LEARNING EFFICIENT VISUAL EMBEDDING MODELS UNDER DATA CONSTRAINTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Mert Bülent Sarıyıldız
September 2019

LEARNING EFFICIENT VISUAL EMBEDDING MODELS UNDER DATA CONSTRAINTS

By Mert Bülent Sarıyıldız

September 2019

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Selim Aksoy(Advisor)

R. Gökberk Cinbiş(Co-Advisor)

Emre Akbaş

A. Ercüment Çiçek

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

LEARNING EFFICIENT VISUAL EMBEDDING MODELS UNDER DATA CONSTRAINTS

Mert Bülent Sarıyıldız
M.S. in Computer Engineering
Advisor: Selim Aksoy
Co-Advisor: R. Gökberk Cinbiş
September 2019

Deep learning models require large-scale datasets to learn rich sets of low and mid-level patterns and high-level semantics. Therefore, given a high-capacity neural network, one way to improve the performance of a model is increasing the size of the dataset which the model is trained over on. Considering that it is easy to get the amount of computational power required to train a network, data becomes a serious bottleneck in scaling up the existing machine learning pipelines. In this thesis, we look into two main data bottlenecks that rise in computer vision applications: I. the difficulty of finding training data for diverse sets of object categories, II. the complication of utilizing data containing sensitive user information for the purpose of training neural network models. To address these issues, we study zero-shot learning and decentralized learning schemes, respectively.

Zero-shot learning (ZSL) is one of the most promising problems where substantial progress can potentially be achieved through unsupervised learning, due to distributional differences between supervised and zero-shot classes. For this reason, several works investigate the incorporation of discriminative domain adaptation techniques into ZSL, which, however, lead to modest improvements in ZSL accuracy. In contrast, we propose a generative model that can naturally learn from unsupervised examples, and synthesize training examples for unseen classes purely based on their class embeddings, and therefore, reduce the zero-shot learning problem into a supervised classification task. The proposed approach consists of two important components: I. a conditional Generative Adversarial Network that learns to produce samples that mimic the characteristics of unsupervised data examples, and II. the Gradient Matching (GM) loss that measures the quality of the gradient signal obtained from the synthesized examples. Using our GM

loss formulation, we enforce the generator to produce examples from which accurate classifiers can be trained. Experimental results on several ZSL benchmark datasets show that our approach leads to significant improvements over the state of the art in generalized zero-shot classification.

Collaborative learning techniques provide a privacy-preserving solution, by enabling training over a number of private datasets that are not shared by their owners. However, recently, it has been shown that the existing collaborative learning frameworks are vulnerable to an active adversary that runs a generative adversarial network (GAN) attack. In this work, we propose a novel classification model that is resilient against such attacks by design. More specifically, we introduce a key-based classification model and a principled training scheme that protects class scores by using class-specific private keys, which effectively hides the information necessary for a GAN attack. We additionally show how to utilize high dimensional keys to improve the robustness against attacks without increasing the model complexity. Our detailed experiments demonstrate the effectiveness of the proposed technique.

Keywords: zero-shot learning, meta learning, generative models, privacy-preserving machine learning, collaborative learning, classification, generative adversarial networks.

ÖZET

VERİ KISITLAMALARI ALTINDA VERİMLİ GÖRÜNTÜ GÖMME MODELLERİNİ ÖĞRENME

Mert Bülent Sarıyıldız
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Selim Aksoy
İkinci Tez Danışmanı: R. Gökberk Cinbiş
Eylül 2019

Derin öğrenme modelleri, düşük ve orta seviye örüntüleri ve üst seviye anlambilimden oluşan zengin kümeleri öğrenmek için büyük ölçekli veri kümelerini gerektirir. Bu nedenle, yüksek kapasiteli bir sinir ağı göz önüne alındığında, bir modelin performansını iyileştirmenin bir yolu, modelin üzerinde çalıştığı veri kümesinin boyutunu artırmaktır. Bir ağı eğitmek için gereken hesaplama gücü miktarını elde etmenin kolay olduğu göz önüne alındığında, veriler mevcut makine öğrenme çözümlemelerinin yükseltilmesinde ciddi bir tıkanıklık haline gelir. Bu tezde, bilgisayarlı görme uygulamalarında ortaya çıkan iki ana veri darboğazını inceliyoruz: (I) çeşitli nesne kategorileri kümesinde eğitim verisi bulma zorluğu, (II) sinir ağı modellerini eğitim amacıyla hassas kullanıcı bilgisi içeren verilerin kullanılmasının zorluğu. Bu konuları ele almak için sırasıyla sıfır örnek ile öğrenme ve merkezi olmayan öğrenme şemalarını inceliyoruz.

Sıfır örnek ile öğrenme, denetimli ve sıfır örnekli sınıfları arasındaki dağılım farklılıkları nedeniyle denetimsiz öğrenme yoluyla önemli ilerlemenin potansiyel olarak gerçekleştirilebileceği en umut verici sorunlardan biridir. Bu nedenle, birkaç çalışma, ayrımcı alan uyarlama tekniklerinin sıfır örnek ile öğrenmeye dahil edilmesini araştırmaktadır; ancak bu, sıfır örnek ile öğrenmenin doğruluğunda mütevazi gelişmelere yol açmaktadır. Buna karşılık, denetlenmeyen örneklerden doğal olarak öğrenebilecek ve görünmeyen sınıflara yönelik eğitim örneklerini yalnızca sınıf gömülerine dayanarak sentezleyen bir üretici modeli öneriyoruz. Bu nedenle, sıfır örnek ile öğrenme problemini denetlenen bir sınıflandırma görevine indirgiyoruz. Önerilen yaklaşım iki önemli bileşenden oluşmaktadır: (I) denetlenmemiş veri örneklerinin özelliklerini taklit eden örnekler üretmeyi öğrenen koşullu bir üretici grup ağı ve (II) sentezlenen örneklerden elde edilen gradyan sinyalinin kalitesini ölçen gradyan eşleşmesi kaybı. Gradyan eşlemesi

kayıp formülasyonumuzu kullanarak, üreticiyi doğru sınıflandırıcıların eğitilebileceği örnekler üretmesi için zorlarız. Çeşitli sıfır örnek ile öğrenme ölçütü veri seti üzerinde elde edilen deneysel sonuçlar, yaklaşımımızın gelişmiş sıfır örnek ile öğrenme modellerine göre genelleştirilmiş sıfır örnekli sınıflandırmada önemli gelişmelere yol açtığını gösteriyor.

İşbirlikçi öğrenme teknikleri, sahipleri tarafından paylaşılmayan bazı özel veri kümeleri üzerinde eğitim sağlayarak gizlilik koruma çözümü sunar. Bununla birlikte, son zamanlarda, mevcut işbirlikçi öğrenme çerçevelerinin üretken bir çekişmeli ağ saldırısı yapan aktif bir düşmana karşı savunmasız olduğu gösterilmiştir. Bu çalışmada, bu tür saldırılara karşı tasarım açısından dayanıklı yeni bir sınıflandırma modeli öneriyoruz. Daha belirgin olarak, bir çekişmeli üretken ağ saldırısı için gerekli bilgileri etkin bir şekilde saldırgandan saklayan, sınıfa özgü özel anahtarlar kullanarak sınıf puanlarını koruyan bir anahtar tabanlı sınıflandırma modeli ve ilkeli bir eğitim programı sunuyoruz. Ayrıca, model karmaşıklığını arttırmadan saldırılara karşı sağlamlığı artırmak için yüksek boyutlu anahtarların nasıl kullanılacağını gösteriyoruz. Detaylı deneylerimiz önerilen tekniğin etkinliğini göstermektedir.

Anahtar sözcükler: Sıfır örnek ile öğrenme, meta öğrenme, üretken modeller, gizliliği koruyan makine öğrenmesi, işbirlikçi öğrenme, sınıflandırma, çekişmeli üretken ağlar.

Acknowledgement

I consider myself lucky as I had a chance to know and work with Dr. R. Gökberk Cinbiş. It had been a great pleasure to study with him as his student and to explore with him as his co-worker. I am grateful to him for his guidance, support, patience and most importantly trust in me.

I give my special thanks to Dr. Selim Aksoy who kindly agreed to be my supervisor in Bilkent University, Dr. Erman Ayday for his precious contributions to the second project we conducted in the context of this thesis and Dr. Emre Akbaş and Dr. A. Ercüment Çiçek for being in my thesis committee.

I would like to thank all my roommates in EA-427 including Caner, Ali Burak, Onur, Bulut, Gencer, Yarkın with whom I spent enjoyable time and Ebru Ateş and Nebahat Karakaya who were always kind to me and helped me countless times.

Also I would like to thank Dr. Cihan Topal for inspiring me to research in computer vision and Dr. Sermetcan Baysal, who is the husband of my dear friend İpek Baysal, for encouraging me to follow this adventure.

Finally, I would like to thank the Computer Engineering Department of Bilkent University and TÜBİTAK (Scientific and Technological Research Council of Turkey) for providing funding for my studies. This work was supported in part by the TÜBİTAK Grant 116E445. The numerical calculations reported were partially performed at TÜBİTAK ULAKBİM High Performance and Grid Computing Center, the servers of Bioinformatics and Computational Genomics Group at Bilkent University, and the servers of ImageLab at Middle East Technical University.

Of course, this was all possible thanks to the endless support and love of the people behind me, *my family*, especially my mom and Merve. I am indebted to them.

to *Mervet* —

Contents

1 Introduction **1**

1.1 Data Bottleneck in Learning to Recognize Real-Life Objects . . . 2

1.2 Data Bottleneck in Learning Under Privacy Constraints 4

1.3 Contributions 7

1.4 Outline 8

2 Literature Review **9**

2.1 Zero-shot Learning 9

2.2 Privacy Preserving Machine Learning 10

3 Gradient Matching Networks for Zero-shot Learning **14**

3.1 Preliminaries 15

3.2 Model 16

3.2.1 Unsupervised GAN 16

3.2.2 Gradient Matching Loss 18

3.2.3 Supervision by Conditional Discriminator 23

3.2.4 Feature Synthesis 23

3.3 Experiments 24

4 Key-Protected Classification for Collaborative Learning 36

4.1 Background 37

4.1.1 Collaborative Learning of Participants 37

4.1.2 Generative Adversarial Network 39

4.1.3 GAN Attack in Collaborative Learning 40

4.2 Model 41

4.2.1 Key Protected Classification Model 41

4.2.2 Learning with Restricted Class Key Access 43

4.2.3 Class Key Generation 45

4.2.4 Learning with High Dimensional Keys 47

4.3 Experiments 49

4.3.1 Experimental setup 49

4.3.2 Collaborative Learning Evaluation 50

4.3.3 Preventing GAN Attack 52

4.3.4 Shared Classes Among Participants 57

4.3.5 Evaluating the Key Based Regression Loss 59

5 Conclusions **61**

A Softmax Over Infinitely-many Classes **63**

List of Figures

1.1 Illustration of turning zero-shot learning into supervised N-way classification problem. 3

3.1 Illustration of the wiring difference between the attribute concatenation (AC) and the latent noise (LN) types of generator networks. 17

3.2 Illustration of the compute chain in the Gradient Matching Network. 22

3.3 Analysis of GMN regarding impact of the number of synthesized features on (from left to right) **T-1**, **u** and **s** scores on CUB, SUN and AWA datasets. 32

3.4 Analysis of cWGAN, f-CLS-WGAN-Bilinear, cycle-WGAN-Bilinear and GMN models regarding impact of the number of synthesized features on **h** scores on CUB and AWA datasets. 33

3.5 Examples from unseen CUB classes that are misclassified with a cWGAN-based classifier but correctly recognized when we include $\mathcal{L}_{\text{GM}}$ into generator training. 34

3.6 **u** scores of validation samples from the SUN dataset over the training iterations of $\mathcal{L}^{\text{S}}_{\text{WGAN}}$ (GMN_{in}§), $\mathcal{L}_{\text{GM}}$ (GMN_{in}‡), $\mathcal{L}^{\text{S}}_{\text{WGAN}} + \mathcal{L}_{\text{GM}}$ (GMN_{in}) and $\mathcal{L}^{\text{S+U}}_{\text{WGAN}} + \mathcal{L}_{\text{GM}}$ (GMN_{tr}). 35

4.1 The comparison of the compute chains constructed during the vanilla (left, a) and the modified key-protected (right, b) collaborative learning frameworks. 38

4.2 The correlation between two random vectors with respect to their dimension d. 47

4.3 Mean participant accuracies obtained in collaborative learning with 2, 3, and 5 participants over the MNIST and the Olivetti Faces datasets. 51

4.4 Demonstration of GAN attack when the attacker has access to the exact key of the class that it is attacking, on the (a) MNIST and (b) Olivetti Faces datasets. 54

4.5 Approximating the maximum Euclidean distance between any class key and $\psi(c_{\text{attack}})$ necessary for the adversary to succeed in attack. 55

4.6 GAN attack results on the MNIST dataset using random attack keys. 56

4.7 GAN attack results on the Olivetti Faces dataset using random attack keys. 56

4.8 Analysis of keys when multiple participants use different class keys for the same class. 58

List of Tables

3.1 Statistics for the benchmark datasets used in zero-shot learning experiments. 24

3.2 Quantitative evaluation of GMN over the strong baselines. 28

3.3 Comparison of GMN against the baselines and the state-of-the-art on the benchmarks from [1]. 30

4.1 Key-based regression versus cross entropy. 60

Chapter 1

Introduction

In order to tackle a particular machine learning problem efficiently, it is crucial to extract descriptive high-level representations of the input data [2]. This, *i.e.* learning a feature extractor that can *generalize* comprehensively to the input space for the problem of interest, often requires processing *tons of* data with *expressive* models. However, while we can easily increase the complexity of embedding models to learn more powerful and sophisticated feature extractors, collecting large datasets is a bottleneck in developing machine learning-based solutions for a diverse set of problems.

In this study, we focus on two of the main causes of data bottlenecks in learning reliable and robust visual embedding models. We argue that in most cases, I. it is difficult to *prepare* training data for every possible objects which we want a visual embedding model to be generalized to, and II. it is not always possible to gather a publicly-available dataset to learn a visual embedding model for a specific problem when the *privacy* of subjects is of concern. We treat these two problems independently and propose a novel methodology for each one that improves the state-of-the-art in its respective domain. In the following sections, we elaborate more on these problems and introduce our approaches.

1.1 Data Bottleneck in Learning to Recognize Real-Life Objects

There has been tremendous progress in visual recognition models over the past several years, primarily driven by the advances in deep learning. In particular, convolutional neural networks wired in deep architectures [3, 4, 5, 6] have led to remarkable performances in object recognition tasks. The state-of-the-art approaches in deep learning, however, predominantly rely on the availability of a large set of carefully annotated training examples. For instance, the excellent classification models trained on the ILSRVC datasets utilize over a thousands of labeled training examples *per class* [7]. The need for such large-scale datasets poses a significant bottleneck against building comprehensive recognition models of the visual world, especially due to the long-tailed distribution of object categories [8].

Recently, there has been a significant research interest in overcoming this difficulty. Prominent approaches for this purpose include semi-supervised learning, *i.e.* improving supervised classification through leveraging unlabeled data [9, 10], few-shot learning, *i.e.* learning from few labeled samples [11, 12] and zero-shot learning (ZSL) for modeling novel classes without training samples [13, 14, 15]. In the first work we performed within the context of this thesis, we focus on the ZSL problem, where the goal is to extrapolate a classification model learned from *seen classes*, *i.e.* those with labeled examples, to *unseen classes* with no labeled training samples. In order to relate classes to each other, they are commonly represented as *class embedding* vectors constructed from side information. Such class embedding vectors can be constructed in several different ways, such as by manually defining attributes that characterize visual and semantic properties of objects [16, 17, 18, 19, 20, 21, 22, 23, 24], by adapting vector-space embeddings of class names [25, 26, 27], or by representing the position of classes in a relevant taxonomy tree as vectors [28]. Given the class embeddings, the ZSL problem boils down to modeling relations between a visual feature space, *i.e.* images or features extracted from some deep convolutional network, and a class embedding

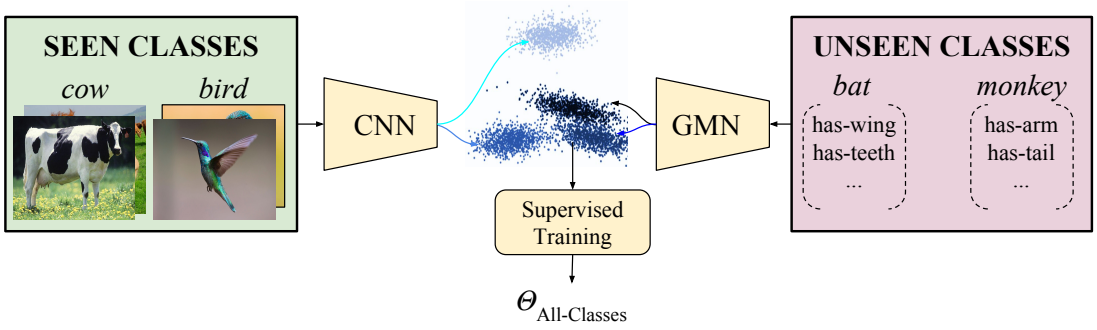


Figure 1.1: Illustration of turning zero-shot learning into supervised N-way classification problem. By using a generative model, we can generate training samples for zero-shot (*unseen*) classes, then train a supervised classifier over the union of this synthetic set and the training set of *seen* classes.

space [29, 28, 30, 31, 32, 33, 34, 35].

