accuracy has a negative correlation with the speed-up coefficient.

For ImageNet-1k we adopt 7 values for S, from 10 to 100, since $K = 1000$. From the perspective of efficiency, the speed-up coefficient reaches a maximum when $S = 50$ which is a little higher than the square root of K. The efficiency approximately conforms to the theoretical derivation. The accuracy and super accuracy reach their maximum when $S = 10$ and are negatively correlated with S. The result follows a similar pattern to the result of ImageNet-100. Even though hierarchical PPCA for $T = 1$ can be computationally efficient, its accuracy doesn't surpass the flat classification.

For ImageNet-10k, considering that $K = 10450$, we consider the following S values: 50, 100, 200, 300. From the perspective of efficiency, the speed-up coefficient reaches the optimum of 43.5 when $S = 100$. The hierarchical PPCA reaches the best super accuracy of 0.306 when $S = 50$.

Table 5.4: Evaluation of speed-up and accuracy for different values of the parameter T from Algorithm 1.

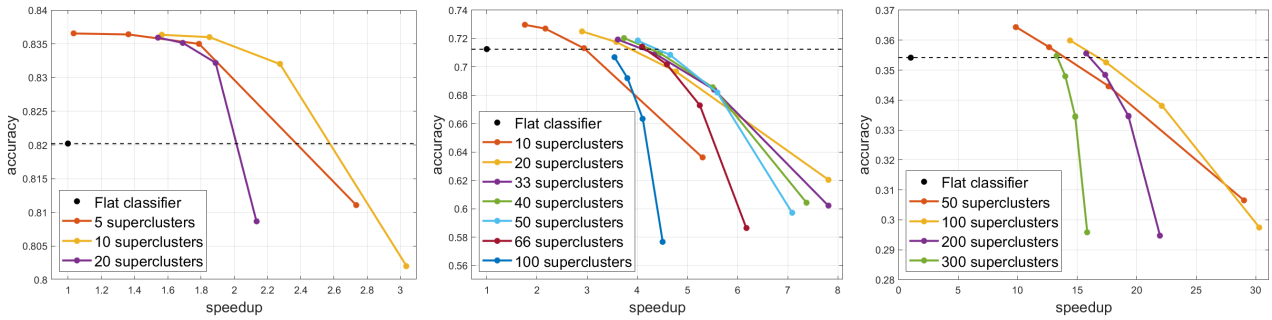
T	Density	Speed-up	Accuracy	Super Accuracy
ImageNet-100, $S = 10$, $q = r = 20$				
Flat	1.0	1.0	0.820	-
1	0.229	4.4	0.802	0.921
2	0.339	2.9	0.832	0.975
3	0.440	2.3	0.835	0.988
4	0.540	1.9	0.836	0.993
ImageNet-1k, $S = 33$, $q = r = 20$				
Flat	1.0	1.0	0.713	-
1	0.095	10.5	0.602	0.753
2	0.148	6.8	0.687	0.897
3	0.197	5.1	0.711	0.946
4	0.247	4.0	0.721	0.969
ImageNet-10k, $S = 100$, $q = r = 20$				
Flat	1.0	1.0	0.354	-
1	0.023	43.5	0.298	0.665
2	0.035	28.4	0.338	0.800
3	0.047	21.1	0.353	0.858
4	0.060	16.8	0.360	0.891

In summary, the S experiments indicate the efficiency of the Hierarchical PPCA for $T = 1$ reaches an optimum when the $S = \sqrt{K}$. Overall, the accuracy and super accuracy have a negative correlation with S. The overall accuracy of Hierarchical PPCA is slightly inferior to flat classification. The super accuracy results indicate that super-classes are interpretable for classification.

5.2.2 Improving the Super Accuracy.

Hierarchical PPCA assigns the samples to the T most likely super-classes. The failure of super-classification significantly damages the overall classification accuracy. The S experiment indicates that Hierarchical PPCA is inferior to flat classification on overall accuracy when $T = 1$. The overall error is composed of errors from the image classification and errors from the super-class classification. The Hierarchical PPCA can increase its super-class accuracy by assigning samples to multiple most likely clusters. Experiments aim to explore whether increasing T benefits the overall accuracy and what is the impact on the classification speed.

Table 5.4 shows the effectiveness of this strategy. The overall accuracy increases monotonically with T. For ImageNet-100 , Hierarchical PPCA starts to surpass flat classification on overall accuracy when $T = 2$. Meanwhile, the super-class accuracy increases significantly, from 0.92 to



ImageNet-100 ImageNet-1k ImageNet-10k

Figure 5.1: Classification accuracy vs speed-up for different hierarchical classifiers.

