A GENERALIZED EIGENGAME WITH EXTENSIONS TO DEEP MULTIVIEW REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalized Eigenvalue Problems (GEPs) encompass a range of interesting scientific computing problems. Canonical Correlation Analysis (CCA) and Partial Least Squares (PLS) are two such examples of GEPs which are often used to learn representations of multiview data. Development of efficient stochastic approaches to these problems would allow them to scale to large datasets. Furthermore, existing deep learning based extensions of CCA require large minibatch sizes in the stochastic setting to achieve good performance. Inspired by recent formulations of Principal Components Analysis and GEPs as games with differentiable utilities, we develop an alternative game theoretic approach to solving GEPs in which all constraints are softly enforced by Lagrange multipliers. We show that our approach shares much of the theoretical grounding of the previous game theoretic approaches but has fewer hyperparameters, is faster to converge, and permits extension to general function approximators like neural networks for certain GEPs including CCA. We demonstrate the effectiveness of our method for solving GEPs using canonical multiview datasets and demonstrate state-of-the-art performance for the Deep CCA problem for multiview representation learning.

1 INTRODUCTION

Generalized Eigenvalue Problems (GEPs) are often characterised by the equation:

$$AV = \lambda BV \tag{1}$$

where $A \in \mathbb{R}^{d \times d}$ is symmetric and $B \in \mathbb{R}^{d \times d}$ is symmetric, positive definite. GEPs generalize the standard eigenvalue problem where $B = I$. One way to address GEPs is to assume (or ensure by regularization) that B is invertible in order to form the related Eigenvalue Problem:

$$B^{-1}AV = \lambda V \tag{2}$$

However in the stochastic setting (where we only have access to sample estimates $\hat{A}$ and $\hat{B}$), even if $\hat{B}$ is invertible, the inversion risks introducing bias by Jensen's inequality (Jensen, 1906; Gemp et al., 2020) or intuitively when $\hat{B}$ is near singular so that gradients of equation 2 with respect to V become extremely large. Similarly, for positive definite B, we can write a related eigenvalue problem:

$$B^{-\frac{1}{2}}AB^{-\frac{1}{2}}W = \lambda W \tag{3}$$

Whose solution is related to the solution of the generalized eigenvalue problem by $W = B^{\frac{1}{2}}V$. However in the stochastic setting we encounter similar problems to the form in equation 2

1.1 CANONICAL CORRELATION ANALYSIS

Multiview representation learning methods are useful for learning representations of data with multiple sets of features, or 'views'. Many multiview learning approaches also fall under the broader topic of unsupervised learning and can be used to generate robust low-dimensional representations or to discover relationships between views. They have been widely applied across a range of fields such as Neuroimaging (Krishnan et al., 2011), Finance (Cassel et al., 2000), and Imaging Genetics (Hansen et al., 2021) in order to find associations between views in large datasets.

Canonical Correlation Analysis (CCA) (Hotelling, 1992) is used in multiview machine learning to identify subspaces in at least two different views that are highly correlated. CCA can be understood by the constrained optimisation problem:

$$\text{maximize}_{V_x,V_y}\mathbb{E}\left[x^\top V_x V_y^\top y\right] \text{ subject to } V_x^\top \mathbb{E}[x^\top x]V_x = I_k, V_y^\top \mathbb{E}[y^\top y]V_y = I_k$$

Where V_x and V_y are typically referred to as weights and k is a specified number of latent components. By constraining the dimensions of the subspaces to unit variance, we ensure that the covariance objective is equal to correlation. By using Lagrange multipliers to enforce the constraint we can form the Lagrangian function:

$$\text{maximize}_{V_x,V_y}\mathbb{E}\left[x^\top V_x V_y^\top y\right] + \langle \lambda_x, (I_k - V_x^\top \mathbb{E}[x^\top x]V_x)\rangle + \langle \lambda_y, (I_k - V_y^\top \mathbb{E}[y^\top y]V_y)\rangle \quad (4)$$

We can combine the first order conditions into a system of equations to give the Generalized Eigenvalue Problem form of CCA:

$$\begin{pmatrix} 0 & \mathbb{E}[x^\top y] \\ \mathbb{E}[y^\top x] & 0 \end{pmatrix} \begin{pmatrix} V_x \\ V_y \end{pmatrix} = \lambda \begin{pmatrix} \mathbb{E}[x^\top x] & 0 \\ 0 & \mathbb{E}[y\top y] \end{pmatrix} \begin{pmatrix} V_x \\ V_y \end{pmatrix} \quad (5)$$

Where matrices A and B are formed from covariance blocks. Depending on the circumstance, we may be interested in the top-k generalized eigenvectors and associated values or more simply the top-k subspace (any arbitrary rotation of the space spanned by the top-k eigenvectors).

1.2 PARTIAL LEAST SQUARES

The Partial Least Squares (PLS) Wold et al. (1984) problem can also be formulated as a similar GEP but with B replaced by the identity matrix. PLS is equivalent to finding the singular value decomposition (SVD) of the covariance matrix $X^\top Y$. It has an interpretation as a ridge regularised CCA where the covariance matrices $\mathbb{E}[x^\top x]$ and $\mathbb{E}[y^\top y]$ are modified to assume that there is no collinearity between variables.

1.3 DEEP CCA

The original Deep CCA formulation (Andrew et al., 2013) is based on the the generalized eigenvalue form in equation 3 and has the objective:

$$\text{maximize}\,\text{trace}(\hat{\Sigma}_{XX}^{-1/2}\hat{\Sigma}_{XY}\hat{\Sigma}_{YY}^{-1/2}) \quad (6)$$

Where $\hat{\Sigma}_{XX}$ and $\hat{\Sigma}_{YY}$ are the sample covariance matrices of neural network representations of two views X and Y. The objective effectively maximises the top-k eigenvalues of equation 3 for the learnt representations. The authors proposed to use the full batch L-BFGS optimizer (Liu & Nocedal, 1989) which helps to ensure the positive definiteness of B and reduces bias in gradient estimates, however this limits the effective use of Deep CCA for large datasets which do not fit in memory.

In this work we build on recent work which reformulates top-k eigenvalue and generalized eigenvalue problems as games in which each component is learnt as part of a coupled 'EigenGame' in which players maximise an objective and are penalized for not remaining orthogonal to each other (Gemp et al., 2020; 2021; 2022). Unlike previous EigenGame approaches and recent work on Riemannian optimization for the CCA problem (Meng et al., 2021), we relax the constraints and instead force them using an implicit Lagrange multiplier method. We show empirically that our method can converge faster to better solutions than previous work. Furthermore, for certain GEPs including CCA our method can be extended to deep learning functional forms. Our contributions can be summarised as developing:

- An alternative, valid EigenGame form for solving GEPs in the stochastic setting with fewer hyperparameters than previous forms and no constraints which we refer to as δ-EigenGame
- Two extensions of our proposed method to the Deep CCA problem (DCCA-SGHA and DCCA-EigenGame) which address the biased gradients of the original DCCA form and both demonstrate state-of-the-art performance on canonical multiview representation learning tasks

2 An Alternative Generalized EigenGame

Our solution solves the general form of the generalized eigenvalue problem in equation 7 and adds an explicit hierarchy so that $\hat{V}_1$ solves the top-1 problem ($k = 1$) and $V_{1:k}$ solve the top-k problem. The solution to this form of the GEP presents 3 challenges. The first challenge is to find directions of maximal variance $\hat{v}_i^\top A \hat{v}_i$, the second is the unit variance constraint $\hat{v}_i^\top B \hat{v}_i = 1$, and the third is a generalized orthogonality constraint $\hat{v}_i^\top B \hat{v}_j = 0$, where $\hat{v}_i$ is an estimate of column i of V. Our solution treats the first challenge as the objective and configures the second and third challenges as soft penalties rather than constraints. By using a hierarchical game structure, as introduced by Gemp et al. (2020), we enable ordering of $V_{1:k}$ by their generalized eigenvalues.