However, ZSL models typically suffer from the domain shift problem [36] due to distributional differences between seen and unseen classes. This can significantly limit the *generalized zero-shot learning* (GZSL) accuracy where test samples may belong to any of the seen or unseen classes [37]. Towards addressing this problem, several recent work have proposed generative models that can synthesize training samples for unseen classes and learn a classifier from real and/or synthesized examples [38, 39, 40, 41]. Therefore, a bias towards seen classes can be reduced considerably.

Similarly, in this work, our goal is to learn a generative model that can synthesize samples for any class of interest, purely based on the embedding vector of the class. Once the generative model is learned, we augment the set of seen class examples by the set of unseen class examples sampled from the generative model. The final classification model is then built by training a classifier over the real and synthetic training examples. Therefore, in a sense, we reduce ZSL to a supervised learning problem, as illustrated in Fig 1.1.

Just like any other example-synthesis based ZSL approach, however, the accuracy of the resulting classifier heavily depends on the diversity and fidelity of the

training examples synthesized by the generative model. For this reason, we specifically focus on two directions: I. leveraging unseen examples to implicitly model the manifold of each unseen class, II. ensuring that generative model produces data using which we can train an accurate classifier.

In order to leverage unlabeled examples, we propose as a Generative Adversarial Network (GAN) [42] based formulation. In particular, in contrast to recent GAN-based example synthesis approaches [39, 40, 41], our approach allows utilizing an unconditional GAN discriminator, which naturally extends to incorporating unlabeled training examples. In this way, we aim to learn a generator that produces samples that mimic the characteristics of both seen and unseen classes.

In order to learn to generate better training data, we propose a novel loss function that behaves as a quality inspector on the synthetic samples. More specifically, we aim to guide the generator towards minimizing the classification loss of synthetic example-driven classification models. For this purpose, we derive the *gradient matching loss* as a proxy for the classification loss, which measures the discrepancy between the gradient vectors obtained using the real versus synthetic samples. We refer to our complete model that incorporates this loss term as *Gradient Matching Network* (GMN).

We show that our final classification models lead to state-of-the-art results on ZSL benchmark datasets Caltech-UCSD Birds (CUB) [43], SUN Attributes (SUN) [44] and Animals with Attributes (AWA) [18] using the challenging and realistic Generalized ZSL (GZSL) evaluation protocols [37, 1].

1.2 Data Bottleneck in Learning Under Privacy Constraints

Most deep learning approaches rely on training over large-scale datasets and computational resources that makes the utilization of such datasets possible. While

large-scale public datasets, such as ImageNet [45], Celeb-1M [46], and YouTube-8M [47], have a fundamental role in deep learning research, it is typically difficult to collect a large-scale dataset for problems that involve processing of sensitive information. For instance, data privacy becomes a significant concern if one considers training models over personal messages, pictures or health records.

To enable training over large-scale datasets without compromising data privacy, decentralized training approaches, such as *collaborative learning framework* (CLF) [48], *federated learning* [49], *personalized learning* [50, 51] approaches have been proposed. These training schemes enable multiple parties (private data holders) to train a single neural network model without sharing their sensitive, private data with each other.

In the second work of this thesis, we consider improving privacy protection mechanism of CLFs. In a CLF, a target model is trained in a distributed way, where each *participant* contributes to training without sharing its (sensitive) data with other participants. More specifically, each participant hosts only its own training examples, and a central server, called the *parameter server*, combines local model updates into a shared model. Therefore, the training procedure effectively utilizes the data owned by all participants. At the end, the final model parameters are shared with all participants.

However, there are cases where the original CLF approach fails to preserve data privacy due to the knowledge embedded in the final model parameters. In particular, [52] shows that the parameters of a neural network model trained on a dataset can be exploited to partially reconstruct the training examples in that dataset, which is called a *passive attack*. To mitigate this threat, one may consider partially corrupting the model parameters by adding noise into the final model [53]. The study by [48] also shows that differential privacy [54] can be incorporated into CLF in a way that guarantees the indistinguishability of the participant data by perturbing parameter updates during training. However, such prevention mechanisms may introduce a difficult trade-off between classifier accuracy versus data privacy level for training. Several other differential privacy based approaches, which introduce noise injection methods [55, 56] or training

frameworks [57], have recently been proposed.

It has been shown that collaborative learning approaches can also be vulnerable to *active attacks i.e.*, training-time attacks [58]. More specifically, a training participant can construct a generative adversarial network (GAN) [59] such that its GAN model learns to reconstruct training examples of one of the other participants over the training iterations. For this purpose, the attacker defines a new class for the joint model, which acts as the GAN discriminator, and utilizes the samples generated by its GAN generator when locally updating the model. In this manner, the attacker effectively forces the victim to release more information about its samples, as the victim tries to differentiate its own data from attacker class during its local model updates. To the best of our knowledge no solution—other than introducing differential privacy to the CLF, has previously been proposed against the GAN attack[1].

In this work, we propose a novel classification model for collaborative learning that prevents GAN attacks by design. First, we observe that GAN attacks depend on the classification scores of the targeted classes. Based on this observation, we define a classification model where class scores are protected by class-specific keys, which we call *class keys*. Our approach generates class keys independently within each training participant and keeps the keys private throughout the training procedure. In this manner, we prevent the access of the adversary to the target classes, and therefore the GAN attack. We also demonstrate that the dimensionality of the keys directly affect the security of the proposed model, much like the length of passwords. We observe, however, that naively increasing the key dimensions can greatly increase the number of model parameters, and therefore, reduce the data efficiency and the classification accuracy. We address this issue by introducing a *fixed neural network layer* that allows us to use much higher dimensional keys without increasing the model complexity. We experimentally validate that our approach prevents GAN attacks while providing effective collaborative learning on the MNIST [60] and Olivetti Faces [61] datasets, both of which are challenging datasets in the context of privacy attacks due to their

[1][50] claims that their methodology can reduce the efficacy of the GAN attack, however leaves the analysis for a future work.

relative simplicity, and therefore, the ease of reconstructing the samples in them.

1.3 Contributions

To sum up, our contributions can be summarized as follows.

In the first part of the thesis, I. we formulate a novel proxy loss function, which we call gradient matching loss, for zero-shot learning. This new objective enables us to learn data distributions of individual classes on the embedding space of a pre-trained ResNet-101 network. This way we synthesize training samples for unseen classes and thus turn zero-shot learning into supervised classification problem by combining the synthetic training set of unseen classes with the real training set of seen classes. II. We show that this formulation, combined with the recent advances in generative modeling, achieves state-of-the-art results on the three most commonly used benchmark datasets for zero-shot learning, namely Caltech-UCSD Birds (CUB) [43], SUN Attributes (SUN) [44] and Animals with Attributes (AWA) [18].

In the second part of the thesis, I. We formalize a novel classification model for collaborative learning frameworks where we decouple the end-to-end classifier learning into the shared representation learning and the private class prediction steps secured by class keys. Making the class predictions private within each participant enables us to prevent the GAN attacks, while learning a shared image embedding model which generalizes across the private data hosted by all participants. II. We derive a principled training formulation for collaboratively learning the proposed model when participants are allowed to access only their own class keys. III. We show that high-dimensional keys can be used to improve the robustness against attacks and introduce the idea of using randomly generated fixed neural network layers to map image representations to higher-dimensional spaces without increasing the number of learnable parameters. IV. We investigate the key-based classification setup that we propose for the purpose of supervised

training where all the data is centralized. In this regard, we show that our regression-like loss formulation achieves comparable results with the discriminative cross-entropy loss on CIFAR-10/100 datasets [62].

1.4 Outline

The rest of the paper is organized as follows. Chapter 2 provides an overview of the most relevant prior works in both zero-shot learning and privacy-preserving machine learning research. Chapters 3 and 4 explain the details of and validate the Gradient Matching Networks for Zero-shot Learning and Key-Protected Classification for Collaborative Learning approaches that we propose to overcome the two notorious data bottlenecks in learning competent visual embedding models. Finally, Chapter 5 concludes the thesis with a brief discussion.

Chapter 2

Literature Review

In this section, we provide an overview of the literature in zero-shot learning and machine learning attacks.

2.1 Zero-shot Learning

Over the years, a number of ZSL approaches have been proposed. For example, [13] models the joint probability of attributes, [15] models the conditional distribution of features given attributes, [63] uses semantic knowledge bases for attribute classification, [64, 65, 66] build convex combinations of seen class classifiers, [28, 30, 31, 32, 33, 34, 35] learn a compatibility function between features and class embeddings. Similarly, [67, 68, 69] learn a mapping from semantic embeddings to visual features, and, [70, 71, 72] learn a data-driven metric for comparing similarities between features and semantic embeddings. Alternatively, transductive approaches have been proposed to benefit from unlabeled data [73, 36, 74, 69]. Such discriminative techniques, however, typically assume that each unlabeled example belongs to one of the unseen (or seen) classes, which can be an unrealistic assumption in practice.

Recently, the use of contemporary generative models in zero-shot learning settings has gained attention. [75] proposes training a conditional Variational Auto-Encoder (cVAE), that learns to generate samples according to given class embeddings. [74] extends this notion with trainable class conditional latent spaces. [41] also develops a cVAE except that their model learns a separate semantic embedding regressor/discriminator. [38] evaluates several generative models for learning to generate training examples. [40] adopts cycle consistency loss of cycle-GAN into zero-shot learning to regularize feature synthesis network. [76] uses a separate reconstructor, discriminator and classifier all targeting at visual features to remedy domain-shift problem. Slightly different from mainstream approaches, [77] introduces diffusion regularization to increase utility of features. [39] proposes a WGAN [78] based formulation that uses a discriminative supervised loss function, in addition to the unsupervised adversarial loss. In this model, the supervised loss enforces the WGAN generator to produce samples that are correctly classified according to a pre-trained classifier of seen classes.

Among the aforementioned works, [39] is the one closest to the Gradient Matching Network (defined in Chapter 3) in the sense that we also train a conditional WGAN towards synthesizing training samples. However, our approach has two major differences. First, we use the proposed gradient matching loss, which aims to directly maximize the value of the produced training examples by measuring the quality of the gradient signal obtained over the synthesized examples. Second, our model learns an unconditional discriminator *i.e.*, the discriminator network does not rely on a semantic embedding vector. This permits us to explore the incorporation of unlabeled training examples into training in a semi-supervised fashion.

2.2 Privacy Preserving Machine Learning

Attacks against machine learning mechanisms and privacy preserving machine learning methods have become a popular research area over the recent years. In [79], authors discuss different types of adversarial attacks and countermeasures

against them. In this section, we describe the most relevant attacks and countermeasures to our work.

We focus on reconstruction attacks, in which the goal of the adversary is to reconstruct the training samples of the other participants in a distributed learning setting. Fredrikson *et al.* show this threat via a passive attack, in which the adversary does not actively attack during the learning process, but it tries to reconstruct the training samples from the final model parameters [52]. In [80], authors develop passive and active inference attacks to exploit the leakage of sensitive information during the learning process. In particular, they show the risk of membership inference and attribute inference about training data (*i.e.*, infer properties that hold for a subset of the training data). In a more recent work, Hitaj *et al.* [58] devise a powerful active attack against collaborative learning frameworks. In this attack, one of the participants in the collaborative learning framework is assumed to be an adversary. The adversary tries to exploit one of the classes belonging to other participants (victim) by using a generative adversarial network (GAN) [59]. This attack is powerful in the sense that the adversary can actively influence the victim to release more details about its samples during the training process. The GAN attack is aimed for global data distribution among all clients, therefore it is challenging to run this attack for specific clients. Following this idea, Wang *et al.* [81] propose another GAN based attack, in which there is a malicious server which leaks model updates of a particular participant (the victim), and a multi-task discriminator that leads GAN. This way, the attacker may choose the victim intentionally. By doing so, the authors show how to discriminate category, reality, and client identity of input samples.

Several countermeasures have been proposed to mitigate the attribute inference attacks for machine learning mechanisms. One line of research on this direction is based on the differential privacy concept. Differential privacy proposed by Dwork *et al.* [82] aims at providing sample indistinguishability with respect to the outputs of an algorithm (or a neural network model) when its input is slightly changed. This indistinguishability criterion is then used as a proxy measure to quantitatively evaluate how well the algorithm (or the model) can protect the

privacy of a subject data. This concept is formalized for empirical risk minimization in machine learning problems by Chaudhuri *et al.* [83, 53]. Song *et al.* and Rajkumar *et al.* apply differential privacy to stochastic gradient descent-based optimization problems [84, 85, 86]. Following that, Shokri *et al.*, Abadi *et al.*, Phan *et al.* and Papernot *et al.* propose several ways of employing differential privacy for large-scale machine learning problems in the forms of I. structured noise addition [56, 48, 55] and II. 2-step training methodology [57]. Using the differential privacy concept together with trusted hardware, in [87], authors introduce a privacy-preserving deep learning framework called Myelin. Similar to differential privacy-based approaches, in [88], to preserve the privacy of training samples, authors propose using an obfuscate function to add random noise (or new samples) to the training data before using it for model generation. Although the use of differential privacy in machine learning problems has shown promising results, most of the differential privacy-based approaches offer a trade-off between the privacy and the utility of data, *e.g.* increasing the protection reduces the utility of data, and hence the performance of the algorithm. Therefore, overall, privacy-preserving machine learning without recognition performance compromise remains as an unsolved problem.

Another line of research advocates the use of decentralized training schemes for privacy. These strategies enable large-scale machine learning for scenarios, in which private datasets that contain sensitive information are hosted by multiple parties and therefore cannot be shared. There are two main streams of approaches: I. multiple parties train a single model on the fly by means of contributing to the model updates individually using their private data [48, 49] and II. multiple parties learn separate models over their private data, and then the final model is constructed by aggregating the information stored in the different models [89, 86, 90]. Besides, any of the methods considered under this line can still adopt differential privacy or secure multi-party computation techniques [91, 92], in which model updates can be encrypted before sharing them with others.

Other than these two major lines of research, the use of cryptographic tools has also been proposed for privacy-preserving machine learning [79, 93, 94, 95, 96]. Several differential privacy-based solutions have been proposed to prevent the

GAN attack, in which a noise signal is added to gradients during the learning phase in order to achieve differential privacy, and hence prevent the GAN attack [97, 98]. Different from these works, our key-protected classification model (defined in Chapter 4) provides a countermeasure against the GAN attack without requiring a trade-off for utility.

Chapter 3

Gradient Matching Networks for Zero-shot Learning

In the conventional supervised learning setup, a classification function is trained to discriminate a fixed number of object classes. However, it is not practical to employ the classifier for only a limited set of objects due to the concerns mentioned in Section 1.1. As a way to tackle this problem, zero-shot learning (ZSL) aims at developing classification functions that can generalize beyond known object categories. Following this research line, in this chapter, we propose a ZSL approach to improve the specific data bottleneck issues discussed in Section 1.1. Recently, a prior version of this work has appeared in [99].

Organization of this chapter is as follows. Section 3.1 gives the problem definition and notation used throughout the chapter, Section 3.2 details the proposed model and Section 3.3 empirically evaluates the model.

3.1 Preliminaries

In ZSL, the goal is to learn a classifier on a set of *seen* classes for which we have training samples, and then to use this function to predict the class labels of test samples belonging to *unseen* classes for which we have no training data. In addition to the conventional ZSL, in generalized zero-shot learning (GZSL), the test samples may also belong to the seen classes. To enable knowledge transfer to novel classes, one can define an auxiliary (semantic embedding) space $\mathcal{A}$, in which both seen and unseen classes can be uniquely identified. This way, the classifier can be formulated as a compatibility function $f(x, a; \theta_f) : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$, which estimates the degree of confidence that the input image (or it's embedding) $x \in \mathcal{X}$ belongs to the class represented by the embedding $a \in \mathcal{A}$, using the model with parameters θ_f. Given the compatibility function, the classifier over all classes can be constructed.

We start by defining a set of seen classes $\mathcal{Y}_{\text{s}} = \{1, \ldots, C_s\}$ and a set of unseen classes $\mathcal{Y}_{\text{u}} = \{C_s + 1, \ldots, C_s + C_u\}$ such that $\mathcal{Y}_{\text{s}} \cap \mathcal{Y}_{\text{u}} = \emptyset$ and $\mathcal{Y}_{\text{all}} = \mathcal{Y}_{\text{s}} \cup \mathcal{Y}_{\text{u}}$. For each class in $\mathcal{Y}_{\text{all}}$, there is a unique class embedding vector $a \in \mathbb{R}^{d_a}$, and we denote the set of all class embeddings by $\mathcal{A}_{\text{all}}$. Thus $\mathcal{A}_{\text{s}}$ and $\mathcal{A}_{\text{u}}$ represent embeddings of seen and unseen classes, respectively. $\mathcal{D}_{\text{train}} = \{(x, a) \,|\, x \in \mathcal{X}_{\text{s}}, a \in \mathcal{A}_{\text{s}}\}$, is the training set containing N examples, where each training example consists of an image embedding vector $x \in \mathbb{R}^{d_x}$ extracted using a CNN pre-trained on ImageNet, and a class embedding vector a that corresponds to the class embedding of the object in the image. Here, $\mathcal{X}_{\text{s}}$ denotes the set of all labeled data points. During training, our approach can optionally utilize a set of unlabeled examples, denoted by $\mathcal{X}_{\text{u}}$.

