DEEPWAK: END-TO-END DIFFERENTIABLE SPARSE ENSEMBLE CLUSTERING USING DENOISING

A PREPRINT

Keira Wiechecki
Center for Genomics & Systems Biology
New York University

January 8, 2024

ABSTRACT

Clustering is a special case of sparse dictionary learning where all features are discrete. Here we introduce Denoising by Deep learning of a Partitoned Weighted Affinity Kernel (DeePWAK).

1 Introduction

Though applications of deep learning to classification is well established, self-supervised classification has been much less thoroghly explored. Deep clustering is an active area of research[3]. Similarly to DeepCluE[2], we use an ensemble clustering method. But rather than creating a single consensus partition, we aim to maximise independence between submodels.

Our approach uses what we believe to be a previously unexplored combination of an information bottleneck with self-supervised denoising. Like ClAM[4] it is end-to-end differentiable. Unlike ClAM,

1.1 Notation

Capital letters indicate matrices. Subscripts indicate indices. Superscripts indicate dimensionality. A circumflex indicates a reconstruction of data by a predictor. Lowercase Greek letters indicate tunable parameters. Capital Greek letters indicate parameter spaces. For parameters θ, $\theta^{m \to d}$ indicates parameters for a model that accepts an input of dimension m and returns an output of dimension d. $\underset{n \to m}{\text{function}}$ indicates a layer with input dimension n, output dimension m, and activation function function.

It's necessary to distinguish between a model architecture and its parameters. We use the notation $\mathcal{F}(\theta)$ to indicate a model architecture $\mathcal{F}$ parameterized by θ. $\mathcal{F}(\theta)(X)$ indicates passing data X to the model $\mathcal{F}(\theta)$. We write this as a curried function to emphasize that $\mathcal{F}(\theta)$ is stateful.

1.2 Type Definitions

We define a `Model` as having type

$$\texttt{Model} := \exists m, d : \mathbb{N}.\mathbb{R}^m \to \mathbb{R}^d \tag{1}$$

$$\texttt{Architecture} := \forall m, d : \mathbb{N}.\mathbb{R}^{m \to d} \to \texttt{Model}(m, d) \tag{2}$$

In other words, a `Model` has an input dimension m and an output dimension d. For any input $X \in \mathbb{R}^m$, it applies a sequence of operations parameterized by $\theta \in \mathbb{R}^{m \to d}$ which transform X to an output $E \in \mathbb{R}^d$.

We define a `Loss` function as having type

$$\texttt{Loss} := \forall m, d : \mathbb{N}.\texttt{Model}(m, d) \to \mathbb{R}^m \to \mathbb{R}^d \to \mathbb{R} \tag{3}$$

For any model $\mathcal{F}$: `Model` parameterized by $\theta^{m \to d}$, it accepts inputs $X \in \mathbb{R}^m$ and $E \in \mathbb{R}^d$. It returns a scalar representing the similarity between $\mathcal{F}(\theta)(X)$ and E.

Additionally, a `Model` must implement an $\mathrm{update!}$ function, which has type

$$\forall \mathcal{F} : \texttt{Model}, \mathcal{L} : \texttt{Loss}. \tag{4}$$

1.3 noise2self

Batson & Royer[1] identify a class of denoising functions which can be optimised using only unlabeled noisy data.

Let $J \in \mathcal{J}$ be independent partitions of noisy data X. Let $\mathcal{F}(\theta)$ be a family of predictors of X_J with tunable parameters $\theta \in \Theta$ that depends on its complement X_{J^C}

$$\hat{X}_J = \mathcal{F}(\theta)(X_J^C) \tag{5}$$

In other words, $\mathcal{F}$ predicts each data point X_J from some subset of the data excluding X_J.