$$\text{maximize}_{\hat{V}} \ \text{trace}(\hat{V}^\top A \hat{V}) \text{ subject to } \hat{V}^\top B \hat{V} = I_k \tag{7}$$

We will construct updates and utilities for each player in the game by noting the similarity between the μ-EigenGame variant (Gemp et al., 2021) (which was designed to solve eigenvalue problems i.e. where $B = I$) and the Stochastic Generalized Hebbian Algorithm (SGHA) (Chen et al., 2019). Specifically, μ-EigenGame was applied to the PCA problem where $B = I$ and $A = \mathbb{E}[x^\top x]$. In this case the update for the SGHA is:

$$\Delta_i^{\text{SGHA}} = A\hat{v}_i - \sum_j \hat{v}_j \left(\hat{v}_j^\top A \hat{v}_i\right) \tag{8}$$

Whereas the update for μ-EigenGame is:

$$\Delta_i^{\mu} = A\hat{v}_i - \sum_{j<i} \hat{v}_j \left(\hat{v}_j^\top A \hat{v}_i\right) \tag{9}$$

Where the major difference is that μ-EigenGame introduces an explicit hierarchy in order to find the ordered eigenvectors of A whereas SGHA only finds the top-k subspace. Now consider the SGHA update where B is an arbitrary matrix:

$$\Delta_i^{\text{SGHA}} = A\hat{v}_i - \sum_j B\hat{v}_j \left(\hat{v}_j^\top A \hat{v}_i\right) \tag{10}$$

In order to combine the benefits of SGHA and μ-EigenGame, we propose updates with an explicit hierarchy which we refer to as δ-EigenGame:

$$\Delta_i^\delta = \overbrace{A\hat{v}_i}^{\text{Reward}} - \overbrace{B\hat{v}_i\left(\hat{v}_i^\top A\hat{v}_i\right)}^{\text{Variance Penalty}} - \overbrace{\sum_{j<i} B\hat{v}_j\left(\hat{v}_j^\top A\hat{v}_i\right)}^{\text{Orthogonality Penalty}} \tag{11}$$

Where the first term maximises $\hat{v}_i^\top A\hat{v}_i$, the second term constrains $\hat{v}_i^\top B\hat{v}_i = 1$ at convergence, and the third ensures the generalized orthogonality of the solutions.

2.1 Defining a Game

Like μ-EigenGame we note that these updates are not the gradients of a utility function but rearranging we can form a pseudo-utility:

$$u_i^\delta = \hat{v}_i^\top A\hat{v}_i + y_{ii}\text{⬢}(1 - \hat{v}_i^\top B\hat{v}_i) - 2\sum_{j<i} y_{ij}\text{⬢}\hat{v}_j^\top B\hat{v}_i \tag{12}$$

$$\tag{13}$$

where $y_{ij} = \hat{v}_i^\top A\hat{v}_j$ is a lagrange multiplier and ⬢ is the stop gradient operator from Gemp et al. (2021). We show how this pseudo-utility relates to the familiar projection deflation method in appendix C

In order to fully define our algorithm as a game, we must additionally define a strategy space for our k players who select v_i from the space of real vectors $\mathbb{R}^d$ where d is the dimensionality of A and B in order to maximise the utilities defined in equation 43. Similarly to previous work Gemp et al. (2022), we can prove that the top-k eigenvectors of equation 7 coincide with the equilibrium of this game.

Theorem 2.1. *Assuming the top-k generalized eigenvalues of the generalized eigenvalue problem defined in equation 7 are positive and distinct, their corresponding generalized eigenvectors form the unique, strict Nash equilibrium of $\delta - EigenGame$. The proof follows from the game definition and the lemma in Appendix A which shows that v_i is the unique correct response to correctly learnt v_{-i} as in Gemp et al. (2022).*

Analagously to previous work (Gemp et al., 2020), the shape of the utility functions of each players can also be described in terms of an angular deviation from the optimum by defining $\hat{v}_i = m(\cos(\theta_i)\, v_i + \sin(\theta_i)\, \Delta_i)$ where $m = \hat{v}_i^\top B\hat{v}_i$:

$$u_i\left(\hat{v}_i, v_{j<i}\right) = u_i\left(mv_i, v_{j<i}\right) - \sin^2(\theta_i)(u_i\left(mv_i, v_{j<i}\right) - u_i\left(m\Delta_i, v_{j<i}\right)) \tag{14}$$

This result is proved in appendix B.

2.2 Complexity and Implementation

Like previous variants of EigenGame, our algorithm has a per-iteration cost of $\mathcal{O}(bdk^2)$ and can similarly leverage parallel computing in both the model components (players) and data to achieve a theoretical complexity of $\mathcal{O}(dk)$ (Gemp et al., 2022). Furthermore, and unlike the γ-EigenGame variant which can be applied to GEPs our algorithm has just one hyperparameter (learning rate) which makes optimisation extremely simple. We provide pseudo-code for δ-EigenGame in algorithm 1.

3 A Deep EigenGame for Deep CCA

Gemp et al. (2020) noted that the objectives of their players were all generalized inner products which should extend to general function approximators. However, it was unclear how to translate the constraints in the problem to the neural network setting (and likewise for any EigenGame form

Algorithm 1 δ-EigenGame

Input: Stream of data with minibatch size m $\left(X_t \in \mathbb{R}^{m \times d_x}, Y_t \in \mathbb{R}^{m \times d_y}\right)$, learning rate η
for $t = 1$ **to** T **do**
  for $i = 1$ **to** k **do**
    Construct unbiased estimates for A and B
    Rewards $\leftarrow 2A\hat{v}_i$
    Penalties $\leftarrow \sum_{j \leq i} B\hat{v}_j \hat{v}_j^\top A\hat{v}_i$
    $\tilde{\nabla}_i \leftarrow$ Rewards - Penalties
    $\hat{v}_i' \leftarrow \hat{v}_i + \eta_t \tilde{\nabla}_i$
  end for
end for

which is optimized on a manifold). In contrast our work is constraint free but can still be written completely as generalized inner products for certain GEPs. To see this, consider the δ-EigenGame utilities in inner product form:

$$u_i^\delta = 2\langle \hat{v}_i, A\hat{v}_i \rangle - \langle \hat{v}_i, B\hat{v}_i \rangle \langle \hat{v}_i, A\hat{v}_i \rangle - 2\sum_{j<i} \langle \hat{v}_i, B\hat{v}_j \rangle \langle \hat{v}_j, A\hat{v}_i \rangle \tag{15}$$

Now noting the block structure of CCA in equation 5, we can separate these inner products into functions of X and Y. Denoting the subset of the vector $\hat{v}_i$ applied to X as $\hat{v_{x}}_i$ and $\hat{v_{y}}_i$ for the subset applied to Y we can define:

$$\langle \hat{v}_i, A\hat{v}_j \rangle = \langle X\hat{v_{x}}_i, Y\hat{v_{y}}_j \rangle + \langle X\hat{v_{x}}_j, Y\hat{v_{y}}_i \rangle \tag{16}$$

$$\langle \hat{v}_i, B\hat{v}_j \rangle = \langle X\hat{v_{x}}_i, X\hat{v_{x}}_j \rangle + \langle Y\hat{v_{y}}_i, Y\hat{v_{y}}_j \rangle \tag{17}$$

