Learning Macro Variables with Auto-encoders

Maitreyi Swaroop*
Department of Mathematics
Indian Institute of Technology
Kharagpur,
maitreyi.swaroop@mila.quebec

Eric Elmoznino
Department of Computer Science
Mila & Université de Montréal
Montréal, QC H2S 3H1
eric.elmoznino@gmail.com

Dhanya Sridhar
Department of Computer Science
Mila & Université de Montréal
Montréal, QC H2S 3H1
dhanya.sridhar@mila.quebec

Abstract

Most causal variables that we reason over, in both science and everyday life, are coarse abstractions of low-level data. However, despite their importance, the field of causality lacks a precise theory of abstract "macro" variables and their relation to low-level "micro" variables that can account for our intuitions. Here, we define a macro variable as something that (a) is simpler than its micro variable, (b) shares mutual information with its micro variable, and (c) is related to other macro variables via simple mechanisms. From this definition, we propose DeepCFL: a simple self-supervised method that learns macro variables and their relations. We empirically validate DeepCFL on synthetic tasks where the underlying macro variables are known, and find that they can be recovered with high fidelity. Given that the individual components of DeepCFL leverage standard and scalable techniques in deep learning, our preliminary results are encouraging signs that it can be successfully applied to real-world data.

1 Introduction

How does the kinetic energy of particles that make up an object affect our sensory neurons when we touch these objects? We might notice that as the mean kinetic energy of the particles – call this temperature – increases past a particular value, our sensory neurons send signals of a particular configuration – call this pain. To summarize this relationship, we might say "high temperatures cause pain when touched." Like with this example, often in science, the causal relationship between one complex system and another is sufficiently captured in terms of coarse variables that we define.

Building on such long-standing intuitions, Chalupka et al. [2017] formalized the process of abstracting such coarse summaries – called macro variables – from low-level measurements – called micro variables – with a clustering-based approach termed causal feature learning (CFL). Concurrently, in the field of deep learning, representation learning offers an alternative for compressing observed variables into macro variables. In representation learning, the goal is to learn "generative factors" that explain all of the observed variation.

CFL and representation learning output qualitatively different macro variables because of their differing objectives. On one hand, representation learning seeks to reconstruct the observations

*Corresponding author

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

perfectly and thus typically recovers macro variables that maximize the mutual information with the observed micro variables. In contrast, CFL clusters micro variables that have the same effect on a (potentially complex) outcome, constructing discrete macro variables that contain significantly less information than the observations. On the other hand, interpreting the outputs of CFL is as hard as deciphering which observed features a clustering algorithm focused on to produce each cluster. In contrast, we can interpret variables inferred via representation learning by perturbing each one and reconstructing the corresponding observation. The overarching goal of abstracting complex systems may be best served if we could combine the differing advantages of CFL and deep learning: focusing only on causal explanations versus on reconstructing low-level signals, admitting only discrete macro variables or offering more generality, being less or more interpretable.

In this paper, we take the perspective that causal feature learning offers a useful definition of macro variables, but propose a learning algorithm that extends standard representation learning approaches. In doing so, we equip CFL with three new advantages: going beyond discrete macro variables, learning complex macro variables that are nonlinear functions of their micro variables, and directly interpreting macro variables via reconstruction. The practical contribution of this paper is a method we call DeepCFL, an approach to learning macro variables that contain information about the observed micro variables, but which also satisfy a key property of CFL: the macro variables output by DeepCFL only capture enough information necessary to capture relationships between the space of possible causes and the space of possible effects.

DeepCFL works by translating the desiderata of CFL into a differentiable constrained objective that can be optimized using gradient-based approaches. DeepCFL learns to encode two sets of micro-level observations (the "cause space" and the "effect space" in the parlance of CFL) into macro variables so that the "cause" macro variables explain the "effect" ones in terms of simple functions [2]. Sparse linear functions are one such choice we explore in this work. The preference for simple explanations guides the encoders to find macro variables that are summary functions of micro variables, just as temperature and the concept of pain signaled by the brain offer simple explanations that would not be possible to express in the low-level space. We constrain the prediction objective to find encodings of macro variables that have some mutual information with the micro variables. Practically, this constraint prevents the encoders from collapsing to trivial solutions.

In our preliminary experiments, we implement the DeepCFL objective by extending β-variational auto-encoders (β-VAEs) [Higgins et al., 2016], which approximate mutual information between observations and latents based on a modified evidence lower-bound. We evaluate DeepCFL on tasks that explore the relationship between English and Kannada MNIST digits where we know the correct macro variables, conducting ablation studies to study the impact of each component in DeepCFL. We compare DeepCFL to the vanilla β-VAEs and show that the components of DeepCFL are important for learning correct causal variables.

2 Background

We start by establishing some notation, and then reviewing CFL and representation learning in terms of the notation we introduce.