The optimal θ is given by

$$\underset{\theta}{\overset{\Theta}{\mathrm{noise2self}}}[\mathcal{F}(\theta), X] := \underset{\theta}{\overset{\Theta}{\mathrm{argmin}}}[\sum_{J}^{\mathcal{J}} \mathbb{E}||X_J - \mathcal{F}(\theta)(X_{J^C})||^2] \tag{6}$$

1.4 Graph diffusion

Our choice of $\mathcal{F}$ is adapted from DEWAKSS[5]. The parameters we want to tune generate a graph G from embeddings E. The adjacency matrix of any graph can be treated as a transition matrix (or weighted affinity kernel) by setting the diagonal to 0 and normalizing columns to sum to 1. We call this the WAK function. For each embedding, an estimate is calculated based on its neighbors in the graph. This can be expressed as matrix multiplication.

$$\hat{E} := \mathrm{WAK}(G)E^\top \tag{7}$$

Though DEWAKSS uses a k-NN graph, any adjacency matrix will do. A clustering can be expressed as a graph where points within a cluster are completely connected and clusters are disconnected.

Let $C^{c \times n}$ be a matrix representing a clustering of n points into c clusters. Let each column be a 1-hot encoding of a cluster assignment for each point. We can obtain a partition matrix $P^{n \times n}$ by

$$P := C^\top C \tag{8}$$

Informally, this can be equated to position-independent attention with data points as tokens and the batch size as the context window.

2 Architecture

The DeePWAK constructor has the type signature

$$\mathrm{DeePWAK} := \forall m, d, c : \mathbb{N} \to (\mathbb{R}^m \to \mathbb{R}^d) \to (\mathbb{R}^m \to \mathbb{R}^c) \to (\mathbb{R}^d \to \mathbb{R}^m) \to \mathbb{R}^m \to \mathbb{R}^m \tag{9}$$

It consists of an encoder, partitioner, and decoder.

The architecture is inspired by dot product attention, but has a few key differences. Because it is position-independent, Q and K are simply transposes of each other. Each head has a separate encoder and decoder.

Algorithm 1 DeePWAK constructor

data DeePWAK
encoder : $\exists m, d : \mathbb{N} \to \mathbb{R}^m \to \mathbb{R}^d$
partitioner : $\exists m, c : \mathbb{N} \to \mathbb{R}^m \to \mathbb{R}^c$
decoder : $\exists d, m : \mathbb{N} \to \mathbb{R}^d \to \mathbb{R}^m$

Algorithm 2 DeePWAK application

function DeePWAK$(\mathcal{E}(\theta), \mathcal{P}(\pi), \mathcal{D}(\phi))(X : \mathbb{R}^{m \times n})$
$E \leftarrow \mathcal{E}(\theta)(X)$
$C \leftarrow (\text{softmax} \circ \mathcal{P}(\pi))(X)$
$P \leftarrow C^\top C$
$G \leftarrow \text{WAK}(P)$
$\hat{E} \leftarrow (GE^\top)^\top$
$\hat{X} \leftarrow \mathcal{D}(\phi)(\hat{E})$
return $\hat{X}$

2.1 Multihead DeePWAK

3 A concrete example from microscopy data

Tools for imaging DEWAKSS is poorly suited to

4 Results

4.1 Multihead DeePWAK learns sparse representations

5 Discussion

References

[1] Joshua Batson and Loic Royer. *Noise2Self: Blind Denoising by Self-Supervision*. 2019. arXiv: 1901.11365 [cs.CV].

[2] Dong Huang et al. *DeepCluE: Enhanced Image Clustering via Multi-layer Ensembles in Deep Neural Networks*. 2023. arXiv: 2206.00359 [cs.CV].

[3] Yazhou Ren et al. *Deep Clustering: A Comprehensive Survey*. 2022. arXiv: 2210.04142 [cs.LG].

[4] Bishwajit Saha et al. *End-to-end Differentiable Clustering with Associative Memories*. 2023. arXiv: 2306.03209 [cs.LG].

