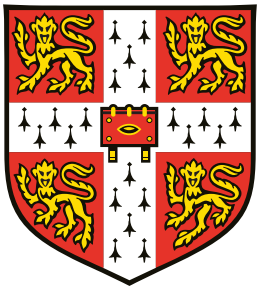


DEEP GENERATIVE MODELING OF MULTIMODAL DATA FROM CORRELATED SOURCES

Louis de Benoist

Wolfson College
University of Cambridge

A thesis submitted for the degree of:
Master of Philosophy (MPhil) in Advanced Computer Science

July 3, 2021

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Declaration

I Louis de Benoist of Wolfson College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14,963

Signed: Louis de Benoist

Date: June 18, 2021

Acknowledgements

I would like to thank my supervisor, Dr Bianca Dumitrascu, for her continued guidance and feedback throughout this project. Without her, none of this would have been possible. In addition, I wish to thank my mother and father, along with my little brother and sister, for always believing in me.

Abstract

We propose a deep generative model for learning shared latent representations of multi-view data when the data sources are correlated. The model generalizes deep probabilistic canonical correlation analysis (DPCCA) through an additional graph regularization term. We evaluate our method on synthetic data and find that it surpasses DPCCA when there is prior knowledge about source to source interaction. In addition, we show that our model can exploit spatial structural correlation in image data, paving the way for numerous biological applications.

We also introduce a non-probabilistic variant of the model, which extends graph regularized canonical correlation analysis (gCCA) through a reframing of the loss function to facilitate the use of deep learning methods.

Contents

1 Introduction **2**

2 Background **6**

2.1 Canonical Correlation Analysis 7

2.1.1 Classical formulation 7

2.1.2 Deep CCA 9

2.1.3 Generalizations to $M > 2$ views 9

2.2 Probabilistic CCA 10

2.3 Deep Probabilistic CCA 12

2.3.1 Generating one modality from the other 15

2.4 Exploiting Source Information in CCA 17

2.4.1 Graph Regularized CCA 17

3 Deep Models for Correlated Sources **20**

3.1 Graph Regularized Deep CCA 21

3.2 Graph Regularized Deep PCCA 23

3.3 Spatially Correlated Sources 25

3.3.1 Space Informed gDPCCA 26

4 Experiments **29**

4.1 Pseudo Gene Expression 29

4.1.1 Data Generation 30

4.1.2 Methodology 32

4.1.3 Results 33

4.2 Bimodal MNIST 35
4.2.1 Data Generation 35
4.2.2 Methodology 37
4.2.3 Results 37
4.3 Colored MNIST 45
4.3.1 Data Generation 46
4.3.2 Methodology 48
4.3.3 Results 48
4.4 What about gDCCA? 51
4.4.1 Iris Dataset 51

5 Case Study: Tension in Epithelial Cells 54
5.1 Motivation 54
5.2 Tension Inference with gDPCCA 56
5.3 Pressure Inference 57
5.3.1 Data Generation 57
5.3.2 Methodology 61
5.3.3 Results 62

6 Looking Forward 64

List of Figures

1.1 Multiple views are observed from a given source. In this example, the source can be understood in two ways: it can be perceived visually or auditorily. 3

2.1 Visualization of CCA. As we can see the two views are projected to a shared space in which we seek to maximize the correlation (i.e. minimize the distance between the normalized projected vectors) 8
2.2 Graph representation of PCCA 11
2.3 Overview of the DPCCA architecture 13
2.4 Overview of the graph regularization for CCA. In this example, sources are animals, of which we have two views: images and sounds. $\mathbf{E}_1$ and $\mathbf{E}_2$ map the two views to some shared space where they are correlated (like-colored arrows denote projections of two views of the same source in the shared space). The graph on the right gives some prior information on the relationship between the sources: lions and jaguars. Since the two are related in the graph, the projections of the two sources will be pushed closer together in the shared space. 18

3.1 Overview of the different methods introduced thus far and our proposed algorithms. 21

3.2 Visualization of how the loss for gDCCA is computed at every iteration (for every group of elements for which we know the source to source correlation). 22
3.3 Visualization of how the loss for gDPCCA is computed at every iteration (for every group of elements for which we know the source to source correlation). 24
3.4 Visualization of how spatially correlated patches in an image can be used with gDPCCA. Each of the small patches has an "image view" and some other view (e.g. a physical quantity). The patches in the image are related through their relative distances. If we assume that the source objects are related in a way that is captured by the spatial correlation for this view, then we can use some kernel function based on the distance between patches to build the correlation graph $\mathcal{G}$. . . 27

4.1 Procedure for sampling multi-view synthetic data points coming from a shared source graph. Using the source graph's adjacency matrix G as covariance, we sample d points. This leads to multiple d-dimensional centroids. Choosing a centroid uniformly and sampling around it leads to a latent variable z, from which we can generate two views $\mathbf{x}_m \in \mathbb{R}^{d_m}$ using some $d_m \times d$ matrix W_m. 30
4.2 Projection of the latent space on the first two PCA components of data synthetically generated from a graph with 200 distinct sources, from which we drew 3 samples. As we can see in (a), most of the variance in the latent space is determined by the centroids, making separation between the sources in $\mathcal{G}$ feasible. As is the case in (b), points can also be clustered according to which sample they were collectively drawn from. . . 31

4.3 Loss averaged over 5 runs on synthetic data when varying the γ parameter for gDPCCA. When γ is 0, the model reduces to DPCCA. 33
4.4 Illustration of the two different views generated from a given image in the MNIST dataset 35
4.5 Loss averaged over 3 runs on Bimodal MNIST when varying the γ parameter for gDPCCA. 39
4.6 Switching between modalities by sampling from the learned latent space. The sample and the input that was used to generate it are combined to produce the final image. As before, view 1 denotes the top half and view 2 the second half. For example, in (a), the top half of the resulting image is sampled knowing the bottom half. 40
4.7 Sampling the top view knowing the bottom view. In (a) and (b), we see two different sampled reconstructions obtained when knowing the bottom part. Each image is the combination of the original bottom view and the model's attempt at generating a matching top. The bottom's ambiguity could lead to the reconstruction of either a 0 or a 3, despite the ground truth being a 0. 41
4.8 Visualization of the shared latent space for gDPCCA with $\gamma = 10^{-9}$ 42
4.9 Visualization of the view-specific latent space corresponding to the first view (the top half of each image) for gDPCCA with $\gamma = 10^{-9}$. The 2-component LDA transform was applied to project the data to two dimensions. 43

4.10 Visualization of the view-specific latent space corresponding to the second view (the bottom half of each image) for gDPCCA with $\gamma = 10^{-9}$. The 2-component LDA transform was applied to project the data to two dimensions. 45
4.11 Illustration of the two different views generated from a given image in the colored MNIST dataset 46
4.12 Loss on Colored MNIST data when varying the γ parameter for gDPCCA. 48
4.13 2D PCA projections of the latent spaces when $\gamma = 0.0005$ on the colored MNIST dataset. 50
4.14 CCA loss and graph regularization loss on the Iris dataset for models with different values of γ. 52
4.15 Shared embeddings of the model when $\gamma = 100$. In both (a) and (b), we can see the embedded samples. In (a), the color denotes the view while, in (b), it denotes the class to which each sample belongs. 53

5.1 Illustration of the data processing needed for tension inference to be done with gDPCCA. 56
5.2 Illustration of the process for generating spatially correlated patches of synthetic cells. 58
5.3 Loss on the pressure dataset when varying the γ parameter for gDPCCA. 62
5.4 Sampled patches from the trained gDPCCA model. . . 62

List of Tables

4.1 Average loss for gDPCCA with different graph regularization levels (controlled by the hyperparameter γ). . . 34
4.2 Average loss for gDPCCA on Bimodal MNIST with different graph regularization levels (controlled by the hyperparameter γ). 38

Chapter 1

Introduction

"Everything we see is a perspective, not the truth." This quote, which dates back almost two thousand years, paraphrases a famous passage from Roman emperor Marcus Aurelius's Meditations.

In a courtroom, a witness's visual identification of the suspect might be enough to convict. However, our eyes are often deceiving. In many cases, DNA, audio recordings, videos, and ballistic reports are also gathered and analyzed. With all of this evidence, the prosecution is somehow able to then pinpoint the truth — to ascertain one's guilt "beyond a reasonable doubt." But ultimately, what is the *truth*?

The truth can be understood as a convergence of perspectives: the more we know about an event or an object, the more accurately we can describe it.

This goes back to Leibniz's *Principle Identity of Indiscernibles* [5], the converse of his undisputed *Principle of Indiscernibility of Identicals*, which states that two objects which share all the same properties are identical. This simple statement, although often taken to be axiomatic, is widely debated in the philosophical community.

One famous counterargument was posed by Max Black: suppose we

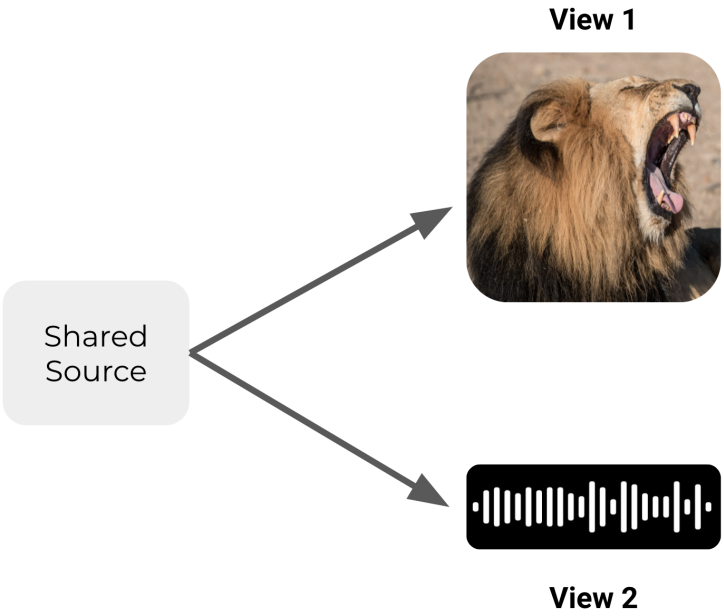


Figure 1.1: Multiple views are observed from a given source. In this example, the source can be understood in two ways: it can be perceived visually or auditorily.

place two identical spheres in a symmetrical universe without any observer, preventing the existence of any differentiating property based on relative position [8]. In this case, the balls would have all the same properties and, yet, be distinct.

However, object equality, in the Leibnizian sense, inherently depends on the observer. Diderot famously exemplified this through his "blind man" thought experiment [19]. According to him, blindness does not hold one back from perceiving the world; instead, the world is merely viewed through a different lens. In the case of Black's balls, if we suppose that one of the balls is a different color, a blind man would assert that the balls are identical. To him, the observer, all the properties are the same. The observer is thus a fundamental part of the equation [6]. In mathematics, the same idea applies when we consider latent variable models. A source object, which we call the *latent variable*, can be perceived in vari-

ous ways, through some mathematical transformations that yield one specific view of the source. As we can see in Figure 1.1, a given source yields various perspectives (or views) which we then observe.

An interesting question, therefore, is whether we can reverse this process. Given the observed views, can we recover the source, this *truth* from which the views were derived? This, in essence, is the focus of this dissertation. In particular, we will introduce ways of answering this question from the standpoint of deep generative modeling.

Given a set of paired views, the objective is to derive algorithms that enable us to generate what one view is likely to be given the other.

This problem has a wide range of applications. Indeed, multimodal data is quite common and can be found anywhere from biology (e.g. histology slides paired with gene expression) to image captioning, or even language translation [14].

Chapter 2, the background chapter, will begin by laying down the classical methods used for finding correlation between modalities and how, recently, these methods were adapted to deep learning using autoencoders.

Then, in **Chapter 3**, we will introduce two novel algorithms that exploit additional knowledge of source to source interaction. For instance, does the knowledge that tigers and lions are similar help us improve how we generate sounds of animals given their image?

In **Chapter 4**, we will then explore how these new methods improve the state of the art and are general enough to be used in various domains, including biophysics. In particular, we will look at how the spatial correlation between patches in images can be used to improve their latent representations.

Lastly, **Chapter 5** will take as case study the problem of tension inference in epithelial cells and discuss ways in which our new al-

gorithm could be used to improve current methods.

Chapter 2

Background

In the following, we will outline the foundations upon which our work is based. This background chapter will allow us to then ease into our two novel algorithms.

One of the most common settings in multimodal learning is one in which we know, in advance, a set of paired data points that come from different modalities, such as paired audio-visual signals or images of a common target taken from different viewpoints.

Suppose that we have access to data sampled from two views $\mathbf{x}^1$ and $\mathbf{x}^2$, which both come from a shared source $\mathbf{s}$. In particular, we know the points $(\mathbf{x_i^1})_{i=1}^N$ and $(\mathbf{x_i^2})_{i=1}^N$ along with a direct correspondence between modes: $\mathbf{x_i^1}$ and $\mathbf{x_i^2}$ correspond to the same source $\mathbf{s_i}$.

The goal is to understand the underlying source shared by the two views. To do so, most methods rely on finding a mapping to an approximation of the source space. Once suitable transformations have been determined, this formulation then enables switching between modalities, provided that we learn a probability distribution from which we can sample.

In this chapter, we will introduce the following three methods: CCA, PCCA, and DPCCA — all of which build on top of each other. Canon-

ical Correlation Analysis (CCA) finds a linear mapping that minimizes the distance between matching points in the shared space. Probabilistic Canonical Correlation Analysis (PCCA) reformulates CCA in a probabilistic way, leading to a meaningful latent space with a probabilistic structure. Deep Probabilistic Canonical Correlation Analysis (DPCCA) leverages the modeling power of deep learning to enable PCCA to be useful for more complex problems involving images, for example.

Finally, we will discuss graph regularized CCA (gCCA), which enables the inclusion of source to source correlation when determining the latent space mappings.

2.1 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA), first introduced by Hotelling [17][16], is a well established method in statistics which relies on finding linear transformations between different modalities and a shared d-dimensional latent space.

2.1.1 Classical formulation

Consider a set of N source vectors $\{s_i \in \mathbb{R}^\rho\}_{i=1}^N$, which we place in a matrix $\mathbf{S} \in \mathbb{R}^{\rho \times N}$. For each source, we have access to centered data from M different views, $\{\mathbf{X}_i \in \mathbb{R}^{d_i \times N}\}_{i=1}^M$ [10].

The classical formulation of CCA introduced by Hotelling [17] takes $M = 2$ and looks for $\mathbf{U}_1 \in \mathbb{R}^{d_1 \times d}$ and $\mathbf{U}_2 \in \mathbb{R}^{d_2 \times d}$ that minimize the Euclidean distance between the linear projections $\mathbf{U}_1^T\mathbf{X}_1$ and $\mathbf{U}_2^T\mathbf{X}_2$ — the *canonical variables* — subject to an orthonormality constraint to ensure that the projections in each of the directions are uncor-

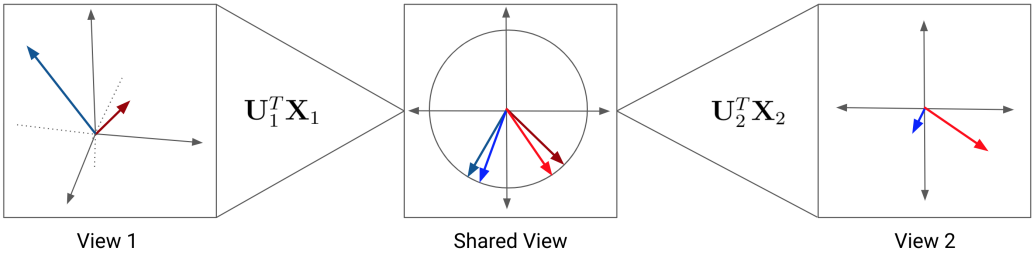


Figure 2.1: Visualization of CCA. As we can see the two views are projected to a shared space in which we seek to maximize the correlation (i.e. minimize the distance between the normalized projected vectors)

related [10][30],

$$\min_{\mathbf{U}_1,\mathbf{U}_2} ||\mathbf{U}_1^T\mathbf{X}_1 - \mathbf{U}_2^T\mathbf{X}_2||_F^2 \tag{2.1}$$

$$\text{s.t } \mathbf{U}_m^T(\mathbf{X}_m^T\mathbf{X}_m)\mathbf{U}_m = \mathbf{I} \text{ for } m \in \{1,2\} \tag{2.2}$$

The canonical variables themselves can be seen as d-dimensional approximations of S.

As can be seen in Figure 2.1, CCA can be viewed as the problem of finding transformations that maximize the similarity between the vectors once they are projected in the latent space. As measure of similarity, the *cosine similarity* metric is being used, which simply translates to saying that two non-zero vectors u and v are similar if the cosine of the angle between them is small, i.e.

$$\textbf{similarity}(\mathbf{u},\mathbf{v}) = \frac{\langle \mathbf{u},\mathbf{v}\rangle}{||\mathbf{u}||||\mathbf{v}||} \tag{2.3}$$

which takes values between 0 (least similar) and 1 (most similar). As we can see, this is equivalent to minimizing the Euclidean distance between the vectors in the latent space once they have been normalized.