Notation. We consider two matrices of observations termed $\boldsymbol{Y}_l$ and $\boldsymbol{X}_l$. Each observation in $\boldsymbol{X}_l$, called a micro variable, is a vector $\boldsymbol{x}_l \in \mathcal{X}^{d_x}$ (and correspondingly, $\boldsymbol{y}_l \in \mathcal{Y}^{d_y}$). The goal is to learn encoding functions $\boldsymbol{x}_h = g_x(\boldsymbol{x}_l)$ and $\boldsymbol{y}_h = g_y(\boldsymbol{y}_l)$ that map micro variables to latent macro variables $\boldsymbol{y}_h$ and $\boldsymbol{x}_h$ that satisfy some desiderata.

Causal feature learning. In prior work [Chalupka et al., 2017], a macro variable $\boldsymbol{x}_h \in \{1 \dots C\}$ that captures one of C categories, or equivalence classes, satisfies the desiderata that,

$$\boldsymbol{x}_h = g_x(\boldsymbol{x}_l) = g_x(\boldsymbol{x}'_l) \Leftrightarrow p(\boldsymbol{y}_l|\boldsymbol{x}_l) = p(\boldsymbol{y}_l|\boldsymbol{x}'_l), \text{ for all } \boldsymbol{y}_l \in \mathcal{Y}^{d_y}. \tag{1}$$

Similarly, a macro variable $\boldsymbol{y}_h \in \{1 \dots E\}$ captures one of E equivalence classes so that,

$$\boldsymbol{y}_h = g_y(\boldsymbol{y}_l) = g_y(\boldsymbol{y}'_l) \Leftrightarrow p(\boldsymbol{y}_l|\boldsymbol{x}_l) = p(\boldsymbol{y}'_l|\boldsymbol{x}_l), \text{ for all } \boldsymbol{x}_l \in \mathcal{X}^{d_x}. \tag{2}$$

The causal feature learning (CFL) algorithm described in Chalupka et al. [2017] first estimates the conditional density $p(\boldsymbol{y}_l|\boldsymbol{x}_l)$ and then performs clustering based on the criteria above to output macro

[2] Although CFL discusses "cause" and "effect" spaces, the algorithm can be used to explain associations rather than causal relationships between micro variables.

variables. At a higher level, in CFL, macro variables explain all the variation in the distribution $p(\boldsymbol{y}_l|\boldsymbol{x}_l)$, both in terms of its outputs and its inputs. The learning algorithm outputs variables that stand in bijective relation to the state space of the true discrete macro variables, assuming accurate density estimation.

Representation learning. A macro variable $\boldsymbol{x}_h \in \mathcal{H}^{d_m}$, where $d_m \leq d_x$, output by a representation learning procedure (approximately) satisfies the property that,

$$\boldsymbol{x}_l = f \circ g_x(\boldsymbol{x}_l), \tag{3}$$

where $f : \mathcal{H}^{d_m} \to \mathcal{X}^{d_x}$ is a decoding function that maps macro variable values to their corresponding micro variable state. The macro variable $\boldsymbol{y}_h$ is defined analogously, with respect to $\boldsymbol{y}_l$. Notably, representation learning methods aim to compress the variation in each set of observations independently, not seeking to explain the modes of $p(\boldsymbol{y}_l|\boldsymbol{x}_l)$ as CFL does. The decoding function $f(\cdot)$ also provides a way of interpreting what macro variables $\boldsymbol{x}_h$ or $\boldsymbol{y}_h$ "mean": perturb one dimension, decode the new latent to an observation, and use domain knowledge to interpret what that axis of change means. Although representation learning procedures do not place restrictions on $\mathcal{H}^{d_m}$, to leverage auto-differentiation tools to optimize Eq. 3, we typically work with $\boldsymbol{x}_h \in \mathbb{R}^{d_m}$.

β-VAE. One approach to representation learning, which we extend in this work, is the β-VAE, introduced by Higgins et al. [2016]. For micro variables $\boldsymbol{X}_l$, a β-VAE assumes that the macro variable $\boldsymbol{x}_h \sim g_x(\boldsymbol{x}_l)$ is a stochastic function parameterized by the neural network $q_\phi(\cdot|\boldsymbol{x}_l)$. The decoding function $f(\boldsymbol{x}_h)$ with parameters θ in a β-VAE is another neural network that parameterizes $p_x(\boldsymbol{x}_l|\boldsymbol{x}_h)$. β-VAEs implement Eq. 3 by maximizing the evidence lower bound (ELBO), a lower bound to $\log p_x(\boldsymbol{x}_l)$. It is,

$$ELBO(\phi, \theta) = \mathbb{E}[p(\boldsymbol{x}_l|\boldsymbol{x}_h; \theta)] - \beta D_{KL}(q_\phi(\boldsymbol{x}_h|\boldsymbol{x}_l), p(\boldsymbol{x}_h)), \tag{4}$$

where $D_{KL}(\cdot)$ is the Kullback-Liebler (KL) divergence and $p(\boldsymbol{x}_h))$ is a prior on the macro variables. Given $\boldsymbol{x}_h \in \mathbb{R}^{d_m}$, a standard choice for the prior is a multivariate standard Normal distribution.

In the next section, we introduce DeepCFL, a differentiable approach to learning macro variables that combines the criteria underlying CFL with the flexibility of representation learning.