[5] Andreas Tjärnberg et al. "Optimal tuning of weighted kNN- and diffusion-based methods for denoising single cell genomics data". In: *PLOS Computational Biology* 17.1 (Jan. 2021), pp. 1–22. DOI: 10.1371/journal.pcbi.1008569. URL: https://doi.org/10.1371/journal.pcbi.1008569.

Algorithm 3 DeePWAKBlock

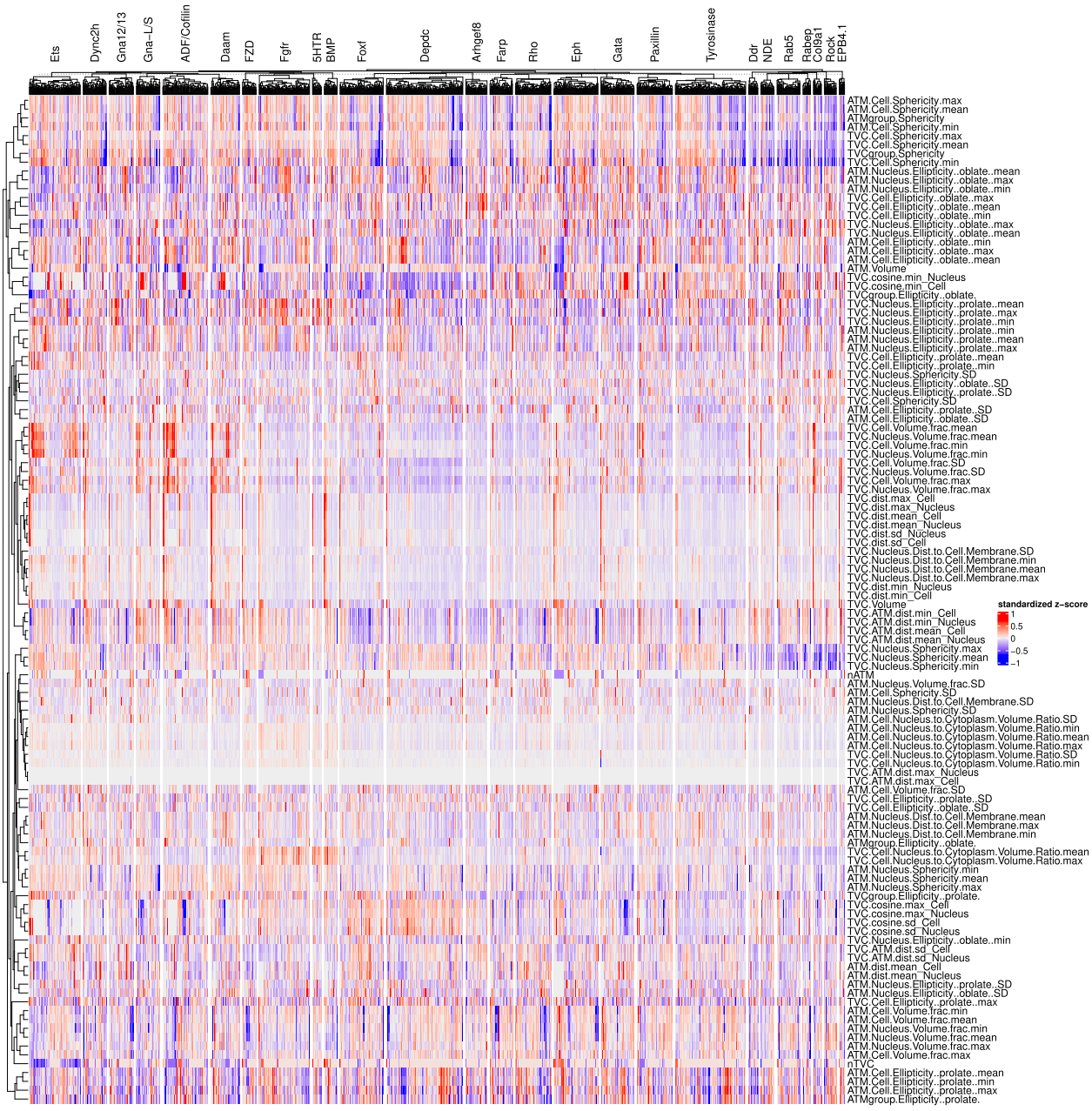
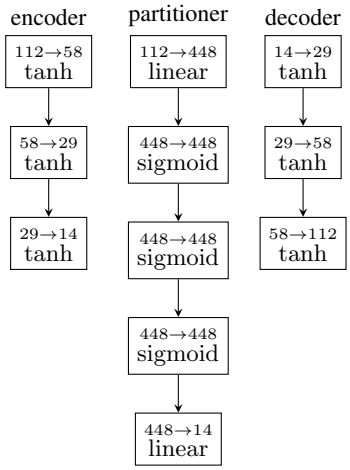
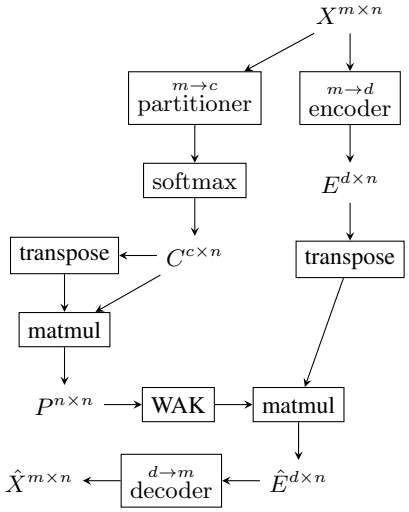