3.2 Model

3.2.1 Unsupervised GAN

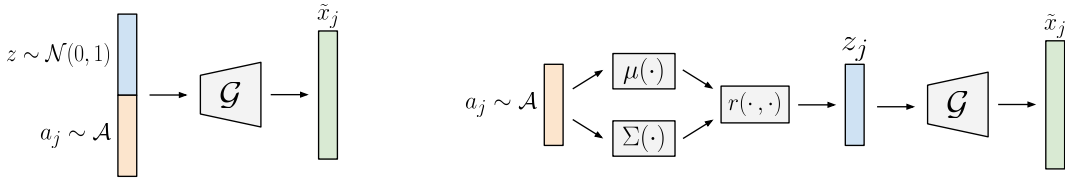
Our generative model is built upon the WGAN [100] as in [39]. Different from vanilla GAN [42], WGAN optimizes the Wasserstein distance using Kantorovich Rubinstein duality, instead of optimizing the Jensen-Shannon divergence. It is shown that enforcing discriminators to be 1-Lipschitz provides more stable gradients for generators. Even though clipping the weights of discriminators serves this purpose, it leads to unstable training for the WGAN. Instead, [78] propose to apply gradient penalty to discriminators to control their Lipschitz norm, which we use as our starting point:

$$\mathcal{L}_{\text{WGAN}} = \mathop{\mathbb{E}}_{x \sim P_r} [\mathcal{D}(x)] - \mathop{\mathbb{E}}_{\tilde{x} \sim P_g} [\mathcal{D}(\tilde{x})] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim P_{\hat{x}}} \left[(\|\nabla_{\hat{x}} \mathcal{D}(\hat{x})\|_2 - 1)^2 \right], \tag{3.1}$$

where, P_r is the true data distribution, P_g denotes generator outputs and $\hat{x}$ is the interpolation between x and $\tilde{x}$.

Note that Equation 3.1 does not involve any label information regarding either the real samples from the data distribution $x \sim P_r$ or fake ones synthesized by the generator $\tilde{x} \sim P_g$. In order to generate a sample $\tilde{x}$, a noise vector shall be sampled from a prior distribution and then fed into the generator in a purely unsupervised manner. In our case, however, we aim to produce training samples for the unseen classes using the generative model. For this purpose, we need to train a generator network such that it takes a combination of the noise vector and class embeddings as input, and therefore, produces class-specific samples according to the side information given by the class embedding.

A simple scheme for combining the noise and class embedding vectors is to concatenate them [39]. In this scheme, a noise vector is sampled from unit-Gaussian and concatenated with semantic class embeddings. However, we can also aim to model the latent distributions corresponding to classes and then take samples from these latent distributions instead. For this purpose, inspired from [74], we propose to define a conditional multivariate Gaussian distribution $\mathcal{N}(\mu(a), \Sigma(a))$,



(a) Concatenation of the semantic class embedding and noise vectors.

(b) Modeling class conditional latent noise distributions using semantic class embeddings. $r(\cdot,\cdot)$ denotes the re-parametrization [101].

Figure 3.1: Illustration of the wiring difference between the attribute concatenation (AC) and the latent noise (LN) types of generator networks.

where $\mu(a) = \mathcal{W}_{\text{mu}} a + b_{\text{mu}}$ and $\Sigma(a) = \mathbb{I}_{d_z} \exp(\mathcal{W}_{\text{log-cov}} a + b_{\text{log-cov}})$ estimate an $\mathbb{R}^{d_z}$ dimensional Gaussian noise mean and covariance conditioned on class embeddings, $\mathcal{W}_{\text{mu}}$ and $\mathcal{W}_{\text{log-cov}}$ are linear transformation matrices, b_{mu}, $b_{\text{log-cov}}$ are bias vectors, and $\mathbb{I}_{d_z}$ is a $d_z \times d_z$-dimensional identity matrix. Therefore, in order to generate a sample of class j, we first compute $\mu(a_j)$ and $\Sigma(a_j)$, take a noise sample from $\mathcal{N}(\mu(a), \Sigma(a))$ and then feed the noise into the generator network. To make the sampling process differentiable, we use the re-parameterization trick [101, 102]. In this manner, we make $\mathcal{W}_{\text{mu}}$, $\mathcal{W}_{\text{log-cov}}$, b_{mu} and $b_{\text{log-cov}}$ end-to-end differentiable and train them as an integral part of the generative network. The difference between attribute concatenation and modeling latent noise distributions are illustrated in Figure 3.1.

We conjecture that the efficiency of constructing such a latent noise space depends on at least the following factors. I. This strategy requires semantic class embeddings to be discriminative enough so that the generator can model distinct class distributions on the image embedding space. II. The number of classes whose distributions we want to model affects the structure of the latent space. Therefore, in order to maximize the performance, the choice of attribute concatenation or modeling a latent noise space can be determined on a held-out validation data. In Section 3.3 we compare both strategies.

3.2.2 Gradient Matching Loss

So far the aforementioned approach lacks definition of any supervisory signal, which is crucial for learning a correct conditional generative model (See the *Ablation Study* in Section 3.3). One possible solution is to measure the correctness of the resulting samples for the seen classes using the loss function of a pre-trained classification model, which is the approach used in [39]. However, we argue that classification guidance does not necessarily lead to the synthesis of a good training set as it measures the loss of the samples w.r.t. the pre-trained model, rather than the expected loss of the model trained on them. For instance, if the generator learns to generate only confidently classified examples, the classification loss given by the pre-trained model will be low, even though the resulting training set lacks examples near class boundaries, *i.e.* the support vectors. In fact, [103, 104] report that conditional GAN models tend to produce degenerate class conditional examples when they are trained to minimize the loss of a pre-trained classifier.

Based on these observations we propose that instead of aiming to produce samples that are correctly classified by a pre-trained model, we should focus on learning to generate training examples that lead to accurate classification models. For this purpose, one can consider training the generative model by minimizing the final loss of a tentative classification model trained over the synthetic samples. Here, the tentative classifier would be iteratively trained via a gradient based optimizer over a number of model update steps, within each training iteration for the generative model. Since all computational blocks are differentiable, such an approach would allow training the generative model in an end-to-end fashion such that it learns to generate training examples from which accurate classification models can be built.

However, based on our preliminary experiments, we have observed that such a naive strategy performs poorly for two important reasons. First, normally a large number of model update steps are needed to be able to train the tentative classifier. However, integrating a long compute chain of model update steps within the generative model training procedure not only slows down the training

procedure very significantly, but also leads to vanishing gradient problems. Second, using an unrealistically small number of classifier update steps due to the aforementioned problems, on the other hand, encourages the generative model to produce unrealistic samples that aim to "quickly" minimize the final loss over the few classification model update steps.

Instead, we address these issues by focusing on maximizing the correctness of individual model updates. We observe the simple fact that in the case where a generative model learns true class manifolds, partial derivatives of a loss function with respect to classification model parameters over a large set of synthetic examples would be highly correlated with those over a large set of real training examples.

Following these observations, we propose to minimize the approximation error of the gradients obtained over the synthetic samples of seen classes. More specifically, we propose to learn a generative model $\mathcal{G}$ such that it maximizes the correlation between gradients over the synthetic samples and those over the real samples. To formalize this idea, we first define the aliases g_r and g_s for the expected gradient vectors over the real and synthesized examples, respectively:

$$g_\mathrm{r}(\theta) = \mathop{\mathbb{E}}_{(x,a)\sim\mathcal{D}_\mathrm{s}} \left[\nabla_{\theta_f} \mathcal{L}_\mathrm{CLS}(f, x, a; \theta_f = \theta)\right], \tag{3.2}$$

$$g_\mathrm{s}(\theta) = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathcal{G}(a),a\sim\mathcal{A}_\mathrm{s}} \left[\nabla_{\theta_f} \mathcal{L}_\mathrm{CLS}(f, \tilde{x}, a; \theta_f = \theta)\right]. \tag{3.3}$$

Here, $\mathcal{L}_\mathrm{CLS}(f, x, a)$ is the loss function used in training the compatibility function $f(x, a; \theta_f)$. Throughout the training procedure, we approximate g_r and g_s over sample batches.

Since the most important information conveyed by the gradient vector is the direction towards the local minima rather than the absolute scale of the gradient vectors, we measure the discrepancy between g_r and g_s via the cosine similarity between two vectors. Finally, we formalize the *gradient matching loss* $\mathcal{L}_\mathrm{GM}$ as the expected cosine distance between the g_r and g_s, computed over all possible compatibility model parameters θ:

$$\mathcal{L}_\mathrm{GM} = \mathop{\mathbb{E}}_{\theta}\left[1 - \frac{g_\mathrm{r}(\theta)^\mathrm{T} g_\mathrm{s}(\theta)}{\|g_\mathrm{r}(\theta)\|_2 \|g_\mathrm{s}(\theta)\|_2}\right]. \tag{3.4}$$

In our experiments, we approximate the expectation by sampling θ_f vectors obtained over the training iterations while learning the compatibility function via gradient descent over real training examples. Then our final objective becomes

$$\theta^*_{\mathcal{G}}, \theta^*_{\mathcal{D}} = \arg\min_{\theta_{\mathcal{G}}, \theta_{\mathcal{D}}} \{\mathcal{L}_{\text{WGAN}} + \beta \mathcal{L}_{\text{GM}}\}, \tag{3.5}$$

where β is a simple weight hyper-parameter to be tuned on a validation set. We refer to a generative model trained within this framework as Gradient Matching Network (GMN).

Given the true generative model of the data distribution and a representative training set, the correlation between $g_{\text{r}}(\theta)$ and $g_{\text{s}}(\theta)$ is expected to be high, independent of the compatibility model parameters θ. Therefore, in principle, any compatibility function model can be utilized within the gradient matching loss. In our experiments, we use cross-entropy as $\mathcal{L}_{\text{CLS}}$ and implement the compatibility function f as a bilinear model:

$$f(x, a; W, b) = (Wx + b)^{\text{T}} a. \tag{3.6}$$

The compatibility matrix $W \in \mathbb{R}^{d_a \times d_x}$ and the bias vector $b \in \mathbb{R}^{d_a}$ corresponds to θ. Note that when f has the bilinear form, it learns multi-modal embeddings between (x, a) pairs in the training set. Therefore, the performance of f as well as the efficiency of the gradient matching loss depend not only on the image embeddings x but also on the semantic class embeddings a. For the completeness, we also experiment with compatibility functions having a linear form:

$$f(x, a; W, b) = Wx + b. \tag{3.7}$$

This time $W \in \mathbb{R}^{|\mathcal{Y}_{\text{all}}| \times d_x}$ and bias vector $b \in \mathbb{R}^{\mathcal{Y}_{\text{all}}}$.

We note that while optimizing $\mathcal{L}_{\text{GM}}$ by a batch gradient descent update rule, it is important to compute $g_{\text{r}}(\theta)$ and $g_{\text{s}}(\theta)$ over real and synthetic samples of the same class, respectively. This makes sure that the genarator effectively learns class manifolds separately. Otherwise, although matching the aggregated gradients ∇_{θ_f} of a batch of samples that belong to different classes is still a valid supervision for the generator to learn the data distribution, it becomes difficult for the generator to learn individual class distributions.

Furthermore, thanks to our gradient matching loss, we can decouple the class label supervision from $\mathcal{L}_{\text{WGAN}}$ objective. This way, depending on the availability of unlabeled training data, $\mathcal{L}_{\text{WGAN}}$ term in Equation 3.5 can be computed either over seen class embeddings and samples ($\mathcal{L}^{\text{S}}_{\text{WGAN}}$), or, over all classes ($\mathcal{L}^{\text{S+U}}_{\text{WGAN}}$) possibly in a transductive way:

$$\mathcal{L}^{\text{S}}_{\text{WGAN}} = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathcal{G}(a),a\sim\mathcal{A}_{\text{s}}}[\mathcal{D}(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathcal{X}_{\text{s}}}[\mathcal{D}(x)] + \lambda\mathcal{L}_{\text{GP}} \tag{3.8}$$

$$\mathcal{L}^{\text{S+U}}_{\text{WGAN}} = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathcal{G}(a),a\sim\mathcal{A}_{\text{all}}}[\mathcal{D}(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathcal{X}_{\text{all}}}[\mathcal{D}(x)] + \lambda\mathcal{L}_{\text{GP}}, \tag{3.9}$$

where $\mathcal{L}_{\text{GP}}$ is the gradient penalty term in Equation 3.1. Here, in the case of Equation 3.9, $\mathcal{D}_{\text{train}}$ also includes $\mathcal{X}_{\text{u}}$. Unlike most of the transductive zero-shot learning approaches, we do not assume that unseen examples belong solely to the unseen classes: while such an assumption can possibly provide a significant advantage in training, it is unrealistic in most scenarios. The compute graph summarizing our approach is depicted in Figure 3.2.

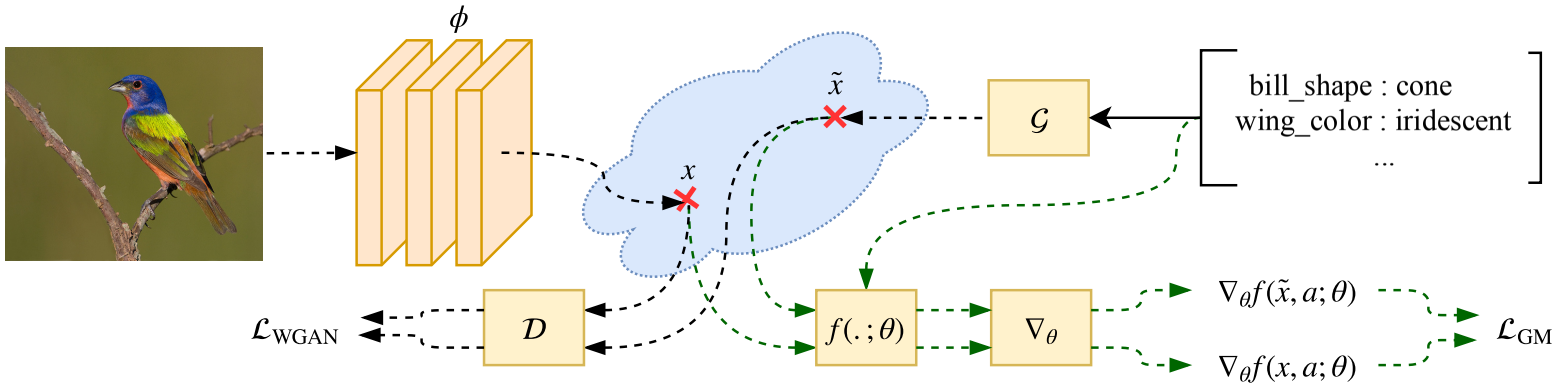


Figure 3.2: Illustration of the compute chain in the Gradient Matching Network. ϕ is a pre-trained CNN. $\mathcal{G}$ is the generator and it synthesizes features for any class using its semantic embedding. $\mathcal{D}$ represents the discriminator network. f is the compatibility function. ∇ denotes gradient operator in the compute graph. Paths through which only data of seen classes flow when $\mathcal{D}$ is unconditional are colored in green. (Best viewed in color.)

3.2.3 Supervision by Conditional Discriminator

Up to now, the source of supervision for a generator network is defined by an auxiliary loss function minimized by the generator network itself during training. However, we can also condition a discriminator network on either one-hot class labels or semantic embedding vectors so that it can also learn relations between visual features and semantic embeddings [39, 40, 41, 76]. To do that we slightly change Equation 3.1 as follows:

$$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} = \mathbb{E}\left[\mathcal{D}(x,a)\right] - \mathbb{E}\left[\mathcal{D}(\tilde{x},a)\right] + \lambda\,\mathbb{E}\left[(\|\nabla_{\hat{x}}\mathcal{D}(\hat{x},a)\|_2 - 1)^2\right].$$

We note that this conditional form of the discriminator network can only be trained using training samples of seen classes. In other words, it cannot be utilized over unsupervised samples in a semi-supervised or transductive settings.

In our experiments, we comprehensively evaluate the impact of training with different GAN loss versions ($\mathcal{L}^{\mathrm{S}}_{\mathrm{WGAN}}, \mathcal{L}^{\mathrm{S+U}}_{\mathrm{WGAN}}, \mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}}$) and their combinations with gradient matching loss.

3.2.4 Feature Synthesis

Once we train our generative model, we synthesize training examples for both seen and unseen classes by providing their class embeddings into the generative network as input, and then we combine the resulting $\mathcal{D}_{\mathrm{fake}}$ with $\mathcal{D}_{\mathrm{train}}$ to form our final training set $\tilde{\mathcal{D}} = D_{\mathrm{train}} \cup \mathcal{D}_{\mathrm{fake}}$. Once all samples are generated, we train the multi-class classification model based on the compatibility function by simply minimizing the cross-entropy loss over all (seen+unseen) classes. Finally, we utilize the resulting f to perform ZSL and GZSL.