3 The DeepCFL approach

To introduce DeepCFL, we first lay out intuitions about what makes a good causal macro variable. From these, we then derive the different components of our model and training objective.

A macro variable is simpler than its micro variable. By definition, a macro variable should be a more abstract, coarse representation of its micro variable. Chalupka et al. [2017] frame macro variables in CFL as discrete clusters of micro variables. While this can be true in particular situations, we argue that in the general case a macro variable must simply (a) be a function of a micro variable and (b) have lower representational complexity than it. Note that this subsumes the definition provided in Chalupka et al. [2017], but also includes many other possibilities, such as continuous macro variables with lower dimensionality than their micro variables. For instance, temperature is a 1-dimensional continuous macro variable who's value is a function of a high-dimensional micro variable (particle velocities).

To satisfy this requirement in DeepCFL, we make each macro variable a function of its micro variable, giving us $\boldsymbol{x}_h = g_x(\boldsymbol{x}_l)$ and $\boldsymbol{y}_h = g_y(\boldsymbol{y}_l)$. Importantly, we parameterize g_x and g_y using deep neural networks (DNNs), given that they are universal function approximators and that we do not wish to constrain the space of possible macro variables that our method can discover. To ensure that the macro variables are simpler than their micro variables, we limit the output dimensionality of g_x and g_y.

A macro variable shares mutual information with its micro variable While macro variables are simple and abstract, they must nevertheless describe *something* about their micro variables. For instance while temperature is a single summary statistic of particle motion, knowing the temperature nevertheless drastically reduces the space of possible particle states – in other words, the mutual information $I(\boldsymbol{X}_h; \boldsymbol{X}_l) > 0$ and $I(\boldsymbol{Y}_h; \boldsymbol{Y}_l) > 0$.

In deep learning, common approaches to maximizing the mutual information between an input variable and its representation include (a) reconstruction [e.g., Kingma and Welling, 2013] and (b) self-supervised learning [e.g., Chen et al., 2020, Grill et al., 2020]. To avoid specifying *ad hoc* data augmentations that are typically required in self-supervised learning, in DeepCFL, we build on the β-VAE, which Alemi et al. [2018] showed is a variational approximation to $I(\boldsymbol{X}_h; \boldsymbol{X}_l)$. We use the objective in Eq. 4 to simultaneously learn the macro variables $\boldsymbol{x}_h \sim g_x(\boldsymbol{x}_l)$ and $\boldsymbol{y}_h \sim g_y(\boldsymbol{y}_l)$.

Macro variables are related by a simple mechanism. Part of what makes abstractions useful is that mechanisms at the high-level are simpler and more interpretable than those at the low-level. For instance, there is certainly a relationship between the kinetic energy of all the particles that make up an object and the activity of our sensory neurons when we touch it. However, it would be impossible for a human to describe this relationship in its entirety at the low-level. At the same time, not all possible abstractions are useful. For instance, we don't construct concepts for "the variance in particle kinetic energy" and "the variance of sensory neural activity" because there is either (a) no reliable predictive relationship between these variables or (b) a complex relationship. We therefore require that macro variables be related by simple and predictive mechanisms (e.g., temperature and pain). This requirement extends the definition provided by Chalupka et al. [2017] (Eqs. 1 and 2).

To satisfy this requirement in DeepCFL, we model a mechanism $\hat{\boldsymbol{y}}_h = f_{xy}(\boldsymbol{x}_h)$ that attempts to predict $\boldsymbol{y}_h$ from $\boldsymbol{x}_h$. Importantly, we restrict f_{xy} to a simple class of functions (e.g., sparse linear functions) and train all models end-to-end to minimize the prediction error between $\boldsymbol{y}_h$ and $\hat{\boldsymbol{y}}_h$. Satisfying this objective thus constrains both g_x and g_y to discover macro variables that are related by a simple mechanism.

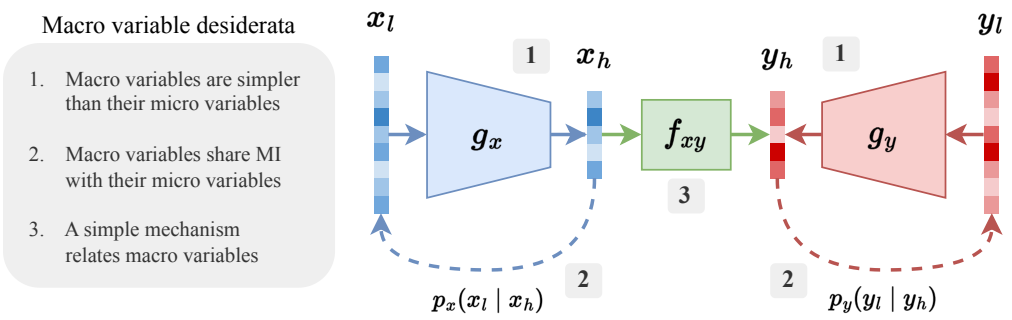

Figure 1: Desiderata for macro variables and our resulting model. Macro variables are constrained to be simpler than micro variables through a dimensionality bottleneck (1). Mutual information between macro and micro variables is maximized using β-VAEs (2). The mechanism relating macro variables must be simple, which we enforce by restricting f_{xy} to a simple class of functions (3).