0.975. Compared to flat classification, Hierarchical PPCA can reach better accuracy while only using 22% of the PPCA neurons. Table 5.4 also indicates that the density increases linearly with T.

The results on ImageNet-1k with $S = 33$ follow a similar pattern. The accuracy and super accuracy follow a positive correlation with T. The overall accuracy increased from 0.601 to 0.721 as T increased to 2. The Hierarchical PPCA starts to surpass the performance of flat classification when T reaches 3 while the density is 0.247. The Hierarchical PPCA can achieve superior performance with less than 25% of neurons.

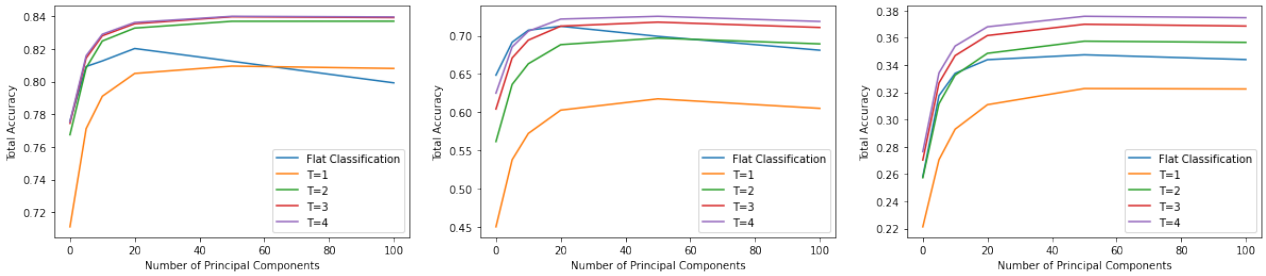
The experiment on ImageNet-10k is conducted with $S = 100$. The overall accuracy increased from 0.298 to 0.36 as T increased to 4. The hierarchical accuracy starts to surpass the flat accuracy when T increases to 3 while the density is 0.047. The density increases linearly with T.

In summary, the strategy of increasing T elevates the super-class accuracy and overall accuracy by sacrificing some computation efficiency. The experimental results indicate that for medium datasets like ImageNet-100 and ImageNet-1k, Hierarchical PPCA can achieve equivalent results with not more than 20% of the neurons used. The computational cost increases approximately linearly with increasing T.

Figure 5.1 reveals the relationship between accuracy and speed-up for different numbers S of super-classes. The relationship between accuracy and speed up coefficient follows a negative logarithmic trend. Compared to flat classification, the hierarchical classification can obtain a better accuracy with between 2 times and 15 times speed up, depending on the dataset.

Table 5.5: Accuracy vs. number q of principal components.

			q					
Method	S	T	0	5	10	20	50	100
ImageNet-100								
flat classification	-	-	0.776	0.809	0.813	0.820	0.812	0.799
Hierarchical PPCA, $r = q$	10	1	0.711	0.723	0.791	0.805	0.810	0.808
Hierarchical PPCA, $r = q$	10	4	0.775	0.816	0.829	0.836·	0.840	0.840
ImageNet-1k								
flat classification	-	-	0.648	0.692	0.707	0.713	0.699	0.681
Hierarchical PPCA, $r = q$	33	1	0.449	0.536	0.572	0.603	0.617	0.605
Hierarchical PPCA, $r = q$	33	4	0.625	0.685	0.706	0.722	0.726	0.719
ImageNet-10k								
flat classification	-	-	0.288	0.335	0.349	0.354	0.348	0.333
Hierarchical PPCA, $r = q$	100	1	0.221	0.271	0.293	0.311	0.323	0.322
Hierarchical PPCA, $r = q$	100	4	0.277	0.334	0.354	0.368	0.376	0.375



ImageNet-100 ImageNet-1k ImageNet-10k

Figure 5.2: Classification accuracy vs speed-up for different numbers of principal components $q = r$.

5.2.3 Number of Principal Components.

The number of principal components (PC) is an essential hyperparameter. Theoretically, q represents the dimension of the (linear) manifold to the observations of each class belonging to. Also considered is the case $q = 0$, in which the PPCA neurons will degenerate into RBF neurons

$$r_k(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}_k\|^2, \tag{5.2}$$

where classification is based on the nearest Euclidean distance to the mean $\boldsymbol{\mu}_k$ instead of the Mahalanobis distance (2.7). The experiment aims to explore how the number of principal components influences accuracy.