2.1.2 Deep CCA

CCA can be made nonlinear by replacing the linear maps with neural networks. Indeed, this is exactly what Deep Canonical Correlation Analysis (DCCA) does [1]. The objective is to maximize the correlation between the embeddings of the two views,

$$\arg\max_{\theta_1,\theta_2} \textbf{similarity}\left(\mathbf{E}_1(\mathbf{X}_1), \mathbf{E}_2(\mathbf{X}_2)\right) \tag{2.4}$$

where $\mathbf{E}_m$ is parametrized by weights θ_m, which are optimized via gradient descent.

2.1.3 Generalizations to $M > 2$ views

CCA was originally conceived for two views. However, multiple generalizations exist to $M > 2$ views [10], such as sum-of-correlations MCCA [25] and maximum-variance MCCA [20]. In particular, the maximum-variance formulation minimizes the distance between each of the canonical variables and an approximation of the shared space $\hat{\mathbf{S}}$ (which it also looks for as part of the optimization) instead of minimizing the distance between the canonical variables themselves [10][20],

$$\min_{\{\mathbf{U}_m\}_{m=1}^M, \hat{\mathbf{S}}} \sum_{m=1}^{M} ||\mathbf{U}_m^T \mathbf{X}_m - \hat{\mathbf{S}}||_F^2 \tag{2.5}$$

subject to $\hat{\mathbf{S}}$ being orthonormal (i.e. $\hat{\mathbf{S}}^T\hat{\mathbf{S}} = \mathbf{I}$). It was shown [10] that if we assume full rank for each view's sample covariance matrix, then, given $\hat{\mathbf{S}}$, for each $m \in \{1, \cdots, M\}$, the optimal $\mathbf{U}_m$ is given by

$$\mathbf{U}_m = (\mathbf{X}_m\mathbf{X}_m^T)^{-1}\mathbf{X}_m\hat{\mathbf{S}}^T \tag{2.6}$$

Meanwhile, $\hat{\mathbf{S}}$ can be found by substituting Equation 2.6 into Equation 2.5 and optimizing with respect to $\hat{\mathbf{S}}$. As a result, the rows of $\hat{\mathbf{S}}$ are the first d eigenvectors of $\sum_{m=1}^{M} \mathbf{X}_m^T(\mathbf{X}_m\mathbf{X}_m^T)^{-1}\mathbf{X}_m$ [10].

2.2 Probabilistic CCA

At the turn of the century, much like what Tipping and Bishop had done for PCA [7], Bach and Jordan did for CCA: reformulating it as a probabilistic model [4]. Probabilistic Canonical Correlation Analysis (PCCA), like the original CCA, considers two views of a single source. For reasons that will become evident in the next section, we will slightly modify our notation and use $\mathbf{y}^1$ and $\mathbf{y}^2$ to refer to samples from the two views (instead of using $\mathbf{x}^1$ and $\mathbf{x}^2$). The key idea is that elements of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are linearly transformed random samples from the source latent spaces: one view specific and one shared by both views.

Suppose that $\mathbf{y}^m \in \mathbb{R}^{d_m}$. PCCA is an instance of the more general model for finding dependencies given by [22],

$$\mathbf{y}^1 = \mathbf{f}(\mathbf{z}|\mathbf{W}^1) + \mathbf{g}(\mathbf{z}^1|\mathbf{B}^1) + \mathbf{u}^1 \tag{2.7}$$

$$\mathbf{y}^2 = \mathbf{f}(\mathbf{z}|\mathbf{W}^2) + \mathbf{g}(\mathbf{z}^2|\mathbf{B}^2) + \mathbf{u}^2 \tag{2.8}$$

where $\mathbf{u}^m \sim \mathcal{N}(0, \mathbf{\Psi}^m)$ is some noise. In particular, PCCA, like CCA, makes a linear assumption, leading $\mathbf{f}$ and $\mathbf{g}$ to be written as $\mathbf{f}(\mathbf{z}|\mathbf{W}^m) = \mathbf{W}^m\mathbf{z}$ and $\mathbf{g}(\mathbf{z}^m|\mathbf{B}^m) = \mathbf{B}^m\mathbf{z}^m$, with $\mathbf{W}^m, \mathbf{B}^m \in \mathcal{M}_{d_m,d}(\mathbb{R})$ and $\mathbf{\Psi}^m \in \mathcal{M}_{d_m,d_m}(\mathbb{R})$.

With all of these assumptions in mind, PCCA can be defined as the following model [22][14],

$$\mathbf{y}^m \sim \mathcal{N}(\mathbf{W}^m\mathbf{z} + \mathbf{B}^m\mathbf{z}^m,\ \mathbf{\Psi}^m) \tag{2.9}$$

$$\mathbf{z}, \mathbf{z}^m \sim \mathcal{N}(0, \mathbf{I}_d) \tag{2.10}$$

The orthonormality constraint in CCA corresponds to an independence assumption on the latent variables with unit variance in PCCA (since the covariance matrix is diagonal, there is no covariance between the latent variables) — however, the latent variables themselves are not constrained to be orthonormal [13].

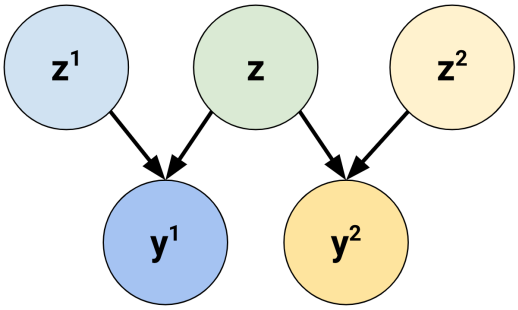


Figure 2.2: Graph representation of PCCA

As we can see in Figure 2.2, PCCA has three latent variables: $\mathbf{z}$, which is shared by both views, and $\mathbf{z}^1$ and $\mathbf{z}^2$ which are dependent on the particular view. Thus, any observed $\mathbf{y}^m$ is a linear combination of some shared and view-specific latent variables.

Fitting the model can be done by considering the problem in terms of factor analysis and using the Expectation Maximization (EM) procedure [22][12]. Indeed, PCCA can equivalently be written in the generic factory analysis form [12]

$$\mathbf{y} = \mathbf{\Lambda}\mathbf{z}' + \mathbf{u}, \quad \text{where} \quad \mathbf{u} \sim \mathcal{N}(0, \mathbf{\Psi}) \tag{2.11}$$

by taking Equations 2.7 and 2.8 with the linear choices for f and g mentioned above, and combining them using matrices defined as follows [22] [14],

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}^1 \\ \mathbf{y}^2 \end{bmatrix}, \qquad \mathbf{\Lambda} = \begin{pmatrix} \mathbf{W}^1 & \mathbf{B}^1 & 0 \\ \mathbf{W}^2 & 0 & \mathbf{B}^2 \end{pmatrix} \tag{2.12}$$

$$\mathbf{z}' = \begin{bmatrix} \mathbf{z} \\ \mathbf{z}^1 \\ \mathbf{z}^2 \end{bmatrix}, \qquad \mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}^1 & 0 \\ 0 & \mathbf{\Psi}^2 \end{pmatrix} \tag{2.13}$$

Then, the optimal values for $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ can be found using the EM

algorithm, the iterative updates of which are given by [12]

$$\mathbf{\Lambda}_{\text{new}} = \left(\sum_{i}^{N} \mathbf{y}_i \mathbb{E}[\mathbf{z}'|\mathbf{y}_i]^T \right) \left(\sum_{i}^{N} \mathbb{E}[\mathbf{z}'\mathbf{z}'^T|\mathbf{y}_i] \right)^{-1} \tag{2.14}$$

$$\mathbf{\Psi}_{\text{new}} = \frac{1}{N} \mathbf{Diag} \left\{ \sum_{i=1}^{N} \mathbf{y}_i \mathbf{y}_i^T - \mathbf{\Lambda}_{\text{new}} \mathbb{E}[\mathbf{z}'|\mathbf{y}_i] \mathbf{y}_i^T \right\} \tag{2.15}$$

There have been some nonlinear extensions of PCCA, such as Kernel PCCA (KPCCA) [27], yet they require a lot of fine-tuning and are not as popular.

2.3 Deep Probabilistic CCA

The probabilistic formulation of CCA is limited in its assumption that the input data points are vectors in $\mathbb{R}^{d_m}$. What about the case in which the data is an image, a graph, or a manifold?

Extracting meaningful information from such domains typically requires convolutional layers which agglomerate neighborhood information. As an example, consider image inputs. Convolutional Neural Networks (CNNs) learn the parameters for two-dimensional filters which are convolved with the image. This patch-based mechanism is necessary in order to learn a translation invariant representation for the data (i.e. the positioning of the object in the image is irrelevant). This suggests that flattening the images and feeding them into PCCA will be relatively useless.

A solution to this problem was recently proposed by Gundersen et al. [14]. In their paper, they introduce Deep Probabilistic Canonical Correlation Analysis (DPCCA), which sandwiches PCCA in between encoder/decoder neural networks that are trained to extract useful information from complex objects (such as images).

From a bird's eye point of view, the process is as follows (as can

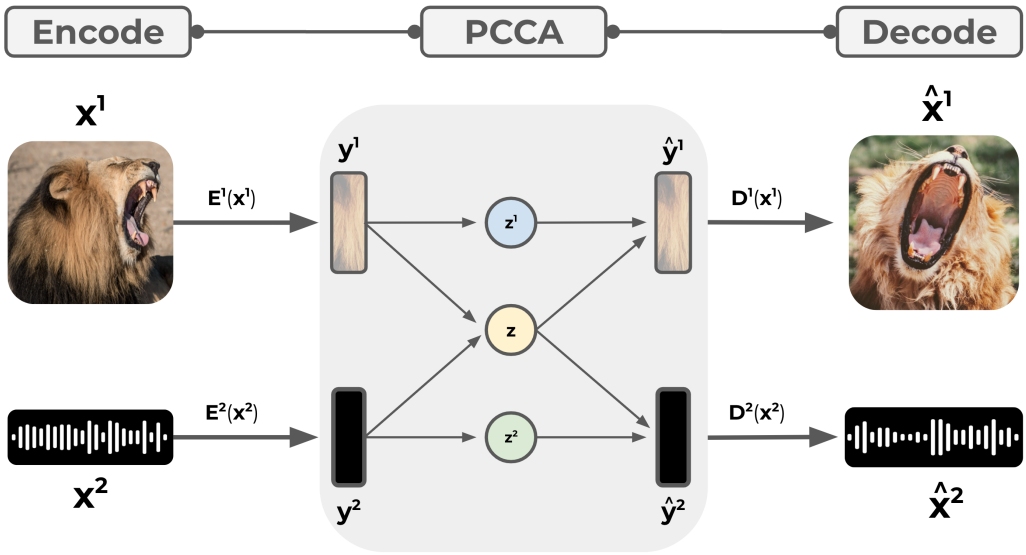


Figure 2.3: Overview of the DPCCA architecture

be seen in Figure 2.3). For each view, DPCCA learns a mapping from the input object to some vector embedding (one for each of the modalities). Using these two embeddings, it then approximately fits PCCA on the batch by iteratively updating the parameters using the EM algorithm. Then, using PCCA, it samples the reconstructed embeddings, and learns a mapping from each of them to the corresponding initial objects.

More formally, like in the previous section, let us fix some latent dimension $d \in \mathbb{R}$ and embedding dimensions d_m, where $m \in \{1, 2\}$. Suppose that we have data points $(\mathbf{x}_i^1, \mathbf{x}_i^2)$ from two different views. Then, the inputs for PCCA are given by

$$\mathbf{y}_i^1 = \mathbf{E}_1(\mathbf{x}_i^1), \quad \mathbf{y}_i^2 = \mathbf{E}_2(\mathbf{x}_i^2) \tag{2.16}$$

where $\mathbf{E}_m$ is the encoder for view $m \in \{1, 2\}$ — for image inputs, this could be CNNs, for graph inputs, GNNs, and so on. The dimension of the embedding for view m, denoted d_m, is arbitrary.

Then, with the embeddings $(\mathbf{y}_i^1, \mathbf{y}_i^2)$, we apply a fixed number of iter-

ations of the EM algorithm, given by Equations 2.14 and 2.15. Once the PCCA model has been fit, we can simply sample $\hat{\mathbf{y}}_i^1$ and $\hat{\mathbf{y}}_i^2$, reconstructed embeddings, using the general factor analysis form in Equation 2.11. Sampling from the two reconstructions then merely comes down to applying Equation 2.11 with the updated parameters $\mathbf{\Lambda}_{\text{new}}$ and $\mathbf{\Psi}_{\text{new}}$.

However, the operation "sample $\mathbf{u}$ from $\mathcal{N}(0, \mathbf{\Psi}_{\text{new}})$" is not a differentiable operation, which prevents the use of gradient descent. For the purpose of training an end-to-end model, this is an imperative necessity. Luckily, a common way of circumenting this problem is the so-called "reparametrization trick", which is common to most variational autoencoder architectures [21][14].

The trick itself relies on the affine transformation property of multivariate Gaussian distributions. The idea is to look for some matrix $\mathbf{L}$ which allows us to factor out $\mathbf{\Psi}_{\text{new}}$ from the distribution of $\mathbf{u}$,

$$\mathbf{u} \sim \mathcal{N}(0, \mathbf{\Psi}_{\text{new}}) \iff \mathbf{u} \sim \mathbf{L}\, \mathcal{N}(0, \mathbf{I}_{d_1+d_2}) \tag{2.17}$$

Solving this is quite simple; we distribute $\mathbf{L}$ and obtain the necessary condition,

$$\mathbf{u} \sim \mathcal{N}(0, \mathbf{\Psi}_{\text{new}}) \iff \mathbf{u} \sim \mathcal{N}(0, \mathbf{L}\mathbf{L}^T) \tag{2.18}$$

which implies that $\mathbf{\Psi}_{\text{new}} = \mathbf{L}\mathbf{L}^T$ — this coincides with finding the Cholesky decomposition of $\mathbf{\Psi}_{\text{new}}$. The only condition for doing so is that $\mathbf{\Psi}_{\text{new}}$ is positive definite. If we recall the EM parameters update for $\mathbf{\Psi}$ in Equation 2.15, we see that $\mathbf{\Psi}_{\text{new}}$ is a diagonal matrix. This implies that the Cholesky decomposition is trivial and is given by the diagonal matrix with elements given by

$$\mathbf{L}_{k,k} = \sqrt{\mathbf{\Psi}_{\text{new}k,k}}, \quad k \in \{1, \cdots, d_1 + d_2\} \tag{2.19}$$

We can now consider $\epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_1+d_2})$ (which does not depend on any

parameters) and apply the transformation,

$$\hat{\mathbf{y}} = \mathbf{\Lambda}\mathbf{z}' + \mathbf{L}\epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_1+d_2}) \tag{2.20}$$

in order to obtain the two sampled reconstructions: $\hat{\mathbf{y}} = [\hat{\mathbf{y}}^1, \hat{\mathbf{y}}^2]$.

Once the reconstructions have been obtained, the next step is to reconstruct the inputs. Given $\hat{\mathbf{y}}^m$, a decoder neural network $\mathbf{D}_m$ is tasked with reconstructing $\mathbf{x}^m$,

$$\hat{\mathbf{x}}^m = \mathbf{D}_m(\hat{\mathbf{y}}^m) \tag{2.21}$$

Training DPCCA consists in minimizing the reconstruction error for both modalities via stochastic gradient descent, i.e. how close each $\hat{\mathbf{x}}^m$ is to $\mathbf{x}^m$,

$$\mathcal{L}_\Theta = \sum_{i=1}^{N} \left\{ ||\hat{\mathbf{x}}_i{}^1 - \mathbf{x}_i^1||_2^2 + ||\hat{\mathbf{x}}_i{}^2 - \mathbf{x}_i^2||_2^2 \right\} \tag{2.22}$$

where Θ, the parameters with respect to which we compute the gradient, is the combination of the parameters for the encoder/decoder neural networks and the PCCA parameters ($\mathbf{\Lambda}$ and $\mathbf{\Psi}$).

2.3.1 Generating one modality from the other

Suppose now that the DPCCA model has been succesfully fit on some training data — the encoders and decoders are trained and we have the optimal PCCA parameters $\mathbf{\Lambda}^\star$ and $\mathbf{\Psi}^\star$. Where this model shines is in its ability to sample one modality when we know the other. Going back to our example in Figure 2.3, if we have images of animals, how do we go about generating corresponding sounds?

Suppose, without loss of generality, that we are given $\mathbf{x}^1$ and want to generate a reasonable sample from the second view.

First, we need to obtain embeddings to be used as inputs into PCCA, which can be done using the encoder for the first view and assuming

no prior knowledge for the second (by letting it contain zeros),

$$\mathbf{y}^1 = \mathbf{E}_1(\mathbf{x}^1) \tag{2.23}$$

$$\mathbf{y}^2 = \mathbf{0}_{\mathbb{R}^{d_2}} \tag{2.24}$$

Then, since PCCA assumes that the inputs are zero-centered, $\mathbf{y}^1$ is shifted by its mean so that it is centered at 0.

Then, using the factor analysis form introduced in the previous section, $\mathbf{y}^1$ and $\mathbf{y}^2$ are concatenated into $\mathbf{y} = [\mathbf{y}^1, \mathbf{y}^2]$.