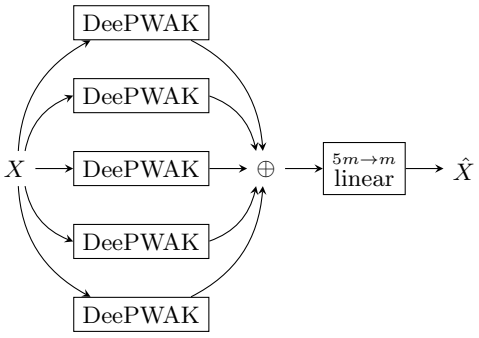
Figure 1: Microscopy data after preprocessing.



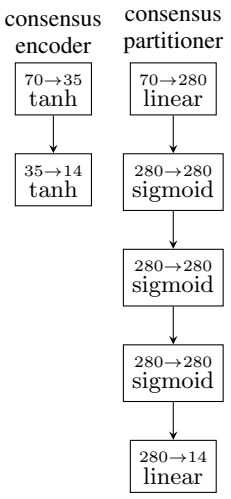
(a)



(b)



(c)



(d)

Figure 2: (a) Architectures of encoder, partitioner, and decoder. (b) Architecture of one DeePWAK head. (c) Architecture of one DeePWAK block. (d) Architectures of pooled encoder and pooled clusterer.