Actually, there are two PC parameters: the number q of PCs in the PPCA image models and the number r of PCs in the super-class PPCA models.

Table 5.6: Accuracy vs. number r of principal components for super classification.

			r					
Method	S	T	0	5	10	20	50	100
ImageNet-100								
flat classification	-	-	0.776	0.809	0.813	0.820	0.812	0.799
Hierarchical PPCA, $q = 20$	10	4	0.833	0.836	0.836	0.836	0.836	0.836
Hierarchical PPCA, $q = r$	10	4	0.775	0.816	0.829	0.836·	0.840	0.840
ImageNet-1k								
flat classification $q = r$	-	-	0.648	0.692	0.707	0.713	0.699	0.681
Hierarchical PPCA, $q = 20$	33	4	0.701	0.716	0.721	0.724	0.726	0.726
Hierarchical PPCA, $q = r$	33	4	0.625	0.685	0.706	0.722	0.726	0.719
ImageNet-10k								
flat classification $q = r$	-	-	0.288	0.335	0.349	0.354	0.348	0.333
Hierarchical PPCA, $q = 20$	100	4	0.351	0.363	0.365	0.367	0.369	0.369
Hierarchical PPCA, $q = r$	100	4	0.277	0.334	0.354	0.368	0.376	0.375

Experiment 1, $\mathbf{r = q}$. In a first experiment, we set $r = q$ and vary q. Table 5.5 and Figure 5.2 present the accuracy for different numbers q of PC used in both the image class and the super-class models ($r = q$). The overall accuracy of Hierarchical PPCA and flat classification doesn't increase with q monotonically. There is an optimum number of principal components where overall accuracy reaches a maximum. This optimum point indicates the classifier's capacity of utilizing variation to improve the classification. The optimum point for flat classification is $q = 20$ while for Hierarchical PPCA, the optimum point is $q = r = 50$. The results of the optimum point indicate that Hierarchical PPCA has a larger capacity to utilize variation. Figure 5.2 indicates that Hierarchical PPCA with $T = 4$ starts to surpass the flat classification on accuracy where q reaches 10. With more variation encoded, the Hierarchical PPCA performs better than the flat classifier. When $q = 0$, the PPCA neurons degrade to RBF neurons. Table 5.5 shows that PPCA neurons are superior to RBF neurons for both flat and hierarchical classifiers.

Experiment 2, changing r when $\mathbf{q = 20}$. Super-classes represent more general concepts in hierarchical PPCA. They are generated by the minimization of the sum of the distance to the image classes. In Experiment 1 above, we explored the relationship between the number of principal components for all PPCA neurons ($q = r$) and the overall accuracy. However, the super-classes may have different semantic properties than image classes. This experiment is designed to explore how the amount of variation encoded for super-classes influences the overall accuracy. We restrict the number of PCs q for the image classes to $q = 20$, which is the optimum point for flat classifiers while changing the number r of PCs for the super-classes.

Table 5.7: Hierarchical PPCA accuracy vs. number r of principal components for super classification, when $T = 4$.

Number of PCs r for super classification	0PC	1PC	2PC	3PC
ImageNet-100 Overall Accuracy	0.833	0.836	0.836	0.836
ImageNet-100 Super Accuracy	0.993	0.997	0.998	0.997
ImageNet-1k Overall Accuracy	0.701	0.708	0.710	0.712
ImageNet-1k Super Accuracy	0.936	0.945	0.947	0.950
ImageNet-10k Overall Accuracy	0.351	0.359	0.361	0.362
ImageNet-10k Super Accuracy	0.902	0.916	0.921	0.924

Table 5.6 indicates that the overall accuracy doesn't change significantly with the number r of PCs for the super-classes. By comparison with the other experiment and with flat classification, we can conclude that changing the variation encoded for super-classes doesn't influence the classification much. Super-classes have a different semantic property from image classes and the super classification is not sensitive to changing the number of principal components like image classes.

To reveal the semantic characteristics of PPCA neurons, we explore the overall accuracy when a small amount of variation is encoded for super-classes, i.e. $r \leq 3$, keeping $q = 20$ and $T = 4$. Table 5.7 reveals the performance of PPCA neurons is better than RBF neurons for super classification. For medium datasets like ImageNet-100, the performance reaches the optimum when $r = 1$. For large-scale datasets like ImageNet-1k and ImageNet-10k, more encoded information facilitates the super classification thus improving the overall accuracy.