Now, we simply need to sample $\hat{\mathbf{y}}^1$ and $\hat{\mathbf{y}}^2$ from the fitted PCCA model. This can be done by first computing $\mathbb{E}[\mathbf{z}'|\mathbf{y}]$, the expected value of $\mathbf{z}'$ knowing $\mathbf{y}$,

$$\mathbb{E}[\mathbf{z}'|\mathbf{y}] = \mathbf{\Lambda}^T(\mathbf{\Psi} + \mathbf{\Lambda}\mathbf{\Lambda}^T)^{-1}\mathbf{y} \tag{2.25}$$

which is derived using the joint normality of $\mathbf{y}$ and $\mathbf{z}'$ [12].

In order to sample $\hat{\mathbf{y}}$, we can then simply use Equation 2.11, replacing $\mathbf{z}'$ with the expected $\mathbf{z}'$, knowing $\mathbf{y}$. Hence,

$$\hat{\mathbf{y}} = \mathbf{\Lambda}^\star\mathbb{E}[\mathbf{z}'|\mathbf{y}] + \mathbf{u} \tag{2.26}$$

where $\mathbf{u}$ is sampled from $\mathcal{N}(0, \mathbf{\Psi}^\star)$. Now that we have the reconstruction, we can separate it into $[\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2] = \hat{\mathbf{y}}$. Using the decoder for the second view, we obtain the final reconstructed view,

$$\hat{\mathbf{x}}^2 = \mathbf{E}_2(\hat{\mathbf{y}}_2) \tag{2.27}$$

The same process can be used to generate the first view knowing the second. If, instead, we wish to generate two views at random, we simply set $\mathbf{y} = 0_{\mathbb{R}^{d_1+d_2}}$ and apply the same process.

2.4 Exploiting Source Information in CCA

The methods that we discussed in the previous sections were limited to the case in which we have multiple views of a given source. In the algorithms that we considered (CCA, PCCA, and DPCCA), the data points were treated independently of each other. The objective was to find a latent representation which captured how the two views were linked. However, what if, in addition, the sources themselves were correlated? Could this additional information enable us to learn a more meaningful latent space? This setting is much more realistic. In the real world, we rarely come across completely independent samples; objects are usually related in some way. For example, if we were trying to match animals with their sounds, we naturally would expect species in the same genus to be closer to each other in the latent space (a lion should be closer to a jaguar than to a hawk).

This relationship could be learned directly by the algorithms, despite their independence assumptions. In the case of lions, it is not hard for the algorithms to figure out: "lions and jaguars look and sound almost the same, therefore the sources are similar." In other cases, however, it might not be as obvious. Therefore, adding another layer of knowledge might be extremely beneficial.

The key challenge is to thus harness the additional global information about how the individual objects relate to each other in addition to how they relate to their respective views.

2.4.1 Graph Regularized CCA

Now that we have established that the goal is to leverage some additional information about the sources in order to mold the latent representation in way that more adequately represents this source-source correlation, we need a way to represent this relationship.

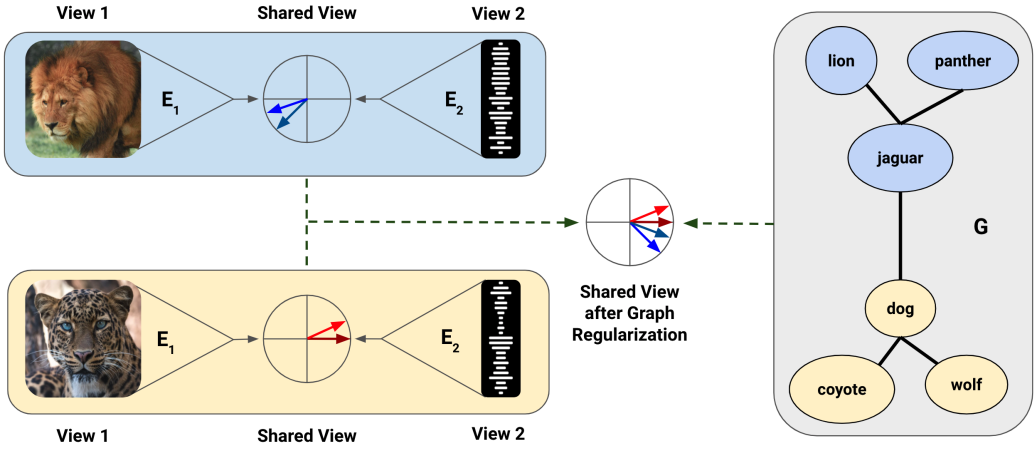


Figure 2.4: Overview of the graph regularization for CCA. In this example, sources are animals, of which we have two views: images and sounds. $\mathbf{E}_1$ and $\mathbf{E}_2$ map the two views to some shared space where they are correlated (like-colored arrows denote projections of two views of the same source in the shared space). The graph on the right gives some prior information on the relationship between the sources: lions and jaguars. Since the two are related in the graph, the projections of the two sources will be pushed closer together in the shared space.

Graphs are the perfect candidates for this as they can represent the correlation between two sources using weighted edges between nodes.

Formally, we define a graph $\mathcal{G}$ as a tuple $(\mathcal{S}, \mathcal{W})$ in which $\mathcal{S} = \{1, \cdots, N\}$ is a set of source nodes and $\mathcal{W} = \{w_{i,j}\}_{(i,j)\in\mathcal{S}^2}$ is a set of edge weights, connections between source vertices.

One of the key assumptions that we make is that the sources are smooth over the graph. More specifically, the intensity of the connection $w_{i,j}$ between source nodes i and j should be proportional to their Euclidean distance. Essentially, we want the weights to bring in some additional information about how closely related (or not) two source nodes might be.

Graph Regularized CCA (gCCA) [11] leverages this additional graph

information in order to fit a more representative version of CCA. To do so, it includes this source-source knowledge via a graph regularization term,

$$\textbf{gCCA regularizer} := \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}||\mathbf{E}_1(\mathbf{x}_i^1) - \mathbf{E}_2(\mathbf{x}_j^2)||^2 \tag{2.28}$$

where $\mathbf{E}_m$, the encoder for view m, linearly projects points from view m to the source space approximation. The full CCA minimization objective is then given by its original formulation with the addition of this regularizer scaled by some hyperparameter γ that determines how much importance we place on the source-source correlation determined by the graph.

The way that the regularizer works is quite simple. If sources i and j are very similar, then $w_{i,j}$ will be quite high and, therefore, both sources will be pushed closer together in the shared space — i.e. the projection of the first view of source i will be made closer to the projection of the second view of source j.

Chapter 3

Deep Models for Correlated Sources

As we discussed in the background section, there are multiple models that are able to find a latent representation that captures cross-view correlation. We saw that CCA could be formulated in a probabilistic manner via PCCA which, itself, could be included as part of a more powerful model that leverages the non-linear modeling power neural networks (DPCCA). In addition, we explored how additional knowledge about source to source correlation could be used to inform the latent representation inferred in CCA. In particular, we saw that this information could be provided through the use of a graph regularization term in CCA's loss function, yielding the gCCA algorithm.

A natural extension to think about is one which combines the strength of gCCA with that of DPCCA. The key motivation for such a combination is that it would enable DPCCA to wield additional information about how the sources, themselves, are correlated. This would be the best of both worlds — complicated objects, such as images or graphs, could be embedded and correlated in some lower-dimensional space, in a way that takes into account the object-

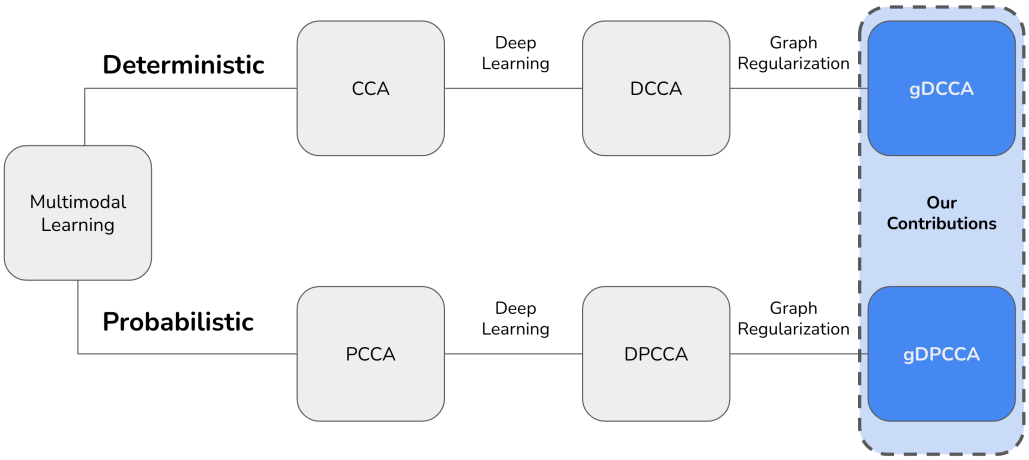


Figure 3.1: Overview of the different methods introduced thus far and our proposed algorithms.

object relationship.

To this end, we introduce two new algorithms: Graph Regularized Deep Canonical Correlation Analysis (gDCCA) and Graph Regularized Deep Probabilistic Canonical Correlation Analysis (gDPCCA). The former is an extension of DCCA, while the latter extends DPCCA.

As we will see in this chapter, the key difference between the two models is that the latter, a generative model, enables sampling in the latent space which, crucially, allows us to switch from one modality to the other (like in DPCCA).

3.1 Graph Regularized Deep CCA

Our first novel algorithm, Deep Graph Regularized Canonical Correlation Analysis (gDCCA), can be seen as a graph regularized extension of DCCA. Like in gCCA, the objective is to learn a latent representation for both views that includes source-source correlation in the form of a graph regularizer.

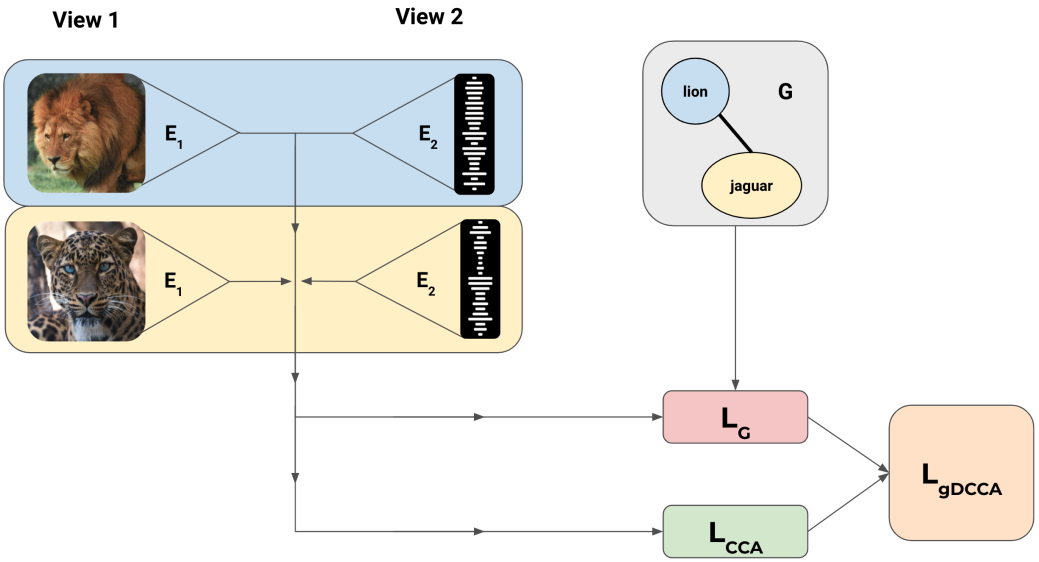


Figure 3.2: Visualization of how the loss for gDCCA is computed at every iteration (for every group of elements for which we know the source to source correlation).

As before, we suppose that we have access to data coming from two views. However, unlike in gCCA, we do not register all of the data at once — instead, we pass in sets of paired views, each of which contains some source-source correlation information.

Therefore, at each training step t, we consider a set of source nodes $\mathcal{S}_t = \{1, \cdots, N\}$, for which we have multiple paired views $(\mathbf{x}_i^1, \mathbf{x}_i^2)^t$.

Like in gCCA, we have an encoder for each view $\mathbf{E}_m$, which is made non-linear using multi-layer perceptrons. Given the incremental nature of such a model, we cannot use the exact same objective funtion as gCCA (since we cannot analytically solve for the minimum, we need to use gradient descent). The encoders for each view both directly map to $\mathbb{R}^d$, the shared latent space. The goal is to maximize the correlation between the embeddings of the two views, as in CCA. We thus go back to the "roots" and directly compute the correlation using the cosine similarity (see Equation 2.3, like in DCCA. Maximizing the correlation thus implies having to minimize

the following loss function,

$$\mathcal{L}_{CCA} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\langle \mathbf{E}_m(\mathbf{x}_i^1), \mathbf{E}_m(\mathbf{x}_i^2)\rangle}{||\mathbf{E}_m(\mathbf{x}_i^1)||||\mathbf{E}_m(\mathbf{x}_i^2)||} \tag{3.1}$$

Geometrically speaking, this consists of pushing $\mathbf{E}_m(\mathbf{x}_i^1)$ and $\mathbf{E}_m(\mathbf{x}_i^2)$ so that they are pointed in the same direction.

In addition to maximizing the correlation, as the name would imply, we also include knowledge of source to source correlation by imposing some graph regularization like in Equation 2.28. Therefore, as in Chapter 2.4, we also accept a graph $\mathcal{G}_t = (\mathcal{S}_t, \mathcal{W}_t)$, where $\mathcal{W}_t = \{w_{i,j}^t\}_{(i,j)\in\mathcal{S}^2}$ is a set of edge weights for the source nodes $\mathcal{S}_t$ obeying the rule:

$$w_{i,j}^t > w_{i,k}^t \implies \text{source i is more similar to j than k} \tag{3.2}$$

The term itself, dependent on the sources present at time t, is defined as

$$\mathcal{L}_{\mathcal{G}}^{gDCCA} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}^t||\mathbf{E}_1(\mathbf{x}_i^1) - \mathbf{E}_2(\mathbf{x}_j^2)||^2 \tag{3.3}$$

Then, for a whole training batch, the total graph regularization term, $\mathcal{L}_{\mathcal{G}}^{gDCCA}$, is the sum of every $\mathcal{L}_{\mathcal{G}_t}^{gDCCA}$ in the batch. The complete loss for gDCCA is thus the combination of the CCA loss and the graph regularization, weighted by some constant $\gamma \in \mathbb{R}^+$,

$$\mathcal{L}_{gDCCA} = \mathcal{L}_{CCA} + \gamma\mathcal{L}_{\mathcal{G}}^{gDCCA} \tag{3.4}$$

3.2 Graph Regularized Deep PCCA

Our second novel algorithm, and arguably the most applicable of the two, is Graph Regularized Deep Probabilistic Canonical Correlation Analysis (gDPCCA). At its core, gDPCCA is an extension of DPCCA which allows for additional source-source information

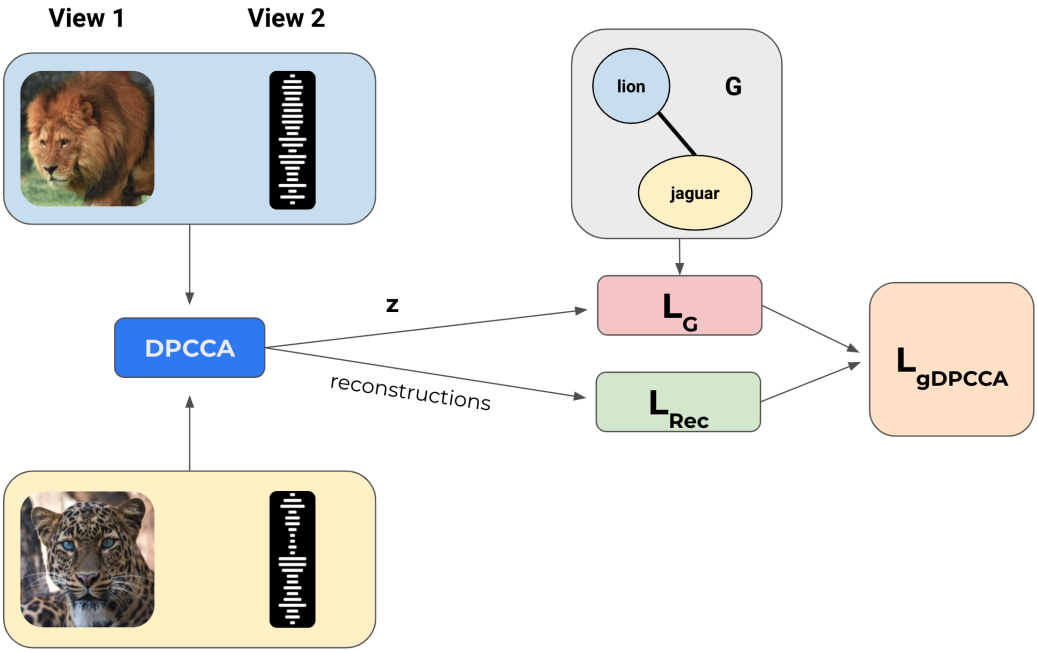


Figure 3.3: Visualization of how the loss for gDPCCA is computed at every iteration (for every group of elements for which we know the source to source correlation).

in the form of a graph regularizer. Importantly, unlike gDCCA, gDPCCA is a probabilistic generative model, making it possible to sample from the latent space and, thus, to switch between views.

As before, we suppose that we have access to data coming from two views. However, unlike in DPCCA, we do not directly feed in a set of pairs $(\mathbf{x}_i^1, \mathbf{x}_i^2)$. Instead, globally speaking, we want to pass in sets of paired views, each of which contains some source-source correlation information.