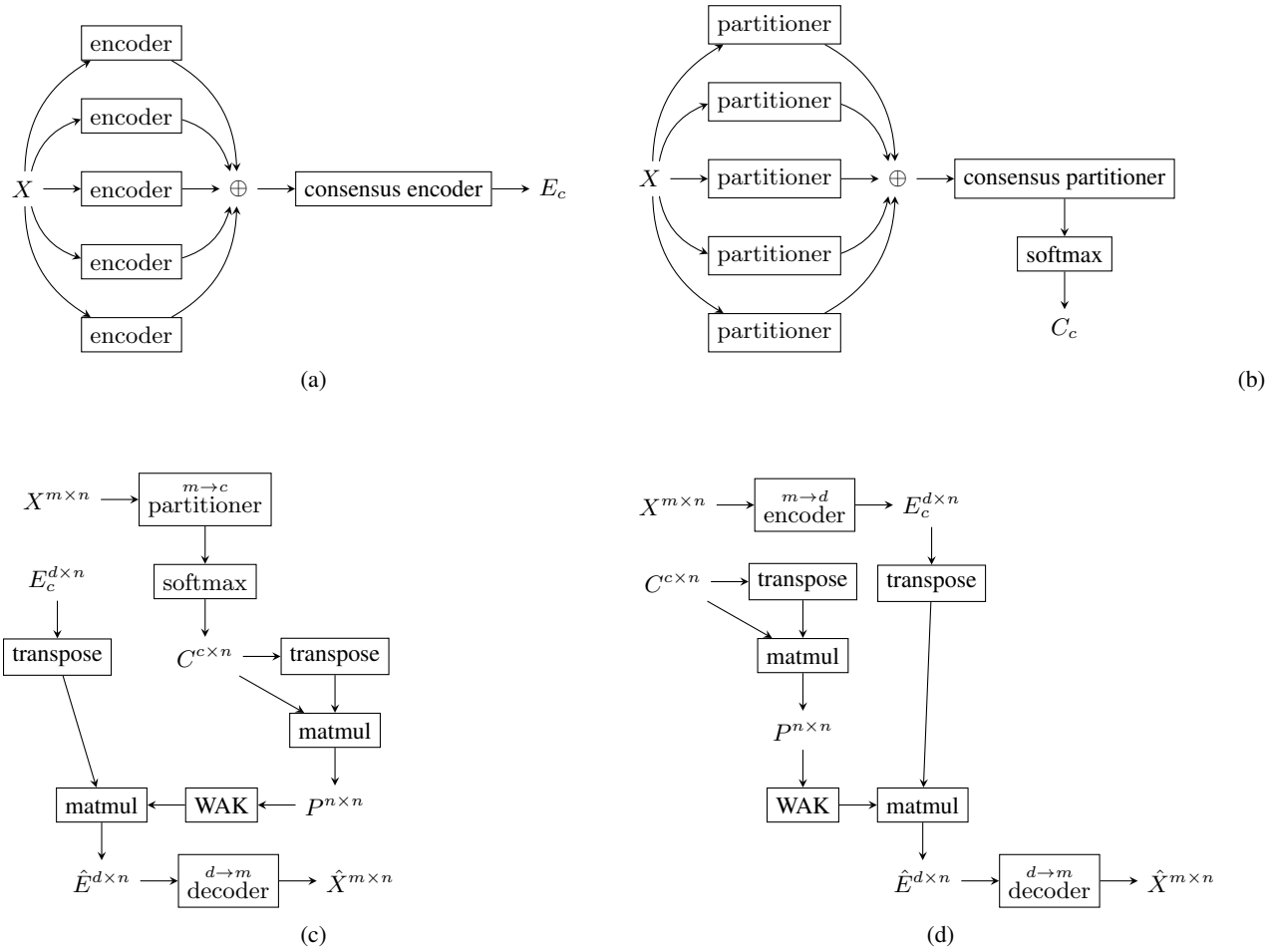


Figure 3: (a,b) Calculation of consensus E and C, respectively.