3.3 Experiments

In this section, we present an experimental evaluation of the proposed approach. First we briefly explain our experimental setup, then we evaluate important GMN variants and compare with the state of the art. We additionally analyze our model via a detailed ablation study, including an evaluation on the effect of using synthesized training examples.

Table 3.1: Statistics for the benchmark datasets used in zero-shot learning experiments. n_{attr} denotes number of attributes and $|.|$ indicates cardinality of a set. The last column shows *average number of samples per class* (ANOSPC) over each of the datasets. We use the splits proposed in [1].

	n_{attr}	$\lvert\mathcal{Y}_u\rvert$	$\lvert\mathcal{Y}_s\rvert$	$\lvert\mathcal{X}_u\rvert$	$\lvert\mathcal{X}_s\rvert$	*ANOSPC*
CUB [43]	312	50	150	2967	8821	59
SUN [44]	102	72	645	1440	12900	20
AWA [18]	85	10	40	5685	24790	609

Datasets. We evaluate our model in the three commonly used benchmark datasets, namely Caltech-UCSD Birds-200-2011 (CUB) [43], SUN Attribute (SUN) [44] and Animals with Attributes (AWA) [18]. CUB and SUN are fine-grained image datasets which contain 200 bird species and 717 scene categories, respectively. They are particularly challenging for ZSL & GZSL as they contain relatively fewer images per class, making it difficult to model intra-class variations efficiently. AWA is a coarse-grained dataset consisting of images belonging to 50 animals. AWA contains a relatively small set of classes, which makes generalization to unseen classes more difficult. A summary is given in Table 3.1.

In our comparisons, we utilize the splits, class embeddings and evaluation metrics proposed in [1] for standardized ZSL and GZSL evaluation. Following [1, 38, 39], we use the 2048-dimensional top pooling units of a ResNet-101 pretrained on ImageNet-1K as the image representation. We do not apply any pre-processing to these features. We use class-level attributes as class embeddings. For CUB experiments, we additionally use 1024-dimensional character-based CNN-RNN features [105] as in [1, 40]. As a pre-processing step, we ℓ_2 normalize the class embeddings.

Evaluation. Once we train a particular generative model, *e.g.* conditional WGAN or GMN, on a particular dataset $\mathcal{D}_{\text{train}}$, we follow two different strategies to evaluate the generator network. I. We synthesize n_{zsl}, $n_{\text{gzsl-u}}$ and $n_{\text{gzsl-s}}$-many samples for each class to create a separate augmented dataset $\mathcal{D}^{\text{zsl}}_{\text{fake}}$, $\mathcal{D}^{\text{u}}_{\text{fake}}$, $\mathcal{D}^{\text{s}}_{\text{fake}}$ for training separate models for the ZSL, GZSL-u and GZSL-s tasks, respectively. II. We create $\mathcal{D}^{\text{a}}_{\text{fake}}$ containing n_{a} synthetic sample per class, to train a single model that performs all tasks, *i.e.* classifying seen and unseen class examples. The former evaluation procedure enables us to get a complete understanding about how networks perform in each task independently. Whereas, the latter is the way to compare ourselves with the state-of-the-art. Depending on the dataset, n_{zsl}, $n_{\text{gzsl-u}}$, $n_{\text{gzsl-s}}$ and n_{a} can be either integers denoting number of synthetic samples generated for only unseen classes or tuple of integers indicating that synthetic samples are generated for seen classes as well. For instance, on the AWA dataset, where there is a significant imbalance among training classes, we additionally synthesize examples for the seen classes to obtain equivalent number of training examples for the seen classes. These numbers are considered as hyper-parameters and therefore tuned on the validation splits. We set sample spaces in a way that they are comparable with the *ANOSPC* value of each dataset (Table 3.1) and they maximize the performance on the target task. Then we stack each $\mathcal{D}_{\text{fake}}$ together with the training set $\mathcal{D}_{\text{train}}$ to form $\tilde{\mathcal{D}}$ on which we train f defined in Equation 3.6. We also tune the hyper-parameters of this classifier, *i.e.* learning rate and number of training iterations, for each experimental setup separately on the validation splits as well. Once the final classifier is trained, we assign a label to a test sample by considering only the scores of unseen classes $\mathcal{Y}_{\text{u}}$ for ZSL; we consider the scores of all classes $\mathcal{Y}_{\text{all}}$ when determining the label for GZSL. As evaluation scores we compute normalized mean top-1 accuracies **T-1**. We compute two metrics for GZSL experiments: GZSL-u (**u**) and GZSL-s (**s**) are normalized GZSL **T-1** accuracies of unseen and seen classes, respectively . Finally we compute their harmonic means by $\mathbf{h} = \frac{2 \times \mathbf{u} \times \mathbf{s}}{\mathbf{u} + \mathbf{s}}$ [1].

Implementation details. In our experiments, we realize $\mathcal{G}$ and $\mathcal{D}$ as simple MLPs that have 1 or 2 hidden layers with 1024, 2048 or 4096 units. Both networks have ReLU activation functions. We consider the dimension of noise spaces

d_z as another hyper-parameter. While minimizing $\mathcal{L}_{\text{GM}}$, we also update the classification model, but unlike the $\mathcal{G}$ and $\mathcal{D}$ models, we regularly re-initialize the classification model every N iterations. We tune all the hyper-parameters on the validation sets. While developing the GMN algorithm and its network we observed the followings:

I. Constraining noise means by applying tanh activation, *i.e.* $\mu(a) = \tanh(\mathcal{W}_{\text{mu}}a + b_{\text{mu}})$, slightly improves generalization performance.

II. Using an identity covariance matrix, ie. $\Sigma(a) = \mathbb{I}$, performs equally well.

III. Minimizing l_{p} loss between the gradient vectors, *i.e.* $\|g_{\text{r}}(\theta) - g_{\text{s}}(\theta)\|_p$ in addition to maximizing their cosine similarity leads to slightly better results.

IV. It is important to use a classification loss for $f(\cdot\,;\theta)$ that gives non-zero values until θ is re-initialized. For instance, using hinge loss requires careful attention during training, as it is possible to get zero loss values and when it happens g_{r} and g_{s} become vague.

Therefore, we tune these design choices on the validation sets as well.

GMN versus strong baselines. In Table 3.2, we present a detailed evaluation of the GMN variants and compare them against strong baselines. In order to carefully observe the variants on each task, we train a separate model for each task following the evaluation scheme described above.

The first part of the table shows baselines with no GM-loss: I. training f with available seen class samples only, II. training $\mathcal{G}$ model w.r.t. unconditional discriminator based on the seen class examples ($\mathcal{L}^{\text{S}}_{\text{WGAN}}$) plus the loss of a pre–trained classifier ($\mathcal{L}_{\text{CLS}}$), III. training $\mathcal{G}$ model with conditional discriminator over the seen classes($\mathcal{L}^{\text{S}}_{\text{cWGAN}}$).

The second part of the table shows GMN variants, where the GM-loss is used in combination with I. an unconditional seen-class only discriminator ($\mathcal{L}^{\text{S}}_{\text{WGAN}}$), II. a conditional discriminator over the seen classes ($\mathcal{L}^{\text{S}}_{\text{cWGAN}}$), III. an unconditional

discriminator over seen and unseen classes ($\mathcal{L}_{\text{WGAN}}^{\text{S+U}}$). In all cases (except the very first one), the resulting $\mathcal{G}$ model is used for generating n_{zsl}, $n_{\text{gzsl-u}}$ and $n_{\text{gzsl-s}}$-many synthetic training examples for each unseen class. Only the last experiment runs in a transductive setting. Note that we do not report any results with $\mathcal{L}_{\text{WGAN}}^{\text{S}}$-only or $\mathcal{L}_{\text{WGAN}}^{\text{S+U}}$-only training as they lead to unusable $\mathcal{G}$ due to the lack of any class supervision (see Figure 3.6).

From the results, we observe that all generative model based methods significantly improve over the simple real data only baseline. This result shows that generating unseen class examples via $\mathcal{G}$ helps training f models that are particularly better at Generalized ZSL. We also observe that using a conditional discriminator leads to better results than training with the loss function of a pre-trained classifier, as we hypothesize in Section 3.2.3.

From the results in the second part of Table 3.2, we observe that the proposed gradient matching loss consistently improves all ZSL and **u** scores by marginally sacrificing the **s** scores. In particular we observe that GM loss is significantly a better way for training $\mathcal{G}$ than minimizing the classification loss of a pre-trained classifier. In addition, we observe that $\mathcal{L}_{\text{GM}}$ improves over the conditional discriminator based $\mathcal{G}$ training. Finally, we observe that unsupervised discriminator (over all classes) combined with the GM loss (over seen class examples only) gives a strong approach for training the conditional generative model in a transductive way.

Table 3.2: Quantitative evaluation of GMN over the strong baselines.

	Zero-Shot Learning			**Generalized Zero-Shot Learning**								
	CUB	**SUN**	**AWA**	**CUB**			**SUN**			**AWA**		
Method	**T-1**	**T-1**	**T-1**	**u**	**s**	**h**	**u**	**s**	**h**	**u**	**s**	**h**
Train only with real samples ($\mathcal{D}_s$)	56.8	60.7	62.3	26.9	67.6	38.4	23.4	36.3	28.4	13.4	78.1	22.9
$\mathcal{L}^{S}_{WGAN} + \mathcal{L}_{CLS}$	58.3	61.4	70.0	47.0	71.0	56.5	47.7	**41.2**	44.2	47.8	78.7	59.5
$\mathcal{L}^{S}_{cWGAN}$	60.6	62.6	72.0	55.9	71.1	62.6	53.6	41.1	46.5	55.2	79.1	65.0
$\mathcal{L}^{S}_{WGAN} + \mathcal{L}_{GM}$	61.9	63.8	70.4	55.8	70.7	62.4	53.8	40.9	46.5	52.1	78.8	62.7
$\mathcal{L}^{S}_{cWGAN} + \mathcal{L}_{GM}$	**64.6**	64.1	73.9	57.9	**71.2**	63.9	55.2	40.8	46.9	63.2	78.8	70.1
$\mathcal{L}^{S+U}_{WGAN} + \mathcal{L}_{GM}$ (transductive)	**64.6**	**64.3**	**82.5**	**60.2**	70.6	**65.0**	**57.1**	40.7	**47.5**	**70.8**	**79.2**	**74.8**

GMN versus state-of-the-art. Finally, we present a comparison of GMN with the recently proposed state-of-the-art non-transductive ZSL approaches on the benchmarks from [1]. The results are given in Table 3.3.

f-CLS-WGAN-{ALE/DEVISE/Softmax} models from Xian *et al.* [39] and cycle-{WGAN/CLS-WGAN} models from Felix *et al.* [40] employ "attribute concatenation" (AC) type of generator networks. Besides, f-CLS-WGAN-Softmax [39] and cycle-{WGAN/CLS-WGAN} models [40] train classifiers that have linear forms as in Equation 3.7. In addition to these, we implement f-CLS-WGAN-Bilinear, cycle-WGAN-Bilinear models with "latent noise" (LN) type of generator networks and bilinear classifiers. These models correspond to lines 10 and 11 in Table 3.3 respectively. This way we compare GMN variants, where the compatibility function is either linear or bilinear and the generator networks are either AC or LN types (lines 12-15), extensively with the most recent leading approaches.

In the light of the results, we see that gradient matching outperforms both minimizing the loss of a pre-trained classifier and minimizing the cycle-consistency loss in the context of zero-shot learning when a single model is trained to perform all tasks. Overall, we observe that GMN leads to significant improvements over the state-of-the-art in terms of h-scores on all datasets.

Table 3.3: Comparison of GMN against the baselines and the state-of-the-art on the benchmarks from [1]. The results are taken from the papers, except for [38] (which is obtained by us using the the authors' implementation) and $\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{CLS}}$ (our implementation). "Linear" and "Bilinear" denote the type of f (see Section 3.2.1). "LN" and "AC" stand for "latent noise" and "attribute contatenation" denoting the the way we merge noise and semantic class embeddings in the generator networks (see Section 3.2.2).

				Zero-Shot Learning			**Generalized Zero-Shot Learning**								
				CUB	**SUN**	**AWA**	**CUB**			**SUN**			**AWA**		
				T-1	**T-1**	**T-1**	**u**	**s**	**h**	**u**	**s**	**h**	**u**	**s**	**h**
1	*Zhang et al.* [76] '18			52.6	61.7	67.4	31.5	40.2	35.3	41.2	26.7	32.4	38.7	74.6	51.0
2	*Bucher et al.* [38] '17			57.8	60.4	66.3	28.8	55.7	38.0	40.5	37.2	38.8	2.3	**90.2**	4.5
3	*Xian et al.* [39] - `DEVISE` '18			60.3	60.9	66.9	52.2	42.4	46.7	38.4	25.4	30.6	35.0	62.8	45.0
4	*Xian et al.* [39] - `ALE` '18			61.5	62.1	68.2	40.2	59.3	47.9	41.3	31.1	35.5	47.6	57.2	52.0
5	*Xian et al.* [39] - `Softmax` '18			57.3	60.8	68.2	43.7	57.7	49.7	42.6	36.6	39.4	57.9	61.4	59.6
6	*Verma et al.* [41] '18			59.6	63.4	69.5	41.5	53.3	46.7	40.9	30.5	34.9	56.3	67.8	61.5
7	*Felix et al.* [40] - `cycle-WGAN` '18			57.8	59.7	65.6	46.0	60.3	52.2	48.3	33.1	39.2	56.4	63.5	59.7
8	*Felix et al.* [40] - `cycle-CLSWGAN` '18			58.4	60.0	66.3	45.7	61.0	52.3	49.4	33.6	40.0	56.9	64.0	60.2
9	Bilinear	LN	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}}$	61.7	62.7	67.3	45.6	59.2	51.5	50.6	30.3	37.9	53.5	72.0	61.4
10	Bilinear	LN	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{CLS}}$ [39]	61.9	61.5	66.4	45.5	58.9	51.4	49.9	30.0	37.4	52.7	71.0	60.5
11	Bilinear	LN	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{CYCLE}}$ [40]	62.2	62.9	68.2	51.1	54.9	52.9	52.0	31.2	39.0	55.4	70.1	61.8
12	Bilinear	LN	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{GM}}$ (*Ours*)	**67.0**	**63.6**	**72.0**	**54.7**	58.4	**56.5**	42.5	35.5	38.7	**61.1**	71.3	65.8
13	Linear	LN	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{GM}}$ (*Ours*)	63.1	58.9	70.1	48.5	**62.8**	54.7	42.0	**39.3**	40.7	57.1	81.3	**67.1**
14	Bilinear	AC	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{GM}}$ (*Ours*)	65.7	62.6	69.7	53.8	58.2	55.9	43.2	36.2	39.4	54.8	74.1	63.0
15	Linear	AC	$\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{GM}}$ (*Ours*)	63.8	61.1	66.8	45.8	65.5	53.9	**53.2**	33.0	**42.8**	46.8	84.8	60.3

Effect of the feature generation. Having verified the effectiveness of our framework, we now evaluate how the size of the synthetic train data affect the final ZSL and GZSL performances. For this purpose, we perform two sets of experiments.

In the first set, we train two generative models on each dataset by optimizing $\mathcal{L}_{\text{WGAN}}^{\text{S}}$ and $\mathcal{L}_{\text{WGAN}}^{\text{S+U}}$, respectively. Then we create six different synthetic datasets $\tilde{\mathcal{D}}_u^i$ with each model by sampling for each class $\{10, 25, 50, 150, 200, 250\}$ features on CUB and SUN, and $\{500, 600, 700, 800, 900, 1000\}$ features on AWA. The goal of this analysis is two-fold. We want to see how **u** and **s** scores vary when we I. increase the size of the synthetic datasets, II. use test samples of unseen classes in an unsupervised way (as in traditional transductive settings). In Figure 3.3, we see that, as the number of features synthesized increases, **u** scores on all datasets increase rapidly. Whereas, ZSL scores develop progressively on the SUN and remain almost fixed on the others. Additionally, we observe a trade-off between **s** scores and the amount of synthesized features. That is, **s** scores decrease dramatically as f is trained with more synthesized features. Moreover, we see that utilizing the samples of unseen classes boost the performance of the final f in favour of the unseen classes. However, this is an expected result since f is no longer biased towards seen classes and the pooled sets $\tilde{\mathcal{D}}^i$ become dominated by the synthesized sets $\tilde{\mathcal{D}}_u^i$. In fact **s** scores on the AWA support this claim, *i.e.* slope of the decrease in **s** scores is much smaller. Because in AWA there are 609 samples per class on average. We observe that synthesizing more features does not necessarily increase the ZSL scores. This might indicate that latent noise spaces become less discriminative due to the WGAN updates or the generator might be learning to model the feature space irrespective of the noise distributions. And finally, the gap between $\mathcal{L}_{\text{WGAN}}^{\text{S}}$ and $\mathcal{L}_{\text{WGAN}}^{\text{S+U}}$ on AWA suggests that GMN struggles in finding generalizable latent spaces when there are small set of classes and attributes.