Therefore, we can see that the δ-EigenGame for CCA can be fully represented by generalized inner products. We can replace $X\hat{v_{x}}_i$ and $Y\hat{v_{y}}_i$ with general function approximators $Z_x = f(X)$ and $Z_y = g(Y)$ and in this work we will use neural networks to represent these functions. By plugging these into the utility function we have:

$$u_i^\delta = 4Z_{x_i}^\top Z_{y_i} - 2Z_{x_i}^\top Z_{y_i}(Z_{x_i}^\top Z_{x_i} + Z_{y_i}^\top Z_{y_i}) - 2\sum_{j<i}(Z_{x_i}^\top Z_{y_j} + Z_{x_j}^\top Z_{y_i})(Z_{x_i}^\top Z_{x_j} + Z_{y_i}^\top Z_{y_j}) \tag{18}$$

We could set up k neural networks for each view to optimise these utilities but by constructing the matrices $A = Z_x^\top Z_y + Z_y^\top Z_x$ and $B = Z_x^\top Z_x + Z_y^\top Z_y$ we can simplify the sum of all of the players utilities to give the scalar objective in equation 19.

$$U^{\text{dcca-eg}} = 2\,\mathrm{trace}(A) - \mathrm{diag}(B)^\top \mathrm{diag}(A) - 2\,\mathrm{trace}(A^\top(\mathrm{triu}(B)) \tag{19}$$

Where triu is an operator which sets the principal diagonal and lower triangle of a matrix to zero. By differentiating this objective with respect to the neural networks f and g we can use stochastic gradient descent with backpropagation to optimise this utility (as shown in pseudocode in algorithm 3).

Additionally, we can remove the explicit hierarchy to give an objective which resembles the original SGHA defined in equation 20. We refer to these methods as DCCA-EigenGame and DCCA-SGHA respectively.

$$U^{\text{dcca-sgha}} = \mathrm{trace}(A) - \mathrm{trace}(A^\top(B - I)) \tag{20}$$

Algorithm 2 Deep CCA EigenGame

Input: Stream of data with minibatch size m $\left(X_t \in \mathbb{R}^{m \times d_x}, Y_t \in \mathbb{R}^{m \times d_y}\right)$, neural networks $f_\theta(X)$, $g_\psi(Y)$, learning rate η
for $t = 1$ **to** T **do**
  $Z_x \leftarrow f_\theta(X), Z_y \leftarrow g_\psi(Y)$
  $A \leftarrow Z_x^\top Z_y + Z_y^\top Z_x$
  $B \leftarrow Z_x^\top Z_x + Z_y^\top Z_y$
  Rewards $\leftarrow 2\,\mathrm{trace}(A)$
  Penalties $\leftarrow \mathrm{diag}(B)^\top \mathrm{diag}(A) - 2\,\mathrm{trace}(A^\top(\mathrm{triu}(B))$
  $U \leftarrow$ Rewards - Penalties
  $\tilde{\nabla}_f \leftarrow \frac{\partial U}{\partial f}, \tilde{\nabla}_g \leftarrow \frac{\partial U}{\partial g}$
  $\hat{\theta}' \leftarrow \hat{\theta} + \eta \tilde{\nabla}_f, \hat{\psi}' \leftarrow \hat{\psi} + \eta \tilde{\nabla}_g$
end for

4 Related Work

Our work further bridges the Generalized Hebbian Algorithm (Sanger, 1989) and recent work reformulating classical optimization problems by using a game-theory inspired approach (Gemp et al., 2020) which lends itself to parallel computing. Arguably the core idea behind the variants of EigenGame and the Generalized Hebbian Algorithm is to penalize any lack of orthogonality rather than enforce it at each iteration. Furthermore, previous EigenGames introduce an explicit hierarchy of components which allows them to simultaneously find the top-k subspace as well as the top-k eigenvectors. In the stochastic minibatch setting using soft penalties has shown benefits for both a stochastic variant of the Generalized Hebbian Algorithm in Li et al. (2017) (SGHA), itself interpretable as a primal-dual GEP algorithm (Chen et al., 2019), and EigenGame approaches (Gemp et al., 2021). Comparison to γ-EigenGame (Gemp et al., 2022) is particularly relevant as it is the only EigenGame which can solve GEPs. In a sense their method approximates the GEP form in equation 2 whereas our method approximates the form in equation 7. Compared to γ-EigenGame, our method has fewer hyperparameters and can be extended to deep learning for certain GEPs. We give further comparisons to previous EigenGames in appendix D and E.

There has been a particular interest in the stochastic CCA and stochastic PLS problems. Arora et al. (2012; 2016) provide an extensive review of methods for stochastic PLS. They further developed a method for finding the top-k subspace based on Matrix Stochastic Gradient Descent (MSG) with provable guarantees however their experiments show that the stochastic power method, a variant of Oja's algorithm (Oja, 1982) (also referred to as Oja in Gemp et al. (2020) for the PCA problem), obtains similar or better performance empirically. Furthermore, the efficiency of their MSG method is based on rank-constrained incremental SVD (Arora et al., 2012; Brand, 2002) which does not scale to larger minibatches.

There have been several approaches to the top-1 CCA problem in the stochastic setting. Li & Jordan (2021) derive a scaled stochastic gradient descent method which replaces the quotient term in the gradient of the GEP rayleigh quotient and appears similar to the first player of γ-EigenGame. Bhatia et al. (2018) similarly developed an algorithm for the top-1 setting inspired by Oja's algorithm. These approaches cannot be extended to the top-k problem. Arora et al. (2017) extended their MSG method for finding the top-k CCA subspace but with the same drawbacks as the MSG for PLS when using larger minibatches. There have been a few approaches which are inspired by the alternating least squares (ALS) form of CCA (Golub & Zha, 1995) which solve the global problem by block coordinate descent with the subproblems optimised using stochastic methods (Ma et al., 2015; Ge et al., 2016; Xu & Li, 2019; 2021). Finally, Meng et al. (2021) develop an algorithm for streaming CCA which stochastically approximates the principal components of each view in order to stochastically approximate the top-k CCA problem which they refer to as RSG+. The closest methods in motivation and performance to our work are the γ-EigenGame, SGHA, RSG+ and Stochastic Power methods.

With respect to Deep CCA, Wang et al. (2015b) observed that despite the biased gradients, the original Deep CCA objective could still be used in the stochastic setting for large enough minibatches,

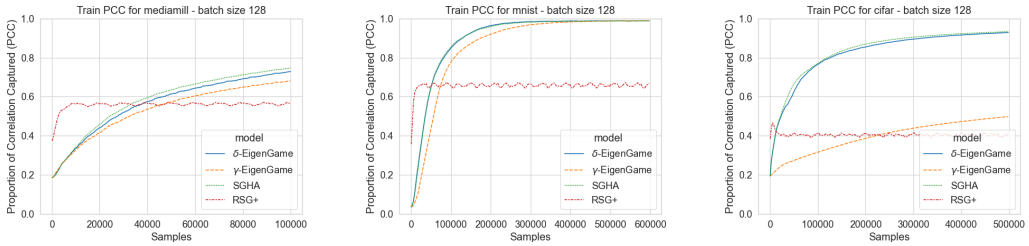

Figure 1: CCA with stochastic minibatches: proportion of correlation captured with respect to Scipy ground truth by δ-EigenGame vs prior work. The maximum value is 1.

a method referred to in the literature as stochastic optimization with large minibatches (DCCA-STOL). Wang et al. (2015c) also built on the ALS form of CCA which adaptively approximated the covariance of the embedding for each view in order to whiten the targets of a regression in each view. This mean square error type loss can then be decoupled across samples in a method called non-linear orthogonal iterations (DCCA-NOI). To the best of our knowledge this method is the current state-of-the-art for Deep CCA optimisation using stochastic minibatches. Deep CCA and self-supervised learning (SSL) are closely related since typical SSL algorithms use an X and Y which are augmented versions of the same image while f and g typically share weights as described in Lyu et al. (2021). We expand on this in appendix H.