Like in gDCCA, at each training step t, we are given some sources S_t, paired views $(\mathbf{x}_i^1, \mathbf{x}_i^2)^t$, and some graph $G_t = (\mathcal{S}_t, \mathcal{W}_t)$ describing source to source correlation.

At every time step, we encode the pairs $(\mathbf{x}_i^1, \mathbf{x}_i^2)^t$ as $(\mathbf{y}_i^1, \mathbf{y}_i^2)^t$, fit PCCA, and sample their reconstructions $(\hat{\mathbf{y}}_i^1, \hat{\mathbf{y}}_i^2)^t$, after which we can reconstruct the inputs (exactly like in DPCCA, see Chapter 2.3).

Like in Equation 2.28, we wish to move the projections of the sources in the shared latent representations closer to each other depending on the graph weights. DPCCA, we recall, is a probabilistic model which has three latent variables that are learned over time: two view-specific latent variables, $\mathbf{z}^1$ and $\mathbf{z}^2$, and one shared variable that captures cross-modality information, $\mathbf{z}$. We wish to target the latter. Indeed, $\mathbf{z}$ represents the source space for which we possess additional graph-based information.

We can define the gDPCCA graph regularization term at time t as follows,

$$\mathcal{L}_{\mathcal{G}_t} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j}^t ||\mathbf{z}_i^t - \mathbf{z}_j^t||^2 \tag{3.5}$$

where $\mathcal{L}_{\mathcal{G}_t}$ denotes the graph regularizer. Like in gDCCA, the total graph regularization term, $\mathcal{L}_{\mathcal{G}}$, is the sum of every $\mathcal{L}_{\mathcal{G}_t}$ in a given batch. The complete loss for gDPCCA is thus the combination of the DPCCA loss and the graph regularization, weighted by some constant $\gamma \in \mathbb{R}^+$,

$$\mathcal{L}_{gDPCCA} = \mathcal{L}_{DPCCA} + \gamma \mathcal{L}_{\mathcal{G}} \tag{3.6}$$

As we can see, when γ is set to 0, gDPCCA reduces to DPCCA. The larger it is, the more we privilege the additional source-source correlation to inform the latent representation.

Having this additional information in the form of regularization thus makes it both simple to modify and interpretable. As we will see in the experiments, the optimal choice for γ will greatly depend on the problem at hand.

3.3 Spatially Correlated Sources

The graph regularizer which we introduced for gDCCA and gDPCCA is quite broadly defined. What, exactly, then, is this source from which we can extract additional information during training? Al-

though there are various ways of including source interaction, one of particular interest is spatial correlation between features in small patches of an image. The graph in this case would just be some distance based measure of similarity between the patches. Such a setting has numerous applications, notably in biology, where multimodal image data (e.g. histology) are quite common.

We will first discuss how to extend Graph Regularized PCCA to exploit spatial information by explicitly defining the graph.

3.3.1 Space Informed gDPCCA

Space informed gDPCCA is, first and foremost, a special case of gDPCCA. Hence, once again we have a dataset made up of two views of some source along with knowledge about source to source correlation. However, we now make the assumption that one of the views is an image (arbitrarily, we suppose it is view 1).

The pillar upon which gDPCCA is based is the correlation graph $\mathcal{G}$. In practice, since it is quite difficult to know about source to source correlation, we will infer it using correlation between the objects for view 1. A very common setting, especially in biology, is one in which we can obtain image patches from a larger image, for which we can measure some physical quantities.

We recall that $\mathbf{G}$, the adjacency matrix of $\mathcal{G}$, contains the correlation between sources. In terms of some positive semi-definite kernel $\mathcal{K}$, this can be written as,

$$\mathbf{G} = \begin{pmatrix} \mathcal{K}(s_1, s_1) & \cdots & \mathcal{K}(s_1, s_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(s_N, s_1) & \cdots & \mathcal{K}(s_N, s_N) \end{pmatrix} \tag{3.7}$$

As we can see in Figure 3.4, we can construct $\mathbf{G}$ by letting $\mathcal{K}$ depend on the Euclidean distance between the patches. In other words, if

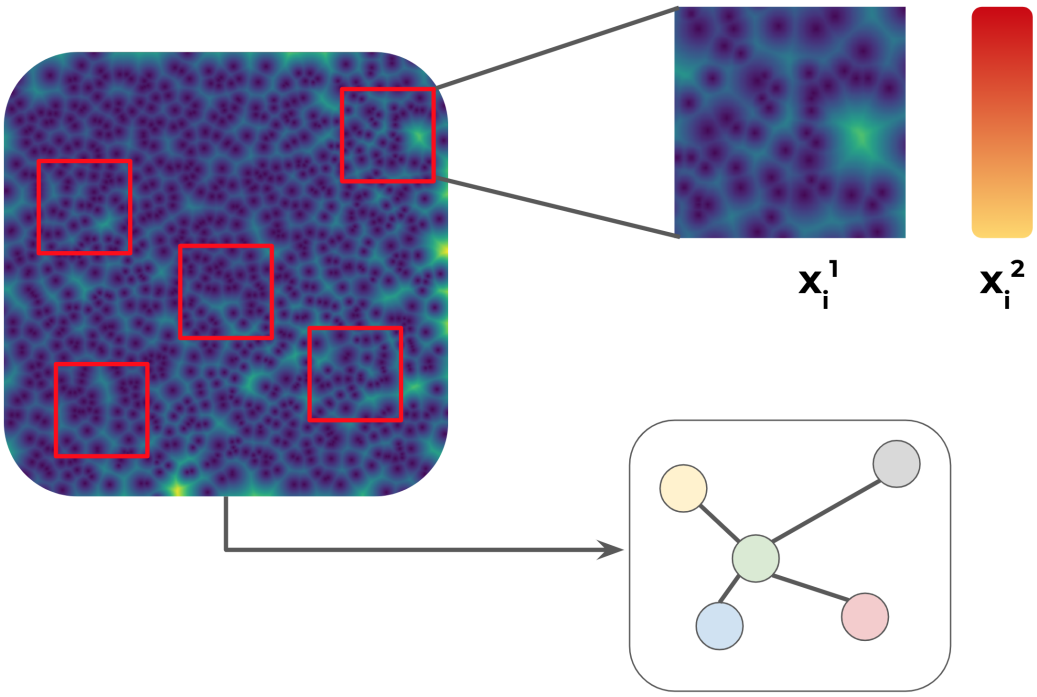


Figure 3.4: Visualization of how spatially correlated patches in an image can be used with gDPCCA. Each of the small patches has an "image view" and some other view (e.g. a physical quantity). The patches in the image are related through their relative distances. If we assume that the source objects are related in a way that is captured by the spatial correlation for this view, then we can use some kernel function based on the distance between patches to build the correlation graph $\mathcal{G}$.

we let $\mathbf{r}_i \in \mathbb{R}^2$ denote the position of the center of the patch, then

$$\mathcal{K}(\mathbf{s}_i, \mathbf{s}_j) = \mathcal{H}\Big(||\mathbf{r}_i - \mathbf{r}_j||^2\Big) \tag{3.8}$$

The choice of $\mathcal{H}$ will vary based on the problem at hand. In fact, it is entirely dependent on what we believe the underlying correlation is and how fast we think it changes through space. If we assume some underlying Gaussian mechanism, then an radial basis func-

tion (RBF) kernel could be used,

$$\mathcal{H}(d) = \exp\left(-\frac{d}{\sigma^2}\right) \tag{3.9}$$

where $\sigma \in \mathbb{R}$ is some parameter that determines how fast the function decreases.

Chapter 4

Experiments

In this chapter, we will look into multiple experiments in order to both benchmark the efficacy of gDPCCA and to see more precisely in which cases it might be of use. To this end, we will begin by evaluating our method on a synthetic dataset constructed from a source-source graph. Then, we will explore how binary information can improve performance by constructing a bi-modal derivative of the well known MNIST dataset [24]. We will then see a limitation of gDPCCA by considering a set of spatially correlated colored MNIST digits. Finally, we will also evaluate the performance of gDCCA.

4.1 Pseudo Gene Expression

This first dataset has a data generation mechanism that explicitly depends on source to source correlation, making it ideal to test whether gDPCCA is able to leverage additional graph knowledge when compared to DPCCA. It is analagous to how gene expression is correlated in cells [29].

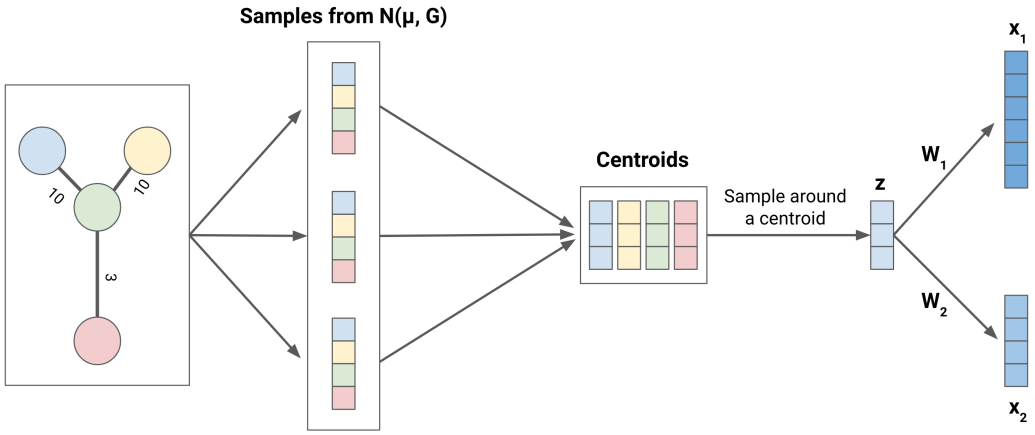


Figure 4.1: Procedure for sampling multi-view synthetic data points coming from a shared source graph. Using the source graph's adjacency matrix $\mathbf{G}$ as covariance, we sample d points. This leads to multiple d-dimensional centroids. Choosing a centroid uniformly and sampling around it leads to a latent variable $\mathbf{z}$, from which we can generate two views $\mathbf{x}_m \in \mathbb{R}^{d_m}$ using some $d_m \times d$ matrix W_m.

4.1.1 Data Generation

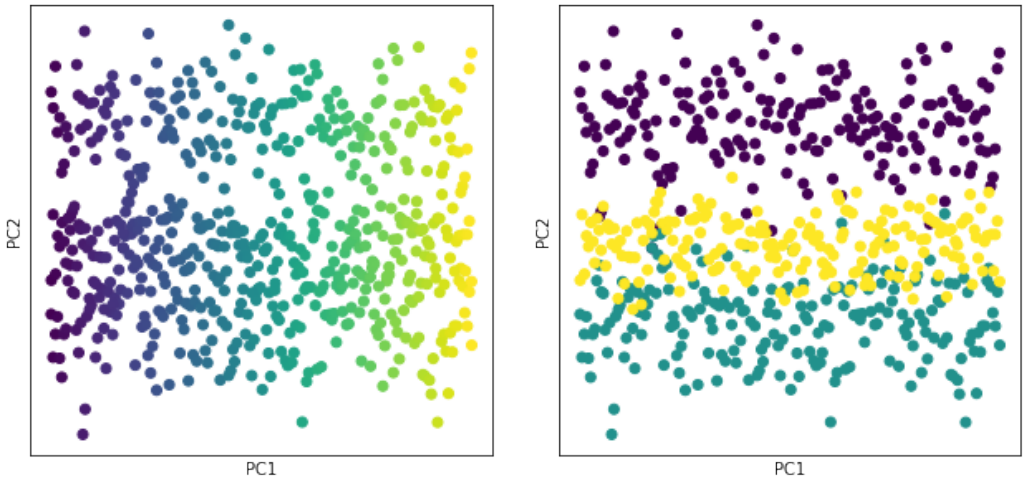
The mechanism to generate the data is quite simple. Suppose that we have a graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ modeling source interaction between sources $\mathcal{S} = \{1, \cdots, N\}$. As before, $\mathcal{E} = \{w_{i,j}\}_{(i,j)\in\mathcal{S}^2}$.

We can build the adjacency matrix for $\mathcal{G}$ as $\mathbf{G}_{i,j} = w_{i,j}$. The idea is to use $\mathbf{G}$ as a covariance matrix from which we sample d sets of correlated points,

$$C_i \sim \mathcal{N}(\mu, \mathbf{G}), \text{for } i \in \{1, \cdots, d\} \tag{4.1}$$

where each C_i is a set of values which are correlated according to the specified source graph and $\mu \in \mathbb{R}^N$ determines where those values are initially positioned in space.

Agglomerating the C_i in a matrix, $\mathbf{C} = [C_1, \cdots, C_d]$, we now have a $N \times d$ matrix whose rows correspond to d-dimensional centroids for the class represented by source node i. Visually speaking, creating



(a) Same color → generated from the same centroid

(b) Same color → same sample from the graph

Figure 4.2: Projection of the latent space on the first two PCA components of data synthetically generated from a graph with 200 distinct sources, from which we drew 3 samples. As we can see in (a), most of the variance in the latent space is determined by the centroids, making separation between the sources in $\mathcal{G}$ feasible. As is the case in (b), points can also be clustered according to which sample they were collectively drawn from.

the centroids is akin to sampling multiple versions of $\mathcal{G}$ and stacking them on top of each other, yielding a multi-dimensional set of points correlated according to the values contained in $\mathcal{G}$.

Using those d-dimensional centroids, we can sample points — the latent variables — around each of them, which we can then transform to obtain two views of each sampled centroid. For centroid $\mathbf{C}_i$, we sample $\mathbf{z}_i$ and obtain $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$ as follows

$$\mathbf{x}_i^m = \mathbf{W}^m \mathbf{z}_i + \epsilon_m, \quad m \in \{1, 2\} \tag{4.2}$$

where $\epsilon_m \sim \mathcal{N}(0, \mathbf{I}_{d_m})$ is some Gaussian noise. From the construction of the two views $\mathbf{x}_i^m$, it follows that their shared source $\mathbf{z}_i$ is going to be correlated with some other $\mathbf{z}_j$ (obtained from C_j) according to

$\mathcal{G}$.

Gradually generating a dataset compatible with gDPCCA involves repeating the above process as many times as needed, indexing each of the elements with some time step t.

In Figure 4.2, we can see projections of the latent space on the first two principal components, the directions which maximize the variance in the data. Figure 4.2.a and Figure 4.2.b are the same plots, except with a different labeling of the points.

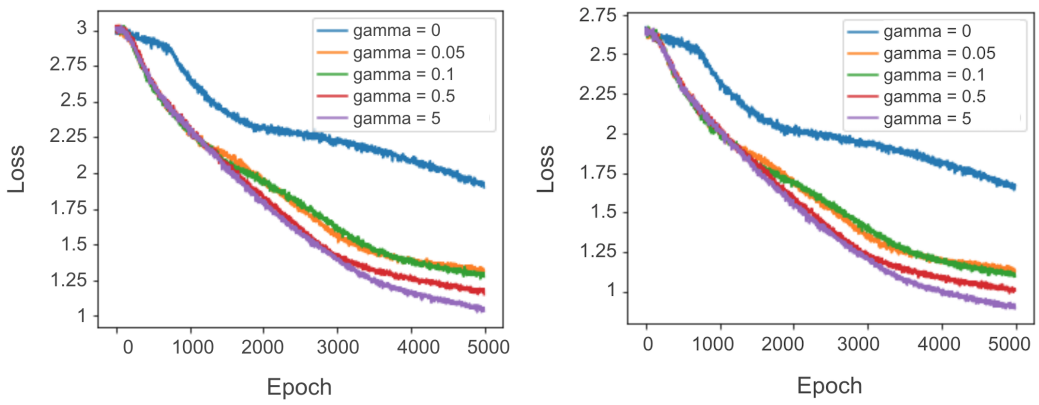
Points in Figure 4.2.a are the same color provided they come from the same centroid (i.e. they represent the same source in $\mathcal{G}$). The fact that such a clustering is possible in the latent space implies that it is feasible for algorithms to distinguish between sources (in this case, there are 200 correlated sources).

Meanwhile, points in Figure 4.2.b share the same color if they were generated from the same C, i.e. if they were all sampled at the same time. As we can see, we sampled three different C, yielding three different sets of 200 points (one for each source). Each cluster is distributed in the same way, according to $\mathcal{G}$.

4.1.2 Methodology

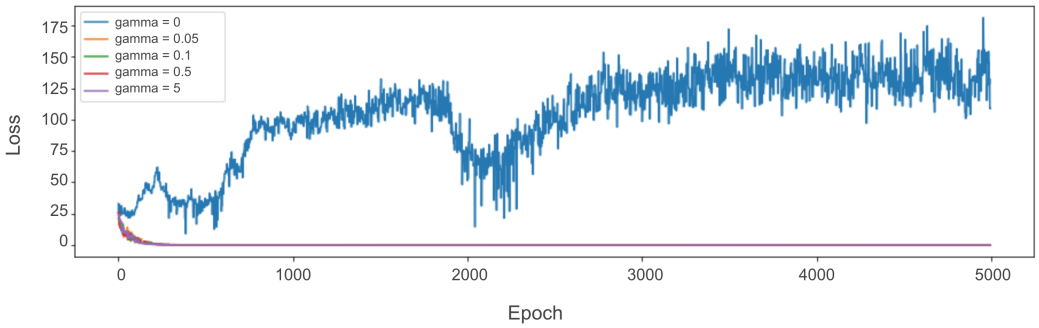
Given this data generation mechanism, the question is: does gDPCCA improve performance over generic DPCCA. To investigate, we trained multiple gDPCCA models, varying the γ parameter to see which amount of graph regularization (if any) is helpful in finding a latent representation useful for the task of reconstructing the two views.