Complete model. Taking all of the desiderata above, we end up with the model illustrated in Fig. 1. The model consists of 2 β-VAEs (one for $\boldsymbol{x}$ and another for $\boldsymbol{y}$) and a prediction function $\hat{\boldsymbol{y}}_h = f_{xy}(\boldsymbol{x}_h)$, which we parameterize here using sparse linear functions ($\boldsymbol{W}^\top \boldsymbol{x}_h$ where we penalize the L1 norm of $\boldsymbol{W}$). We then optimize all models end-to-end by minimizing the following loss (see Fig. 1 to match colors with model components):

$$\mathcal{L} = -ELBO(g_x, p_x) - ELBO(g_y, p_y) + \lambda \frac{||f(\boldsymbol{x}_h) - \boldsymbol{y}_h||^2}{var(\boldsymbol{y}_h)}, \tag{5}$$

where we divide by $var(\boldsymbol{y}_h)$ in the last term so that the prediction loss is relative to the scale of $\boldsymbol{y}_h$ (which is arbitrary). In sum, successfully minimizing this loss should provide us with macro variables that are simple abstractions of their micro variables and are related by a simple mechanism. In addition, we also obtain encoders for abstracting the micro variables (g_x and g_y), a mechanism that relates the macro variables of the two domains (f_{xy}), and decoders that can be used to interpret the macro variables through reconstruction (p_x and p_y).

4 Empirical Studies

Our preliminary empirical study of DeepCFL focuses on how important each loss term in the DeepCFL objective (Eq. 5) is for learning macro variables that satisfy our aforementioned desiderata. To design experiments where we know what the correct macro variables are, we use images of handwritten digits in two languages as the micro variables $\boldsymbol{X}_l$ and $\boldsymbol{Y}_l$. The preliminary findings suggest that the particular components of DeepCFL are crucial for finding the macro variables that correctly relate two complex spaces of observations.

Dataset. For our preliminary studies and for ease of interpretability, in each experiment we use images of handwritten English digits from the MNIST dataset (LeCun and Cortes [2010]) as $\boldsymbol{X}_l$, and images of handwritten Kannada digits (Prabhu [2019]) as $\boldsymbol{Y}_l$. Both are datasets of 28×28 dimensional images. The correct macro variables capture the digit identities, while other generative factors of images like the stylistic differences in the handwritten digits are irrelevant in this setting.

We pair each English digit with a randomly chosen digit of the same number in Kannada, creating a one-to-one correspondence between macro variables $\boldsymbol{x}_h$ and $\boldsymbol{y}_h$. In the Appendix B, we include results for a second experiment where we randomly pair all the *odd* English digits to one Kannada digit, and all *even* English digits to another Kannada digit, creating a many-to-one mapping.

Model architecture We use the same architecture to parameterize both $(\boldsymbol{g}_{\boldsymbol{x}}, p_{\boldsymbol{x}})$ and $(\boldsymbol{g}_{\boldsymbol{y}}, p_{\boldsymbol{y}})$ - a CNN VAE with 4 convolutional layers, a single linear layer, and a 10-dimensional latent space. Since we are working with categorical macro variables, we also pass the latents through a softmax layer as an inductive bias that encourages better distinction between different macro variables. The prediction map f_{xy} is a single linear layer with no bias or activation. For a comparison of experimental results with and without the softmax layer, and for different values of the latent dimension, see Appendix A.

Metrics. To quantitatively measure how well our model recovers the true macro variables, we use the *silhouette score* of the learned macro variable representations. The silhouette score measures the goodness of clustering of the representations into digit classes. Silhouette scores lie in the range $(-1, +1)$, with $+1$ implying perfect clustering. We also perform a t-SNE analysis of the input data and latent variables to qualitatively evaluate the learned representations.

4.1 Experiment

We set the dimensions of the learned latent representations to 10, i.e. $\boldsymbol{x}_h \in \mathbb{R}^{10}, \boldsymbol{y}_h \in \mathbb{R}^{10}$. Ideally, the learned representations for both domains should only capture the digit identity for each digit. Since we pass the latent representations through a softmax layer, the 10 dimensional latent vectors should ideally be a one-hot-vector indicating digit identity or some permutation of a one-hot-vector, for both $\boldsymbol{x}_h$ and $\boldsymbol{y}_h$. Thus our benchmark is the case where f_{xy} is the identity matrix.