5 Experiments

5.1 Stochastic CCA and PLS

In this section we compare δ-EigenGame to previous methods for solving CCA and PLS in the stochastic setting. We use the Scipy (Virtanen et al., 2020) package to solve the population GEPs as a ground truth value. We use the proportion of correlation captured (PCC) and proportion of variance (PV) captured by the learnt subspace as compared to this population ground truth (defined mathematically in appendix F) which both by definition have a maximum value of 1. We additionally use a variant of the number of correct consecutive eigenvectors metric from Gemp et al. (2020; 2021) in order to demonstrate that δ-EigenGame is able to recover both the top-k eigenvectors as well as the top-k subspace. Specifically, we show the number of correct consecutive PLS weight vectors v_x and v_y as measured by cosine similarity greater than 0.95.

For both the CCA and PLS problem we optimise for the top-8 eigenvectors for the MediaMill, Split MNIST and CIFAR datasets (replicating the experiments from Gemp et al. (2022); Meng et al. (2021) with double the number of components). The MediaMill dataset (Snoek et al., 2006) was originally used for evaluating stochastic CCA by Arora et al. (2017) while the latter two datasets are formed from left and right halves of the canonical datasets (LeCun et al., 2010; Krizhevsky et al., 2009). With the same initialization for all methods, we trained for 10 epochs on each dataset with a batch size of 128 and illustrate the models with the best performance in the validation set. We give further details on the hyperparameters in appendix F.3.

5.1.1 CCA with stochastic minibatches

In this experiment we compare our method to RSG+, γ-EigenGame, and SGHA for the stochastic CCA problem.

Figure 1 shows that for all three datasets, our method exhibits faster convergence on a per-iteration basis as well as higher PCC with the exception of SGHA for the MNIST data. Our method generally performs comparably to SGHA algorithm which is consistent with their close theoretical relationship, however δ-EigenGame recovers ordered vectors in addition to the subspace (see appendix I.1).

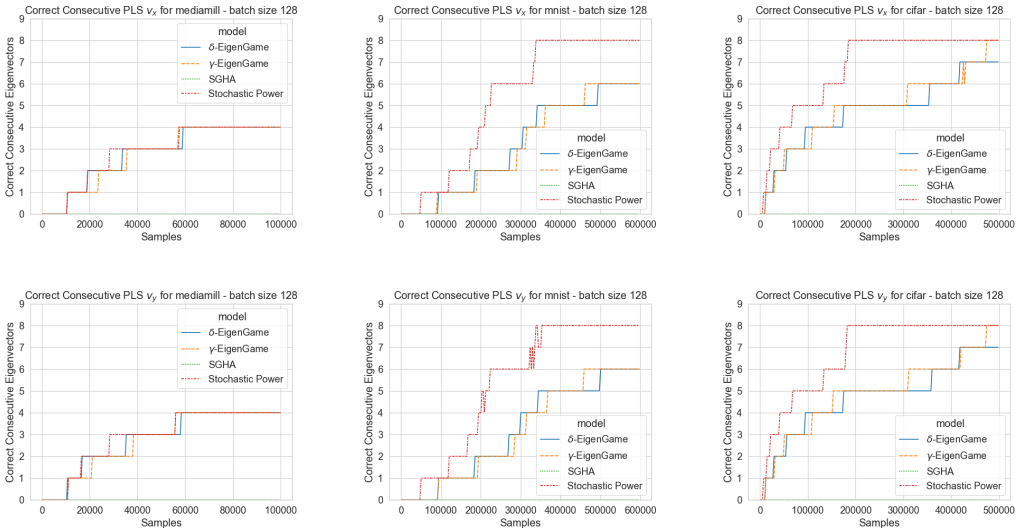

Figure 3: PLS with stochastic minibatches: correct consecutive weights v_x (top) and v_y (bottom) with respect to Scipy ground truth by δ-EigenGame vs prior work. The maximum value is 8.

5.1.2 PLS WITH STOCHASTIC MINIBATCHES

In this experiment we compare our method to the Stochastic Power method, γ-EigenGame, and SGHA for the stochastic PLS problem.

Figure 2 shows that all of the methods perform similarly in terms of variance captured across the datasets. While the stochastic power method is very fast to converge in the MNIST and CIFAR data, it solutions can be suboptimal. The performance of δ-EigenGame is arguably more suprising for the PLS problem because both the Stochastic Power method and γ-EigenGame explicitly enforce the constraint set at each iteration whereas δ-EigenGame only enforces the constraint via penalty terms. For the PLS problem we additionally show in figure 3 that all of the methods except for SGHA are able to extract correct, ordered weight vectors. The strong performance of the stochastic power method on this task likely relates to the enforcement of orthogonality at each iteration and is consistent with observations of the stochastic variant of Oja's algorithm for PCA in Gemp et al. (2020).

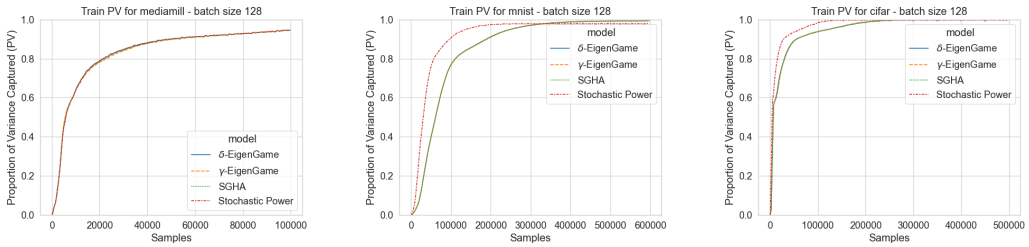

Figure 2: PLS with stochastic minibatches: proportion of variance captured with respect to Scipy ground truth by δ-EigenGame vs comparitors. The maximum value is 1.

5.2 DEEP CANONICAL CORRELATION ANALYSIS WITH STOCHASTIC MINIBATCHES

In this section we compare DCCA-EigenGame and DCCA-SGHA to previous methods for optimizing DCCA in the stochastic setting. We replicated an experiment from Wang et al. (2015c) and

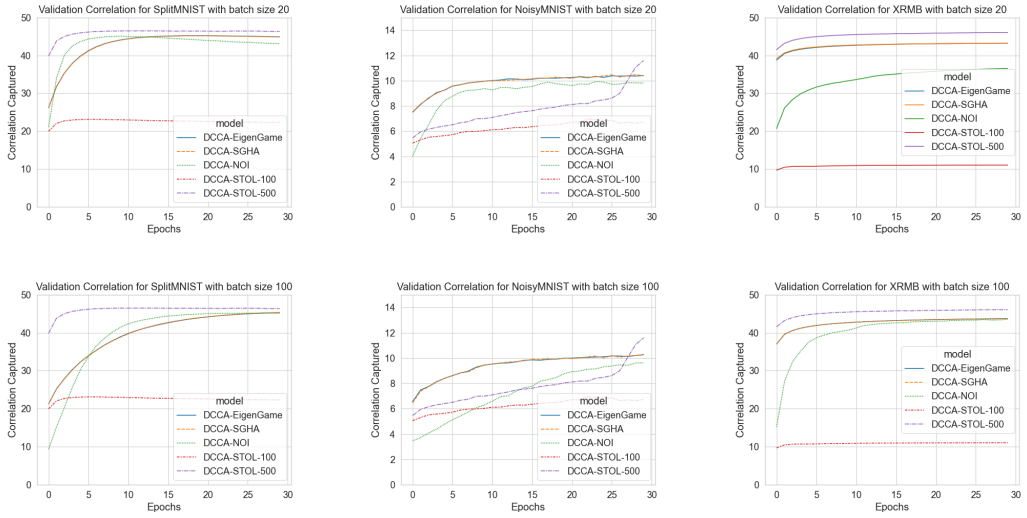

Figure 4: Total correlation captured by the 50 latent dimensions in the validation data. The maximum value is 50. The top row show results for batch size 20 and the bottom row show results for batch size 100

compare our proposed methods to DCCA-NOI and DCCA-STOL. Like the previous work, we use the total correlation captured (TCC) of the learnt subspace as a metric.