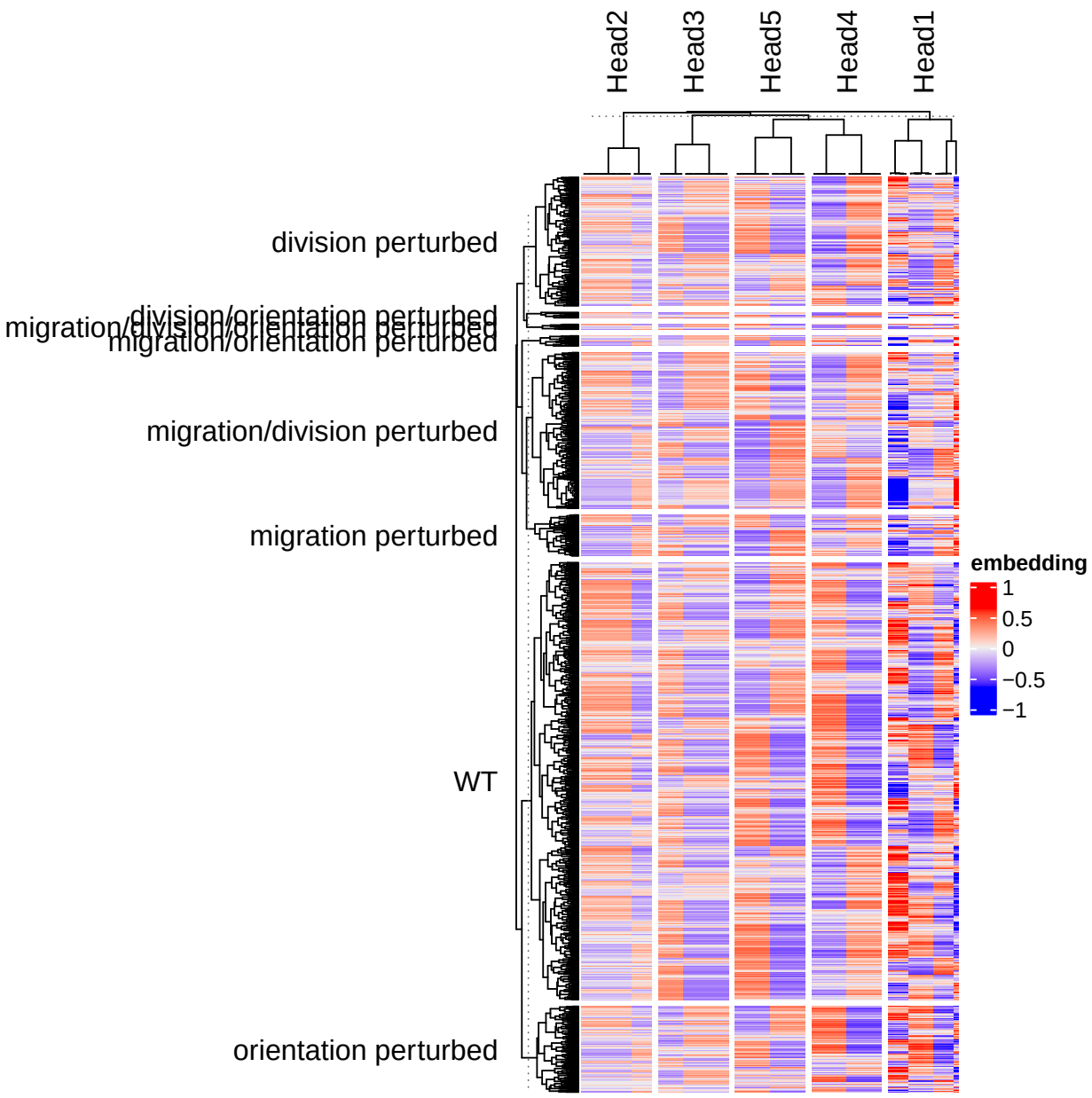


Figure 4: DeePWAK learns sparse embedding values.