We already know what the latent representation should look like (see Figure 4.2). Given that the mechanism used to generate it directly relies on source to source interaction, it should be no surprise that gDPCCA offers an advantage.



(a) View 1 Loss

(b) View 2 Loss



(c) Graph Regularization Loss

Figure 4.3: Loss averaged over 5 runs on synthetic data when varying the γ parameter for gDPCCA. When γ is 0, the model reduces to DPCCA.

To evaluate our claim, we initialized and trained four otherwise identical models with difference γ parameters. Specifically, we set $\gamma \in \{0, 0.05, 0.1, 0.5, 5\}$. We trained each model 4 times, the results of which we averaged in Figure 4.3. Each model was initialized with the same weights for each run. The weights, however, were modified at every run in order to obtain a fair comparison.

4.1.3 Results

As we can see in Figure 4.3, there does appear to be some differences in convergence for the view-reconstruction losses when γ is modified. In this specific case, the fastest convergence occurs when

γ	**View 1 Loss**	**View 2 Loss**	**Graph Regularization**
0	1.92	1.68	161
0.05	1.34	1.15	0.0048
0.1	1.30	1.11	0.0028
0.5	1.18	1.01	0.0027
5	1.05	0.90	0.0000553

Table 4.1: Average loss for gDPCCA with different graph regularization levels (controlled by the hyperparameter γ).

$\gamma = 5$, followed by the other models with decreasing γ. This suggests that gDPCCA performs better as the graph regularization increases. Furthermore, gDPCCA with any $\gamma > 0$ performs better than DPCCA (i.e. gDPCCA with $\gamma = 0$).

This result is not unsurprising given the artificial mechanism which we used to generate the data. Since the points in a sample are correlated according to some graph prior, it appears logical that the inclusion of this underlying relationship would yield better performance.

However, we observe marginal returns when it comes to how much the graph regularizer affects performance. The initial increase from $\gamma = 0$ to $\gamma = 0.05$ has a significant impact on the losses, whereas the massive increase from $\gamma = 0.5$ to $\gamma = 5$ only slightly improves the loss, as we can see in Table 4.1. Even more interesting is the fact that the graph regularization loss significantly decreases in that interval. This suggests that the graph regularization is only informative up to a certain point This could be due to the stochasticity of the problem — no matter how much we know, there will always be some random fluctuations which we cannot account for.

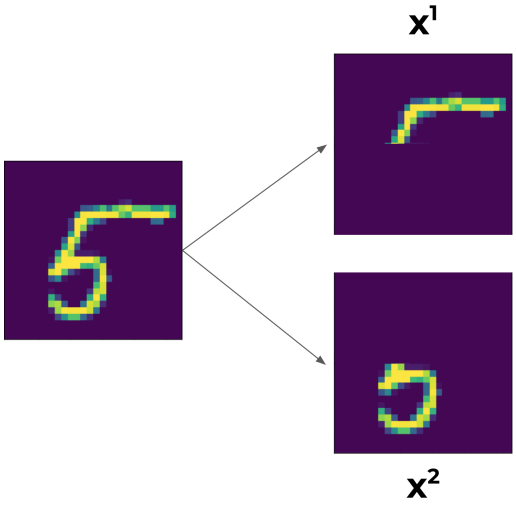


Figure 4.4: Illustration of the two different views generated from a given image in the MNIST dataset

4.2 Bimodal MNIST

In the first section, we saw that the graph regularizer in gDPCCA could be used to significantly improve convergence when the underlying sources are generated via a mechanism that artificially forces them to be correlated. A natural follow-up question to ask is: what if we intuitively know that the sources are correlated, but do not know the exact mechanism? Is gDPCCA still a feasible option when the source graph itself does not contain total information?

4.2.1 Data Generation

In order to answer this question, we can consider one of the most popular datasets in machine learning: MNIST [24]. This dataset contains $60\ 000$ labeled images $(\mathbf{x}, y)_{i=1}^{60\,000}$ of handwritten digits in various styles. Given an image $\mathbf{x}_i \in \mathcal{M}_{28,28}([0,1])$, the label $y_i \in \{0, \cdots, 9\}$ specifies which number it describes.

We can easily construct a multimodal variant of MNIST, as can be seen in Figure 4.4. If we take the image of a digit, we can horizon-

tally split it up into two parts — the top and bottom parts — which correspond to $\mathbf{x}^1$ and $\mathbf{x}^2$, respectively.

The key element which is currently missing is the graph which describes how the sources themselves — in this case, the numbers — are correlated. Even though one could try to add some "expert" knowledge, such as 1 being intuitively closer to 7 than to 3, we will keep things simple. The most natural construction for $\mathcal{G}$, is

$$\mathbf{G}_{i,j} = 1 \quad \text{if } y_i = y_j \tag{4.3}$$

where G the adjacency matrix for $\mathcal{G}$. In other words, images i and j are correlated if they describe the same image; otherwise they are not.

This formulation for $\mathcal{G}$ is clearly nowhere near exact. It does not capture the true correlation between the numbers, it merely describes a general trend in the data which we know to be true. We have incomplete information, but information nonetheless. Is it still possible to use this, albeit limited, knowledge of source to source interaction to speed up convergence?

The most important difference with the previous setting is that we do not truly know if the data is bimodal; in fact, we expect it not to be. The process which generates the two views from the source might be degenerative, i.e. there could be multiple "correct" answers that do not agree with $\mathcal{G}$. For example, the bottom of 7 and 1 is exactly the same. It is impossible in this case for the algorithm to always recover a 1 or 7 when fed the bottom view; it can merely sample views that appear consistent. In this case, we would expect some reconstructed samples to be ones and others to be sevens. It is akin to asking a natural language processing to fill in the blanks for "I like" — there are many possible, equally valid ways of completing the sentence.

4.2.2 Methodology

As we mentioned above, we do not expect this additional knowledge to be indispensable for the model to learn. We merely hope that it might push it in the right direction, which is very different from what we did in Section 4.1 where we knew the exact data generation mechanism.

This means that we cannot fully rely on the information provided by $\mathcal{G}$. As a result, we expect the regularizer to only marginally improve the loss. Since it cannot be used as a guiding force in the optimization, we voluntarily selected small values for γ.

In particular, like in the previous section, we compare DPCCA, the baseline, to gDPCCA with increasing values of γ. Each model was initialized in the same way and trained for the same duration (50 epochs). The training-testing split ratio that we used is 6 to 1. The batch size is an important parameter as it determines how much of the source to source information we can use in that specific cycle. Having bigger batches quadratically increases the size of G, increasing the chance of having meaningful graph regularization. We experimented with multiple sizes (5, 28, 128, and 256), eventually settling on 28 as we empirically found that it balanced convergence and performance).

4.2.3 Results

Effects of γ on convergence

As we can see in Figure 4.5, the convergence is noticeably faster for $\gamma = 10^{-9}$ than the baseline, when $\gamma = 0$. However, unlike in the previous synthetic example, in which any $\gamma > 0$ appeared to yield an increase in performance, the model's convergence is heavily affected by our choice of γ. For example, when $\gamma \in \{10^{-2}, 10^{2}, 10^{5}\}$, the loss barely decreases at all.

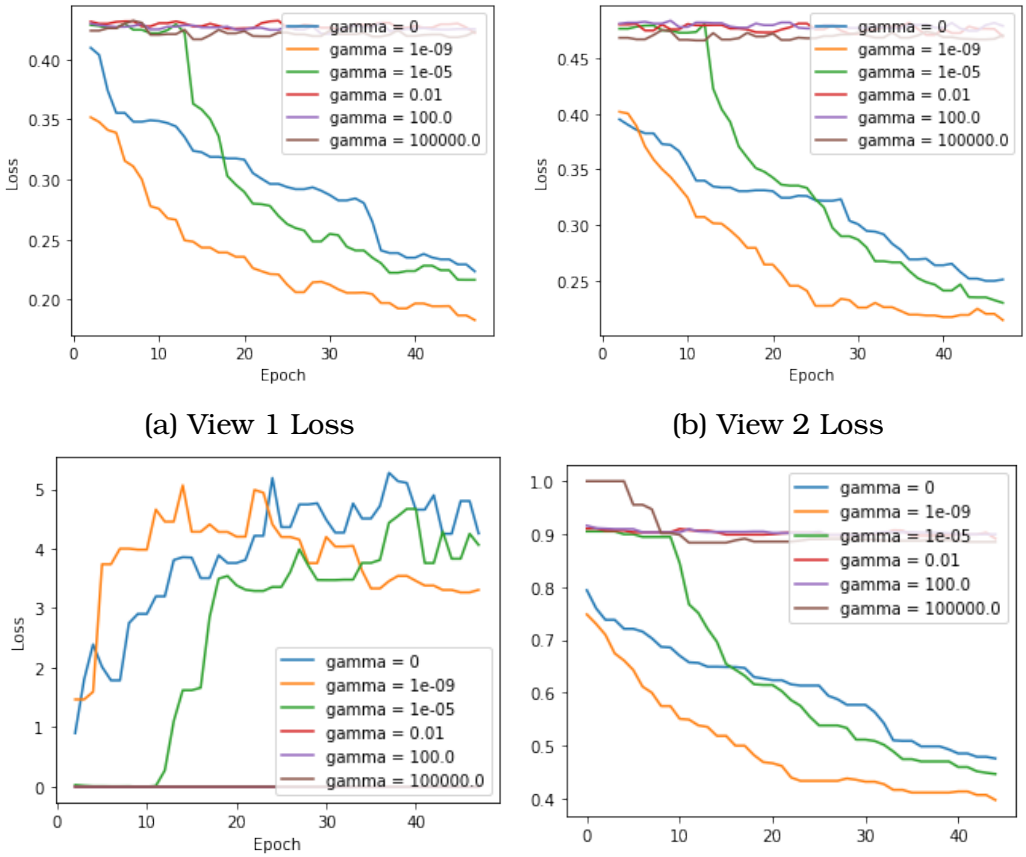
γ	**View 1 Loss**	**View 2 Loss**	**Graph Regularization**
0	0.23	0.24	4.78
10^{-9}	0.18	0.22	3.57
10^{-5}	0.22	0.23	4.09
10^{-2}	0.42	0.47	$3.32 \cdot 10^{-6}$
10^{2}	0.43	0.48	$4.96 \cdot 10^{-7}$
10^{5}	0.42	0.47	$3.42 \cdot 10^{-11}$

Table 4.2: Average loss for gDPCCA on Bimodal MNIST with different graph regularization levels (controlled by the hyperparameter γ).

This confirms our intuition that too much graph regularization when $\mathcal{G}$ does not provide perfect information can prevent the model from converging altogether.

In this case, when the emphasis is on minimizing the graph regularization term, the model tries to ensure that digits in the same class are very close to each other (in a way that is proportional to γ since the entries in G are binary). Since the batches are not very large, the number of objects sharing the same class in each batch is quite small. Consequently, the model has no concrete way of figuring out the relationship between objects since minimizing the loss for views 1 and 2 requires first increasing the graph regularization loss, as we can see in Figure 4.5.c.

Once the ball is rolling in the right direction, so to speak, the graph regularizer can be used to gently adjust the direction in order to reach the bottom of the hill faster. This is the case when $\gamma \in \{10^{-5}, 10^{-9}\}$. Adjustments are made in order to set views 1 and 2 on the right track, which happens around the 8th epoch when $\gamma = 10^{-9}$ and the 15th when $\gamma = 10^{-5}$ (likewise when $\gamma = 0$). Then, the graph regularizer takes over and decreases in a more or less stable way, especially when $\gamma = 10^{-9}$, as can be seen in Figure 4.5.c. This progressive decrease in the graph regularization loss once the



(a) View 1 Loss

(b) View 2 Loss

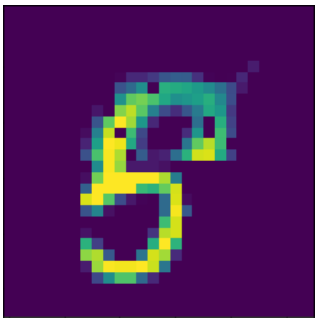
(c) Graph Regularization Loss

(d) Total loss

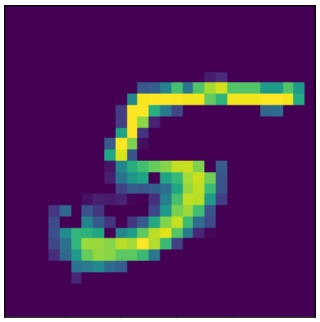
Figure 4.5: Loss averaged over 3 runs on Bimodal MNIST when varying the γ parameter for gDPCCA.

ball has been rolled coincides with a decrease in the loss for both views. This suggests, then, that the direction being followed is mutually beneficial to all parties (which was not the case for the first 10 to 20 epochs when the graph regularization was increasing).

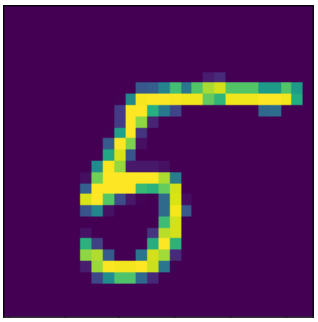
If, however, the graph regularizer dominates the loss function, the model will not even succeed in learning a nicely clustered latent space. Instead, it reduces the scale at which the objects are projected, thus leading to a decrease in the distance between points and, thus, in the graph regularization term. This "trick" that the model learns in this setting induces a worthless latent representa-



(a) View 1 knowing view 2



(b) View 2 knowing view 1



(c) Original image (two views combined)

Figure 4.6: Switching between modalities by sampling from the learned latent space. The sample and the input that was used to generate it are combined to produce the final image. As before, view 1 denotes the top half and view 2 the second half. For example, in (a), the top half of the resulting image is sampled knowing the bottom half.

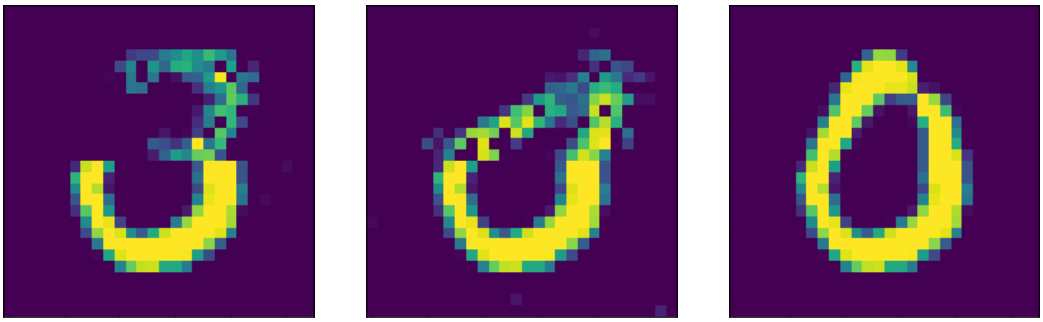
tion. For instance, when $\gamma = 10^5$, most points in the latent representation are spread out a distance on the order of 10^{-6}, which is incredibly small compared to whenever $\gamma \in \{0, 10^{-5}, 10^{-9}\}$ (an order of magnitude of 1).

In short, graph regularization coming from some $\mathcal{G}$ with incomplete (or partially correct) information can still help the model converge, but only once it has already been set on the right path. A smaller term is thus much more favorable in this setting.

What does the model actually learn?

Like we mentioned before, the model has no way of knowing which digit the view comes from in many cases — instead the distribution itself captures this ambiguity.

A good illustration of this can be seen in Figure 4.6. In this example, two views of an oddly shaped 5 are obtained and separately used to sample from the latent space using the trained model. The model attempts to fill in the blanks with the other view that it deems prob-



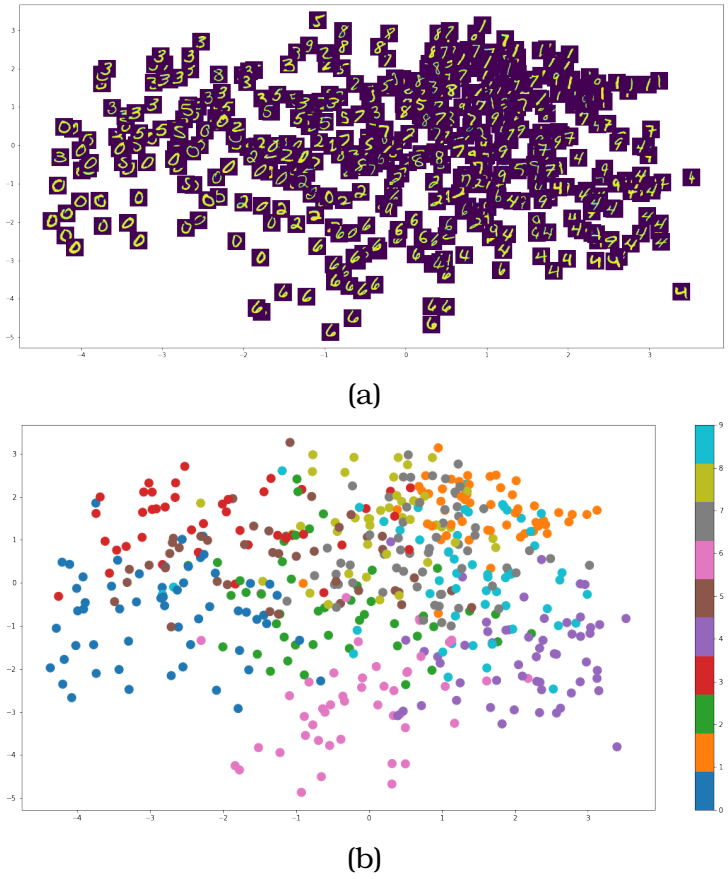
(a) Sample 1 (b) Sample 2 (c) Original image

Figure 4.7: Sampling the top view knowing the bottom view. In (a) and (b), we see two different sampled reconstructions obtained when knowing the bottom part. Each image is the combination of the original bottom view and the model's attempt at generating a matching top. The bottom's ambiguity could lead to the reconstruction of either a 0 or a 3, despite the ground truth being a 0.

able. The images in Figure 4.6 contain the two superimposed views: the true top (resp. bottom) view and the sampled reconstruction of the bottom (resp. top) view.