We use two variants of paired MNIST datasets. The first is identical to the split MNIST dataset in the previous section. The second harder problem is closely related to an experiment by Wang et al. (2015a). Their 'noisy' paired MNIST data takes two different digits with the same class. The first is rotated randomly while the second has additive gaussian noise. Finally, we use the X-Ray Microbeam (XRMB) dataset from Arora et al. (2016). For all of the datasets, we use two encoders with 50 latent dimensions and two hidden layers with size 800 and leaky ReLu activation functions, a similar architecture to that used in Wang et al. (2015c). We compare our proposed method to DCCA-NOI at batch sizes of 20 and 100 as well as DCCA-STOL with minibatch size 100 and 500 (DCCA-STOL cannot be used for batch sizes less than the number of latent dimensions). We give further details on the hyperparameters in the appendix.

In all three datasets, figure 4 shows that DCCA-EigenGame and DCCA-SGHA find higher correlations in the validation data than all methods except DCCA-STOL with $n = 500$ with typically faster convergence in early iterations compared to DCCA-NOI. It is clear from the experiments that in the deep learning context, the hierarchical DCCA-EigenGame does not outperform DCCA-SGHA. This is likely because the players utilities are coupled through the scalar loss function. In this setup, DCCA-EigenGame is perhaps better understood as a one-player game (Muldoon, 2021) which minimizes subspace error (or maximise variance in the DCCA context) subject to generalized orthogonality penalties.

6 Conclusion

We have presented an alternative game theoretic approach to solving GEPs with no hard constraints. Furthermore, we showed that within our framework, certain GEPs lend themselves to general function approximators and we demonstrated that Deep CCA could be formulated as an EigenGame with state-of-the-art performance.

Future work will apply δ-EigenGame to other practically interesting GEPs like Generalized CCA for more than two views. We will also consider how other deep representation learning approaches can be understood and improved through the δ-EigenGame framework.

REFERENCES

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pp. 1247–1255. PMLR, 2013.

Raman Arora, Andrew Cotter, Karen Livescu, and Nathan Srebro. Stochastic optimization for pca and pls. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 861–868. IEEE, 2012.

Raman Arora, Poorya Mianjy, and Teodor Marinov. Stochastic optimization for multiview representation learning using partial least squares. In *International Conference on Machine Learning*, pp. 1786–1794. PMLR, 2016.

Raman Arora, Teodor Vanislavov Marinov, Poorya Mianjy, and Nati Srebro. Stochastic approximation for canonical correlation analysis. *Advances in Neural Information Processing Systems*, 30, 2017.

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Tom Hennigan, Matteo Hessel, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Lena Martens, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/deepmind`.

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Kush Bhatia, Aldo Pacchiano, Nicolas Flammarion, Peter L Bartlett, and Michael I Jordan. Genoja: Simple & efficient algorithm for streaming generalized eigenvector computation. *Advances in neural information processing systems*, 31, 2018.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision*, pp. 707–720. Springer, 2002.

Claes M Cassel, Peter Hackl, and Anders H Westlund. On measurement of intangible assets: a study of robustness of partial least squares. *Total Quality Management*, 11(7):897–907, 2000.

Zhehui Chen, Xingguo Li, Lin Yang, Jarvis Haupt, and Tuo Zhao. On constrained nonconvex stochastic optimization: A case study for generalized eigenvalue decomposition. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 916–925. PMLR, 2019.

Rong Ge, Chi Jin, Praneeth Netrapalli, Aaron Sidford, et al. Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis. In *International Conference on Machine Learning*, pp. 2741–2750. PMLR, 2016.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame unloaded: When playing games is better than optimizing, 2021.

Ian Gemp, Charlie Chen, and Brian McWilliams. The generalized eigenvalue problem as a nash equilibrium. *arXiv preprint arXiv:2206.04993*, 2022.

Ian M. Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame: PCA as a nash equilibrium. *CoRR*, abs/2010.00554, 2020. URL `https://arxiv.org/abs/2010.00554`.

Gene H Golub and Hongyuan Zha. The canonical correlations of matrix pairs and their numerical computation. In *Linear algebra for signal processing*, pp. 27–49. Springer, 1995.

Justine Hansen, Ross Markello, Jacob Vogel, Jakob Seidlitz, Danilo Bzdok, and Bratislav Misic. Mapping gene transcription and neurocognition across human neocortex. *Nature Human Behaviour*, 5:1–11, 09 2021. doi: 10.1038/s41562-021-01082-z.

Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pp. 162–190. Springer, 1992.

Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Anjali Krishnan, Lynne Williams, Anthony McIntosh, and Hervé Abdi. Partial least squares (pls) methods for neuroimaging: A tutorial and review. *NeuroImage*, 56:455–75, 05 2011. doi: 10.1016/j.neuroimage.2010.07.034.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database, 2010.

Chris Junchi Li and Michael I Jordan. Nonconvex stochastic scaled-gradient descent and generalized eigenvector problems. *arXiv preprint arXiv:2112.14738*, 2021.

Xingguo Li, Zhehui Chen, Lin Yang, Jarvis Haupt, and Tuo Zhao. Online generalized eigenvalue decomposition: Primal dual geometry and inverse-free stochastic optimization. In *the 10th NIPS Workshop on Optimization for Machine Learning*, 2017.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Qi Lyu, Xiao Fu, Weiran Wang, and Songtao Lu. Understanding latent correlation-based multi-view learning and self-supervision: An identifiability perspective. In *International Conference on Learning Representations*, 2021.

Zhuang Ma, Yichao Lu, and Dean Foster. Finding linear structure in large datasets with scalable canonical correlation analysis. In *International conference on machine learning*, pp. 169–178. PMLR, 2015.

Zihang Meng, Rudrasis Chakraborty, and Vikas Singh. An online riemannian pca for stochastic canonical correlation analysis. *Advances in Neural Information Processing Systems*, 34:14056–14068, 2021.

Conor Muldoon. Deepmind's eigengame is not a game!, 2021. URL `https://towardsdatascience.com/deepminds-eigengame-is-not-a-game-f580b3709316`.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, pp. 421–430, 2006.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International conference on machine learning*, pp. 1083–1092. PMLR, 2015a.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff A Bilmes. Unsupervised learning of acoustic features via deep canonical correlation analysis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4590–4594. IEEE, 2015b.

Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 688–695. IEEE, 2015c.