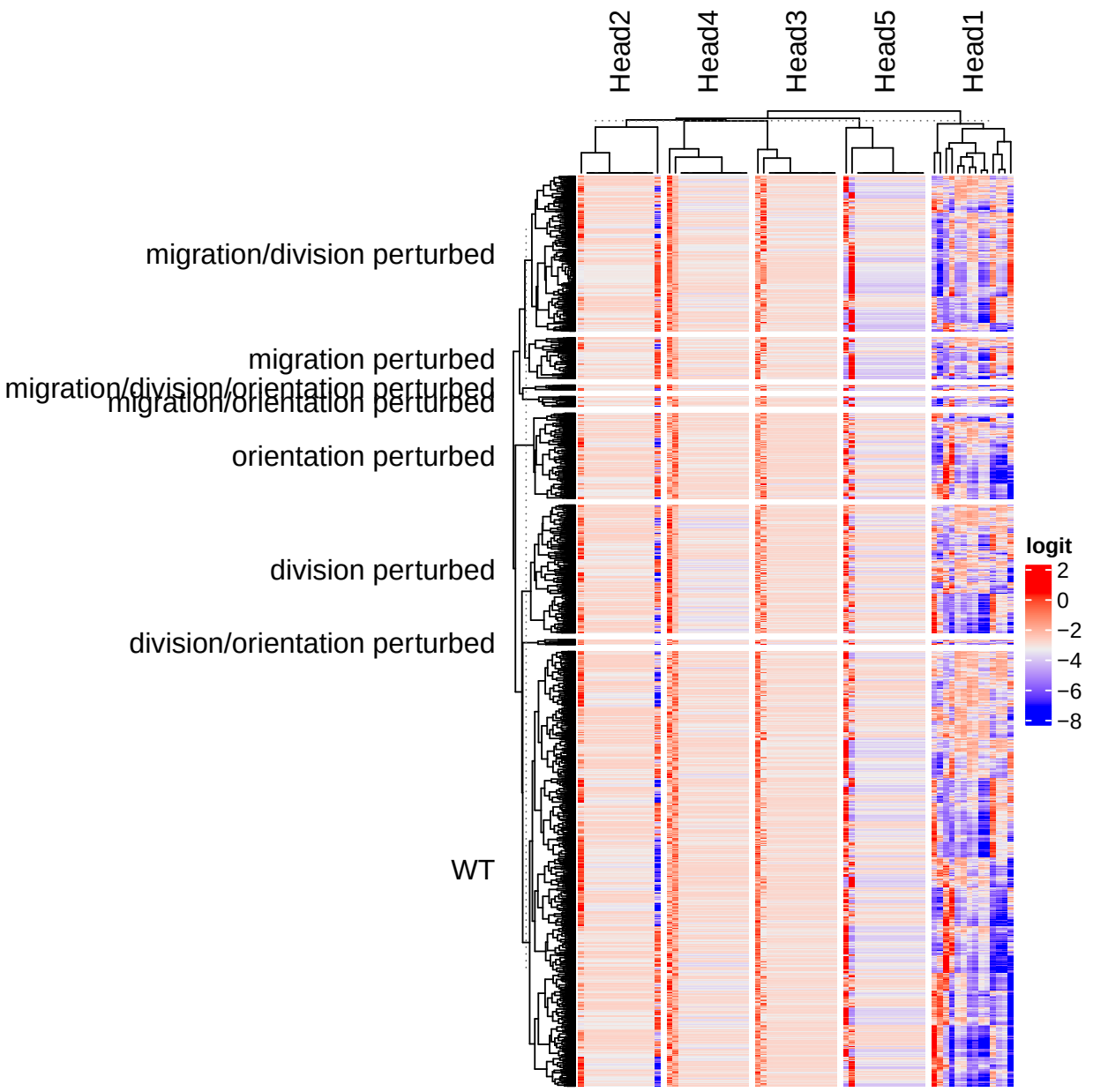


Figure 5: For most heads, two clusters dominate.