Interestingly, the input view in Figure 4.6.a, the bottom view, looks very similar to the bottom part of a 6 (as well as 5). As a result, the model produces a top for the 5 that could also be mistaken for the top part of a 6. If the model were personified, it would be the friend who is constantly trying to please you and, upon hearing that you enjoy fantasy books, decides to wear Gandalf's cloak with Dumbledore's hat on Halloween (the costume could depict either Gandalf or Dumbledore depending on the one you prefer). The top half of the 5, however, cannot be mistaken for anything else — it is clearly a 5 — which explains the highly accurate reconstruction of the bottom part.

A very similar phenomenon can be observed in Figure 4.7. Here, the bottom half of the zero could be mistaken for a 3 as well as a 0. In this figure, we sample multiple times using the bottom part of the zero. As we can see, Figure 4.7.a and Figure 4.7.b are two



(a)

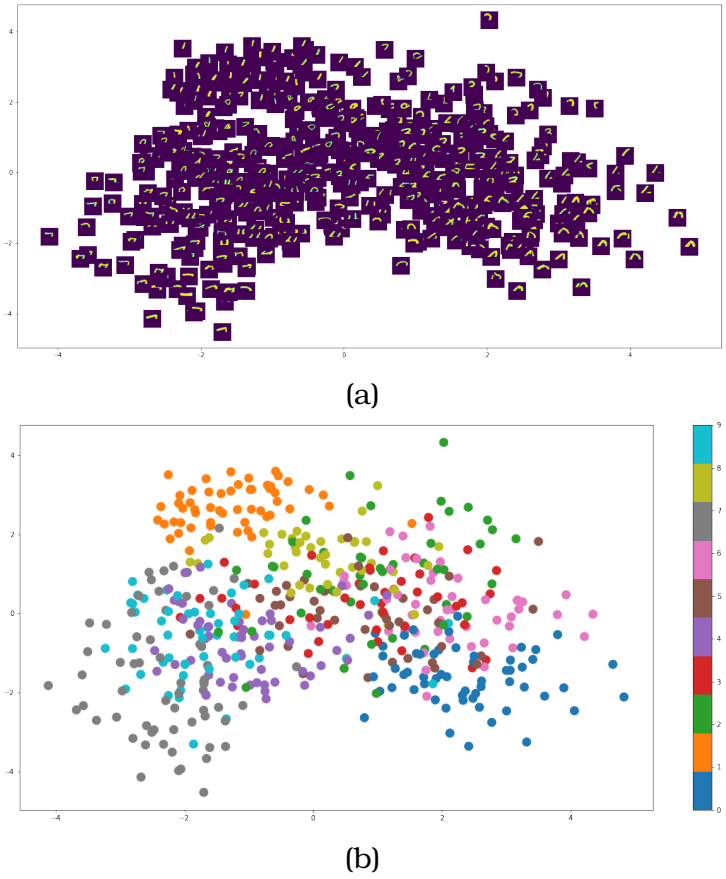
(b)

Figure 4.8: Visualization of the shared latent space for gDPCCA with $\gamma = 10^{-9}$

different samples, one of which reconstructs a 3 and the other a 0.

This can be explained if we look at the latent spaces themselves. As we recall, there are three latent spaces: two view specific spaces and one space shared by both views. The view-specific spaces account for the variation present in a given view. Meanwhile, the shared space learns the source information, the elements common to both views, which is why it is targeted by the graph regularization term.

As we can see in Figure 4.8, the shared latent space clusters the digits in a very particular way. Indeed, clustering is more prevalent in some groups rather than others and there is significant overlap



(a)

(b)

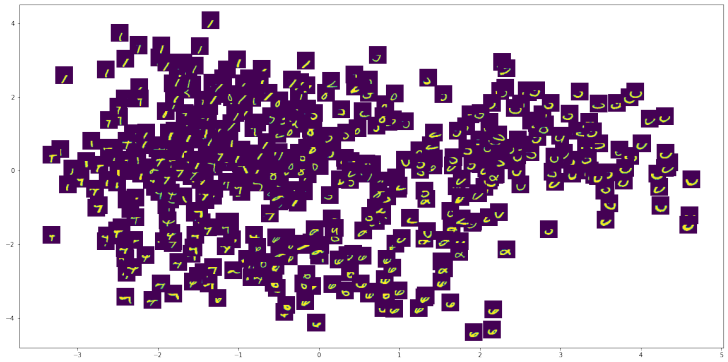
Figure 4.9: Visualization of the view-specific latent space corresponding to the first view (the top half of each image) for gDPCCA with $\gamma = 10^{-9}$. The 2-component LDA transform was applied to project the data to two dimensions.

between some clusters. For example, the group of ones and sevens contain some elements that look very much alike, leading to an intersection between the two groups in which elements could be mistaken for either group. In many other cases (for example, between 9 and 4), intersections between clusters coincide with their elements sharing similar features. The opposite also appears to be true: as one would expect, elements that are far apart in the space do not have much in common. For example, zeros and ones are quite far in the shared latent space, which aligns with our intuition

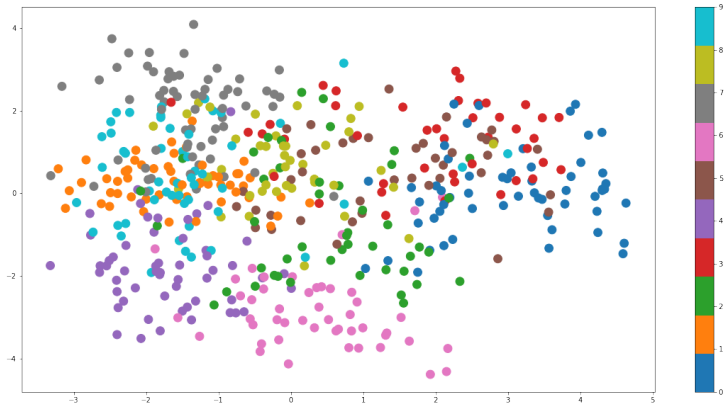
— no matter how we look at a 1, it will look nothing like a zero (and vice versa). Meanwhile, the bottom of a 3 could very well be the same as that of a 5, as we recall from Figure 4.7.

The view-specific latent spaces are also quite insightful. If we look at 4.9.a, we can see how the top parts of digits are related to each other. This "field" has various distinct visually identifiable clusters. The center features oval-like features, which are common to many digits, as we can see in Figure 4.9.b (8, 9, 3, and 4 have top parts that resemble ovals to some extent). Ones are very easily clustered in this space since no other digit contains a straight line in its top half. Most evens and zeros can also be distinguished quite easily (zeros have a fairly unique arch, for instance). The overlap we saw in 4.7 is made clear here: threes and zeros are quite close together in this space.

A similar analysis of the view-specific latent space for the bottom part of the images will yield unsurprising conclusions. For instance, the bottom halves of ones, sevens, and nines are impossible to distinguish with a naked eye (as we can see in Figure 4.9.a) and, yet, a non-negligible amount of clustering can be seen in 4.9.b. This could be due to the graph regularization that was added in the shared space.



(a)



(b)

Figure 4.10: Visualization of the view-specific latent space corresponding to the second view (the bottom half of each image) for gDPCCA with $\gamma = 10^{-9}$. The 2-component LDA transform was applied to project the data to two dimensions.

4.3 Colored MNIST

As we have just seen, gDPCCA can yield better results than DPCCA when images are involved and the knowledge of the source to source correlation is limited to clustering based on the source class (for example, both the top and bottom views of the digit 3 refer to the source object belonging to the class "3"). We will now explore the special case of gDPCCA described in Chapter 3.3.1 in which the source to source correlation is assumed to be derived from the relative spatial positioning of small image patches in a larger image.

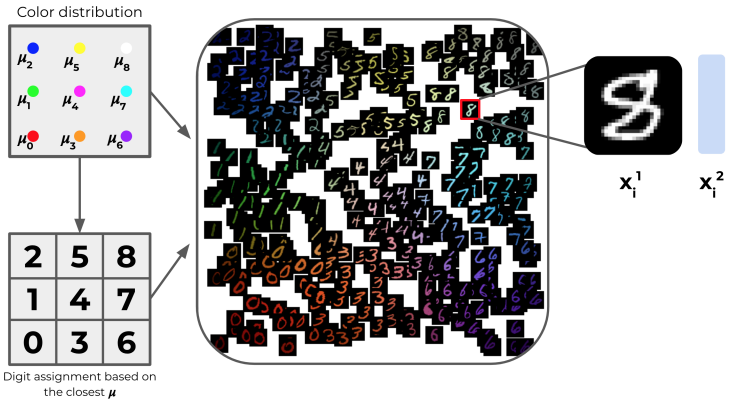


Figure 4.11: Illustration of the two different views generated from a given image in the colored MNIST dataset

We will thus investigate whether the relative positioning of objects in an image can help inform their shared latent representation to enable better reconstruction of the individual views.

In this case, we will consider the smallest possible patches, those which only contain one object without any obstructions. In particular, each source depends on where it is located in the big image, as explained in Chapter 3.3.1. From a given source, two views are realized: a color and an MNIST digit, as can be seen in Figure 4.11.

4.3.1 Data Generation

Like in Chapter 3.3.1, we are working with two different entities: large images and small patches. Each large image contains some MNIST digits arranged in a certain way, each of which has an associated color.

The idea is to have an underlying two-dimensional source distribution which determines the placement of colors and digits in the image. To this end, we will build up the definition for two functions: $\mathbf{F}_{x_1}$ and $\mathbf{F}_{x_2}$, which will respectively assign digits and colors to points in a large image.

Suppose that we split the larger image into 8 squares of the same size and create 8 different multivariate normal distributions, $\{\mathcal{N}(\mu_i, \sigma I_2)\}_{i=1}^{8}$, whose mean μ_i is the center of the i^{th} square, as can be seen in Figure 4.11. The variance σI_2 is kept the same for all distributions.

In addition, each distribution is assigned some color. In particular, distribution i will have a distinct color $\mathbf{c}_i \in \mathbb{R}^3$, where $\mathbf{c}_i \neq \mathbf{c}_j$ when $i \neq j$.

Then, denoting $f_i : \mathbb{R}^2 \to [0, 1]$ the probability density function (pdf) of the i^{th} distribution, we can define the color at pixel $\mathbf{r}$ to be the mixture of the distributions' colors weighted by their pdf evaluated at $\mathbf{r}$,

$$\mathbf{F}_{\mathbf{x}_2}(\mathbf{r}) = \sum_{i=1}^{8} f_i(\mathbf{r})\mathbf{c}_i \tag{4.4}$$

Meanwhile, the digit at that same location $\mathbf{r}$ will be given by the index of the closest distribution,

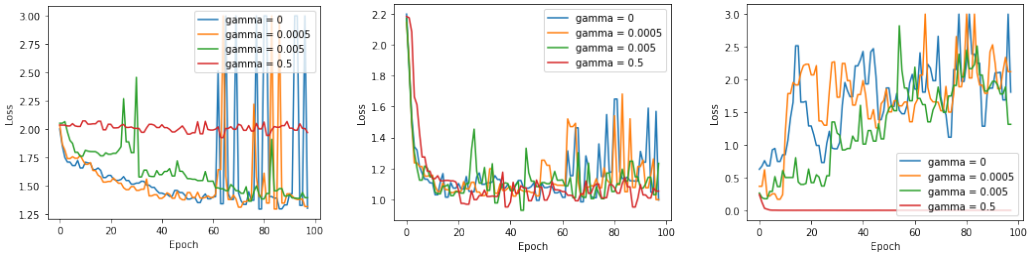
$$\mathbf{F}_{\mathbf{x}_1}(\mathbf{r}) = \mathbf{argmin}_i ||\mathbf{r} - \mu_i|| \tag{4.5}$$

thus making a hard digit cutoff between regions, yielding a square Voronoi diagram, as can be seen in Figure 4.11 (the shape is due to the variance being the same for each distribution and having no covariance between the x and y directions).

Since we now have a mechanism to generate the two views, we fix N, the number of objects we wish to have in a given image. Then, we uniformly sample N positions $\{\mathbf{r}_j\}_{j=1}^{N}$ within the image's dimensions. Then, the data for view $m \in \{1, 2\}$ in the large image at iteration i is given by

$$\mathbf{x}_i^m = [\mathbf{F}_{\mathbf{x}_m}(\mathbf{r}_j)]_{j=1}^{N} \tag{4.6}$$

The graph adjacency matrix G can be constructed like in Equation 3.7. Various similarity kernels can be used to establish the sim-



(a) View 1 Loss (b) View 2 Loss (c) Graph loss

Figure 4.12: Loss on Colored MNIST data when varying the γ parameter for gDPCCA.

ilarity between patches in the larger images on the basis of their Euclidean distance. We chose to use a modified RBF kernel with $\sigma = 1$ to establish the similarity between the centers of two patches, $\mathbf{r}_j$ and $\mathbf{r}_k$ as,

$$\mathcal{K}(r_j, r_k) = \exp\left(-\frac{||\mathbf{r}_j - \mathbf{r}_k||^2}{\sigma^2}\right) \tag{4.7}$$

Repeating this process will generate multiple images with spatially correlated views, the correlation of which is captured by G.

4.3.2 Methodology

We employed the same methodology for this experiment as for the previous two. Namely, each model was trained with varying γ parameters for a fixed number of epochs. Each of the larger images contained 10 sources (each $\mathbf{G}_i$ is thus a 10 by 10 matrix).

4.3.3 Results

For the first time in this analysis, the results do not immediately indicate that graph regularization helps improve performance. In some cases, as we can see in 4.12.a, the loss is much higher when large amounts of graph regularization are used. In particular, this occurs when $\gamma = 0.5$ which, in this case, leads to extremely low graph regularization loss (Figure 4.12.c) and view 2 loss (Figure

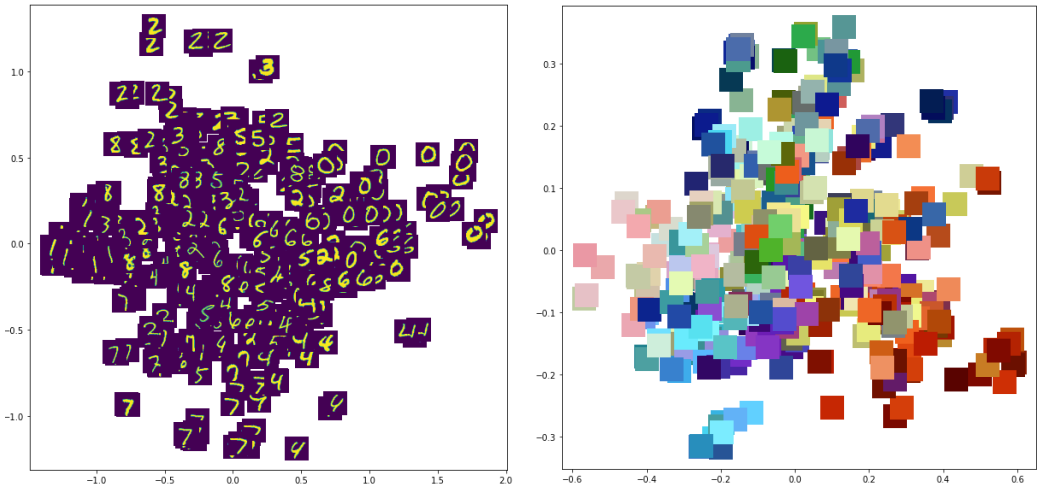
4.12.b), but a steady, non-decreasing, loss for the first view (Figure 4.12.a).

This is not to say that graph regularization only has adverse effects. For instance, when $\gamma = 0.005$, the loss decreases in a smoother way than for $\gamma = 0$ and $\gamma = 0.0005$, especially for the first view (Figure 4.12.a) when the first two models' loss begins to blow up.

A visualization of the three latent spaces obtained when $\gamma = 0.0005$ can be seen in Figure 4.13. As we can see in Figure 4.13.a, there is very clear clustering according to digit identity. Similarly-shaped digits are closer to each other, such as sixes and zeros, for example. Meanwhile, the latent representation for the second view (Figure 4.13.b) clusters according to color in a fairly accurate way. Meanwhile, the shared representation (Figure 4.13.c) captures a combination of these which, in some way resembles the source space (Figure 4.11).