Svante Wold, Arnold Ruhe, Herman Wold, and WJ Dunn, Iii. The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

Zhiqiang Xu and Ping Li. Towards practical alternating least-squares for cca. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhiqiang Xu and Ping Li. On the faster alternating least-squares for cca. In *International Conference on Artificial Intelligence and Statistics*, pp. 1621–1629. PMLR, 2021.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

A NASH PROOF

Lemma A.1. *Given exact parents and assuming the top-k generalized eigenvalues of A and B are distinct and positive, the maximizer of player i's utility is the unique generalized eigenvector v_i (up to sign).*

Proof. For ease of reading the proofs in this appendix, we slightly change notation, and index the normalised solutions to the GEV with superscripts:

$$\langle v^{(i)}, Bv^{(i)}\rangle = 1, \quad \langle v^{(i)}, Av^{(i)}\rangle = \lambda^{(i)}\ \forall i,$$

while we continue to index our estimates with subscripts. We can write our estimates in this basis to define the coefficients $\hat{v}_i = \sum_p w_i^{(p)} v^{(p)}$. Next define

$$m_i = \sum_p (w_i^{(p)})^2 \tag{21}$$

$$z_i^{(j)} = \frac{(w_i^{(j)})^2}{\sum_p (w_i^{(p)})^2} = \frac{(w_i^{(j)})^2}{m_i} \tag{22}$$

so that the vector $z_i = (z_i^{(j)})_j$ takes values in the simplex. Then we have:

$$\langle \hat{v}_i, Av^{(j)}\rangle = \lambda^{(j)} w_i^{(j)} \tag{23}$$

$$\langle \hat{v}_i, A\hat{v}_i\rangle = \sum_p \lambda^{(p)} (w_i^{(p)})^2 \tag{24}$$

$$\langle \hat{v}_i, Bv^{(j)}\rangle = w_j \tag{25}$$

$$\langle \hat{v}_i, B\hat{v}_i\rangle = \sum_p (w_i^{(p)})^2 = m_i \tag{26}$$

Consider the utility function for player i:

$$u_i(\hat{v}_i|\hat{v}_{j<i}) = 2\langle \hat{v}_i, A\hat{v}_i\rangle - \langle \hat{v}_i, B\hat{v}_i\rangle\langle \hat{v}_i, A\hat{v}_i\rangle - 2\sum_{j<i}\langle \hat{v}_i, B\hat{v}_j\rangle\langle \hat{v}_j, A\hat{v}_i\rangle \tag{27}$$

$$= 2\sum_p \lambda^{(p)} w_i^{(p)2} - 2\sum_{j<i}\lambda^{(j)} w_j^2 - (\sum_p \lambda^{(p)} w_i^{(p)2})(\sum_p w_i^{(p)2}) \tag{28}$$

$$= 2\sum_{j\geq i}\lambda^{(j)} w_j^2 - (\sum_p \lambda^{(p)} w_i^{(p)2})(\sum_p w_i^{(p)2}) \tag{29}$$

$$= (2m_i - m_i^2)\sum_{j\geq i}\lambda^{(j)} z_i^{(j)} - m_i^2 \sum_{j<i}\lambda^{(j)} z_i^{(j)} \tag{30}$$

Which is maximised when $m_i = 1$ $z_i^{(j)} = \delta_{ij}$, which implies that $w_i = \pm 1$ and $w_{j\neq i} = 0$. □

Note that in the previous lemma, the utility function took a simple form when we chose the true generalised eigenvectors as a basis; indeed, when using coefficients with respect to the basis, the utility only depended on the generalised eigenvalues and not the basis itself. This simple form shows how our utility interacts naturally with the geometry of the GEV problem. We will now analyse the corresponding simple form of the update steps.

Lemma A.2. *Given exact parents and assuming the top-k generalized eigenvalues of A and B are distinct and positive, the only stable stationary point of the iteration defined by 11 eigenvector v_i (up to sign).*

Proof. Let $v^{(j)}, w_i^{(j)}$ be defined as in the proof of Lemma A.1. Then the update

$$\begin{aligned}
\Delta_i^\delta &= A\hat{v}_i - B\hat{v}_i\left(\hat{v}_i^\top A\hat{v}_i\right) - \sum_{j<i} B\hat{v}_j\left(\hat{v}_j^\top A\hat{v}_i\right) \\
&= A\hat{v}_i - \sum_{j\leq i} B\hat{v}_j\left(\hat{v}_j^\top A\hat{v}_i\right) \\
&= \sum_p \lambda^{(p)} w_i^{(p)} B v^{(p)} - \sum_{j\leq i}\left(\sum_p w_j^{(p)} B v^{(p)}\right)\left(\sum_q \lambda^{(q)} w_i^{(q)} w_j^{(q)}\right) \\
&= \sum_p B v^{(p)}\left\{\lambda^{(p)} w_i^{(p)} - \sum_{j\leq i} w_j^{(p)}\left(\sum_q \lambda^{(q)} w_i^{(q)} w_j^{(q)}\right)\right\}
\end{aligned}$$

If we define the diagonal matrix $\Lambda = \text{diag}(\{\lambda^{(p)}\}_p)$ and the vector $w_i = (w_i^{(p)})_{p=1}^d$ then we can rewrite:

$$\left(\sum_q \lambda^{(q)} w_i^{(q)} w_j^{(q)}\right) = w_i^T \Lambda w_j.$$

Now by orthonormality of the $v^{(p)}$, we can equate coefficients and combine to vector form to obtain the update step for the coefficient vectors, which we shall notate by

$$\Delta(w)_i = \Lambda w_i - \sum_{j\leq i} w_j (w_j^T \Lambda w_i) \tag{31}$$

$$= \left(I - \sum_{j\leq i} w_j w_j^T\right)\Lambda w_i \tag{32}$$

$$= \left(I - \sum_{j<i} w_j w_j^T\right)\Lambda w_i - w_i(w_i^T \Lambda w_i) \tag{33}$$

Only now do we consider the assumption of exact parents. This corresponds to $w_j = e_j \forall j < i$ where e_j is the jth unit vector ($(e_j)_k = \delta_{jk}$). Then

$$\left(I - \sum_{j<i} w_j w_j^T\right) = \begin{pmatrix} 0_{(i-1)\times(i-1)} & 0_{(i-1)\times(i-1)} \\ 0_{(d-i+1)\times(i-1)} & I_{(d-i+1)\times(d-i+1)} \end{pmatrix}$$

So if we write $\bar{\lambda}_i = w_i^T \Lambda w_i$ the update equations become:

$$\Delta(w)_i^{(p)} = -\bar{\lambda}_i w_i^{(p)} \quad \text{for } p < i \tag{34}$$

$$\Delta(w)_i^{(p)} = (\lambda^{(p)} - \bar{\lambda}_i) w_i^{(p)} \quad \text{for } p \geq i \tag{35}$$

So if we observe the qualitative behaviour:

- For $p < i$ the coefficients are shrunk towards zero (for sufficiently small step sizes)
- For $p \geq i$ the coefficients grow/decay depending on their generalised eigenvalue. The larger the eigenvalue, the more the components grow / the less they shrink. So over time only the ith component will be selected.
- The overall magnitude of the solution shrinks faster when $\bar{\lambda}_i$ is large.

A stationary point of the iteration therefore requires $w_i^{(p)} = 0$ for $p < i$. Then for each $p \geq i$ we must have either $\bar{\lambda}_i = \lambda^{(p)}$ or $w_i^{(p)} = 0$. Furthermore, $w_i^{(i)}$ grows at a faster rate than any of the other components, so provided this was non-zero at initialisation, it will be extracted uniquely. Finally, if $w_i^{(p)} = 0 \forall p \neq i$ but $w_i^{(i)} \neq 0$ then we must have $\lambda^{(i)} = \bar{\lambda}_i$ and also $\bar{\lambda}_i = \lambda^{(i)} (w_i^{(i)})^2$; so combining with we get $w_i^{(i)} = \pm 1$, as required. □

We expect it to be possible to conduct an analysis of error propagation by considering the updates of (33) more carefully. Then an extension to the stochastic / mini-batch setting should follow from the literature on stochastic approximation; this should closely mirror the analysis in Gemp et al. (2021), but our algorithm is not restricted to the unit sphere, so we can use standard literature on stochastic approximation instead of the less-standard techniques of Riemannian stochastic approximation; this is another advantage of a 'soft' variance penalty over a hard constraint.