Parameters	x_h	y_h	$\hat{y}_h$
f_{xy} as identity map (fixed) and $\lambda = 0$	0.128	0.204	0.128
f_{xy} as identity map (fixed) and $\lambda \neq 0, \beta_x = \beta_y = 1.25$	0.727	0.745	0.727
f_{xy} as linear map (learned) and $\lambda \neq 0, \beta_x = 1.0, \beta_y = 2.0$	0.536	0.588	0.596

Table 1: Results for Experiment 1. Silhouette scores of learned representations. Here the learned macro variables $\boldsymbol{x}_h, \boldsymbol{y}_h$ are passed through a softmax layer and $\hat{\boldsymbol{y}_h} = f_{xy}(\boldsymbol{x}_h)$. For the case where $\lambda \neq 0$, we increase λ from 1 to 10 over the epochs using a sigmoid scheduler

We study DeepCFL without and with the prediction loss term ($\lambda = 0$ or otherwise) to evaluate how the specific features of DeepCFL affect the learning of correct macro variables compared to standard representation learning. In Table 1 we compare the results of experiments with f_{xy} fixed as the identity matrix to those where f_{xy} is a trainable linear map.

Fig. 2 shows a comparison of the t-SNE plots for the learned macro variables $\boldsymbol{x}_h, \boldsymbol{y}_h$ and the predicted macro variables $\hat{\boldsymbol{y}_h}$ when using our method (f_{xy} is either a fixed identity matrix or a learned linear

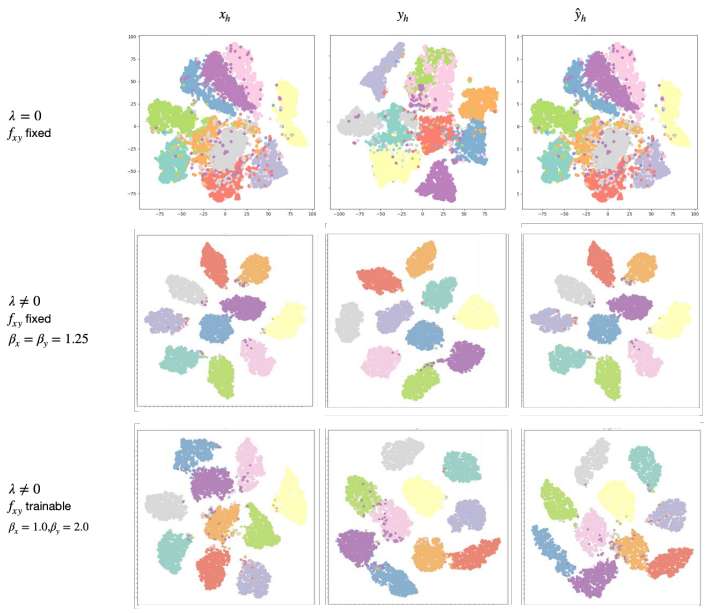

Figure 2: t-SNE plots for experiment 1. Each row contains the t-SNE plots for $\boldsymbol{x}_h, \boldsymbol{y}_h$ and $\hat{\boldsymbol{y}}_h$ respectively (left to right). We observe that the learned representations more easier to disentangle when we include the prediction loss in the training objective.

map) and when removing the prediction loss ($\lambda = 0$). Both the silhouette scores and the t-SNE plots indicate that including the prediction loss leads more accurate recovery of the correct macro variables $\boldsymbol{x}_h$ and $\boldsymbol{y}_h$, which in this setting are the digit identities.

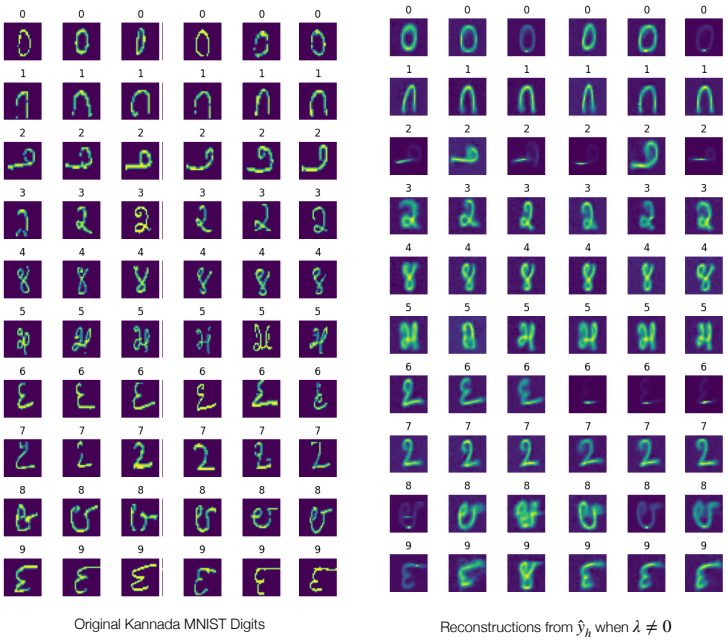

Figure 3: Reconstructions of the digits obtained by passing the predicted $\hat{\boldsymbol{y}}_h = f_{xy}(x_h)$ through the trained $\boldsymbol{y}$ decoder, p_y. Here f_{xy} is a learned linear map.