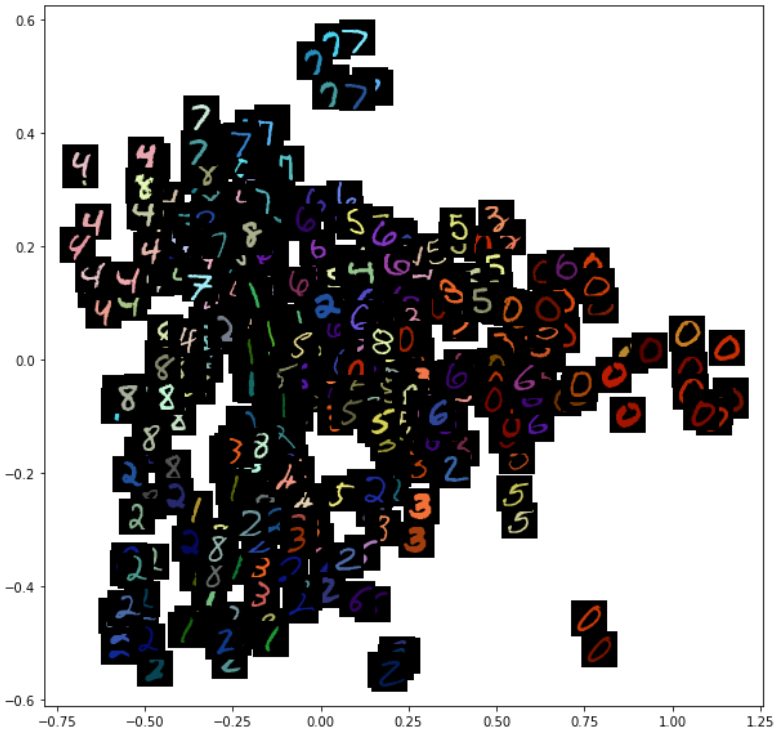
It is likely that the effects of graph regularization are limited due to the sharp switch between classes in the image-generating mechanism. Whereas the color continuously varies according to the source distributions, the digits are abruptly determined by a nearest-neighbor (the neighbor being the distribution mean) mechanism, as can be seen in Figure 4.11. Consequently, part of the graph regularization becomes useless. Had there been a smooth variation in the digits, spatial graph regularization might have been more useful. Instead, binary graph regularization, like in the Bimodal MNIST experiment, could be more beneficial for this dataset.

This experiment thus suggests that space-informed gDPCCA should only be used when one expects continuous spatial variation for both views.



(a) View 1 Latent Space

(b) View 2 Latent Space



(c) Shared Latent Space

Figure 4.13: 2D PCA projections of the latent spaces when $\gamma = 0.0005$ on the colored MNIST dataset.

4.4 What about gDCCA?

Up until now, this evaluation section has been entirely focused on the probabilistic model, gDPCCA. Indeed, given that it enables us to generate new samples from the latent representation, it is far more applicable. However, gDCCA should not so easily be dismissed. Indeed, as we will see in this section, it seems promising when compared to the baseline, DCCA.

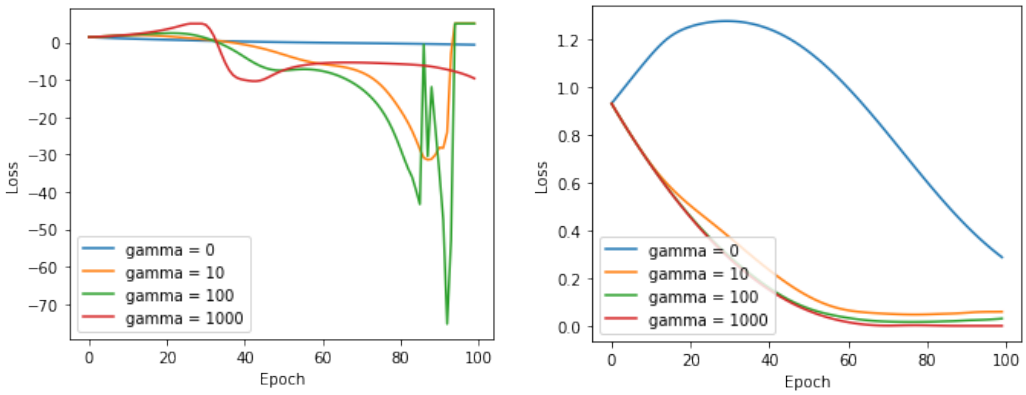
4.4.1 Iris Dataset

The Iris dataset [2] is one of the simplest datasets that we can use to better understand the benefits of gDCCA. It contains 50 observed traits for three different types of Iris (flowers): Iris setosa, Iris virginica and Iris versicolor [2]. Each sample's sepal length, sepal width, petal length, and petal width are recorded, yielding a 4-dimensional feature vector.

Obtaining a bi-modal dataset from the Iris dataset is quite simple; a natural way of doing this is to split the 4-dimensional features into two 2-dimensional vectors, the two views. How well can we correlate these two views? More importantly, how could graph-regularization be included in this example?

A simple way of adding some regularization is to include binary source information, like in the Bimodal MNIST example. More specifically, G is built so that $\mathbf{G}_{i,j} = 1$ if the i^{th} and j^{th} sample both have the same label — if they both describe the same flower. Otherwise, $\mathbf{G}_{i,j}$ is 0.

For the sake of seeing how graph regularization could affect performance, we ran the same initial model for 100 epochs, with varying γ parameters (0, 10, 100, and 1000). We used 3/4 of the data for training and used the rest to evaluate the models' performance. We opted for a 2-dimensional latent space.



(a) CCA Loss

(b) Graph Loss

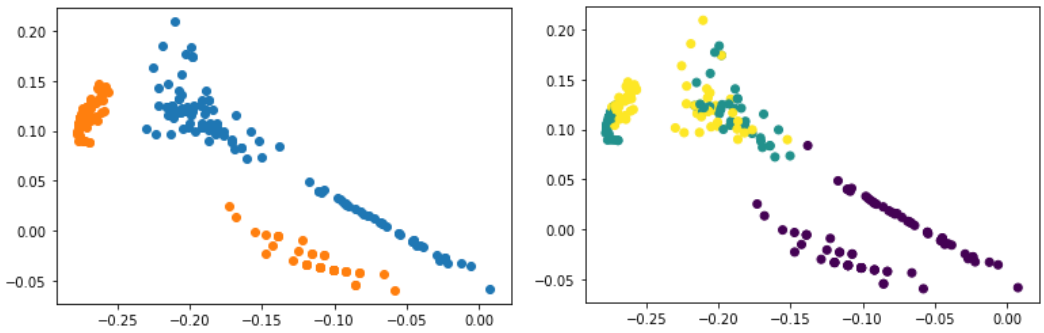
Figure 4.14: CCA loss and graph regularization loss on the Iris dataset for models with different values of γ.

Results

As we can see in Figure 4.14.b, not having graph regularization ($\gamma = 0$) does not prevent the model from figuring out that $\mathcal{L}_{\mathcal{G}}$ should decrease — in this case, the model is just DCCA. This suggests that minimizing the CCA objective indirectly minimizes the graph loss. This is not surprising; we expect the CCA objective to lead to some clustering based on the source's identity (which type of flower the views represent). However, although the CCA loss does decrease when $\gamma = 0$ (it is barely noticeable since the scale is much larger when $\gamma \gg 0$), it is nothing close to what can be achieved with graph regularization (see Figure 4.14.a).

Whenever γ is nonzero, the graph regularization immediately decreases. Thus, the model learns to cluster embeddings based on which source Iris they come from. This additional information is of non-trivial importance in the CCA minimization. As we can see, the right amount of graph regularization ($\gamma = 100$ yielded the best results) leads to a much sharper decrease, leading to significantly smaller minimum.

However, given the small sample size — the dataset is very limited



(a) Color corresponds to view

(b) Color corresponds to class

Figure 4.15: Shared embeddings of the model when $\gamma = 100$. In both (a) and (b), we can see the embedded samples. In (a), the color denotes the view while, in (b), it denotes the class to which each sample belongs.

in size — overfitting happens quite rapidly. Once the minimum has been reached (which typically happens around the 85^{th} epoch, the loss jumps drastically on the testing set.

In Figure 4.15, we can see the structure of the embedding space. We recall that both views are projected to the same space, as can be seen in Figure 4.15.a. As expected, the encoding of the two views in the latent space leads to clustering according to which class they each belong to. In addition, we observe strong correlation between the two views themselves, as can be seen in 4.15.b.

It thus seems reasonable that gDCCA improves the performance of DCCA when there is (even limited) information about source to source correlation.

Chapter 5

Case Study: Tension in Epithelial Cells

In the previous chapter, we explored some of the ways that Graph Regularized Deep PCCA can be applied to synthetic datasets. The data, itself, was not very interesting or insightful. Instead, it provided us with simple test cases to evaluate the effectiveness of our method when compared to the vanilla DPCCA. A natural follow-up question is: in which real-life scenarios would such an algorithm be useful?

This chapter takes one specific example, the problem of estimating tension in epithelial cells, as a case study and discusses how gDPCCA could improve the current state of the art and yield new insights.

5.1 Motivation

Cell shape and development is driven by a set of intracellular and extracellular forces [23]. Gaining a proper understanding of cell and tissue mechanics is thus vital to developmental biologists and, in

particular, those who study organ formation and embryonic morphogenesis [26][23].

In particular, epithelial tissue is composed of two-dimensional monolayers with visible edges between the cells. The way that these monolayers behave is driven by cytoskeletal cortices coupled by intercellular adherens junctions [26]. The shape of an epithelial cell at some point in time is affected by both the intracellular osmotic pressure — which acts to prevent adjacent cells from decreasing the cell's volume — and the balance of local actomyosin cytoskeletal contractility [26][23][28].

One of the main challenges with measuring mechanical properties of cells, such as tension, is that they are usually destructive [18][9][26]. Hence, given the wide range of live imaging data, various computer vision alternatives were proposed, most of which rely on parametrizing the cell network using some tiling (e.g. polygonal) [15]. The state of the art, Variational Approach to Stress Inference (VMSI), considers a circular arc polygonal (CAP) tiling. In particular, they show that there exists a mathematical duality between the two-dimensional CAP tiling and the three-dimensional triangulation formed by the interfacial tensions at equilibrium [26]. By exploiting this duality, they were able to describe an algorithm which can deduce the tensions from a segmented cell graph [26]. One recent implementation of VMSI, TensionMap, streamlines the whole process — given an image of epithelial cells, it returns the corresponding tension map, a CAP tiling with relative tension values on the segmented edges.

However, the main issue with VMSI is that it is quite expensive to evaluate. Given an image with hundreds of cells, it can take up to 5 minutes to obtain the predicted tensions. Deep learning based approaches, which are very quick to evaluate once trained, could thus significantly speed up the tension inference process.

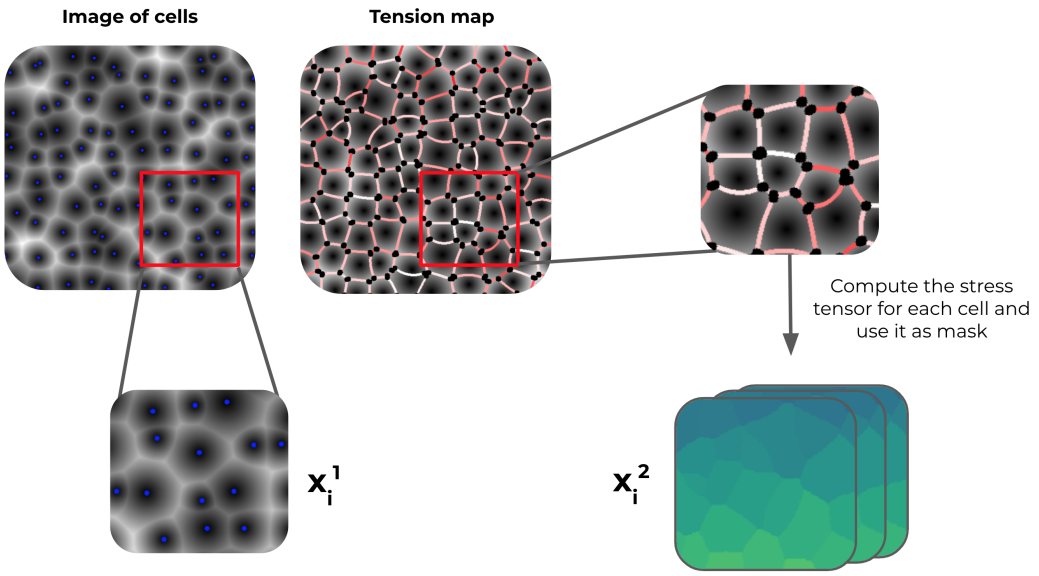


Figure 5.1: Illustration of the data processing needed for tension inference to be done with gDPCCA.

5.2 Tension Inference with gDPCCA

Given the clear problem of speeding up tension inference in images of epithelial cells, gDPCCA is an interesting candidate for solving the problem.

Tension is a clear physical quantity that captures a key component of the mechanical state of a cell in relation to its neighbors. Each cell will have varying tensions at each of its edges. From an algorithmic implementation standpoint, it would be simpler to have one quantity agglomerate these different values.

Luckily, in biology, a more common measure, the stress tensor (which can be computed from the tensions and pressures), does exactly that. A straightforward implementation of gDPCCA in this context would be to have one view contain the cells, while the other contains a stacked set of masks whose number of channels depends on the number of dimensions needed to describe the stress and whose value at any point on the plane will depend on the cell in

that spot. This whole process is illustrated in Figure 5.1.

5.3 Pressure Inference

Given how complex estimating tension is, we first tackle a similar but simpler problem: inferring the pressure in cells. Since the tension is in part determined by the intracellular pressure, successfully predicting pressures would be promising and indicate that gDPCCA could also be useful for tension inference.

5.3.1 Data Generation

Since obtaining single cell data with labeled pressure values is close to impossible, we instead resort to generating data synthetically. The idea is to create realistic cell arrays from generating points with known pressure values. Additionally, since we want to see if gDPCCA can exploit spatial correlation, the pressures themselves are spatially correlated via some underlying two-dimensional Gaussian distribution.

Generating Cell Images Given Pressures

Suppose that we are given a set of points and pressures and asked to generate a realistic image that reflects the chosen values. The pressure difference between cells directly corresponds to the curvature of the edges. For example, if there is no pressure difference, then the edge will be straight, whereas a high pressure difference will yield a highly curved edge. One way of generating cell images is to use a generalized Voronoi construction like in the VMSI paper [26].

The construction is based on the duality explored in the VMSI paper. Given $(q_\alpha, z_\alpha, p_\alpha)$, where $q_\alpha \in \mathbb{R}^2$ is the position of the generating

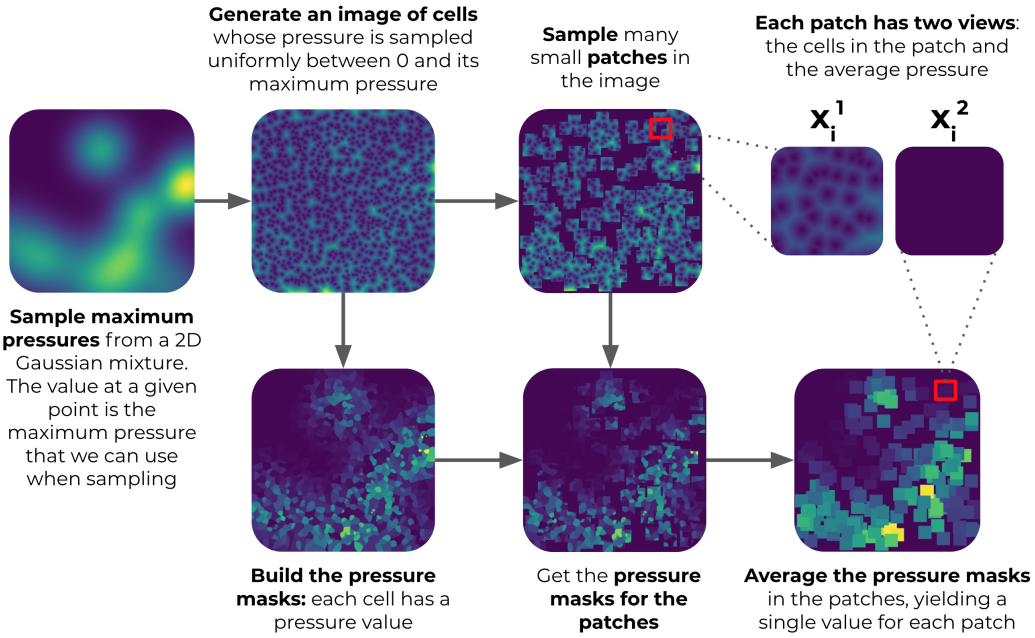


Figure 5.2: Illustration of the process for generating spatially correlated patches of synthetic cells.

points, $z_\alpha \in \mathbb{R}$ is some offset in the z-direction, and p_α is the pressure for cell α. For our purpose, we simplify the process by removing the extra dimension, i.e. getting rid of z_α. Meanwhile, the q_α can be sampled uniformly on the 2×2 grid.

Then, the intensity of each pixel in the resulting image will be given by the (non-Euclidean) distance to the nearest generating point. The key here is that the distance will directly depend on the pressure.