5.3 Conclusion

This chapter introduced a framework for image classification called Hierarchical PPCA, aimed at classifying data with a large number of classes. The framework adopts probabilistic PCA as class models for the classifier and clusters the image classes using a modified k-means approach into a small number of super-classes. During classification, the hierarchical model first classifies the data into a small number of super-classes, then only activates the corresponding image class models after super-classification. For large-scale datasets without hierarchical annotation, the hierarchical PPCA can achieve superior accuracy with a fraction of the computational cost.

Compared with RBF neurons, PPCA neurons are capable of modeling the complexity of semantic variation to gain superior classification accuracy. The Hierarchical PPCA has a stronger capacity to utilize variation to aid in classification and computation efficiency. We have noticed

that with fewer training samples, the Hierarchical PPCA can achieve considerable performance for large-scale datasets.

BIBLIOGRAPHY

Arthur, D. and Vassilvitskii, S. (2006a). How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153.

Arthur, D. and Vassilvitskii, S. (2006b). k-means++: The advantages of careful seeding. Technical report, Stanford.

Bergman, T. J., Beehner, J. C., Cheney, D. L., and Seyfarth, R. M. (2003). Hierarchical classification by rank and kinship in baboons. *Science*, 302(5648):1234–1236.

Bishop, C. M. (1994). Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832.

Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Day, W. H. and Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee.

Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. (2019). Learning without memorizing. In *CVPR*, pages 5138–5146.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102. Springer.

Duchi, J. (2007). Derivations for linear algebra and optimization. *Berkeley, California*, 3(1):2325–5870.

Facebook (2022). SWSL. https://github.com/facebookresearch/semi-supervised-ImageNet1K-models. [Online; accessed 09-02-2022].

Garcez, A. S., Lamb, L. C., and Gabbay, D. M. (2008). *Neural-symbolic cognitive reasoning*. Springer Science & Business Media.

Gardenfors, P. (2004). Conceptual spaces as a framework for knowledge representation. *Mind and matter*, 2(2):9–27.

Gowda, K. C. and Krishna, G. (1978). Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition*, 10(2):105–112.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839.

Jeon, H.-A. (2014). Hierarchical processing in the prefrontal cortex in a variety of cognitive domains. *Frontiers in systems neuroscience*, 8:223.

Jia, Y., Abbott, J. T., Austerweil, J. L., Griffiths, T., and Darrell, T. (2013). Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies. *Advances in Neural Information Processing Systems*, 26.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kumar, S. and Zheng, R. (2017). Hierarchical category detector for clothing recognition from visual data. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2306–2312.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE Trans. on PAMI*, 40(12):2935–2947.

Marszalek, M. and Schmid, C. (2007). Semantic hierarchies for visual object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE.

Minsky, M. L. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34–34.

Newman, S. D., Ikuta, T., and Burns Jr, T. (2010). The effect of semantic relatedness on syntactic analysis: an fmri study. *Brain and language*, 113(2):51–58.

OpenAI (2022). CLIP. `https://github.com/openai/CLIP`. [Online; accessed 09-02-2022].

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Srivastava, N. and Salakhutdinov, R. R. (2013). Discriminative transfer learning with tree-based priors. *Advances in neural information processing systems*, 26.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

Tousch, A.-M., Herbin, S., and Audibert, J.-Y. (2012). Semantic hierarchies for image annotation: A survey. *Pattern Recognition*, 45(1):333–345.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019). Large scale incremental learning.

Zagoruyko, S. and Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *CoRR*, abs/1612.03928.

Zhang, X., Wan, F., Liu, C., Ji, X., and Ye, Q. (2021). Learning to match anchors for visual object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3096–3109.

Zweig, A. and Weinshall, D. (2007). Exploiting object hierarchy: Combining models from different category levels. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE.

BIOGRAPHICAL SKETCH

Boshi Wang received his Bachelor's degree in Statistics in 2016 from the Southwest University of Finance and Economics. Then he joined the Statistics Department at the Florida State University to pursue his Master's degree. In 2018, he passed the qualifying exam and joined the Ph.D. program. Since then, he worked on research in computer vision that included incremental learning, and hierarchical classification. In 2022, with his Ph.D. advisor Dr. Barbu, he published a paper titled 'Scalable Learning with Incremental Probabilistic PCA' in the IEEE International Conference on Big Data.