In the second set, we investigate if the performance gains of GMN result from the number of training samples generated for unseen classes. To do that we train f-CLS-WGAN-Bilinear, cycle-WGAN-Bilinear and GMN models (corresponding

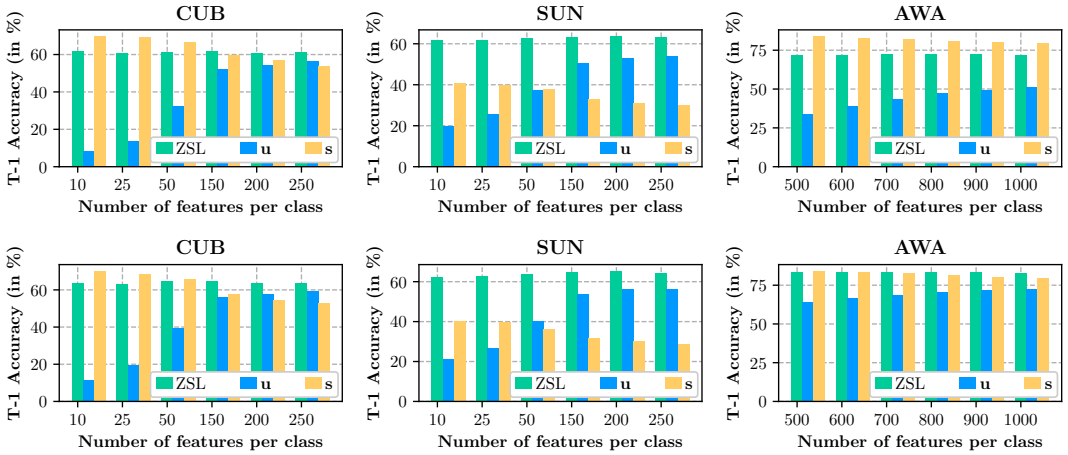


Figure 3.3: Analysis of GMN regarding impact of the number of synthesized features on (from left to right) **T-1**, **u** and **s** scores on CUB, SUN and AWA datasets. Top row is obtained by optimizing $\mathcal{L}^{\text{S}}_{\text{WGAN}}$, while bottom row is $\mathcal{L}^{\text{S+U}}_{\text{WGAN}}$.

to lines 10, 11 and 12 in Table 3.3 respectively) with their best performing hyper-parameters, 3 times with different seeds. Then from each generator network learned; we sample a synthetic dataset for unseen classes, train 5 linear classifiers with different seeds and compute the average scores. In total this amounts to training 15 linear classifiers for each generator network. Finally we compute means and standard deviations of 3 runs for each generator network. We repeat this over AWA and CUB datasets and plot results in Figure 3.4. According to the results, we make the following interesting observations. First, if the hyper-parameters of a conditional Wasserstein-GAN (cWGAN) are tuned carefully, cWGAN itself is a quite strong baseline, *i.e.* it is comparable to f-CLS-WGAN-Bilinear and cycle-WGAN-Bilinear. Second, GMN outperforms f-CLS-WGAN, cycle-WGAN and cWGAN in all cases, proving its robustness against random initializations.

Ablation study. We perform additional analyses to gain further insight about GMN. We begin by elaborating more on the supervision signal conveyed by $\mathcal{L}_{\text{GM}}$, when discriminator is unconditional, and the effect of utilizing samples of unseen classes on **u** scores. To do that we train four different generators over SUN dataset by optimizing $\mathcal{L}^{\text{S}}_{\text{WGAN}}$, $\mathcal{L}_{\text{GM}}$, $\mathcal{L}^{\text{S}}_{\text{WGAN}} + \mathcal{L}_{\text{GM}}$ and $\mathcal{L}^{\text{S+U}}_{\text{WGAN}} + \mathcal{L}_{\text{GM}}$, respectively. Besides, while training the generators we compute **u** scores of samples in

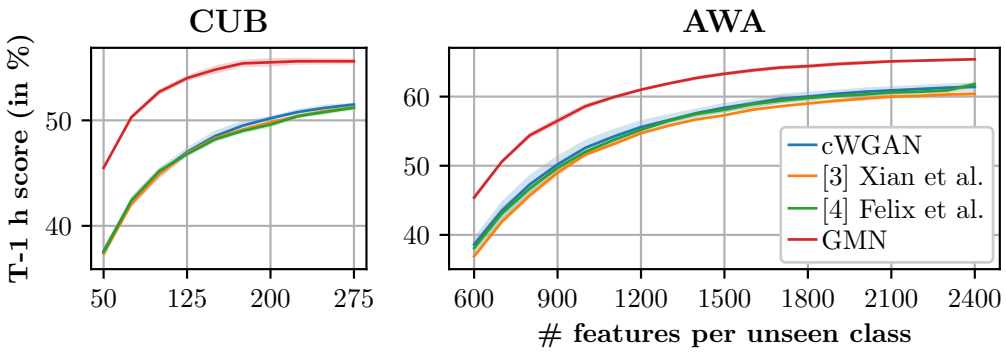


Figure 3.4: Analysis of cWGAN, f-CLS-WGAN-Bilinear, cycle-WGAN-Bilinear and GMN models regarding impact of the number of synthesized features on **h** scores on CUB and AWA datasets.

the validation set. Not surprisingly, when a generator optimizes $\mathcal{L}^{\mathrm{S}}_{\mathrm{WGAN}}$ alone, it cannot distinguish class modes separately. Therefore features synthesized by this network results in a poor classifier. However, a generator optimizing only $\mathcal{L}_{\mathrm{GM}}$ starts learning a mapping from semantic embeddings to prominent visual features. Furthermore, $\mathcal{L}^{\mathrm{S}}_{\mathrm{WGAN}} + \mathcal{L}_{\mathrm{GM}}$ slightly improves the mapping by most likely modeling data manifold more effectively. Finally, as an unconditional discriminator captures statistics of samples belonging to unseen classes by means of optimizing $\mathcal{L}^{\mathrm{S+U}}_{\mathrm{WGAN}}$, classifier performance peaks. We share plots corresponding to each experiment in Figure 3.6.

Next, we show that conditioning a discriminator on semantic embeddings and gradient matching are complementary sources of supervision for a generator network. In our experiments, we see that optimizing $\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}} + \mathcal{L}_{\mathrm{GM}}$, compared to optimizing only $\mathcal{L}^{\mathrm{S}}_{\mathrm{cWGAN}}$, improves almost all performance metrics significantly. We perform a rather qualitative evaluation on CUB dataset to examine the improvements that result from introducing $\mathcal{L}_{\mathrm{GM}}$ term in a plain conditional WGAN setup. For this purpose, we inspect classes that are improved by introducing gradient matching. Significant improvements in certain classes suggest that GMN can help learning better relations between visual features and semantic embeddings. Some of examples are shown in Figure 3.5.

Figure 3.5: Examples from unseen CUB classes that are misclassified with a cWGAN-based classifier but correctly recognized when we include $\mathcal{L}_{\text{GM}}$ into generator training. From top to bottom the classes are *groove billed ani*, *wilson warbler*, *mallard*, *le conte sparrow* and *tropical kingbird*.

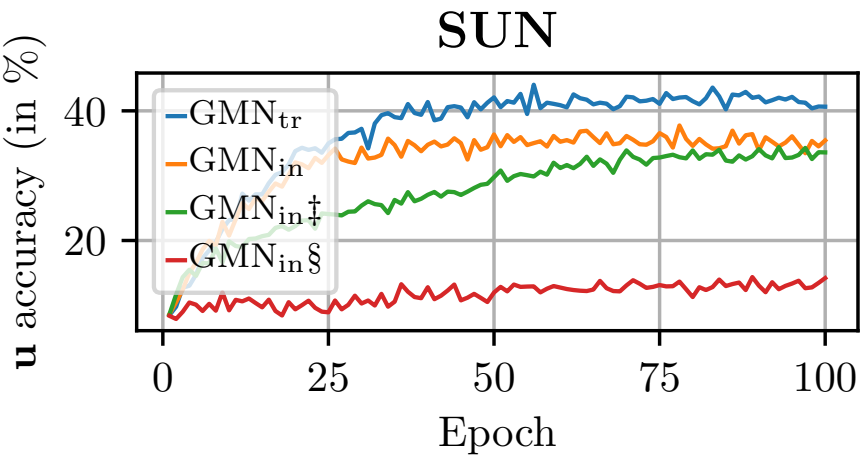


Figure 3.6: **u** scores of validation samples from the SUN dataset over the training iterations of $\mathcal{L}^{S}_{WGAN}$ (GMN$_{in}$§), $\mathcal{L}_{GM}$ (GMN$_{in}$‡), $\mathcal{L}^{S}_{WGAN} + \mathcal{L}_{GM}$ (GMN$_{in}$) and $\mathcal{L}^{S+U}_{WGAN} + \mathcal{L}_{GM}$ (GMN$_{tr}$).

Chapter 4

Key-Protected Classification for Collaborative Learning

In Chapter 3, we discussed zero-shot learning as a way to eliminate the need to collect training samples for each high-level concept we want a visual embedding model to learn. Although zero-shot learning deals with one of the data related concerns raised in Chapter 1, it does not question the integrity of data, *i.e.* all the training and test data is centralized and given to the learner. This chapter takes an orthogonal step and tackles a particular data bottleneck in machine learning appearing when it is not possible to build and share a dataset for a specific problem due to the privacy violations of subjects. As we mentioned in Section 1.2, this issue becomes a limiting factor in developing machine learning models for certain problems such as the ones relying on private health records of patients. To overcome this difficulty a decentralized learning protocol named *collaborative learning frameworks* (CLFs) have been proposed. However, recently it has been shown that CLFs are vulnerable against active machine learning attacks. In this chapter, we propose a novel formulation to make CLFs resilient against such attacks.

Organization of this chapter is as follows. Section 4.1 gives a background on the problem we address in this chapter, Section 4.2 explains the proposed model

and Section 4.3 empirically evaluates the model.

4.1 Background

Our work builds on the collaborative learning framework [48] and tackles the generative adversarial network attack problem [58]. Therefore, before providing the details of our approach, we first provide brief background on CLF and the GAN attack in the following.

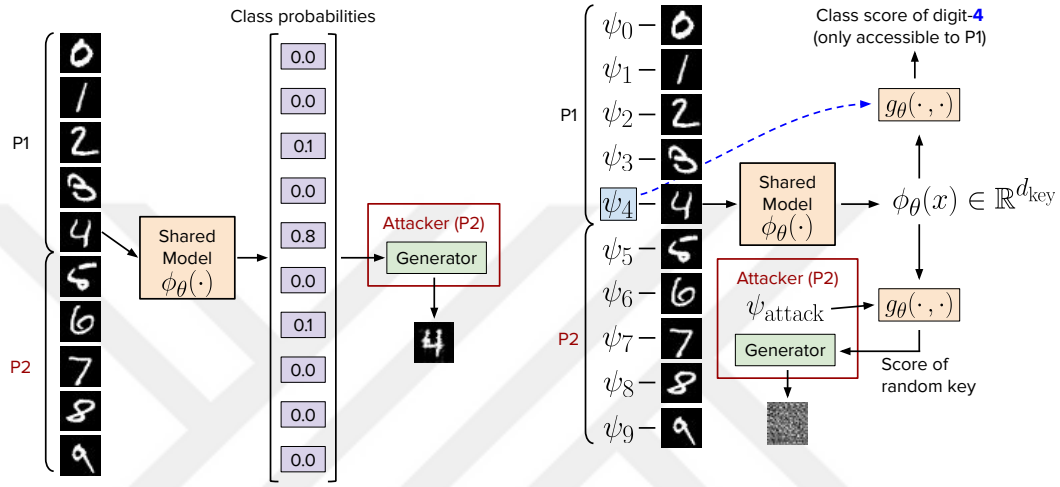
4.1.1 Collaborative Learning of Participants

The goal of CLF is to collaboratively train a shared model over private datasets of several participants such that the model generalizes across all the private datasets. For this purpose, CLF defines a protocol where each participant shares with others information only about its learning progress, rather than the data directly, over the training iterations. Locally, participants train their model as usual using gradient based optimization, but share with others fractions of changes in model parameters, at predefined intervals. The framework is set up among participants based on the following components and the associated policies:

I. A mechanism for participants to share parameter updates. This is typically realized by a trusted third-party parameter server (PS), using which the participants accumulate the model parameters by uploading and downloading fractions of their local parameter changes and central model parameters, respectively.

II. A common objective and model architecture. All participants use the same model architecture and training objective. Typically, participants declare class labels for which they have training data.

III. Meta-parameters. The hyper-parameters of the CLF setup, such as the parameter download fraction (θ_d) and the upload fraction (θ_u), gradient



(a) The compute chain in GAN attack.

(b) The compute chain in our proposed method.

Figure 4.1: The comparison of the compute chains constructed during the vanilla (left, a) and the modified key-protected (right, b) collaborative learning frameworks. In both scenarios, the dataset is split among two participants, P1 (an honest participant) and P2 (an adversary) and they train a shared neural network model by following the collaborative learning framework defined in [48]. In (a), the shared model directly outputs class prediction probabilities $\phi_\theta(x)$ for an image x and this enables participants to train the model by optimizing a discriminative loss, *i.e.* cross entropy between the predicted class probabilities and the ground truth labels of samples. However, outputting the class scores makes the shared model vulnerable to the powerful GAN attack [58]. In (b) the participants create a d-dimensional private class key ψ_c for each of the classes for which they have training samples and the shared model outputs d-dimensional image embedding vector $\phi_\theta(x)$ for an image x. Moreover, instead of minimizing the cross entropy loss, they maximize the cosine similarity $f(\cdot,\cdot)$ between the image embedding vector $\phi_\theta(x)$ and the private key ψ_c of the correct class c. This way, the class probabilities are concealed from the attacker. Still, the adversary can attack by means of creating random class keys $\psi(\text{rand})$ (hoping that the random keys are close enough to any of the real ones created privately by the honest participant) yet it fails as it is very unlikely that the random keys are close to the real ones.

clipping threshold (γ), and the order of participants during training (e.g., round robin, random, or asynchronous) are typically predetermined [48].

Once the framework is established, participants start training on their local datasets in a predetermined order. When a participant takes turn, it first downloads θ_d fraction of parameters from the parameter server and replaces them with its local parameters. After performing several-steps training (for instance, one epoch of training) on its local dataset, participant uploads θ_u fraction of resulting gradients to the parameter server. It is also possible to incorporate differential privacy *i.e.*, by injecting some form of noise to the uploaded gradients, to guarantee a certain level of sample indistinguishability for enhanced privacy protection. But this procedure comes with a trade-off between the level of privacy and the efficiency (performance) of the accumulated final model. We encourage readers to refer to [48] for the details of this learning protocol.

4.1.2 Generative Adversarial Network

Generative adversarial network [59] is an unsupervised learning process for learning a model of the underlying distribution of a sample set. A GAN model consists of two sub-models, called generator and discriminator. The generator corresponds to a function $\mathcal{G}(z;\theta_{\mathcal{G}})$ that aims to map each data point z sampled from a prior distribution, *e.g.* uniform distribution $\mathcal{U}(-1,1)$, to a point $\hat{x}$ in the data space, where $\theta_{\mathcal{G}}$ represents the generator model parameters. Similarly, the discriminator is a function $\mathcal{D}(x;\theta_{\mathcal{D}})$ that estimates the probability that a given x is a real sample from the data distribution $p_{\mathcal{D}}$, where $\theta_{\mathcal{D}}$ represents the discriminator model parameters.

The generator and discriminator models are trained in turns, by playing a two-player mini-max game. At each turn, the generator is updated towards generating samples that are indistinguishable from the real samples according to the current discriminator's estimation:

$$\min_{\theta_{\mathcal{G}}} \ \mathbf{E}_{z\sim p(z)}\big[\log(1-\mathcal{D}(\mathcal{G}(z;\theta_{\mathcal{G}}));\theta_{\mathcal{D}})\big]. \tag{4.1}$$

The discriminator, on the other hand, is updated towards distinguishing the samples given by $\mathcal{G}$ from the real ones:

$$\max_{\theta_{\mathcal{D}}} \mathbf{E}_{x\sim p(x)}\big[\log(\mathcal{D}(x;\theta_{\mathcal{D}})\big] + \mathbf{E}_{z\sim p(z)}\big[\log(1-\mathcal{D}(\mathcal{G}(z;\theta_G));\theta_{\mathcal{D}})\big]. \tag{4.2}$$

GANs have successfully been utilized in numerous problems, *e.g.* see [106, 107].

4.1.3 GAN Attack in Collaborative Learning

[58] devise a powerful GAN-based active attack against collaborative learning frameworks. In this scenario, an adversarial participant takes places during training in a collaborative learning framework setup in order to extract information about some class c_{attack} for which any of the other honest participants has samples[1]. To do that the attacker utilizes the shared model as discriminator network and trains a generator network to capture the data manifold of the class c_{attack}. Besides, the attacker announces an incorrect, unique class c_{fake} to label the examples sampled from the generator network. An overview of this attack is depicted in Figure 4.1a.

In [58] it is shown that such an approach effectively turns collaborative learning framework into a GAN training setup where an adversary takes the following steps:

I. The adversary updates its generator network towards producing samples that are classified as class c_{attack}, by the shared classification model.

II. The adversary takes samples from its generator, labels them as c_{fake} and updates the shared classification model towards classifying the synthetic samples as class c_{fake}.

[1]The adversary may additionally have its own real classes and a real dataset, but for the sake of simplicity, we assume that the adversary works only on its privacy attack.