The results suggest that DeepCFL effectively recovers macro variables that are *simpler* than their micro variables (due to a dimensionality bottleneck), but nevertheless share mutual information with them (qualitatively shown by the accurate reconstructions of the digits from the latent vectors in Fig.3). We also learn the simple mechanism (here, the linear function) that relates the macro variables $\boldsymbol{x}_h$ and $\boldsymbol{y}_h$ (qualitatively shown by the accurate reconstructions of the digits using $\hat{\boldsymbol{y}}_h$ in Fig.3). The learned $\boldsymbol{x}_h$ can reconstruct the English MNIST digits and predict, via the simple mechanism of a linear map, the macro variables of the other domain, which in turn successfully reconstruct their corresponding digits. This demonstrates that our model has successfully learned the desired encoding

and prediction mechanisms. In Appendix B, we show that DeepCFL also works in a different setting, where a many-to-one mapping relates $\boldsymbol{x}_h$ and $\boldsymbol{y}_h$.

5 Conclusions and future work

We introduced DeepCFL, a differentiable method for learning flexible macro variables from low-level micro variables which both satisfies and extends the desiderata of CFL [Chalupka et al., 2017]. Our initial experiments suggest that DeepCFLeffectively learns simple macro variables and a simple mechanism relating them, in particular with discrete macro variables. Our future work will apply our model to real-world CFL benchmark datasets and different macro variable domains (eg. continuous macro variables), explore different methods to maximise mutual information between micro and macro variables, and consider more complex mechanisms (such as quadratic or higher-order polynomials). Finally, our method is adaptable to scenarios with more than two micro variable spaces and could be integrated with causal discovery to learn both the underlying macro variables of a causal graphical model and the mechanisms relating these variables.

6 Acknowledgements

This work is supported by CIFAR.

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International conference on machine learning*, 2018.

Krzysztof Chalupka, Frederick Eberhardt, and Pietro Perona. Causal feature learning: an overview. *Behaviormetrika*, 44:137–164, 2017.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *IEEE Signal Processing Magazine*, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *CoRR*, abs/1908.01242, 2019. URL `http://arxiv.org/abs/1908.01242`.

A Experiment 1: One-to-one correspondence between macro variables

As our experiments are concerned with learning macro variable representations of micro variable data whose true underlying macro variables are discrete and categorical, the primary inductive biases are 1) the use of the softmax layer applied to the learned latent variables in order to amplify the difference between the learned macro variables of different categories and 2) setting the dimensions of the latent variables to the number of categories in the true macro variable space.

In following sections we shall illustrate the validity of these assumptions and illustrate the performance of our model upon their relaxation.

A.1 Use of reconstructions as a qualitative metric

In our experiments we use VAEs for learning the macro variables. The choice of a VAE is not necessary. However, the image reconstructions from the VAE decoders allow us to visually inspect our learned macro variables. As mentioned before, the stylistic differences in the handwritten digits are not relevant macro variables in this setting. We are more concerned with the digit identities. Thus a measure of whether our macro variables meet the competing objectives of being simpler than their micro variables while sharing high mutual information with them, is *how well the reconstructions from the macro variables "smooth out" the stylistic differences between digits of the same class* and generalise over them. The reconstructions obtained by passing the predicted $\hat{\boldsymbol{y}}_h$ through the decoder $\boldsymbol{p}_y$ also indicate the model's proficiency in learning both the prediction mechanism as well as suitable encoding mechanisms for $\boldsymbol{x}_h$ (/ $\boldsymbol{y}_h$) which can successfully predict(/be predicted by) each other.

A.2 Effect of the softmax layer

	f_{xy} as identity map (fixed)			f_{xy} as a linear map (trainable)		
With/without softmax	$\boldsymbol{x}_h$	$\boldsymbol{y}_h$	$\hat{\boldsymbol{y}_h}$	$\boldsymbol{x}_h$	$\boldsymbol{y}_h$	$\hat{\boldsymbol{y}_h}$
With softmax ($\lambda \neq 0$)	0.712	0.738	0.712	0.330	0.413	0.436
Without softmax ($\lambda \neq 0$)	0.331	0.3545	0.331	0.144	0.298	0.335

Table 2: Comparison of results for experiment 1 when we include/do not include a softmax layer in the forward pass of the encoders g_x, g_y.

Table 2 shows that the inclusion of a softmax layer leads to a notably improved performance in terms of distiguishing macro variables between different classes and encouraging the similarity of macro variables within the same class, as evidenced by the silhouette scores.

It is to be noted that the t-SNE plots in Fig. (4) for both without and with softmax are quite similar, despite the contrast in their silhouette scores. This similarity arises because t-SNE inherently performs some level of disentanglement while mapping the learned latents to a new latent space. On the other hand, silhouette scores consider the macro (latent) representations as they are, accounting for the observed differences.

We see that additional information and assumptions about the nature of the macro variables can be incorporated into the model design to aid learning macro variables having the desired properties.

A.3 Varying latent dimensions

The second inductive bias is configuring the dimensions of the macro variables to match the number of underlying categories of our data. Our method performs best if the appropriate latent dimensions are known, but is also fairly robust to overestimation, see Fig.5.