B Utility Shape

Lemma B.1. *Let* $\hat{v}_i = m(\cos(\theta_i) v_i + \sin(\theta_i) \Delta_i)$ *where* $\hat{v}_i^\top B \hat{v}_i = m$*, then:*

$$u_i(\hat{v}_i, v_{j<i}) = u_i(m v_i, v_{j<i}) - \sin^2(\theta_i)(u_i(m v_i, v_{j<i}) - u_i(m \Delta_i, v_{j<i})) \tag{36}$$

We will show that this result follows from similar logic to Gemp et al. (2020) once the scaling factor m is accounted for.

Proof. Let $\Delta_i = \sum_{l=1}^d w_l v_l, \|w\| = 1$. Decomposing the utility function for player i we have:

$$
\begin{aligned}
u_i\left(\hat{v}_i, v_{j<i}\right) &= 2\langle \hat{v}_i, A\hat{v}_i\rangle - \langle \hat{v}_i, B\hat{v}_i\rangle\langle \hat{v}_i, A\hat{v}_i\rangle - 2\sum_{j<i}\langle \hat{v}_i, Bv^{(j)}\rangle\langle v^{(j)}, A\hat{v}_i\rangle && (37)\\
&= (2m - m^2)(\cos^2(\theta_i)\lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \lambda\Delta_i\rangle)\\
&\quad - 2m\sum_{j<i}\left\langle \cos\left(\theta_i\right) v_i + \sin\left(\theta_i\right)\Delta_i, Av_j\right\rangle\left\langle \cos\left(\theta_i\right) v_i + \sin\left(\theta_i\right)\Delta_i, Bv_j\right\rangle && (38)\\
&= (2m - m^2)(\cos^2(\theta_i)\lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \lambda\Delta_i\rangle)\\
&\quad - 2m\sum_{j<i}\sin^2(\theta_i)\left\langle \Delta_i, Av_j\right\rangle\left\langle \Delta_i, Bv_j\right\rangle && (39)\\
&= (2m - m^2)\lambda_{ii}\\
&\quad - (2m - m^2)\sin^2(\theta_i)\lambda_{ii} + \sin^2(\theta_i)((2m - m^2)\langle \Delta_i, A\Delta_i\rangle\\
&\quad - 2m\sum_{j<i}\left\langle \Delta_i, Av_j\right\rangle\left\langle \Delta_i, Bv_j\right\rangle && (40)\\
&= (2m - m^2)\lambda_{ii}\\
&\quad - \sin^2(\theta_i)((2m - m^2)\lambda_{ii} + (2m - m^2)\langle \Delta_i, A\Delta_i\rangle - 2m\sum_{j<i}\left\langle \Delta_i, Av_j\right\rangle\left\langle \Delta_i, Bv_j\right\rangle) && (41)\\
&= u_i\left(mv_i, v_{j<i}\right) - \sin^2(\theta_i)(u_i\left(mv_i, v_{j<i}\right) - u_i\left(m\Delta_i, v_{j<i}\right)) && (42)
\end{aligned}
$$

□

C PSEUDO-UTILITY AS A PROJECTION DEFLATION

$$
u_i^{\delta} = 2\hat{v}_i^{\top}[\overbrace{I - \sum_{j<i} B\hat{v}_j\hat{v}_j^{\top}}^{\text{projection deflation}}]A\hat{v}_i - \hat{v}_i^{\top}B\hat{v}_i\hat{v}_i^{\top}A\hat{v}_i \tag{43}
$$

Analogously to the previous work in α-EigenGame (Gemp et al., 2020), the matrix $[I - \sum_{j\leq i} B\hat{v}_j\hat{v}_j^{\top}]$ has a natural interpretation as a projection deflation.

D RELATIONSHIP TO μ-EIGENGAME

We can show that the pseudo-utility of our δ-EigenGame is similar to μ-EigenGame where $B = I$ if we explicitly enforce the constraint $\hat{v}_i^{\top}B\hat{v}_i = \hat{v}_i^{\top}\hat{v}_i = 1$ with the only difference being a scaling of the orthogonality penalties. However δ-EigenGame converges to solutions which approximate the constraint in equation 7 by implicitly using the method of Lagrange multipliers.

$$
\begin{aligned}
u_i^{\delta} &= \hat{v}_i^{\top}A\hat{v}_i + \hat{v}_i^{\top}A\hat{v}_i(1 - \hat{v}_i^{\top}B\hat{v}_i) - 2\sum_{j<i}\hat{v}_i^{\top}A\hat{v}_j\hat{v}_j^{\top}B\hat{v}_i && (44)\\
&= \hat{v}_i^{\top}A\hat{v}_i - 2\sum_{j<i}\hat{v}_i^{\top}\hat{v}_j\hat{v}_j^{\top}A\hat{v}_i && (45)\\
u_i^{\mu} &= \hat{v}_i^{\top}A\hat{v}_i - \sum_{j<i}\hat{v}_i^{\top}\hat{v}_j\hat{v}_j^{\top}A\hat{v}_i && (46)
\end{aligned}
$$

E RELATIONSHIP TO γ-EIGENGAME

We can similarly show that the pseudo-utility of our δ-EigenGame is similar to γ-EigenGame if we explicitly enforce the constraint $\hat{v}_i^{\top}B\hat{v}_i = 1$.

$$u_i^\gamma = \hat{v}_i^\top A \hat{v}_i - \sum_{j<i} (\hat{v}_i^\top \hat{v}_j)^2 \hat{v}_j^\top A \hat{v}_j \tag{47}$$

F Experiment Details

F.1 Proportion of Correlation Captured (PCC)

This is the sum of the canonical correlations of the learnt representation using each stochastic optimisation method as a proportion of the sum of the canonical correlations of the learnt representation using the population ground truth (i.e. the sum of the top-k canonical correlations of X and Y).

F.2 Proportion of Variance (PV)

This is the sum of the singular values of the learnt representation using each stochastic optimisation method as a proportion of the sum of the singular values of the learnt representation using the population ground truth (i.e. the sum of the top-k singular values of the covariance matrix $X^\top Y$).

F.3 Experiment Hyperparameters

F.3.1 Stochastic CCA and PLS

Learning rate was tuned from $\eta = (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$ and γ-EigenGame parameter γ was tuned from the same range. We used Jax (Babuschkin et al., 2020) to optimise the linear CCA and PLS models using the Jaxline framework. We used Biewald (2020) for experiment tracking to develop insights for this paper.

F.3.2 Deep CCA

We trained for 30 epochs on each dataset with batch sizes of 20 and 100. Learning rate was tuned from $\eta = (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$ and the DCCA-NOI parameter ρ was tuned between 0 and 1. We use PyTorch (Paszke et al., 2019) with the Adam optimizer (Kingma & Ba, 2014).