The pressure-scaled distance between a pixel $r_i \in \mathbb{R}^2$ and a generating point is given by

$$d_\alpha(r_i) = ||r_i - q_\alpha||\sqrt{p_\alpha} \tag{5.1}$$

Then, the grayscale value $I(r_i)$ can be written as

$$I(r_i) = \min_\alpha d_\alpha(r_i) \tag{5.2}$$

The image output will have an edge $\mathcal{E}_{\alpha\beta}$ between cells α and β that is given by

$$\mathcal{E}_{\alpha\beta} = \left\{ r_i \mid \sqrt{p_\alpha} d_\alpha(r_i) = \sqrt{p_\beta} d_\beta(r_i) \right\} \tag{5.3}$$

As we can see, if p_α is always 1, then the image reduces to a standard Voronoi diagram (i.e. all the edges will be straight).

Spatially Correlating the Pressures

Now that we have seen how cell images can be generated from a set of generating points and pressures, how do we generate the pressures? As we mentioned before, the advantage of using spatial gDPCCA is that it takes into account how patches are correlated in a larger image.

Thus, the idea is quite simple. We generate very large images from which we extract patches that are somehow spatially correlated. One way to ensure this is to sample the pressures in such a way that there is a smooth transition between regions (i.e. we want regions with low pressure, others with high pressure, and so on). Since the images are generated using the pressure, the small image patches in the larger image will reflect this choice. There will be patches whose cells have higher edge curvature than others.

In order to have such a property, the idea is to have the interval from which we uniformly sample the pressure in a given cell vary smoothly through space.

Suppose that we restrict this interval (i.e. make it smaller) within a region of the image in which we want to generate cells. Then, the sampled pressures will be very close to each other, implying that the pressure difference between cells will be quite small. As a result, the edges between adjacent cells in that region will be fairly straight. The opposite effect will occur if the interval from which we sample the pressures is very large (in this case, the curvature of the edges will be high). Hence, it follows that smoothly varying the length of

this interval will lead to a smooth variation of the expected edge curvature in the image.

More precisely, given N generating points $\{\mathbf{q}_\alpha\}_{\alpha=1}^{N}$, the pressure for cell α can be sampled as

$$p_\alpha \sim \mathcal{U}([0, \mathbf{M}_\alpha]) \tag{5.4}$$

where $\mathbf{M}_\alpha$ denotes the maximum permissible pressure value for α. It is, itself, spatially dependent on the value of the two-dimensional probability density function of some Gaussian mixture evaluated at the point q_α,

$$\mathbf{M}_\alpha = \sum_{k=1}^{K} \left\{ \frac{1}{\sqrt{4\pi^2|\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{q}_\alpha - \mu_k)^T \Sigma_k^{-1} (\mathbf{q}_\alpha - \mu_k) \right) \right\} \tag{5.5}$$

where the parameters $\{\mu_k, \Sigma_k\}_{k=1}^{K}$ are arbitrary (they could be uniformly sampled, for example).

What about the second view? We have large images with cells whose physical features, the edge curvature, vary spatially. A second view which shares this property must thus also depend on the pressure. A natural candidate is a cell segmentation dependent on the pressure inside the cell, as can be seen in Figure 5.2. The two views would thus be two patches: one with the image and the other with the corresponding pressure mask. However, the pressure mask is too "exact" — it is dependent on the exact cell positions, meaning that an excessive amount of training would be needed to extract useful information.

In order to simplify this further, we can turn the second view into a summary of the pressure mask, some vector which captures the essence of the pressure information contained in the mask. One such candidate is the average pressure in the mask. As we can see in Figure 5.2, the average pressure in small enough patches (we

used 28×28) does indeed smoothly vary through the space: patches close together in the original image will have a similar average pressure.

The whole data generation process, including the intermediary images, can be visualized in Figure 5.2.

Getting the Graph

The data is spatially correlated, yet we still need a graph to captures spatial information. Space informed gDPCCA (see Chapter 3.3.1) has us consider the relative distance between patches.

The only thing that we need to do is to come up with the kernel which assigns a value to pairs of patch centers. Knowing that the underlying mechanism used to sample the pressures is Gaussian, using an RBF kernel seems reasonable.

Thus, G is defined like in Equation 3.7 with

$$\mathcal{K}(s_i, s_j) = \exp\left(-\frac{||\mathbf{r}_i - \mathbf{r}_j||^2}{\sigma^2}\right) \tag{5.6}$$

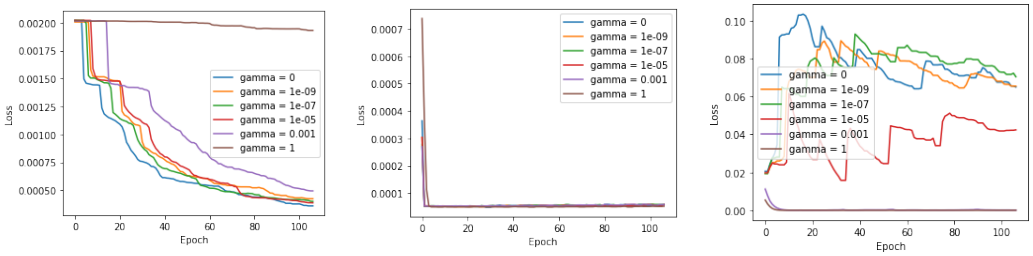
where $\mathbf{r}_i$ is the center of patch i and σ is a parameter to be fixed based on the actual distribution. One way of deducing σ is to evaluate it on different patches and visually see if the similarity is too high or low.

5.3.2 Methodology

Now that we have a way of generating spatially correlated data, we can evaluate gDPCCA for different values of γ. As in previous experiments, all models were initialized in the same way. The models were trained on 130 large images in which every set of grouped elements (patches which are related via a graph) contains 200 patches. The 28×28 patches, themselves, are uniformly generated at every

iteration, yielding an (almost) infinite dataset. For each model, a 10-dimensional latent representation was used. Training was done for 105 epochs.

5.3.3 Results



(a) View 1 Loss (b) View 2 Loss (c) Graph loss

Figure 5.3: Loss on the pressure dataset when varying the γ parameter for gDPCCA.

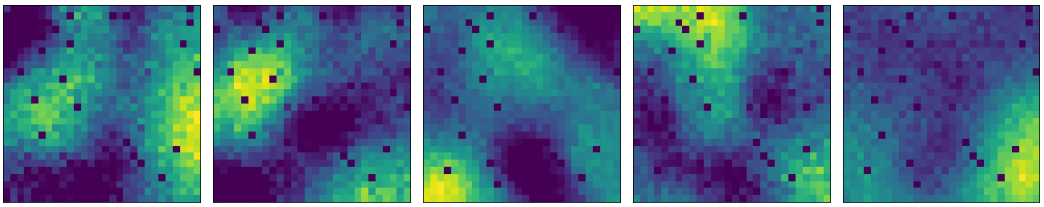


Figure 5.4: Sampled patches from the trained gDPCCA model.

Due to a lack of convergence for either DPCCA or gDPCA, this experiment is inconclusive.

However, surprisingly, graph regularization does not noticeably improve the performance of the baseline DPCCA on the pressure data, as can be seen in Figure 5.3. In fact, as we can see in Figure 5.3.a, it appears that regularization ends up slowing down convergence for the first view. In Figure 5.4, we can see some samples from this view. Although they are still easily distinguishable from the data that was used to train the model, they capture some of the features that characterize these cell-patches.

The same cannot be said for the second view; both DPCCA and gDPCCA seem to struggle to move past the initial drop which, upon closer inspection, corresponds to a performance that is only marginally better than uniformly sampling in some acceptable interval.

This poor performance could be explained by the use of insufficiently complex convolutional neural network architectures for the encoder and decoder networks. The architecture that we used only contained 2 convolutional layers, which could be far too little to capture this subtle information. This also would explain the poor performance for the second view. Figuring out the average pressure in the patch is far from trivial. It requires identifying the edges in the image, computing their curvature, and comparing them.

Another explanation could be the latent dimension, 10, being too small for this dataset. As a result, we also tested 20-dimensional and 30-dimensional latent spaces. However, it had little impact on convergence. All of these indicate that the complexity (or lack thereof) of the network may be to blame. Further research would involve re-running this experiment with more complex neural networks.

Chapter 6

Looking Forward

In this dissertation, we introduced gDPCCA and gDCCA, two novel algorithms for bimodal correlation. In this chapter, we will discuss some current challenges as well as some potential future research directions.

What's next for gDPCCA?

As we saw in the last two chapters, gDPCCA seems promising when one is working with image data for which there exists some underlying phenomena that is spatially correlated. However, as we saw, choosing the right hyperparameters — such as the size of the latent space, the batch size, or γ — can be a long and tedious process. It seems that our methods are sensitive to such changes. In future works, it would be interesting to conduct a proper sensitivity analysis and, perhaps, find ways to improve gDPCCA's usage in practical cases.

An additional factor to take into account is how the graph G is built and what information it contains. As we saw, in some cases, spatial positioning might be useful to inform the latent representation, but not always. The kernel that is used to weigh the distance between

points is a very crucial component. More often than not, as we saw in the experiments, enforcing sparcity in G is desired. In the future, looking into how to improve the construction of G could lead to a significant performance boost.

Tension Inference

As was mentioned in the previous chapter, which took as case study the question of infering tension in epithelial cells, we were unable to apply our method to an actual image-tension dataset.

One of the primary reasons for this is the time it takes to generate the dataset. The TensionMap package, which I developed for a previous course and continued to work on while undergoing this project, suffers from speed issues (due to optimization being rather slow in Python). In future works, I hope to improve the package to be able to realistically generate a dataset.

Another reason was the ambitious nature of such a case study. A more feasible task to tackle was pressure inference, which we performed in the previous chapter. Given the ambiguous nature of the results, it remains unknown whether a similar method will work for tension inference.

Beyond tension inference, it would also be interesting to consider how mechanical properties of cells and gene expression levels are linked. A similar approach could be developed using gDPCCA in future works in order to approach this question.

Extension to $m > 2$ modalitites

The methods that we discussed were limited to working with bimodal data. What about when the number of modalities is greater than 2? It would be interesting to extend these methods to include an arbitrary number of modalities. For example, we could consider a generalization of PCCA [3] in which the generative model is defined

as,

$$\mathbf{y}_m = \mathbf{W}^m\mathbf{z} + \mathbf{B}^m\mathbf{z}^m + \mathbf{u}^m \tag{6.1}$$

for each view $m \in \{1, \cdots, M\}$.

What about unpaired data?

Throughout this dissertation, we made the assumption that the data views were paired. However, in many applications, the data that we are working with is unpaired — we just know that the two datasets contain different views of a common source. As a result, different methods need to be developped.

One such method [31] trains m different autoencoders, one for each view. The novelty, which leads to a shared representation, is the addition of an extra divergence term which forces the latent representations to match. It would be interesting to compare the shared latent representation learned by this method with the one learned by gDPCCA.

Bibliography

[1] Galen Andrew et al. "Deep Canonical Correlation Analysis". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 2013, pp. 1247–1255. URL: http://proceedings.mlr.press/v28/andrew13.html.

[2] D. F. Andrews and A. M. Herzberg. "Iris Data". In: *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. New York, NY: Springer New York, 1985, pp. 5–8. ISBN: 978-1-4612-5098-2. DOI: 10.1007/978-1-4612-5098-2_2. URL: https://doi.org/10.1007/978-1-4612-5098-2_2.

[3] Cédric Archambeau and Francis Bach. "Sparse probabilistic projections". In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller et al. Vol. 21. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper/2008/file/d93ed5b6db83be78efb0d05ae420158e-Paper.pdf.

[4] Francis Bach and Michael Jordan. "A probabilistic interpretation of canonical correlation analysis". In: (May 2005).

[5] Jonathan Bennett. "GW Leibniz, Discourses on Metaphysics". In: (2007). URL: https://www.earlymoderntexts.com/assets/pdfs/leibniz1686d.pdf.

[6] Louis de Benoist et al. "A Historical Analysis of the Identity of Indiscernibles". In: (2020). URL: https://medium.com/literally-literary/a-historical-analysis-of-the-identity-of-indiscernibles-8a4f411f9a13.

[7] C M Bishop and M E Tipping. "Probabilistic Principal Component Analysis". In: (2001). PPCA.

[8] Max Black. "The Identity of Indiscernibles". In: *Mind* (1952).

[9] I. Bonnet et al. “Mechanical state, material properties and continuous description of an epithelial tissue”. In: *J R Soc Interface* 9.75 (2012), pp. 2614–2623.

[10] Jia Chen, Gang Wang, and Georgios B. Giannakis. “Graph Multiview Canonical Correlation Analysis”. In: *IEEE Transactions on Signal Processing* 67.11 (2019), 2826–2838. ISSN: 1941-0476. DOI: 10.1109/tsp.2019.2910475. URL: http://dx.doi.org/10.1109/TSP.2019.2910475.

[11] Jia Chen et al. “Canonical Correlation Analysis of Datasets With a Common Source Graph”. In: *IEEE Transactions on Signal Processing* 66.16 (2018), 4398–4408. ISSN: 1941-0476. DOI: 10.1109/tsp.2018.2853130. URL: http://dx.doi.org/10.1109/TSP.2018.2853130.

[12] Zoubin Ghahramani and Geoffrey E. Hinton. *The EM Algorithm for Mixtures of Factor Analyzers*. Tech. rep. 1997.

[13] Gregory Gundersen. *Probabilistic Canonical Correlation Analysis in Detail*.

[14] Gregory Gundersen et al. “End-to-end Training of Deep Probabilistic CCA on Paired Biomedical Observations”. In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*. Ed. by Ryan P. Adams and Vibhav Gogate. Vol. 115. Proceedings of Machine Learning Research. PMLR, 2020, pp. 945–955. URL: http://proceedings.mlr.press/v115/gundersen20a.html.

[15] H. Honda, H. Yamanaka, and M. Dan-Sohkawa. “A computer simulation of geometrical configurations during cell division”. In: *J Theor Biol* 106.3 (1984), pp. 423–435.

[16] H. Hotelling. “The most predictable criterion.” In: *Journal of Educational Psychology* 26 (1935), pp. 139–142.

[17] Harold Hotelling. “Relations Between Two Sets of Variates”. In: *Biometrika* 28.3/4 (1936), pp. 321–377. ISSN: 00063444. URL: http://www.jstor.org/stable/2333955.

[18] M. S. Hutson et al. “Forces for morphogenesis investigated with laser microsurgery and quantitative modeling”. In: *Science* 300.5616 (2003), pp. 145–149.

[19] Margaret Jourdain. “Diderot’s Early Philosophical Works”. In: (1916). URL: https://archive.org/details/diderotsearlyphi010275mbp/page/n13.

[20] J. R. Kettenring. “Canonical Analysis of Several Sets of Variables”. In: *Biometrika* 58.3 (1971), pp. 433–451. ISSN: 00063444. URL: http://www.jstor.org/stable/2334380.

[21] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[22] Aarto Klami and Sami Kaski. "Probabilistic approach to detecting dependencies between data sets". In: *Neurocomputing* 72 (2008), pp. 39–46.

[23] Thomas Lecuit and Pierre-François Lenne. "Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis". In: *Nature Reviews Molecular Cell Biology* 8.8 (2007), pp. 633–644. DOI: 10.1038/nrm2222. URL: https://doi.org/10.1038/nrm2222.

[24] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[25] Pierre Legendre. "Numerical Ecology: Developments and Recent Trends". In: *Numerical Taxonomy*. Ed. by Joseph Felsenstein. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 505–523. ISBN: 978-3-642-69024-2. DOI: 10.1007/978-3-642-69024-2_56. URL: https://doi.org/10.1007/978-3-642-69024-2_56.

[26] Nicholas Noll, Sebastian J. Streichan, and Boris I. Shraiman. "Variational Method for Image-Based Inference of Internal Stress in Epithelial Tissues". In: *Phys. Rev. X* 10 (1 2020), p. 011072. DOI: 10.1103/PhysRevX.10.011072. URL: https://link.aps.org/doi/10.1103/PhysRevX.10.011072.

[27] Reza Rohani Sarvestani and Reza Boostani. "FF-SKPCCA: Kernel probabilistic canonical correlation analysis". In: *Applied Intelligence* 46.2 (2017), pp. 438–454. DOI: 10.1007/s10489-016-0823-x. URL: https://doi.org/10.1007/s10489-016-0823-x.

[28] Martin P Stewart et al. "Hydrostatic pressure and the actomyosin cortex drive mitotic cell rounding". In: *Nature* 469.7329 (2011), 226—230. ISSN: 0028-0836. DOI: 10.1038/nature09642. URL: https://doi.org/10.1038/nature09642.

[29] Valentine Svensson, Sarah Teichmann, and Oliver Stegle. *SpatialDE - Identification of spatially variable genes*. Nov. 2017. DOI: 10.1101/143321.

[30] Steven Van Vaerenbergh, Javier Via, and Ignacio Santamaria. "A kernel canonical correlation analysis algorithm for blind equalization of oversampled Wiener systems". In: Nov. 2008, pp. 20 –25. DOI: 10.1109/MLSP.2008.4685449.

[31] Karren Dai Yang et al. “Multi-Domain Translation between Single-Cell Imaging and Sequencing Data using Autoencoders”. In: *bioRxiv* (2019). DOI: 10.1101/2019.12.13.875922. eprint: https://www.biorxiv.org/content/early/2019/12/18/2019.12.13.875922.full.pdf. URL: https://www.biorxiv.org/content/early/2019/12/18/2019.12.13.875922.