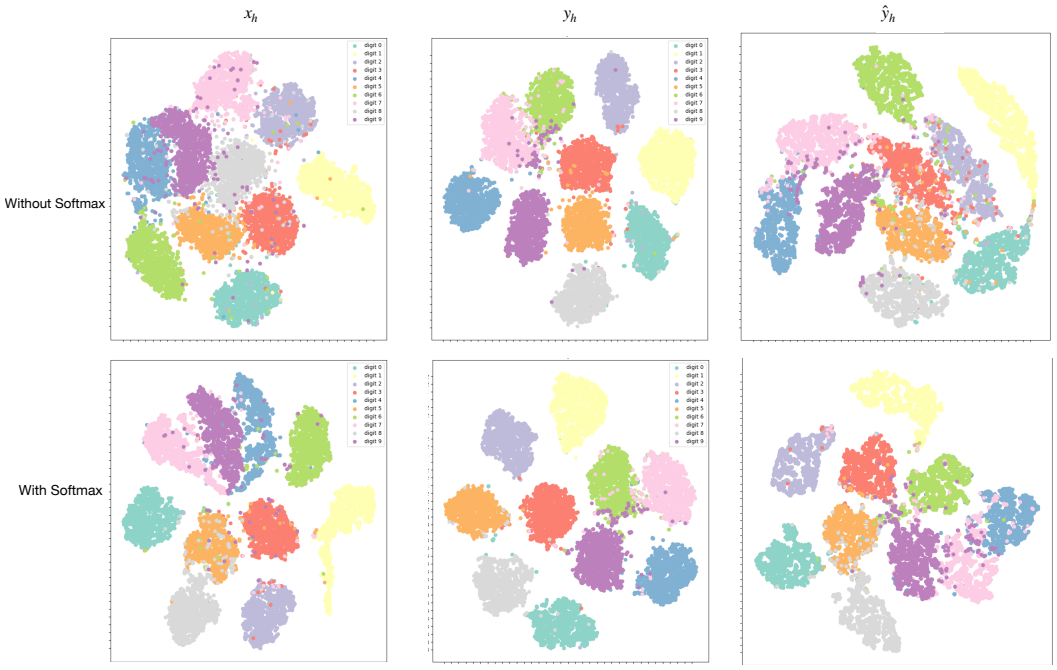

Figure 4: t-SNE plots for experiments in 2 1 without the softmax layer (top row) and with it (bottom row). Each row contains the t-SNE plots for $\boldsymbol{x}_h$, $\boldsymbol{y}_h$ and $\hat{\boldsymbol{y}}_h$ respectively (left to right). Here f_{xy} is a linear map for both cases. While both models perform similarly in terms of t-SNE plots of the learned macro variables and reconstructions of the digits, the latent variables learned from the model with the softmax layer are more disentangled and are more desirable.

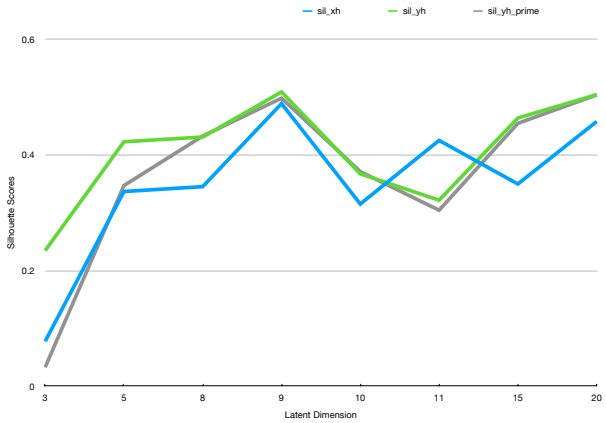

Figure 5: Silhouette scores vs dimensions of macro variables

B Experiment 2: Many-to-one correspondence between macro variables

The causal micro variables $\boldsymbol{X}_l$ are the images of all the handwritten English digits $\boldsymbol{x}_l \in \mathbb{R}^{28\times 28}$. Each odd number digit in the cause domain is (randomly) paired with an image of the number 1 in the Kannada MNIST dataset. Each even number digit in the cause domain is paired with an image of the digit 2 in the Kannada MNIST dataset. We set the dimensions of the learned latent representations to 10 and 2 in the cause and effect domains respectively, i.e. $\boldsymbol{x}_h \in \mathbb{R}^{10}, \boldsymbol{y}_h \in \mathbb{R}^2$.

The architecture design for the $\boldsymbol{x}_l$ VAEs is identical across the two experiments while for the $\boldsymbol{y}_l$ VAEs the only difference is in the output(/input) dimension of the final(/first) linear layer of the encoder(/decoder) (20 for the one-to-one experiment and 4 for the many-to-one experiment).