G Algorithms

G.1 Deep CCA SGHA

Algorithm 3 Deep CCA EigenGame

Input: Stream of data with minibatch size m $\left(X_t \in \mathbb{R}^{m \times d_x}, Y_t \in \mathbb{R}^{m \times d_y}\right)$, neural networks $f_\theta(X)$, $g_\psi(Y)$, learning rate η
for $t = 1$ **to** T **do**
  $Z_x \leftarrow f_\theta(X), Z_y \leftarrow g_\psi(Y)$
  $A \leftarrow Z_x^\top Z_y + Z_y^\top Z_x$
  $B \leftarrow Z_x^\top Z_x + Z_y^\top Z_y$
  Rewards $\leftarrow \mathrm{trace}(A)$
  Penalties $\leftarrow \mathrm{trace}(A^\top(B - I)))$
  $U \leftarrow$ Rewards - Penalties
  $\tilde{\nabla}_f \leftarrow \frac{\partial U}{\partial f}, \tilde{\nabla}_g \leftarrow \frac{\partial U}{\partial g}$
  $\hat{\theta}' \leftarrow \hat{\theta} + \eta \tilde{\nabla}_f, \hat{\psi}' \leftarrow \hat{\psi} + \eta \tilde{\nabla}_g$
end for

H A connection to self-supervised learning methods

Reorganizing our update equation 11, we find that the intuition of our method can also be understood as three terms encouraging variance in A, penalizing variance in B, and discouraging covariance.

$$u_i^\gamma = \overbrace{2\hat{v}_i^\top A\hat{v}_i}^{\text{Invariance}} - \overbrace{\hat{v}_i^\top B\hat{v}_i \left(\hat{v}_i^\top A\hat{v}_i\right)}^{\text{Variance}} - \overbrace{\sum_{j<i} \hat{v}_i^\top B\hat{v}_j \left(\hat{v}_j^\top A\hat{v}_i\right)}^{\text{Covariance}} \tag{48}$$

The motivation is similar to that in recent work in self-supervised learning (Zbontar et al., 2021) and in particular the VICReg method in Bardes et al. (2021). Indeed, recent work has shown links between several self-supervised learning approaches and classical spectral embedding methods (Balestriero & LeCun, 2022), some of which could be represented by GEPs. Like CCA, many self-supervised learning approaches are based on finding a function which is invariant to an image and its augmented version i.e. the learnt representations of both are correlated.

I STOCHASTIC CCA AND PLS WITH SMALLER BATCH SIZES

In this section we repeat the experiments described in the main text with smaller batch sizes (64 and 32). We also demonstrate the correct consecutive weight vector metric at different batch sizes for the CCA problem on the MNIST data.

I.1 CCA

For the CCA problem, results for batch sizes 32 and 64 are broadly similar to those in the main text for batch size 128. In the MNIST data we can see again that there is a tradeoff between speed of convergence in early iterations and the quality of the solution.

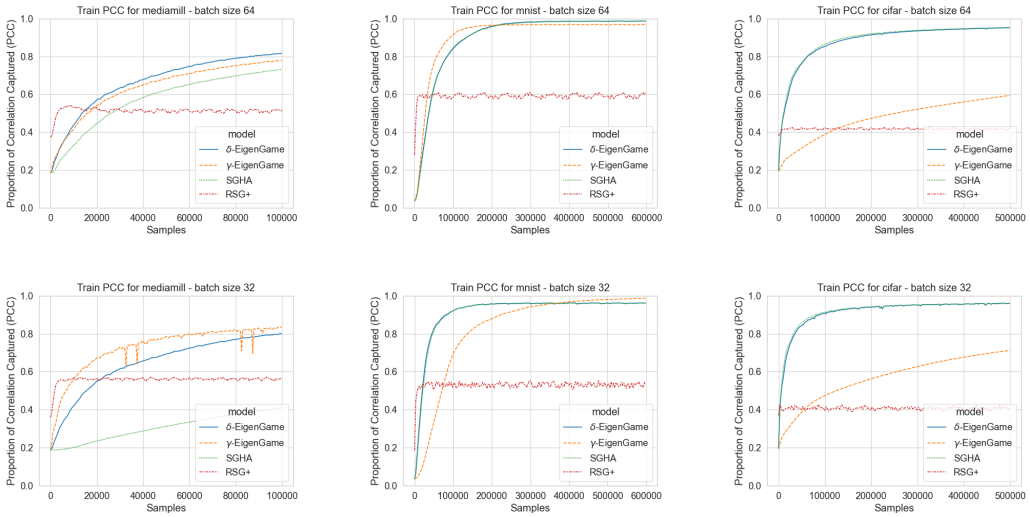

Figure 5: CCA with stochastic minibatches of size 32: proportion of correlation captured with respect to Scipy ground truth by δ-EigenGame vs prior work. The maximum value is 1.

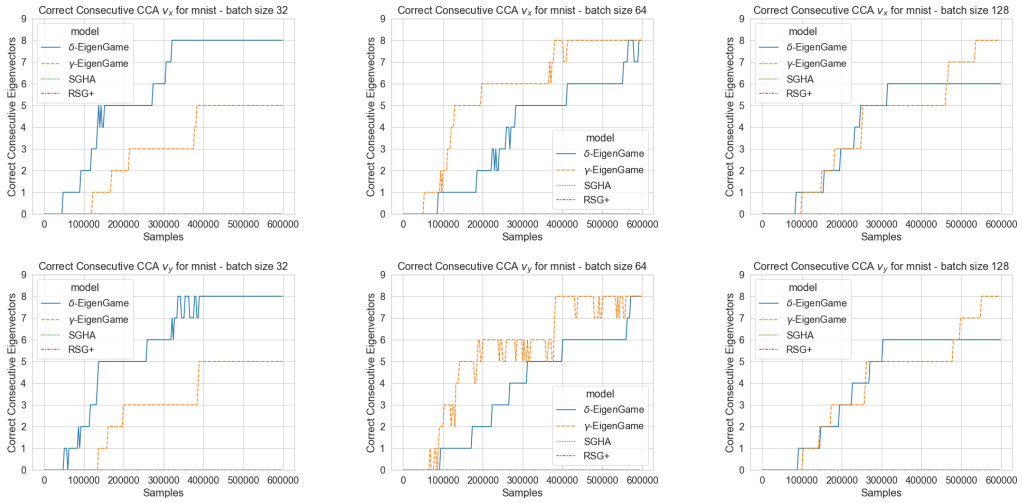

Figure 6: CCA with stochastic minibatches: correct consecutive weights v_x (top) and v_y (bottom) with respect to Scipy ground truth by δ-EigenGame vs prior work. The maximum value is 8.

I.2 PLS

For the PLS problem, results for batch sizes 32 and 64 are broadly consistent with those in the main text for batch size 128.

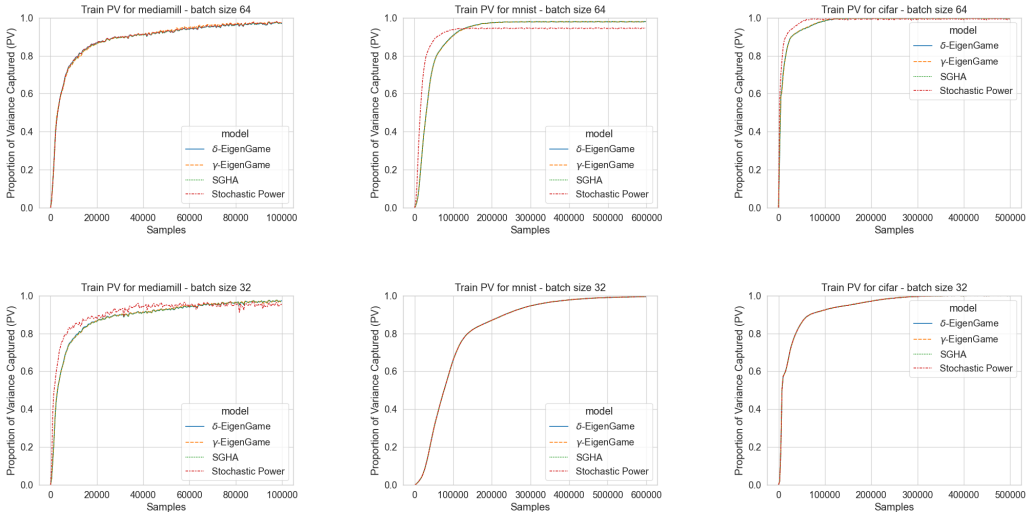

Figure 7: PLS with stochastic minibatches of size 32: proportion of variance captured with respect to Scipy ground truth by δ-EigenGame vs comparitors. The maximum value is 1.