There are two key aspects of the GAN attack. First, throughout the training iterations, the adversary continuously updates its generator, therefore, it can progressively improve its generative model and the reconstructions that it provides. Second, since the adversary defines the class c_{fake} as part of the shared model, the participant that hosts c_{attack} updates the shared model towards minimizing the misclassification of its training examples into class c_{fake}. Over the iterations, this practically forces the victim into releasing more detailed information about the class c_{attack} while updating the shared model [58]. This latter step makes the GAN attack particularly powerful as it influences the training of all participants, and, it is also the main reason why the technique is considered as an active attack. In this paper, privacy model and proposed protection mechanism completely follow the threat model defined in [58]. In our experiments, we simulate the experimental setups of [58, 48] without introducing any extra assumptions.

4.2 Model

In this section, we describe the details of our proposed approach. In Section 4.2.1, we present our class-key protected classification model, as a prevention mechanism against the GAN attack [58]. In Section 4.2.2, we show how such a model can be trained distributedly when each participant has access only to the keys of its own classes, while leveraging the fundamental tools given the distributed learning framework of Shokri *et al.* [48]. In Section 4.2.3, we discuss practical considerations in generating random class keys. Finally, in Section 4.2.4, we propose an extension of our approach that enables efficient incorporation of high dimensional keys towards minimizing the risk of a successful GAN attack within our key-protected classification framework.

4.2.1 Key Protected Classification Model

Our goal is to train a deep (convolutional) neural network based multi-class classifier in a collaborative manner. Our starting point is the observation that the

GAN attack relies on the knowledge of classification scores of samples belonging to the target class c_{attack} throughout the training iterations, as discussed in Section 4.1.3 and in Figure 4.1a. To prevent the attack in a collaborative learning setup, we aim to mathematically prevent each participant from estimating the classification scores for the classes hosted by the other participants. For this purpose, we introduce class-specific keys for all classes and parameterize the classification function in terms of these keys in a way that makes classification score estimation without keys practically improbable.

In our approach, we require each participant to generate random class keys for its classes during initialization, and keep them private until the end of the training process. We denote the key for class c by $\psi_c \in \mathbb{R}^{d_{\text{key}}}$, where d_{key} is the predetermined dimensionality of each key. In order to protect the model using class keys, we first define the network with model parameters θ as an embedding function $\phi_\theta(x)$ that maps each given input x, *e.g.* an image, from the source domain to a d_{key}-dimensional ℓ_2-normalized vector. Then, we define the classification score for class c by a simple dot product between the embedding output and the class key:

$$g_\theta(x, \psi_c) = \langle \phi_\theta(x), \psi_c \rangle,$$

where $\psi_c, \phi_\theta(x) \in \mathbb{R}^{d_{\text{key}}}$. We note that the class keys are analogous to classification weight vectors, or equivalently, the components of the very last fully-connected layer of a standard (mainstream) feed-forward classification neural network. However, unlike the case in a standard neural network model where the classification weights vectors are learned during training, here, these vectors are pre-generated and kept fixed throughout the training process. The training, therefore, takes the form of learning the embedding model $\phi_\theta(x)$.

Therefore, in contrast to a standard classification model where the model produces all class scores, in the proposed formulation, a participant can compute class scores only for the classes for which it has class keys. As a result, a participant does not have access to the class scores that are necessary by definition for the GAN attack. The overview of our proposed approach is given in Figure 4.1b.

Once the embedding model is learned and training is completed, all class keys are made public so that the resulting model can be used to make predictions for all classes. The final classification function takes the form of choosing the class whose key leads to the highest classification score:

$$\arg\max_{c \in C^{\text{all}}} \langle \psi_c, \phi_\theta(x) \rangle, \tag{4.3}$$

where C^{all} is the set of all classes over all participants.

To ensure that the embedding function $\phi_\theta(x)$ provides normalized embedding vectors as required by the definition, an ℓ_2-normalization layer is appended to the corresponding unnormalized embedding network $\phi^u_\theta(x)$, so that:

$$\phi_\theta(x) = \frac{\phi^u_\theta(x)}{\|\phi^u_\theta(x)\|}. \tag{4.4}$$

A clear reason for incorporating ℓ_2-normalization surfaces naturally in the derivation of our training formulation, which we explain in the next subsection.

Finally, we note that the method is presented with the assumption that only a single adversary exists, for the sake of brevity. Our framework, however, naturally handles multiple attackers without requiring any modifications, and, in fact, in Section 4.3, we do present experimental results for multiple attackers.

4.2.2 Learning with Restricted Class Key Access

In this section, we describe our approach for training the classification model, by making updates locally at each participant, using the restricted training set owned by the participants. Suppose that a participant owns n training examples, represented by a set of tuples $(x_i, c_i)_{i=1}^n$. The goal of the local model update is to update the embedding model ϕ such that it minimizes the negative label likelihood of the training examples by regularized risk minimization:

$$\min_\theta - \sum_{i=1}^{n} \log p_\theta(c_i|x_i) + R(\theta) \tag{4.5}$$

where $R(\theta)$ is the regularization function.

We aim to obtain label likelihoods $p_\theta(c|x)$ based on the scoring function $g_\theta(x, \psi)$. However, by design, a participant is not allowed to access keys ψ_c of other classes, therefore, cannot estimate the class probability distribution via a conventional softmax operator over the set of unnormalized class scores. Therefore, in order to define label likelihoods, we generalize the softmax operator by re-defining it as the ratio of exponentiated target class score and the expectation of exponentiated class scores over all possible class keys:

$$p(c|x) = \frac{\exp g_\theta(x, \psi_c)}{\mathbb{E}_\psi[\exp g_\theta(x, \psi)]} \tag{4.6}$$

One way to interpret this definition is applying softmax over infinitely many classes, where class keys follow some probability distribution. Assuming that class keys are obtained by sampling from the d_{key}-dimensional standard normal distribution, the expectation in the denominator can be re-written as:

$$\mathbb{E}_\psi[\exp(g_\theta(x, \psi))] = \int f(\psi) \exp g_\theta(x, \psi)) d\psi \tag{4.7}$$

where $f(\psi)$ is the probability density function for standard multivariate normal distribution. It can be shown that the expectation yields the value $\exp(0.5\|\phi(x)\|^2)$ (See Appendix A). By plugging this result into Eq. 4.5 and re-arranging the terms, we obtain:

$$\min_\theta - \sum_{i=1}^{n} g_\theta(x_i, \psi_{c_i}) + 0.5 \sum_{i=1}^{n} \|\phi_\theta(x_i)\|^2 + R(\theta) \tag{4.8}$$

Here, the first term corresponds to maximizing the classification scores, *i.e.* the sum of inner product values between each pair of sample embedding $\phi(x)$ and the corresponding class key ψ_c. The second term applies ℓ_2 regularization over the $\phi(x)$ embeddings, which limits the scale of the vectors produced by the embedding network $\phi(x)$. Finally, the third term is simple regularization over the model parameters, for which use ℓ_2 regularization.

The second term in Eq. 4.8 is worth discussing in a bit more detail: unlike conventional ℓ_2 regularization over the network parameters (*i.e.* $R(\theta)$), it applies ℓ_2 regularization to the outputs of the embedding network. This term, therefore, can be interpreted as a way to prevent learning a degenerate embedding model

that reduces the objective function by making the embedding vectors, and therefore the classification scores, excessively large. It is also an interesting result in the sense that it appears naturally in our derivation. We note that its weight (0.5) could be altered by using a different temperature constant in Eq. 4.7, which could be another hyper-parameter. However, as denoted before, we opt to fix the scale of the embedding vectors by incorporating a final ℓ_2 normalization layer into the network $\phi(x)$, which forces the network to focus on the directions, and not the scales, of the embedding vectors. This is our primary motivation in incorporating the ℓ_2-normalization layer, and, it converts the second term into a constant value of $0.5n$, which we drop from the objective function.

By plugging in the definitions of all terms into Eq. 4.8 and dropping the second term from Eq. 4.8, we obtain the final form of the objective function:

$$\min_{\theta} -\sum_{i=1}^{n} \langle \phi_\theta(x_i), \psi_{c_i} \rangle + \lambda \|\theta\|^2 \tag{4.9}$$

where λ is the regularization weight. The final formulation can be interpreted as a regression-like training approach that aims to maximize the expected correlation across the sample embeddings and the corresponding class keys.

4.2.3 Class Key Generation

In our approach, even if an adversary gets involved during training, it cannot target a particular class without knowing its class key. However, there is still a chance that the adversary may target an arbitrary class by using a randomly generated key ψ_{attack} as the target key, and, aim to reconstruct the samples belonging to one of the classes without necessarily knowing the identity of the targeted class. In principle, such an attack can be successful if the randomly generated key is sufficiently similar to one of the actual class keys, if *i.e.* $\|\psi_c - \psi_{\text{attack}}\|_2 \leq \delta$ for any c. However, we emphasize that there is no supervisory signal that the attacker can utilize to guess private class keys better than random. It is also not possible to run a GAN attack to reconstruct class keys, due to the lack of any reference score that can be leveraged for this purpose.

To protect the system against such random key-based brute-force attack, it is essential to reduce the probability of approximately replicating class keys by randomly generating keys. Therefore, we need to minimize the probability of generating keys that will lead to scores highly correlated with one of the class scores, without relying on the restrictions on the keys used by the participants, such that an adversary will (most likely) not be successful in training a generative model through a GAN attack.

One may consider addressing this problem by determining all class keys in a centralized manner. However, such a prevention technique is not reliable as it is typically not possible to enforce participants to use the assigned keys, *i.e.* the adversary may still attempt to attack with a random key. Also it is possible that the central server might be compromised. For this reason, we let each participant to generate its class keys independently. To further reduce the probability of key "conflicts", *i.e.* having highly-correlated class key pairs, we opt to using high-dimensional normally-distributed class keys and apply ℓ_2-normalization to them in practice. This design is rooted in the observation that as the key dimensionality increases, ℓ_2 normalized keys tend to be progressively less correlated. This observation can easily be verified in an empirical manner: in Figure 4.2, we randomly generate 100 ℓ_2-normalized vectors, find the maximum of pair-wise dot products of these vectors and plot the maximum of maximums of the pair-wise dot products by repeating this process 1000 times for a variety of d_{key} values. We observe that as the size of keys increase, the maximum overlap across sampled key pairs sharply diminishes. From this empirical finding, we can claim that it is more robust to generate high dimensional vectors to prevent GAN attacks.

We note that while our decentralized scheme addresses the key generation problem to a large-extend, it may possibly lead to complications in two ways. First, if the participants have overlapping classes, they will almost certainly generate and use completely different keys for their shared classes. Fortunately, it turns out that this does not necessarily lead to the failure of the framework and the approach behaves well to a large extend, which we experimentally verify in Section 4.3.

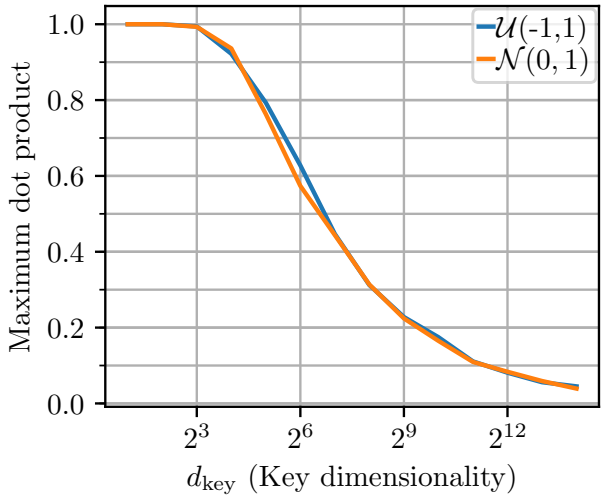


Figure 4.2: The correlation between two random vectors with respect to their dimension d. For $d_{\text{key}} = 2^1, \cdots, 2^{14}$ we randomly generate 100 ℓ_2-normalized vectors by sampling either from $\mathcal{N}(0,1)$ or $\mathcal{U}(-1,1)$. Then, we find maximum of pair-wise dot products of these vectors. We repeat this process 1000 times for each d_{key} and plot the maximum of maximums of the pair-wise dot products.

Second, the use of very high-dimensional keys may lead to training difficulties due to a drastic increase in the model complexity. We show how to address this problem in a technical and principled way in the following subsection.

4.2.4 Learning with High Dimensional Keys

In the proposed framework, as also previously discussed, it is important to have distinctive class keys, with minimal cross-class key correlations for both the quality of the resulting classifier and the success of the GAN-attack prevention mechanism (against the random-key attacks). In order to reduce the probability of having highly-correlated pairs of randomly and independently generated class keys, we aim to use very high dimensional vectors. However, naively increasing the class key dimensionality undesirably increases the number of trainable parameters in the last layer of the $\phi_\theta(x)$ network. Therefore, this increase in the dimensionality of key vectors also increases the complexity of overall model architecture which may (I) slow down training significantly, (II) cause over-fitting

Algorithm 1 Key Protected Classification for Collaborative Learning

Pre-Training Phase:

1: Participants agree on the collaborative learning framework parameters (see Section 4.1.1) and also the dimensionality of private class keys d_{key}.
2: Each participant p generates a random class key ψ_c^p for each class c of its classes.
3: Each attacker participant p generates (at least) one extra placeholder $c_{\text{fake}}^{\text{p}}$ and the corresponding class key $\psi_{\text{attack}}^{\text{p}}$ for running a GAN attack towards some random class key through adversarial training.

Training Phase

4: **for** epoch = 1 to n_{epochs} **do**
5: **for** p in Participants **do**
6: download θ_d fraction of parameters from PS
7: replace downloaded parameters with local ones
8: **if** p is an attacker **then**
9: train the local generator by using $\psi_{\text{attack}}^{\text{p}}$
10: generate M samples from generator
11: label the generated samples as $c_{\text{fake}}^{\text{p}}$
12: merge generated samples with local dataset
13: **end if**
14: train local model with local dataset
15: upload θ_u fraction of differences in parameters to PS
16: **end for**
17: **end for**
18: Participants make their class keys publicly available.

to training samples, (III) and therefore, lead to a poor test performance.

To overcome these problems, we propose to add a *fixed* dense layer with an activation function at the last layer of the architecture. By doing so, parameters of this dense layer are stochastically predetermined and kept unchanged throughout training. Thus, the layer does not impose any extra trainable parameters to be learned. This layer, therefore, effectively maps the preceding low-dimensional embedding vectors to a much higher dimension space. We pre-define the parameters of this layer and keep it shared among all participants.

We experimentally verify the effectiveness of this approach: in Section 4.3.2 and Section 4.3.3 we show that key dimensionality can be increased without deteriorating the participant training convergence using the proposed fixed-mapping

layer and as a result, the random-key GAN attacks can be prevented.

Finally, to summarize the overall framework, we present the list of main steps in Algorithm 1. We note that the underlying collaborative training scheme is the same as the vanilla collaborative learning algorithm of Shokri *et al*. [48]. However, unlike [48], here the participants and the adversary additionally create random private class keys, and, train the proposed key-protected classification scheme instead of a traditional classifier.

4.3 Experiments

In this section we demonstrate that our proposed solution prevents GAN attacks while enabling effective collaborative classifier learning over the participants. We first explain the experimental setup in Section 4.3.1. We verify that our key-based learning formulation performs well in absence of an adversary in Section 4.3.2. We empirically demonstrate that our proposed approach prevents GAN attacks in Section 4.3.3. We evaluate the performance of the collaborative learning approach when there are overlapping classes across the participants in Section 4.3.4. and finally, we demonstrate that key-based regression yields comparable results with cross entropy in Section 4.3.5.

4.3.1 Experimental setup

We perform experiments on the well-known MNIST handwritten digits [60] and AT&T Olivetti Faces [61] datasets. We choose these datasets as they provide a particularly challenging and suitable experimental setup for our purposes: (I) Both of these datasets contain samples that are relatively easy to generate, therefore, they are particularly challenging to protect against a GAN attack. While several improvements have been proposed for improving GANs, *e.g.* [10, 108, 109, 110], it is well-known that GANs can be difficult to train [111].

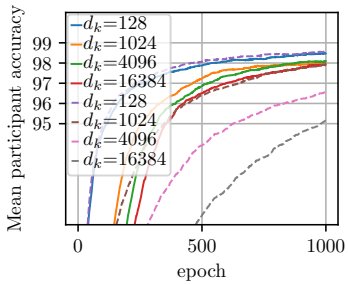
Therefore, we want use datasets where the GAN-attack generator can easily capture relevant data statistics and perform a successful GAN attack. This situation allows us to evaluate the proposed CLF scheme in a challenging scenario. (II) Qualifying the reconstructions obtained by the adversary is relatively easy on these datasets, which allows us to more confidently argue about the success of our formulation.

We observe that collaborative learning with 5 or more participants is challenging as parameters in the PS overfit quickly to the local dataset of any one of these participants, since local models tend to overfit into their local sets as the number of classes per participant decreases. To overcome this, we use plain stochastic gradient descent (SGD) for optimization and instance normalization [112] in the networks. We carefully tune learning rate and λ on validation sets. Besides, we also find that learning with fixed layer is compelling especially when the fixed layer transforms embeddings to a very high dimensional space (*e.g.*, to $\mathbb{R}^{16384}$). For meta-parameters of collaborative learning framework, we set θ_d and θ_u to 1.0 since [58] shows that the GAN attack also works for smaller values of θ_d and θ_u. We also exclude gradient selection mechanism, γ and τ from the experiments. Finally, we note that while increasing the key dimensionality can increase robustness against the GAN attack, setting it to a too large value may lead to numerical instability and/or delay the convergence, therefore, d_{key} should be treated as an hyper-parameter and therefore tuned specifically for each *model architecture* on a separate held out public dataset with relevant content.