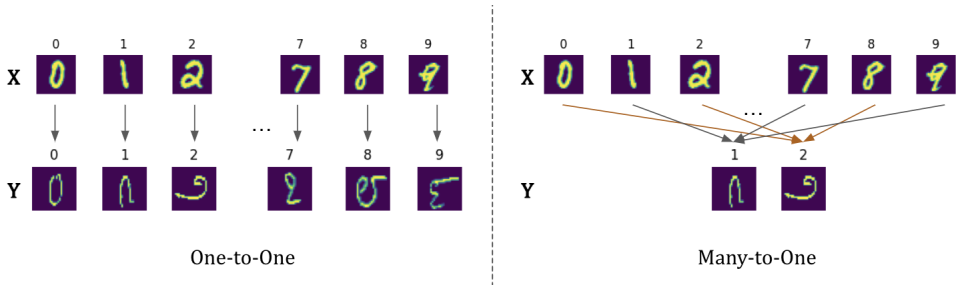

Figure 6: Setup for experiment 1 (left) and experiment 2 (right)

Our benchmark is the model with f_{xy} fixed as the linear map which would map a one-hot-encoded even digit to $[1, 0]^T$ and a one-hot-encoded odd digit to $[0, 1]^T$

$$f_{xy}^{(\text{benchmark})} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \tag{6}$$

.

Our baseline is once again the model learned with the weight of the prediction error λ in (5) set to zero.

We observe that the learned macro variables of different classes are more disentangled when f_{xy} is a linear map than the fixed identity map. Fixing f_{xy} to a sparse linear map forces the learned $\boldsymbol{x}_h$, to compromise on the inter-digit distinctions in order to map to 2 possible $\boldsymbol{y}_h$. In contrast, a learned linear map does not impose such constraints and offers more flexibility to the learned latent (macro) variables.

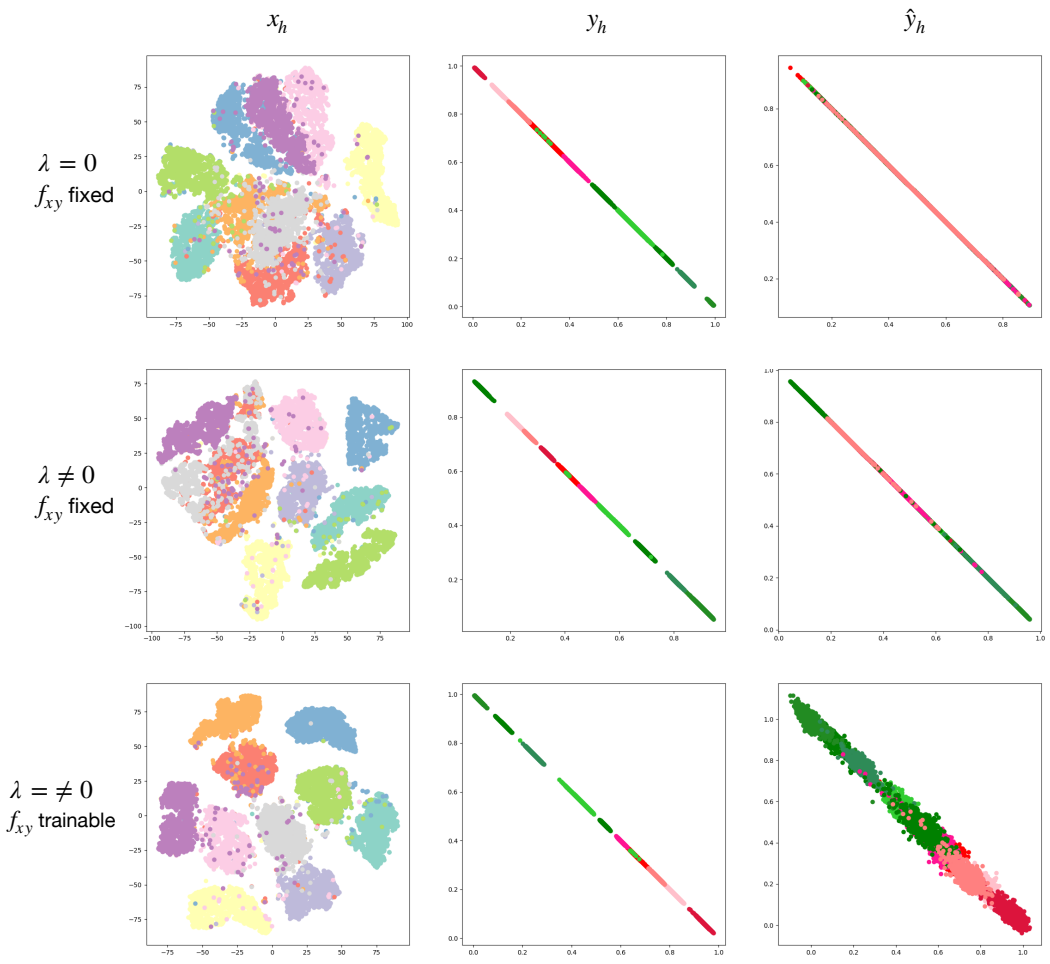

Figure 7: t-SNE plots for experiment 2. Each row contains the t-SNE plots for $\boldsymbol{x}_h$, $\boldsymbol{y}_h$ and $\hat{\boldsymbol{y}}_h$ respectively (left to right). In the plots for $\boldsymbol{y}_h$ and $\hat{\boldsymbol{y}}_h$ the different shades of green correspond with macro variables of the digit 2 paired with even digits in $\boldsymbol{x}$ while the shades of pink correspond with macro variables of the digit 1 paired with odd digits in $\boldsymbol{x}$.