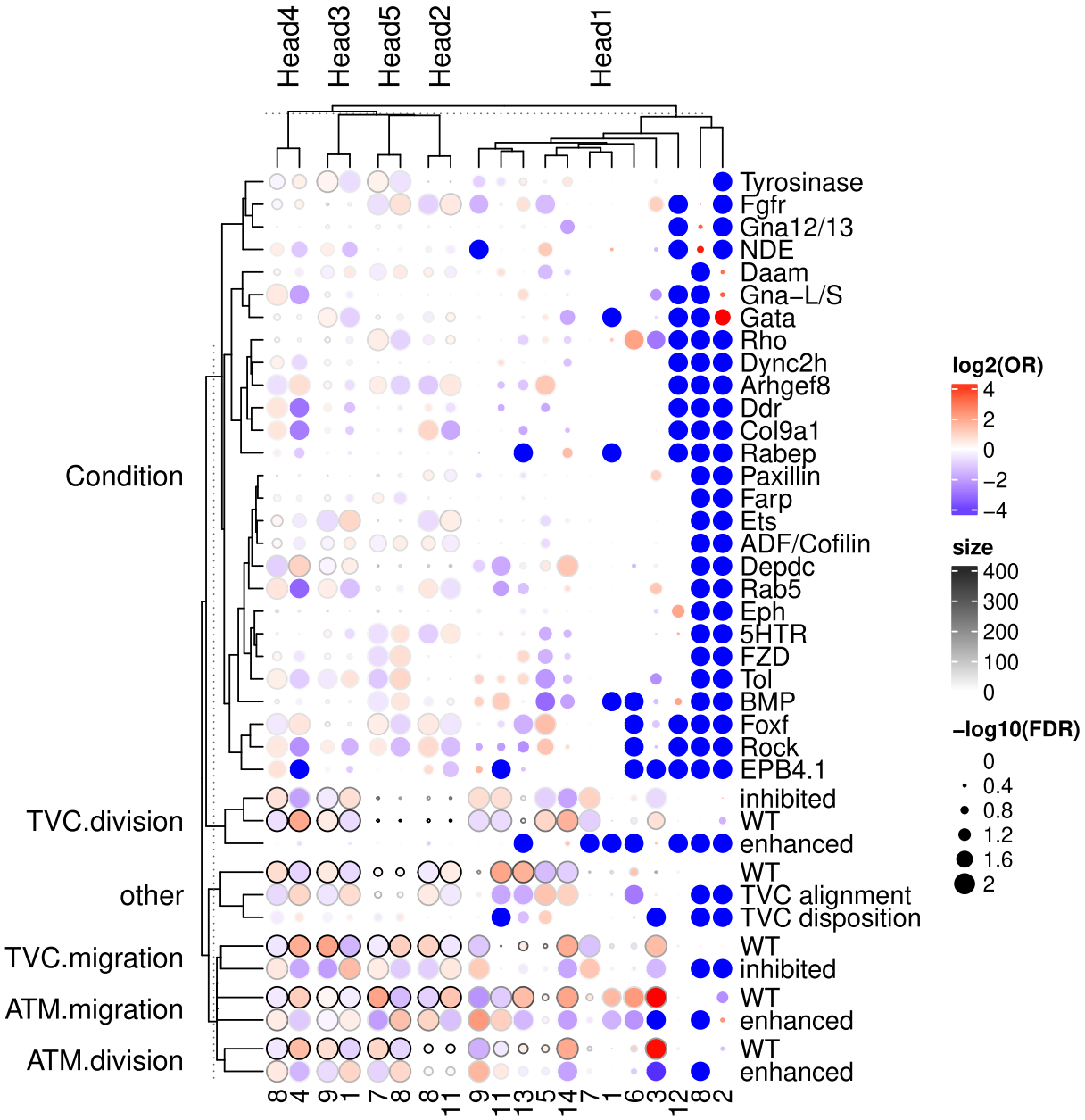


Figure 6: Hypergeometric test for enrichment of phenotype and treatment condition for each cluster in each head. Dot color indicates overrepresentation of each category in a cluster compared to a uniform prior. Dot size indicates statistical significance of the difference. Dot outline shade indicates number of embryos represented by a dot. Head 1 appears "polysemantic". The others appear to distinguish single phenotypic categories.