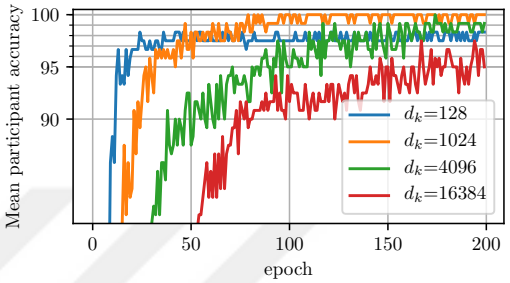
All experiments are implemented in TensorFlow [113].

4.3.2 Collaborative Learning Evaluation

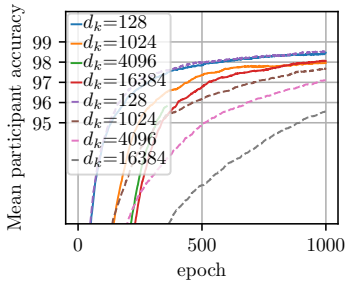
In this section, we evaluate our collaborative learning framework formulation with private class keys, in the absence of an adversary, in order to show that our formulation enables effectively learning a shared model. For this purpose, we examine how the key size d_{key} and a fixed layer affect the test set accuracy of the local models, over the training iterations. To do that, we split MNIST among 2, 3



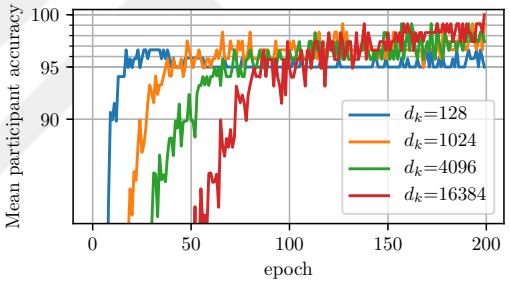
(a) MNIST with 2 participants.



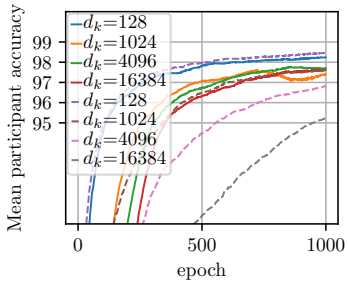
(b) Olivetti Faces with 2 participants.



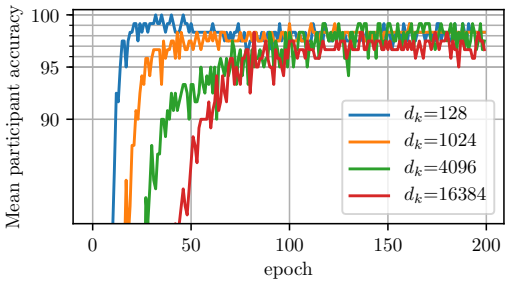
(c) MNIST with 3 participants.



(d) Olivetti Faces with 3 participants.



(e) MNIST with 5 participants.



(f) Olivetti Faces with 5 participants.

Figure 4.3: Mean participant accuracies obtained in collaborative learning with 2, 3, and 5 participants over the MNIST and the Olivetti Faces datasets. The dashed lines in the MNIST figures indicate that there are fixed layers in the local models of participants (see Section 4.2.4 for details). We see that using a fixed layer tends to delay the training convergence with similar final accuracy. In the Olivetti Faces plots, fixed layer based experiments are excluded for brevity.

and 5 participants, run two different collaborative learning frameworks (one with fixed layers and one without fixed layers) for $d_{\text{key}} \in \{128, 1024, 4096, 16384\}$ and report mean participant accuracies (MPA) during the training. We also examine mean participant accuracies with respect to $d_{\text{key}} \in \{128, 1024, 4096, 16384\}$ and number of participants in a collaborative learning framework when there is no fixed layer in local models of participants for the Olivetti Faces dataset. The results of these experiments are given in Figure 4.3.

Observations. We find that even on MNIST it takes considerably longer time to achieve a high classification accuracy compared to the centralized case when applying collaborative learning framework in its vanilla form, *e.g.* without any "hacks" to stabilize and speed up learning. We believe that this is due to the noise introduced by participants when they upload parameters to the server from their local models after performing one epoch of local training. We note that, however, in this work we focus not on improving the training performance of collaborative learning framework, but instead on evaluating the proposed approach within the existing CLF protocol.

MNIST experiments show that fixed layers yield better scores only when $d_{\text{key}} = 128$. For other key dimensionalities and when same hyper-parameters used, we see that a fixed layer delays the convergence time of participants. In the Olivetti Faces experiments we see that in all cases all participants can achieve at least 95% test set accuracy. All these results suggest that fixed layer can be used as an effective tool for increasing the embedding dimensionality without increasing the model complexity, with some penalty mainly in the convergence speed.

From now on, in all experiments, we share a fixed layer among the participants during their local trainings.

4.3.3 Preventing GAN Attack

In this section we evaluate the success of our approach in preventing GAN attacks. We perform three sets of experiments to cover different aspects of our proposed

solution:

I. The class key of one of the classes is given to the adversary.

II. Adversary generates random keys that are δ far (measured in Euclidean distance) from some class key ($\|\psi_{\text{attack}} - \psi_c\| = \delta$ for some c).

III. Adversary generates random keys.

We evaluate samples generated by attackers both qualitatively (by visual inspection) and quantitatively by computing accuracies of the samples using an MNIST model pre-trained on centralized data. For this evaluation, we pre-train a CNN model to classify the MNIST digits over the complete dataset and use this model to classify synthetic samples generated by an adversary.

In Experiment I., we demonstrate the extreme case in which adversary guesses the exact same key of any class in collaborative learning framework. Although this is very unlikely to happen in practice with high-dimensional class keys (due to the fact that high dimensional keys overlap less, as previously discussed in Figure 4.2), we consider this case as a baseline, in which case the GAN attack is expected to be successful. The results for this case over the MNIST and Olivetti Faces datasets are presented in Figure 4.4. In these results, we observe that the attacker succeeds in reconstructing target class images with high visual similarity, as expected. These results verify the effectiveness of the GAN attack in our experimental setup.

In Experiment II., we conduct a study to understand the success of the GAN attack as a function of the minimum similarity between the GAN attack key and one of the actual class keys. For this purpose, we provide the attacker a function that takes the key of the victim class (ψ_{desired}), and a predefined degree of similarity (δ), and, generates an ℓ_2 normalized random key ψ_{attack} such that the generated key approximates the key of the victim class: $\|\psi_{\text{attack}} - \psi_{\text{desired}}\| \approx \delta$. We experiment with several values of $\delta \in [0, 1.3]$ and see that as δ decreases, the attacker starts to generate visually more similar images to the ones in the victim.

(a) MNIST

(b) Olivetti Faces

Figure 4.4: Demonstration of GAN attack when **the attacker has access to the exact key** of the class that it is attacking, on the (a) MNIST and (b) Olivetti Faces datasets. We split the classes among two participants, one being an adversary. In each dataset, we perform 5 different experiments, where the adversary attacks one of the classes owned by the victim. In MNIST (a), the victim owns the digit classes 0-4, and, we directly present the GAN attack results. In Olivetti Faces (b), the victim owns the photos of 20 people, and, we show both the original class samples (upper row) and the corresponding GAN attack reconstructions (lower row). Overall the results demonstrate the effectiveness of the GAN attack, when key-based protection is not utilized.

By visually inspecting the reconstructions of the adversary, we deduce that for $\delta \geq 1.1$ on Olivetti Faces, the GAN attack produces incomprehensible results. In almost all attack experiments we observe that generator of an adversary tends to collapse into a single mode that is either a noise or a meaningful image, depending on the δ value. Consequently, on MNIST, we find a sharp threshold at $\delta = 0.5$ such that when $\delta \geq 0.5$ the synthesized samples become unrecognizable for the pre-trained MNIST classifier. Reconstructions for different values of δ are given in Figure 4.5 for the Olivetti Faces and MNIST datasets.

In Experiment III., we show that our model is robust against the GAN attack when there is no constraint on key generation, *i.e.* all keys are randomly and

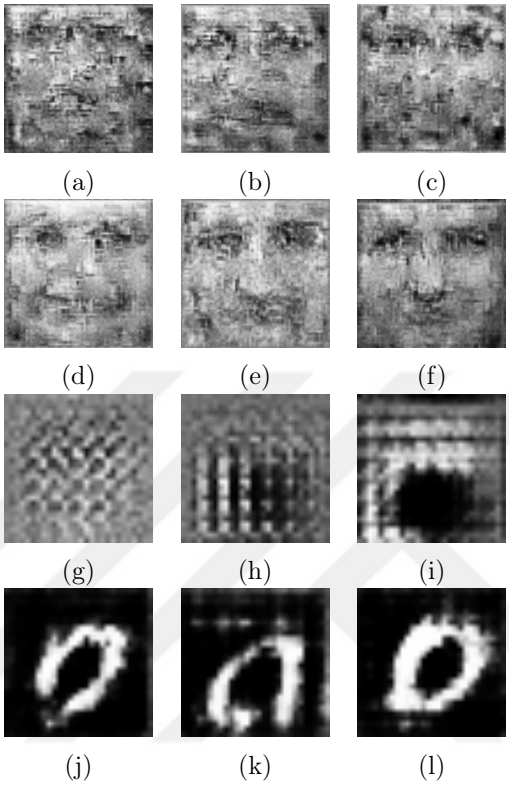


Figure 4.5: Approximating the maximum Euclidean distance between any class key and $\psi(c_{\text{attack}})$ necessary for the adversary to succeed in attack. From (a) to (f) and (g) to (l), reconstructions of the adversary when it generates random keys that are $\delta \in \{1.3, 1.2, 1.1, 1.0, 0.5, 0.1\}$ far from the key of the class whose label is 24 in Olivetti Faces dataset and 0 in MNIST dataset, respectively.

independently generated by the participants, including the guessed ones. We show in Figure 4.6 and Figure 4.7 that for sufficiently large key dimensionalities, the GAN attack fails on both MNIST and Olivetti Faces, respectively. In addition, we observe that the pre-trained MNIST classifier obtains 0% accuracy on the generated examples, which also support our claim that GAN attack fails. This situation occurs in practice since the local generators of the adversaries collapse to singular modes whose samples are unrecognizable for the pre-trained classifier.

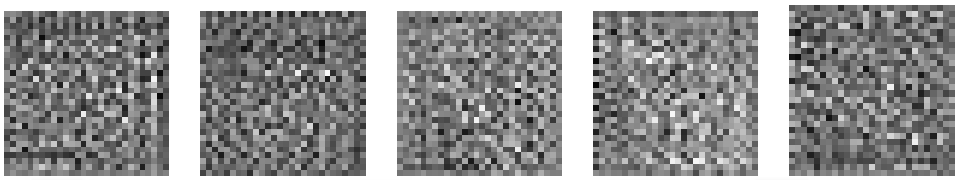


Figure 4.6: GAN attack results on the MNIST dataset using random attack keys. We split MNIST among 5 participants which are also attackers. Figures from left to right correspond to sample reconstructions obtained by each adversary, when $d_{\text{key}} = 16834$. We run the experiments until each participant achieves 97% accuracy on its local dataset. One can see that generators fail at capturing a valid digit mode which indicates that the GAN attack is prevented.

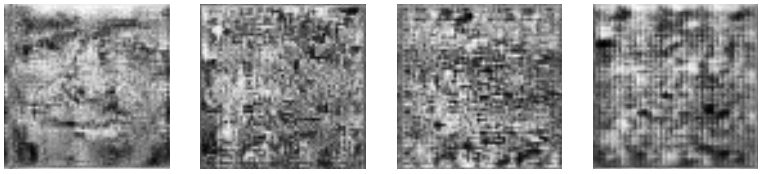


Figure 4.7: GAN attack results on the Olivetti Faces dataset using random attack keys. From left to right, reconstructions of adversary for $d_{\text{key}} \in \{128, 1024, 4096, 16834\}$, respectively. We show that the mode that GAN learns is likely to belong to one of the classes in collaborative learning framework, for small d_{key}. However, as d_{key} become larger, the quality of the GAN attack reconstructions degrade drastically.

4.3.4 Shared Classes Among Participants

So far, we have assumed that there are no overlapping training classes across the participants. However, our approach can be utilized even in the cases where participants own samples from shared classes, without sharing their private class keys. We claim that as class keys becomes nearly orthonormal to each other, a properly trained shared model would learn to map samples belonging to same class but hosted by different participants to a space spanned by the private class keys defined for that class by different participants. In this section, we show that our formulation works well when there exist shared classes among the participants.

Let c be a class shared by the participants i and j. Let there exist two class key vectors in $\mathbb{R}^2$, namely ψ_c^i and ψ_c^j, which are two distinct keys of class c generated by participants i and j, respectively. ϕ_θ is the network that maps input data (*e.g.*, images) into the embedding space. Let X_i^c and X_j^c be the samples that participants i and j have for class c. For our analysis, we assume that the angle between ψ_c^i and ψ_c^j are approximately orthogonal, which is correct in practice when the class keys are high-dimensional, ℓ_2-normalized vectors. In addition, as we expect observing similar examples in X_i^c and X_j^c, we assume, for simplicity, that the sets X_i^c and X_j^c contain a single shared sample x^c.

Then, since $\|\psi_c^i\| = \|\psi_c^j\| = 1$ by definition, and, $\|\phi_\theta(x^c)\| = 1$ due to the ℓ_2-normalization layer at the output of the network, the training objective, *i.e.* maximization of the dot product between the sample embedding and class key, for x^c can be equivalently expressed in terms of minimizing the angle between ψ_c^i and $\phi_\theta(x^c)$ (denoted by α_1), and, between ψ_c^j and $\phi_\theta(x^c)$ (denoted by α_2) [2]. In this simple example devised in $\mathbb{R}^2$, maximizing $\langle \psi_c^i, \phi_\theta(x^c) \rangle$ and $\langle \psi_c^j, \phi_\theta(x^c) \rangle$ with respect to θ by participants i and j iteratively would converge to a model such that $\langle \psi_c^i, \phi_\theta(x^c) \rangle \approx \langle \psi_c^j, \phi_\theta(x^c) \rangle \approx 0.7$ (*i.e.* $\cos(\pi/4)$), and $\alpha_1 \approx \alpha_2 \approx \pi/4$.

In consideration of this behaviour in much higher dimensional spaces, we perform the following experiments. For $d_{\text{emb}} = \{64, 256, 1024, 4096, 16384\}$, we generate three ℓ_2 normalized vectors, namely ψ_c^i, ψ_c^j and $\phi(x^c)$. Then we update

[2] $\langle \psi_c, \phi_\theta(x^c) \rangle = \|\psi_c\|_2 \cdot \|\phi_\theta(x^c)\|_2 \cdot \cos(\alpha) = \cos(\alpha)$

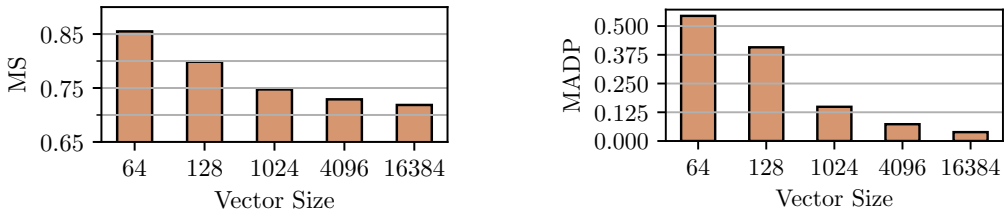


Figure 4.8: Analysis of keys when multiple participants use different class keys for the same class. (Left) We randomly generate ℓ_2 normalized $\psi_c^i, \psi_c^j, \phi(x^c) \in \mathbb{R}^{d_{\text{key}}}$ for $d_{\text{key}} \in \{64, 128, 1024, 4096, 16384\}$. Then we find the optimal $\phi(x^c)$ such that its dot product with ψ_c^i and ψ_c^j are maximized. Bars indicate that as we increase the dimensionality, ψ_c^i and ψ_c^j are more likely to be orthogonal, therefore $\phi(x^c)$ ends up being a vector at 0.7 correlation with each one. MS: Maximum score. (Right) We generate new ℓ_2 normalized vectors in $\mathbb{R}^{d_{\text{key}}}$ for $d_{\text{key}} \in \{64, 128, 1024, 4096, 16384\}$, and check their maximum absolute dot product with the final $\phi(x^c)$. We see that as we increase d_{key}, $\phi(x^c)$ converges towards the space spanned by ψ_c^i and ψ_c^j. This confirms that the approach is likely to behave well when training over shared classes, despite using different private keys across the participants. MADP: Maximum absolute dot product.

$\phi(x^c)$ to maximize $\langle \psi_c^i, \phi(x^c) \rangle$ and $\langle \psi_c^j, \phi(x^c) \rangle$ until convergence, by simple gradient descent. We repeat this procedure 1000 times and plot maximum of the scores obtained either by $\langle \psi_c^i, \phi(x^c) \rangle$ or by $\langle \psi_c^j, \phi(x^c) \rangle$ for all d_{key} in Figure 4.8 (left). We see that scores converge to 0.7 as we increase the class key dimensionality.

We continue our analysis by measuring the dot product of the tuned $\phi(x^c)$ with random class keys representing other classes. Our purpose is to interpret the optimization output of the previous experiment. For each tuned $\phi(x^c)$ we generate ℓ_2 normalized vectors ψ_k^{new} for $k = 1, ..., 1000$ and plot maximum absolute dot products of all ψ_k^{new} and $\phi(x^c)$, in Figure 4.8 (right). Results indicate that when d_{key} is sufficiently high, multiple participants can declare the same set of labels. For each common label, the accumulated model is likely to map samples belonging to the shared class onto a space spanned by the embeddings defined for that class among the participants. The resulting space is likely to be nearly orthogonal to any other class key. At test time, as labels are assigned according to the Equation 4.3, having multiple keys for a class does not constitute an issue, *i.e.* we can simply assign a test sample to the class whose one of the keys maximizes the classification score.

These observations confirm that the approach is likely to behave well when training over shared classes, despite using different private keys across the participants. Overall, the network is likely to map samples to points that are highly correlated with all duplicate keys of their ground-truth classes, and highly uncorrelated with the other ones. In fact, we have empirically verified that training with shared classes perform with no observable issues on both MNIST and Olivetti Faces (example results are omitted for brevity). Our experiments also suggest that when using sufficiently *long* (*i.e.* high-dimensional) keys, generator of each attacker is able to capture some mode that does not match with any of the modes of the classes in the collaborative learning framework, according to fake class key generated by the attacker.

4.3.5 Evaluating the Key Based Regression Loss

We are also curious about the performance of our regression-like objective that we formulate in Equation 4.9 when it is used out of the privacy context, *e.g.* in the supervised classification setup where all the training data is centralized. For this purpose, we train several ResNet-10 [3] models on the CIFAR-10 and CIFAR-100 datasets [62] by optimizing the cross entropy loss and the key-based cosine similarity loss, which we refer to as the *key-based regression*, and compare their results.

In the original ResNet models, there is one fully connected layer mapping non-negative image embeddings (obtained at the output of the last convolutional layer) to classification scores by computing dot products between weight vectors in the fully connected layer and the image embeddings then adding bias terms in the fully connected layer. Instead, in our experiments we compute cosine similarity scores between the weight vectors in the fully connected layer and the image embeddings. In `cross entropy` experiments, the weights of the fully connected layer are trainable, whereas in `key-based regression` experiments we replace the trainable weights with fixed orthonormal class keys, which we obtain by QR decomposition before training starts. Finally, we modify the last

Table 4.1: Key-based regression versus cross entropy.

Objective	**Dataset**	**Top-1**
Key-based regression	CIFAR-10	**94.8**%
Cross entropy	CIFAR-10	94.0%
Key-based regression	CIFAR-100	73.6%
Cross entropy	CIFAR-100	**74.2**%

convolutional layer by replacing the ReLU activation function with the tanh, based on the observation that computing cosine similarities by using non-negative image embeddings $\phi_\theta(x)$ tends to make training unstable.

Top-1 accuracies obtained for both loss functions are shown in Table 4.1. We see that `key-based regression` outperforms cross entropy on CIFAR-10 by a small margin, whereas on `cross-entropy` performs better on CIFAR-100. These results indicate that orthonormal class keys enforce the network to produce sufficiently orthonormal *class codes* at the last fully connected layer, and make the overall training formulation discriminative. We leave an elaborate study for key-based regression in a future work.

Chapter 5

Conclusions

In this thesis, we draw attention to two of the main data-related bottlenecks in existing deep learning solutions for fundamental computer vision problems. We argue that these problems either restrict the use of machine learning algorithms for certain scenarios or cause complex models to be inefficiently limited for a fixed set of objects. Therefore, we conjecture that they pose potential impediments to learning more robust visual embedding models or more accurate decision functions. To address these issues, we perform two studies where we propose novel approaches and validate their performances.

In Chapter 3, we study on a model that turns zero-shot learning into supervised classification problem. The novelty of this work is the gradient matching loss, an objective function optimized in order to train a generative model. We show that the weighted average of the gradient matching loss and Wasserstein GAN with gradient penalty term objective enables the generator network to capture modalities of CNN features given semantic class embeddings. In our experiments, we verify that when the generator is trained with this approach, it is able to synthesize discriminative features which are then used to train state-of-the-art classifiers for generalized zero-shot classification.

The zero-shot learning experimental evaluation protocol that we strictly follow [1] requires image embeddings to be extracted from a ResNet-101 pre-trained on ILSVRC-2012 training set. This powerful ResNet module is highly successful in linearly separating classes in its embedding space. This motivates us to investigate in future, how gradient matching loss performs on other forms of weakly-supervised learning tasks such as semi-supervised and few-shot learning, while training embedding models from scratch.

In Chapter 4, we present a key-based regression loss to perform object classification in order to prevent GAN attacks that target collaborative learning frameworks. More specifically, we introduce a classification model for participants, where random class keys represents classes. This key-based model provides effective learning over the participants and by utilizing high dimensional keys, class scores of an input is protected against an active adversary. This principled training technique restricts the access of each participant to only its own private keys, and, provides a way to increase the key dimensionality without increasing the model complexity. We verify the effectiveness of our formulation by empirically showing that I. the adversary is no longer able to choose which class to exploit, and II. the generator trained by the adversary cannot capture data distribution well enough to reconstruct any class. We believe that the proposed approach makes a step towards making collaborative learning safe and practical, which can potentially have a fundamental impact on learning models in sensitive data domains.

Appendix A

Softmax Over Infinitely-many Classes

In this section, we show the detailed derivation of our softmax generalization to infinitely-many classes. Below, we use the subscript notation on ψ to denote dimensions, instead of classes, and, use d instead of d_{key} for brevity. In order to compute $p(c|x) = \frac{\exp g_\theta(x,\psi_c)}{\mathbb{E}_\psi[\exp(g_\theta(x,\psi))]}$, we need to compute the expectation in the denominator:

$$\mathbb{E}_{\psi\sim\mathcal{N}}\left[\exp(g_\theta(x,\psi))\right] = \int_\psi f(\psi)\exp\left(g_\theta(x,\psi)\right)d\psi \tag{A.1}$$

Since the class keys are assumed to follow multivariate standard normal distribution, the integration can be simplified as follows:

$$\int_\psi f(\psi)\exp\left(g_\theta(x,\psi)\right)d\psi = \int_{\psi_1}...\int_{\psi_d} f(\psi_1)...f(\psi_d)\exp\left(g(x,\psi)\right)d\psi_1...d\psi_d \tag{A.2}$$

By re-arranging the terms and plugging-in Eq. 4.2.1, we obtain:

$$\int_{\psi_d} f(\psi_d)\int_{\psi_{d-1}} f(\psi_{d-1})...\int_{\psi_1} f(\psi_1)\exp\left(\psi^T\phi(x)\right)d\psi_1...d\psi_d \tag{A.3}$$

By converting the exponent of summation into a product of exponents, we obtain:

$$\int_{\psi_d} f(\psi_d)e^{\phi_d(x)\psi_d}\int_{\psi_{d-1}} f(\psi_{d-1})e^{\phi_{d-1}(x)\psi_{d-1}}...\int_{\psi_1} f(\psi_1)e^{\phi_1(x)\psi_1}d\psi_1...d\psi_d \tag{A.4}$$

where $\phi_i(x)$ refers to the i-th dimension of the $\phi(x)$ vector. Here, the inner-most term can be integrated out easily:

$$\int_{\psi_1} f(\psi_1) e^{\phi_1(x)\psi_1} d\psi_1 = \frac{1}{\sqrt{2\pi}} \int_{\psi_1} e^{-\frac{1}{2}\psi_1^2} e^{\phi_1(x)\psi_1} d\psi_1 \tag{A.5}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{\psi_1} e^{-\frac{1}{2}(\psi_1 - \phi_1(x))^2} e^{\frac{1}{2}\phi_1(x)^2} d\psi_1 \tag{A.6}$$

$$= e^{\frac{1}{2}\phi_1(x)^2} \tag{A.7}$$

where the last step is based on the observation that the remaining terms correspond to integrating out Gaussian probability density function with $\mu = \phi_1(x)$ and $\sigma = 1$. Now, we can plug-in this result into Eq. A.4 and repeatedly integrate out each dimension:

$$e^{\frac{1}{2}\phi_1(x)^2} \int_{\psi_d} f(\psi_d) e^{\phi_d(x)\psi_d} ... \int_{\psi_2} f(\psi_2) e^{\phi_2(x)\psi_2} d\psi_2 ... d\psi_d \tag{A.8}$$

$$= \prod_{i=1}^{d} e^{\frac{1}{2}\phi_i(x)^2} = \exp\left(\frac{1}{2}\sum_{i=1}^{d} \phi_i(x)^2\right) = \exp\left(\frac{1}{2}\|\phi(x)\|^2\right) \tag{A.9}$$

Finally, by plugging-in Eq. 4.4, we obtain a constant value of exp(0.5):

$$\mathbb{E}_{\psi\sim\mathcal{N}}\left[\exp(g_\theta(x,\psi))\right] = \exp\left(\frac{1}{2}\|\phi(x)\|^2\right) \tag{A.10}$$

$$= \exp\left(\frac{1}{2}\left\|\frac{\phi_u(x)}{\|\phi_u(x)\|}\right\|^2\right) \tag{A.11}$$

$$= \exp\left(0.5\right) \tag{A.12}$$

which completes the proof. Our generalization of softmax to infinitely-many classes takes the following final form:

$$\frac{\exp g_\theta(x,\psi_c)}{\mathbb{E}_\psi[\exp g_\theta(x,\psi)]} = \frac{1}{\exp 0.5} \exp\left(g_\theta(x,\psi_c)\right). \tag{A.13}$$

Bibliography

[1] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning - the good, the bad and the ugly," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

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

[8] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[9] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014.

[10] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016.

[11] A. Antoniou, A. Storkey, and H. Edwards, "Data Augmentation Generative Adversarial Networks," *arXiv:1711.04340*, 2017.

[12] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to Compare: Relation Network for Few-Shot Learning," *arXiv:1711.06025*, 2017.

[13] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[14] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks.," in *AAAI*, 2008.

[15] X. Yu and Y. Aloimonos, "Attribute-based transfer learning for object categorization with zero/one training example," in *European Conference on Computer Vision*, 2010.

[16] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[17] A. Farhadi, I. Endres, and D. Hoiem, "Attribute-centric recognition for cross-category generalization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[18] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[19] D. Parikh and K. Grauman, "Relative attributes," in *IEEE International Conference on Computer Vision*, 2011.

[20] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang, "Designing category-level attributes for discriminative visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[21] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele, "What helps where–and why? semantic relatedness for knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[22] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *IEEE International Conference on Computer Vision*, 2009.

[23] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Learning multimodal latent attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[24] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Attribute learning for understanding unstructured social activity," in *European Conference on Computer Vision*, 2012.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[26] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing*, 2014.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013.

[28] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[29] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[30] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems*, 2013.

[31] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in Neural Information Processing Systems*, 2013.

[32] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[33] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International Conference on Machine Learning*, 2015.

[34] Y. Fu and L. Sigal, "Semi-supervised vocabulary-informed learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[35] R. Qiao, L. Liu, C. Shen, and A. van den Hengel, "Less is more: Zero-shot learning from online textual documents with noise suppression," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[36] Y. Fu, T. M. Hospedales, T. Xiang, Z. Fu, and S. Gong, "Transductive multi-view embedding for zero-shot recognition and annotation," in *European Conference on Computer Vision*, 2014.

[37] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *European Conference on Computer Vision*, 2016.

[38] M. Bucher, S. Herbin, and F. Jurie, "Generating visual representations for zero-shot classification," in *IEEE International Conference on Computer Vision Workshops*, 2017.

[39] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[40] R. Felix, V. B. G. Kumar, I. Reid, and G. Carneiro, "Multi-modal cycle-consistent generalized zero-shot learning," in *European Conference on Computer Vision*, 2018.

[41] V. Kumar Verma, G. Arora, A. Mishra, and P. Rai, "Generalized zero-shot learning via synthesized examples," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[42] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[44] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[46] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "MS-Celeb-1M: A dataset and benchmark for large scale face recognition," in *European Conference on Computer Vision*, 2016.

[47] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv:1609.08675*, 2016.

[48] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[49] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, 2017.

[50] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[51] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized Collaborative Learning of Personalized Models over Networks," in *International Conference on Artificial Intelligence and Statistics*, 2017.

[52] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[53] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, 2011.

[54] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*, 2011.

[55] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *AAAI Conference on Artificial Intelligence*, 2016.

[56] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[57] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *International Conference on Learning Representations*, 2016.

[58] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.

[61] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *IEEE Workshop on Applications of Computer Vision*, IEEE, 1994.

[62] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[63] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in Neural Information Processing Systems*, 2009.

[64] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," in *International Conference on Learning Representations*, 2014.

[65] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha, "Synthesized classifiers for zero-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[66] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," in *IEEE International Conference on Computer Vision*, 2015.

[67] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[68] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han, "From zero-shot learning to conventional supervised classification: Unseen visual data synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[69] E. Kodirov, T. Xiang, Z. Fu, and S. Gong, "Unsupervised domain adaptation for zero-shot learning," in *IEEE International Conference on Computer Vision*, 2015.

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

[71] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal, "Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach.," in *AAAI*, 2016.

[72] M. Bucher, S. Herbin, and F. Jurie, "Improving semantic embedding consistency by metric learning for zero-shot classiffication," in *European Conference on Computer Vision*, 2016.

[73] J. Song, C. Shen, Y. Yang, Y. Liu, and M. Song, "Transductive unbiased embedding for zero-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[74] W. Wang, Y. Pu, V. K. Verma, K. Fan, Y. Zhang, C. Chen, P. Rai, and L. Carin, "Zero-shot learning via class-conditioned deep generative models," in *AAAI*, 2017.

[75] A. Mishra, M. Reddy, A. Mittal, and H. A. Murthy, "A generative model for zero shot learning using conditional variational autoencoders," *arXiv preprint arXiv:1709.00663*, 2017.

[76] H. Zhang, Y. Long, L. Liu, and L. Shao, "Adversarial unseen visual feature synthesis for zero-shot learning," *Neurocomputing*, 2018.

[77] Y. Long, L. Liu, F. Shen, L. Shao, and X. Li, "Zero-Shot Learning Using Synthesised Unseen Visual Data with Diffusion Regularisation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[78] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[79] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[80] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy*, 2019.

[81] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE Conference on Computer Communications*, 2019.

[82] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, 2006.

[83] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems*, 2009.

[84] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *IEEE Global Conference on Signal and Information Processing*, 2013.

[85] S. Song, K. Chaudhuri, and A. Sarwate, "Learning from data with heterogeneous noise using sgd," in *International Conference on Artificial Intelligence and Statistics*, 2015.

[86] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *International Conference on Artificial Intelligence and Statistics*, 2012.

[87] N. Hynes, R. Cheng, and D. Song, "Efficient deep learning on multi-source private data," *arXiv preprint arXiv:1807.06689*, 2018.

[88] T. Zhang, Z. He, and R. B. Lee, "Privacy-preserving machine learning through data obfuscation," *arXiv preprint arXiv:1807.01860*, 2018.

[89] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *International Conference on Machine Learning*, 2016.

[90] M. Pathak, S. Rane, and B. Raj, "Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers," in *Advances in Neural Information Processing Systems*, 2010.

[91] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, 2017.

[92] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[93] F.-J. González-Serrano, Á. Navia-Vázquez, and A. Amor-Martín, "Training support vector machines with privacy-protected data," *Pattern Recognition*, vol. 72, 2017.

[94] M. Gomez-Barrero, E. Maiorana, J. Galbally, P. Campisi, and J. Fierrez, "Multi-biometric template protection based on homomorphic encryption," *Pattern Recognition*, vol. 67, 2017.

[95] S. Wang and J. Hu, "Design of alignment-free cancelable fingerprint templates via curtailed circular convolution," *Pattern Recognition*, vol. 47, 2014.

[96] A. Anees and Y.-P. P. Chen, "Discriminative binary feature learning and quantization in biometric key generation," *Pattern Recognition*, vol. 77, 2018.

[97] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv preprint arXiv:1802.06739*, 2018.

[98] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and K. Ren, "GANobfuscator: Mitigating information leakage under gan via differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 14, 2019.

[99] M. B. Sariyildiz and R. G. Cinbis, "Gradient matching generative networks for zero-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[100] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017.

[101] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[102] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning*, 2014.

[103] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *International Conference on Machine Learning*, 2017.

[104] T. Miyato and M. Koyama, "cGANs with projection discriminator," in *International Conference on Learning Representations*, 2018.

[105] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[106] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *IEEE International Conference on Computer Vision*, 2017.

[107] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[108] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[109] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf, "AdaGAN: Boosting generative models," in *Advances in Neural Information Processing Systems*, 2017.

[110] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," in *International Conference on Learning Representations*, 2017.

[111] I. Goodfellow, "Neurips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[112] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[113] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems."