Supplemental material for Sum-Product Autoencoding: Encoding and Decoding Representations using Sum-Product Networks

Antonio Vergari
antonio.vergari@uniba.it
University of Bari, Italy

Robert Peharz
rp587@cam.ac.uk
University of Cambridge, UK

Nicola Di Mauro
nicola.dimauro@uniba.it
University of Bari, Italy

Alejandro Molina
alejandro.molina@tu-dortmund.de
TU Dortmund, Germany

Kristian Kersting
kersting@cs.tu-darmstadt.de
TU Darmstadt, Germany

Floriana Esposito
floriana.esposito@uniba.it
University of Bari, Italy

Augmented SPNs

Here is a depiction of the process of augmenting an SPN S into $\overline{S}$ as detailed in the Sum-Product Autoencoding section.

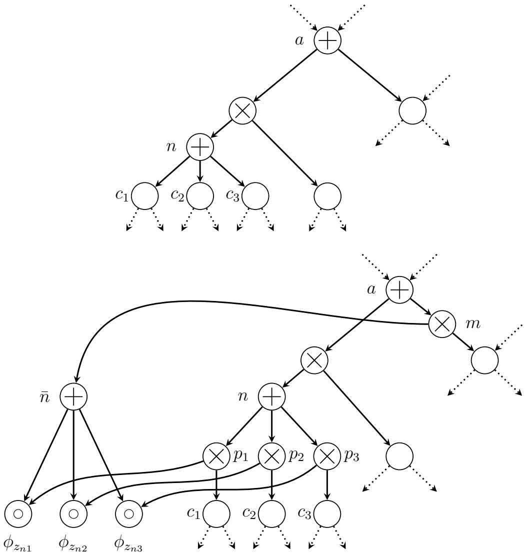

Figure 1: A portion of an SPN S (top), and the corresponding portion while augmenting its sum node n in $\overline{S}$ (bottom). Indicator leaf nodes $\{\phi_{z_{ni}} = \mathbb{1}\{Z_n = z_{ni}\}\}_i$ are introduced as children of both the twin sum node $\overline{n}$ and of link product nodes $\{p_{nc_i}\}$. At the same time product link nodes are set as children of n and as the new parents of $\{c_i\}_i$. Lastly the twin sum node $\overline{n}$ is set as a child of m, the link product node child of a, the conditioning sum node of n (i.e. the ancestor sum node of n, for which n is not an ancestor).

Proofs

Tree path equivalence for encoding and decoding with augmented SPNs

Proposition 1. *Let S be an SPN over $\mathbf{X}$, M its corresponding MPN, and $\overline{M}$ its augmented version over $\mathbf{V} = (\mathbf{X}, \mathbf{Z}_S)$. Let θ' (resp. θ'') be the tree built by computing $f_{\mathsf{MPE}}(\mathbf{x}^i)$ (resp. $g_{\mathsf{MPE}}(f_{\mathsf{MPE}}(\mathbf{x}^i))$) on $\overline{M}$, given a sample $\mathbf{x}^i \sim \mathbf{X}$. Then, it holds that $\theta' = \theta''$.*

Proof. Let $\overline{\mathbf{M}}_{\theta'}$ (resp. $\overline{\mathbf{M}}_{\theta''}$) be the set of nodes in $\overline{M}$ that are also in θ' (resp. θ''). We want to demonstrate that $\overline{\mathbf{M}}_{\theta'} = \overline{\mathbf{M}}_{\theta''}$. We demonstrate this by construction, showing that if a node is added to θ' during the top-down pass of f_{MPE}, then it is also added to θ'' during the top-down pass of g_{MPE} and no other nodes are added to it. We start by the root of $\overline{M}$, which is clearly both in $\overline{\mathbf{M}}_{\theta'}$ and $\overline{\mathbf{M}}_{\theta''}$ and then perform a BFS. Let n_j be the node both in $\overline{\mathbf{M}}_{\theta'}$ and $\overline{\mathbf{M}}_{\theta''}$ currently under consideration. If n_j is a product node, then it must hold that $\forall c \in \mathsf{ch}(n_j) : c \in \overline{\mathbf{M}}_{\theta'} \wedge c \in \overline{\mathbf{M}}_{\theta''}$ by construction. Otherwise, if n_j is a max node, then $\exists! c' \in \mathsf{ch}(n_j) : c' \in \overline{\mathbf{M}}_{\theta'}$ and $\exists! c'' \in \mathsf{ch}(n_j) : c'' \in \overline{\mathbf{M}}_{\theta''}$. We prove that $c' = c''$. n_j must be either a twin sum node (a) or not (b). a) If n_j is a twin max node, then c' and c'' are LV indicators in the form $\phi_{z_{jc'}}$ and $\phi_{z_{jc''}}$ respectively. Since c' has been selected while growing θ' by choosing the LV indicator whose corresponding twin weight was maximum, then the Z_j value in $\tilde{\mathbf{z}}^i = f_{\mathsf{MPE}}(\mathbf{x}^i)$ has been assigned to $z_{jc'}$. Consequently, the only non-zero variable indicator $\phi_{z_{j*}}$ for Z_j, after configuring all leaves according to $\tilde{\mathbf{z}}^i$ in the bottom-up pass of g_{MPE} to build θ'', is $\phi_{z_{jc'}}$ and hence the only one to be added to θ'' is $c' = c''$. b) Analogously, if n_j is a non-twin max node, the same reasoning applies to its children, which are link product nodes. Link product nodes act as "gates" memorizing the traversal of a max node child during encoding and inhibiting all its sibling signals (putting them to zero) during decoding. When link product node p_k, child of max node n_j, is chosen and added to θ', it will then let the corresponding LV indicator $\phi_{z_{jk}}$ be chosen at the next level step of the BFS, consequently storing z_{jk} as the MPE assignment of Z_j in $\tilde{\mathbf{z}}^i$. Again, after configuring all leaves according to $\tilde{\mathbf{z}}^i$ in the bottom-up pass of decoding, $\phi_{z_{jk}}$ will be the only non-zero LV indicator for Z_j and as therefore the corresponding parent link product node p_k will be the only non-zero children of n_j. Therefore, it will be the only one added to θ'' during the top-down pass of decoding.

□

Twin weights choice unaffecting decoding

Let S be an SPN, $\overline{S}$ its augmented version over $\mathbf{V} = (\mathbf{X}, \mathbf{Z}_S)$, and θ' the tree built during the MPE top-down pass on $\overline{S}$. Let $\overline{\mathsf{L}}^{\theta'}_{\mathbf{Z}_S}$ denote the set of leaves of the tree θ' over LVs $\mathbf{Z}_S$. We will denote by $\mathbf{Z}_S^{\theta'_{\mathsf{twin}}}$ (resp. $\mathbf{Z}_S^{\theta'_{\mathsf{link}}}$) the set of LVs for which their indicator in $\overline{\mathsf{L}}^{\theta'}_{\mathbf{Z}_S}$ has a parent in θ' that is a twin sum node $\overline{n}$ (resp. link product node), and by $\mathbf{N}_S^{\theta'}$ all the remaining nodes in θ'. For each LV in $\mathbf{Z}_S^{\theta'_{\mathsf{twin}}}$ its assignment is therefore dependent from the selection of the twin weights $\overline{\mathbf{w}}_n$ equipping its parent (for an example see also the proof of Proposition 1). Nevertheless, it is possible to prove that arbitrarily choosing other values to the twin weights, hence assigning other configurations to $\mathbf{Z}_S^{\theta'_{\mathsf{twin}}}$, does not influence the decoding of the collected assignment $\tilde{\mathbf{v}}^i_{|\mathbf{Z}_S}$, i.e. the partial configuration $\tilde{\mathbf{v}}^i_{|\mathbf{X}}$.

Lemma 1. *Let S be an SPN over $\mathbf{X}$, M its corresponding MPN, and $\overline{M}_1$, resp. $\overline{M}_2$, its augmented version over $\mathbf{V} = (\mathbf{X}, \mathbf{Z}_S)$ equipped with twin weights $\overline{\mathbf{w}}_1$, resp. $\overline{\mathbf{w}}_2$, and $\overline{\mathbf{w}}_1 \neq \overline{\mathbf{w}}_2$. Given a sample $\mathbf{x}^i \sim \mathbf{X}$, it holds that $g_{\overline{M}_1}(f_{\overline{M}_1}(\mathbf{x}^i)) = g_{\overline{M}_2}(f_{\overline{M}_2}(\mathbf{x}^i))$, where $f_{\overline{M}_1}$ and $g_{\overline{M}_1}$, resp. $f_{\overline{M}_2}$ and $g_{\overline{M}_2}$, are the application of f_{MPE} and g_{MPE} to $\overline{M}_1$, resp. $\overline{M}_2$.*

Proof. Let θ'_1 (resp. θ'_2) be the tree path encoded by $f_{\overline{M}_1}$ (resp. $f_{\overline{M}_2}$). Let θ''_1 (resp. θ''_2) be the tree path decoded by $g_{\overline{M}_1}$ (resp. $g_{\overline{M}_2}$). We need to prove that $\overline{\mathsf{L}}^{\theta''_1}_{|\mathbf{X}} = \overline{\mathsf{L}}^{\theta''_2}_{|\mathbf{X}}$: if the leaf sets for decoding are the same, then MPE inference will provide the same decoded $\tilde{\mathbf{x}}^i$ on both $\overline{M}_1$ and $\overline{M}_2$.

It holds that $\forall \mathbf{x}^i \sim \mathbf{X} : \overline{M}_1(\mathbf{v}^i_{|\mathbf{X}}) = \overline{M}_2(\mathbf{v}^i_{|\mathbf{X}})$ since the LVs $\mathbf{Z}_S$ are marginalized out when computing the bottom-up passes for both $f_{\overline{M}_1}$ and $f_{\overline{M}_2}$. Since the activations of the nodes are the same for both $\overline{M}_1$ and $\overline{M}_2$, the construction of θ'_1 and θ'_2 in the top-down pass of the encoding will be synchronized up to the twin sum nodes, leading to $\mathbf{N}_S^{\theta'_1} = \mathbf{N}_S^{\theta'_2}$. While, for each twin sum node $\overline{n} \in \overline{M}_1$ (resp. $\overline{n} \in \overline{M}_2$), the corresponding LV indicator to be put in θ'_1 (resp. θ'_2) is chosen as the one corresponding to its max twin weight. However, since $\mathbf{N}_S^{\theta'_1} = \mathbf{N}_S^{\theta'_2}$, then $\overline{\mathsf{L}}^{\theta'_1}_{|\mathbf{X}} = \overline{\mathsf{L}}^{\theta'_2}_{|\mathbf{X}}$. From Proposition 1, since $\theta'_1 = \theta''_1$ and $\theta'_2 = \theta''_2$, then $\overline{\mathsf{L}}^{\theta''_1}_{|\mathbf{X}} = \overline{\mathsf{L}}^{\theta''_2}_{|\mathbf{X}}$. □

CAT encoding and decoding equivalences

Lemma 2. *Let S be an SPN over $\mathbf{X}$, M its corresponding MPN, and $\overline{M}$ its augmented version. Let θ' (resp. θ) be the tree path encoded by f_{MPE} (resp. f_{CAT}) on $\overline{M}$ (resp. M). Given a sample $\mathbf{x}^i \sim \mathbf{X}$, then $f_{\mathsf{MPE}}(\mathbf{x}^i)_{|\mathbf{Z}_S^{\theta'_{\mathsf{link}}}} = f_{\mathsf{CAT}}(\mathbf{x}^i)_{|\mathbf{Z}_S^{\theta}}$*

Proof. It holds that $\forall \mathbf{x}^i \sim \mathbf{X}$: $\overline{M}(\mathbf{v}^i_{|\mathbf{X}}) = M(\mathbf{x}^i)$, since the LVs in $\overline{M}$ are marginalized out (output 1) when computing the bottom-up pass for f_S and therefore: (i) all twin sum nodes output 1 and (ii) all link product nodes are unaffected by the LV indicator values and they output the same value as their product node children. As a consequence we have that the first bottom-up pass for both f_{MPE} and f_{CAT} leaves the common structure of $\overline{M}$ and M (i.e. all the nodes in M that are also in $\overline{M}$) to output the exactly same values. This will "synchronize" both top-down passes of f_{MPE} on M and of f_{CAT} on M. Indeed, if $n, c_k \in \theta$, where $n \in \mathsf{M}^{\mathsf{max}}, c_k \in \mathsf{ch}(n)$, then $\phi_{z_{nc_k}} \in \overline{\mathsf{L}}^{\theta'}$ because in $\overline{M}$ there is a link product node between n and c_k that is a parent of $\phi_{z_{nc_k}}$ and all its children will be put into θ' according to how $\mathsf{MPEInference}$ works. Therefore LV Z_n will be defined in the same way in both embeddings $f_{\mathsf{MPE}}(\mathbf{x}^i)$ and $f_{\mathsf{CAT}}(\mathbf{x}^i)$. The LVs undefined in $\mathbf{Z}_S^{\theta}$, i.e. those whose corresponding max node is not in θ, will then be ones in $\mathbf{Z}_S^{\theta'} \setminus \mathbf{Z}_S^{\theta'_{\mathsf{link}}} = \mathbf{Z}_S^{\theta'_{\mathsf{twin}}}$. □

Lemma 3. *Let S be an SPN over $\mathbf{X}$, M its corresponding MPN, and $\overline{M}$ its augmented version. Let θ (resp. θ') be the tree path encoded by f_{MPE} (resp. f_{CAT}) on $\overline{M}$ (resp. M). Given a sample $\mathbf{x}^i \sim \mathbf{X}$, then $\overline{\mathsf{L}}^{\theta'}_{|\mathbf{X}} = \mathsf{L}^{\theta}_{|\mathbf{X}}$.*

Proof. It follows by the same construction applied to the proof in Lemma 2. □

Proposition 2. *Let S be an SPN over $\mathbf{X}$, Given a sample $\mathbf{x}^i \sim \mathbf{X}$, then $g_{\mathsf{MPE}}(f_{\mathsf{MPE}}(\mathbf{x}^i)) = g_{\mathsf{CAT}}(f_{\mathsf{CAT}}(\mathbf{x}^i))$.*

Proof. Let θ be the tree path encoded by f_{CAT} on M, θ' be the tree path encoded by f_{MPE} on $\overline{M}$, and θ'' the tree path decoded by g_S on $\overline{M}$. We have to demonstrate that $\mathbf{x}^i \sim \mathbf{X}$: $\overline{\mathsf{L}}^{\theta''}_{|\mathbf{X}} = \mathsf{L}^{\theta}_{|\mathbf{X}}$ since $f_{\mathsf{CAT}}(\mathbf{x}^i)$ outputs a CAT embedding that represents a tree θ unambiguously and g_{CAT} simply traverses it. If the leaves over $\mathbf{X}$ are the same for $\theta^{''}$ and θ, then the MPE inference assignment for both will be the same. From Proposition 1, we have that $\overline{\mathsf{L}}^{\theta''}_{|\mathbf{X}} = \overline{\mathsf{L}}^{\theta'}_{|\mathbf{X}}$. From Lemma 3, it holds that $\overline{\mathsf{L}}^{\theta'}_{|\mathbf{X}} = \mathsf{L}^{\theta}_{|\mathbf{X}}$. Therefore, $\overline{\mathsf{L}}^{\theta''}_{|\mathbf{X}} = \mathsf{L}^{\theta}_{|\mathbf{X}}$. □

CAT-sparse and CAT-dense embeddings

In the main paper we built our encoding f_{CAT} encoding routine by operating like f_{MPE} but on M only (and not $\overline{M}$. By doing so, assignments to some LVs in $\mathbf{Z}_S$ are undefined in $\tilde{z}^i = f_{\mathsf{CAT}}(\mathbf{x}^i)$. This induced *sparsity* can be implemented by assigning a placeholder value, e.g. -1 to the embedding values in $\tilde{z}^i$ corresponding to the aforementioned undefined LVs.

Instead of leaving these LVs unassigned, we can approximate their values by employing Eq.4 even for the max nodes in M that are not in the induced tree θ grew top-down. By doing so we would obtain a *dense* representation for $\tilde{z}^i$, clearly containing more information that the usual sparse version.

In this Appendix, when we will refer to CAT-sparse embeddings, we are denoting the same CAT embeddings as presented in the main text. On the other hand we will use CAT-dense to refer to the here introduced variant.

In the following Lemma we state and demonstrate how g_{CAT} applied to CAT-sparse and CAT-dense embeddings lead to the same reconstruction over $\mathbf{X}$. Later, in the extended experimental section, we investigate both sparse and dense variants empyrically.

Lemma 4. *Let S be an SPN over $\mathbf{X}$, Given a sample $\mathbf{x}^i \sim \mathbf{X}$, then $g_{\mathsf{CAT}}(f_{\mathsf{CAT\text{-}sparse}}(\mathbf{x}^i)) = g_{\mathsf{CAT}}(f_{\mathsf{CATdense}}(\mathbf{x}^i))$.*

Proof. Let θ_d (resp. θ_s) be the tree path grown by g_{CAT} when applied to $f_{\mathsf{CAT\text{-}dense}}(\mathbf{x}^i) = \tilde{\mathbf{z}}^i_d$ (resp. $f_{\mathsf{CAT\text{-}sparse}}(\mathbf{x}^i) = \tilde{\mathbf{z}}^i_s$). Note that $\tilde{\mathbf{z}}^i_d$ differs from $\tilde{\mathbf{z}}^i_s$ only by those entries corresponding to LVs $Z_j \notin \mathbf{Z}^{\theta_s}_S$. These entries are ignored when g_{CAT} materializes θ_s. Therefore the different values in the same entries in $\tilde{\mathbf{z}}^i_d$ do not influence g_{CAT} while growing θ_d. Since θ_d shares the same root of θ_s, they must be equal by construction. □

ACT-full **embeddings**

In addition to ACT embeddings as presented in the main text, comprising activations of *only inner nodes* of an MPN M, we also investigate the variant—denoted as ACT-full—comprising leaf activation information i.e. when $\mathbf{N} = \mathbf{M}$.

Furthermore, as $\mathbf{e}^i_M$ comprises activations for a leaf n, i.e. $\mathbf{e}^i_{M|n}$, we can employ in g_{ACT} a different decoding routine than MPE inference for RVs in $sc(n)$. We define the decoded state for a leaf n as the configuration over its scope that minimizes some distance D over its value through $\hat{\phi}_n$ and $\mathbf{e}^i_{M|n}$:

$$\tilde{\mathbf{x}}_{|\mathsf{sc}(n)} = g_{\mathsf{ACT}}(\mathbf{e}^i_{M|n}) = \operatorname*{argmin}_{\mathbf{u}\sim\mathsf{sc}(n)} D(\hat{\phi}_n(\mathbf{u})||\mathbf{e}^i_{M|n}). \quad (1)$$

This is actually a generalization of our previous decoding routine. Consider the case in which $\mathbf{e}^i_{M|n}$ is the MPE probability value for leaf n, then $\tilde{\mathbf{x}}_{|\mathsf{sc}(n)}$ would be its MPE state. In our experiments with ACT embeddings on discrete data we will employ an L_1 distance as D. Other techniques to obtain perfect decoding with non-bijective leaf pdfs are possible, e.g. by duplicating and splitting distributions or switching from pdfs to CDFs.

Encoding and decoding routines

Here we provide the pseudocode for our SPAE routines: encoding via f_{CAT} (Algorithm 1) and f_{ACT} (Algorithm 2) and decoding via g_{CAT} (Algorithm 3) and g_{ACT} (Algorithm 4). Algorithm 4 lists also the pseudocode for our decoding routine coping with "missing" embedding values, as introduced in the main paper. Moreover, it specifies how to deal with both ACT- and ACT-full embeddings as defined in the previous section of this Appendix. In all algorithms we deal with an MPN M, derived from an SPN S. We denote with $\mathbf{M}^{\mathbf{e}}$ the set of nodes of an MPN M that are not missing in embedding $\mathbf{e}$, a map $a : \mathbf{M}^{\mathbf{e}} \subseteq \mathbf{M} \rightarrow \{1, \ldots, d\}$ references the index of the component in $\mathbf{e}$ corresponding to node n, i.e. $a(n)$.

Algorithm 1 $f_{\mathsf{CAT}}(M, \mathbf{x}, a)$

```
Input: an MPN $M$ over $\mathbf{X}$, a sample $\mathbf{x}$ to encode, a map $a : \mathbf{M}^{\mathsf{max}} \rightarrow \{0, \ldots, |\mathbf{M}^{\mathsf{max}}| - 1\}$
Output: a CAT embedding $\mathbf{z} \in \mathbb{N}^d$ encoding $\mathbf{x}$ through LVs $\mathbf{Z}_M$
$d \leftarrow |\mathbf{M}^{\mathsf{max}}|$
evaluate $M(\mathbf{x})$ bottom-up
$\mathbf{z} \leftarrow -\mathbf{1}_d$        ▷ $-1$ is the placeholder for missing components
$\mathcal{Q} \curvearrowleft \mathsf{root}(M)$ ▷ top-down traversal of $M$ by using a queue $\mathcal{Q}$
while not $\mathsf{empty}(\mathcal{Q})$ do
    $n \curvearrowleft \mathcal{Q}$                                ▷ process current node
    if $n \in \mathbf{M}^{\mathsf{max}}$ then                               ▷ max node
        $c_i \leftarrow \mathrm{argmax}_{k\in\{0,\ldots,|\mathsf{ch}(n)|\}} w_{nc_k} M_{c_k}(\mathbf{x})$
        $z_{a(n)} = i$
        $\mathcal{Q} \curvearrowleft c_i$   ▷ for $\mathsf{CAT}_{\mathsf{dense}}$, all children are followed, i.e. $\forall c \in \mathsf{ch}(n) : \mathcal{Q} \curvearrowleft c$
    else if $n \in \mathbf{M}^{\otimes}$ then                          ▷ product node
        $\forall c \in \mathsf{ch}(n) : \mathcal{Q} \curvearrowleft c$
return $\mathbf{z}$
```

Algorithm 2 $f_{\mathsf{ACT}}(M, \mathbf{N}, \mathbf{x}, a)$

```
Input: an MPN $M$ over $\mathbf{X}$, a subset of the nodes in $M$, $\mathbf{N} \subset \mathbf{M}$, a sample $\mathbf{x}$ to encode, a map $a : \mathbf{N} \rightarrow \{0, \ldots, |\mathbf{N}| - 1\}$
Output: an ACT embedding $\mathbf{e} \in \mathbb{R}^{|\mathbf{N}|}$ encoding $\mathbf{x}$ according to $M$
$d \leftarrow |\mathbf{N}|$
evaluate $M(\mathbf{x})$ bottom-up
$\mathbf{e} \leftarrow \mathbf{0}_d$
for $n \in \mathbf{n}$ do
    $e_{a(n)} \leftarrow M_n(\mathbf{x})$
return $\mathbf{e}$
```

Algorithm 3 $g_{\mathsf{CAT}}(M, \mathbf{z}, a)$

```
Input: an MPN $M$ over $\mathbf{X}$, a CAT embedding $\mathbf{z} \in \mathbb{N}^d$ and a map $a : \mathbf{M}^{\mathsf{max}} \rightarrow \{0, \ldots, |\mathbf{M}^{\mathsf{max}}| - 1\}$
Output: a sample $\tilde{\mathbf{x}} \sim \mathbf{X}$ decoded from $\mathbf{z}$, according to $M$
$\tilde{\mathbf{x}} \leftarrow \mathbf{0}_{|\mathbf{X}|}$
$\mathcal{Q} \curvearrowleft \mathsf{root}(M)$ ▷ top-down traversal of $M$ by using a queue $\mathcal{Q}$
while not $\mathsf{empty}(\mathcal{Q})$ do
    $n \curvearrowleft \mathcal{Q}$                                ▷ process current node
    if $n \in \mathbf{M}^{\mathsf{max}}$ then                               ▷ max node
        $c_{\mathsf{max}} \leftarrow c_{z_{e(n)}}$ such that $c_{z_{e(n)}} \in \{c_i | c_i \in \mathsf{ch}(n) \wedge i = 0, \ldots, |\mathsf{ch}(n)| - 1\}$
        $\mathcal{Q} \curvearrowleft c_{\mathsf{max}}$
    else if $n \in \mathbf{M}^{\otimes}$ then                          ▷ product node
        $\forall c \in \mathsf{ch}(n) : \mathcal{Q} \curvearrowleft c$
    else                                                    ▷ leaf node
        $\tilde{\mathbf{x}}_{|\mathsf{sc}(n)} \leftarrow \mathrm{argmax}_{\mathbf{u}\sim\mathsf{sc}(n)} M_n(\mathbf{u})$
return $\tilde{\mathbf{x}}$
```

Datasets

The 10 datasets employed come from the freely accessible MULAN[1], MEKA[2], and LABIC[3] repositories. They are real world *standard benchmarks* for MLC from text, im-

[1] http://mulan.sourceforge.net/.

[2] http://meka.sourceforge.net/.

[3] http://computer.njnu.edu.cn/Lab/LABIC/LABIC_Software.html.

Algorithm 4 $g_{\mathsf{ACT}}(M, \mathbf{e}, a)$

```
Input: an MPN $M$ over $\mathbf{X}$, an ACT embedding $\mathbf{e} \in \mathbb{R}^d$ and a map $a : \mathsf{M}^{\mathsf{e}} \subseteq \mathsf{M} \rightarrow \{0, \ldots, d-1\}$
Output: a sample $\tilde{\mathbf{x}} \sim \mathbf{X}$ decoded from $\mathbf{e}$, according to $M$
$\tilde{\mathbf{x}} \leftarrow \mathbf{0}_{|\mathbf{X}|}$
$\mathcal{Q} \curvearrowleft \mathsf{root}(M)$ ▷ top-down traversal of $M$ by using a queue $\mathcal{Q}$
while not empty($\mathcal{Q}$) do
    $n \curvearrowleft \mathcal{Q}$                                   ▷ process current node
    if $n \in \mathsf{M}^{\mathsf{max}}$ then                              ▷ max node
        $c_{\mathsf{max}} \leftarrow \mathrm{argmax}_{c \in \mathsf{ch}(n)} w_{nc} v_c$ such that $v_c \leftarrow e_{a(c)}$ if $c \in \mathsf{M}^{\mathsf{e}}$ else $\max_{\mathbf{u} \sim \mathsf{sc}(c)} M_c(\mathbf{u})$
        $\mathcal{Q} \curvearrowleft c_{\mathsf{max}}$
    else if $n \in \mathsf{M}^{\otimes}$ then                          ▷ product node
        $\forall c \in \mathsf{ch}(n) : \mathcal{Q} \curvearrowleft c$
    else                                                  ▷ leaf node
        if $n \in \mathsf{M}^{\mathsf{e}}$ then
            $\tilde{\mathbf{x}}_{\mathsf{sc}(n)} \leftarrow \mathrm{argmin}_{\mathbf{u} \sim (\mathsf{sc}(n))} D(\phi_n(\mathbf{u}) || e_{a(n)})$
        else                                    ▷ (inner embedding)
            $\tilde{\mathbf{x}}_{|\mathsf{sc}(n)} \leftarrow \mathrm{argmax}_{\mathbf{u} \sim \mathsf{sc}(n)} M_n(\mathbf{u})$
return $\tilde{\mathbf{x}}$
```

age, sound and biological domains. Subsets of them have been also used in (Dembczyński et al. 2012; Antonucci et al. 2013; Kong, Ng, and Zhou 2013). They have been binarized as in (Di Mauro, Vergari, and Esposito 2016) by implementing the Label-Attribute Interdependence Maximization (LAIM) (Cano et al. 2016) discretization method[4].

Table 1 reports the information about the adopted datasets, where N, M and L represent the number of attributes, instances, and possible labels respectively. They are divided into five standard folds. Furthermore, for each dataset $\mathcal{D} = \{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^{M}$ the following statistics are also reported: *label cardinality*: $\mathsf{card}(\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{L} y_j^i$, *label density*: $\mathsf{dens}(\mathcal{D}) = \frac{\mathsf{card}(\mathcal{D})}{L}$ and *distinct labels*: $\mathsf{dist}(\mathcal{D}) = |\{\mathbf{y} | \exists(\mathbf{x}^i, \mathbf{y}) \in \mathcal{D}\}|$.

Table 1: Dataset descriptions: number of attributes (N), instances (M), and labels (L).

	domain	N	M	L	card	dens	dist
Arts	text	500	7484	26	1.653	0.063	599
Business	text	500	11214	30	1.598	0.053	233
Cal	music	68	502	174	26.043	0.149	502
Emotions	music	72	593	6	1.868	0.311	27
Flags	images	19	194	7	3.391	0.484	54
Health	text	500	9205	32	1.644	0.051	335
Human	biology	440	3106	14	1.185	0.084	85
Plant	biology	440	978	12	1.078	0.089	32
Scene	images	294	2407	6	1.073	0.178	15
Yeast	biology	103	2417	14	4.237	0.302	198

Learning SPNs

To learn the structure and weights of our SPNs (and hence MPNs), we employ LearnSPN-b (Vergari, Di Mauro, and Esposito 2015), a variant of LearnSPN. LearnSPN-b splits the data matrix slices always into two, both when performing row clustering and checking for RVs independence. With the purpose of slowing down the greedy hierarchical clustering processes, it has proven to obtain simpler and deeper networks without limiting their expressiveness as density estimators. Based on the datasets statistics reported above in Appendix , we define the same ranges for LearnSPN-b hyperparameters both when we learn our SPNs for the $\mathbf{X}$ and the $\mathbf{Y}$. We set the G-test independence test threshold to 5, we limit the minimum number of instances in a slice to split to 10 and we performed a grid search for the best leaf distribution Laplace smoothing value in $\{0.1, 0.2, 0.5, 1.0, 2.0\}$. Note that varying the smoothing coefficient does not require the structure to be learned again. We perform all computations in the $\log$ space to avoid numerical issues.

For the visualizations provided, we employ the binarized version of MNIST provided in (Larochelle et al. 2007). To learn an SPN on it we use again LearnSPN-b by setting the G-test independence test threshold to 20 and the instance threshold to 50, in order to reduce the network size. We then applied the same grid search as above for Laplace smoothing.

For the topic visualization, we used the NIPS[5] bag-of-words dataset. It contains 1,500 documents with a vocabulary above 12k words. We considered the 100 most frequent words and learned a Poisson SPN (PSPN) (Molina, Natarajan, and Kersting 2017) with a minimum number of instances per leaf of 5 documents and independency test $\alpha = 0.1$. This produces a PSPN of 3187 nodes. We scan the network for product nodes having children nodes with a scope length greater than 7. Additionally we retrieve another product node at the same level in the hierarchy as the first one to show different topics variations over similar scopes. Finally, we visualize the MPE word counts for those 4 nodes as wordcloud maps using the python library[6] making each word count proportional to the correponding string size in the map.

SPN model statistics

Statistics for the reference SPN models learned with LearnSPN-b on the $\mathbf{X}$ RVs only are reported in Table 2. Their average (and standard deviations) values over the dataset folds provide information about the network topology and quality: how many nodes are in there (edges + 1), how are they divided into leaves and sum and products and their max depth (as the longest path from the root). The same statistics are reported for the SPNs over RVs $\mathbf{Y}$, then turned in MPNs, in Table 3.

The length of the ACT embeddings extracted from such models is the number of inner nodes from Table 2 for the inner embeddings over $\mathbf{X}$. For ACT-full embeddings over RVs $\mathbf{Y}$, their size can be looked up as the number of all nodes from Table 3. For both CAT-sparse and CAT-dense

[4] The processed versions are freely available at `https://github.com/nicoladimauro/dcsn`.

[5] `https://archive.ics.uci.edu/ml/datasets/bag+of+words`

[6] `https://github.com/amueller/word_cloud`

Table 2: Statistics for the SPN models learned by LearnSPN-b on the $\mathbf{X}$ RVs on the ten datasets. Average and standard deviation values across the five folds reported.

	edges	depth	leaves	inner	sum	prod	scopes
Arts	9241.8	20.2	7412.6	1830.2	605.4	1224.8	1053.6
	±175.4	±1.1	±151.7	±56.2	±19.5	±36.8	±18.6
Business	8569.6	23.4	7029.0	1541.6	507.6	1034.0	971.4
	±228.8	±1.7	±170.7	±73.7	±24.7	±49.1	±22.6
Cal	263.0	7.0	219.8	44.2	14.6	29.6	82.6
	±17.0	±0.0	±18.5	±3.6	±1.1	±2.5	±1.1
Emotions	985.8	13.4	724.6	262.2	87.2	175	147.4
	±36.4	±0.9	±20.2	±20.2	±6.9	±13.3	±4.7
Flags	74.0	7.0	54.6	20.4	6.8	13.6	25.6
	±3.9	±0.0	±1.5	±2.5	±1.7	±0.1	±0.5
Health	7209.2	22.2	5917.0	1293.2	427.8	865.4	899.8
	±249.3	±1.1	±247.4	±21.4	±6.4	±15.0	±7.9
Human	15356.6	19.0	11828.6	3529.0	1170.6	2358.4	1479.2
	±228.9	±1.4	±133.8	±98.7	±32.0	±66.8	±28.8
Plant	3493.8	13.8	2741.8	753.0	247.4	505.6	681.8
	±58.6	±1.1	±42.1	±32.8	±10.7	±22.15	±8.9
Scene	14814.6	15.8	11542.6	3273.0	1089.8	2183.2	1025.6
	±169.1	±1.1	±122.9	±59.9	±20.0	±40.0	±21.8
Yeast	2215.0	18.2	1611.2	604.8	199.6	405.2	262.2
	±96.1	±1.1	±72.4	±28.3	±9.4	±19.0	±3.9

embeddings, on the other hand, their size can found as the number of sum nodes in Tables 2 and 3.

Table 3: Statistics for the SPN models learned by LearnSPN-b on the $\mathbf{Y}$ RVs on the ten datasets. Average and standard deviation values across the five folds reported.

	edges	depth	leaves	inner	sum	prod	scopes
Arts	495.0	17.8	340.6	155.4	50.2	105.2	74.4
	±28.5	±1.1	±21.6	±10.8	±3.9	±7.0	±3.5
Business	414.0	18.6	292.6	122.4	40.2	82.2	65.8
	±18.0	±0.9	±18.1	±5.7	±1.8	±3.9	±2.5
Cal	1840.4	12.6	1428.0	413.4	137.8	275.6	293.6
	±51.2	±0.9	±25.8	±29.6	±9.8	±19.7	±7.8
Emotions	39.2	7.0	24.6	15.6	5.2	10.4	11.2
	±4.5	±0.0	±2.2	±2.5	±1.7	±0.1	±0.8
Flags	25.2	5.4	17.8	8.4	2.8	5.6	9.6
	±4.2	±0.9	±1.8	±2.5	±0.8	±1.7	±0.5
Health	504.2	17.4	355.0	150.2	49.2	101.0	76.4
	±21.6	±1.7	±17.5	±7.6	±2.4	±5.3	±2.1
Human	118.2	14.2	85.2	34.0	11.0	23.0	25.0
	±8.2	±1.1	±5.4	±3.8	±1.6	±2.2	±1.6
Plant	80.0	14.6	57.0	24.0	8.0	16.0	20
	±8.2	±2.2	±6.2	±2.1	±0.7	±1.4	±0.7
Scene	38.4	9.0	24.4	15.0	5.0	10.0	11.0
	±0.5	±0.0	±0.5	±0.0	±0.0	±0.0	±0.0
Yeast	382.4	14.6	241.2	142.2	46.6	95.6	46.4
	±33.4	±0.9	±22.6	±12.8	±4.1	±8.8	±4.2

More experiment details and results

Training details

Learning linear predictors We learn to predict each target feature independently from the others, both when we employ the L_2-regularized logistic regressor (LR) to predict RV $\mathbf{Y}$ directly or when we use a ridge regressor (RR) to predict label embeddings.

To select the best value for the regularization parameter we will perform a grid search for LR in the space $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and for RR in the space $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$[7] for each experiment configuration.

Learning RBMs Concerning RBMs, we train them on the $\mathbf{X}$ alone (or on the $\mathbf{Y}$ alone for the kNN experiments) by using the Persistent Constrastive Divergence (PCD) (Marlin et al. 2010) algorithm, leveraging the implementation available in scikit-learn. For the weight learning hyperparameters we run a grid search for the learning rate in $\{0.1, 0.01\}$, the batch size in $\{20, 100\}$ and let the number of epochs range in $\{10, 20, 30\}$ since no early stopping criterion was available. We then select the best models according to their pseudo-log likelihoods. To generate embeddings from RBMs, we evaluate the conditional probabilities of the hidden units given each sample. To make the comparison fairer we transform these values in the log domain in the same way we do for our SPN and MPN representations.

Learning MADEs For MADEs, following the experimentation reported in (Germain et al. 2015), we employ adadelta to schedule the learning rate during training and fix its decay rate at 0.95; we set the max number of worsening iterations on the validation set to 30 as for RBMs and we employed a batch size of 100 samples. We initialize the weights by employing an SVD-based init scheme.

Other hyperparameters are optimized by a log-likelihood-wise grid search. The gradient dumping coefficient is searched in $\{10^{-5}, 10^{-7}, 10^{-9}\}$, and we employ once the shuffling of mask and orders. Both ReLus and softplus functions are explored as the non-linearities employed for each hidden neuron. We employ a MADE openly available implementation, ported to python3[8].

We learn architectures of three hidden layers comprising 500 and 1000 (resp. 200 and 500) hidden neurons each for the $\mathbf{X}$ (resp. $\mathbf{Y}$). For each reference model, we extract $\mathbf{E_X}$ embeddings by evaluating all the hidden layer activations ($d = 1500$ and $d = 3000$); for the $\mathbf{E_Y}$ case, however, only the last hidden layer embeddings are actually exploited for the prediction ($d = 200$ and $d = 500$).

Learning SAE models Following the experiments in (Wicker, Tyukin, and Kramer 2016), we perform a grid search for the following hyperparameters: the number of layers is chosen in $\{2, 3, 4\}$ and the compression factor $\gamma \in \{0.7, 0.8, 0.9\}$. We employ the Java implementation freely available in MEKA.

We were not able to properly learn SAEs for one dataset, Cal, for all measures, as a numerical error in MEKA prevented the model evaluation, thereby we removed it in the result Table.

We were also not able to train them on the $(\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\mathsf{LR}}{\Rightarrow} \mathbf{Y}$ and hence $((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\mathsf{RR}}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$ scenarios because the learned representations were not available through MEKA.

Learning contractive (CAE) and denoising (DAE) autoencoders For both CAE and DAE models, for all settings, we learned architectures with up to three layers for both the encoder and the decoder network.

[7] We leverage the python implementations for LR and RR from the scikit-learn package (`http://scikit-learn.org/`). Note that in scikit-learn the grid parameter for LR has to be interpreted as an inverse regularization coefficient.

[8] `https://github.com/arranger1044/MADE`.

In particular, we looked at architectures with layers each one made of 500 hidden units for the $(\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\mathsf{LR}}{\Rightarrow} \mathbf{Y}$ scenario, and 200 hidden units in the case of $(\mathbf{X} \overset{\mathsf{RR}}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$. For the last scenario, $((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\mathsf{RR}}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$ we exploited the former for the r models and the latter as the t models. We modeled the size of the representation layer by performing a grid search among network layers of size compressed by a factor in $\{0.7, 0.8, 0.9\}$.

To train them, we employed Adam as the optimized for gradient descent up to 1000 epochs, stopping it earlier if no improvement was found after 50 epochs on validation data. We looked for the batch size in $\{20, 100\}$. For CAEs we searched for the contractive coefficient in $\{0.001, 0.01, 0.1\}$ while we explored the space of $0.1, 0.2, 0.3$ as the percentages of flipping a bit (noise) in the input representation for DAEs.

Reconstruction errors

In this Section we provide the detailed results for the reconstructions of the input for our SPNs turned into MPNs for each train and test portion of each dataset, averaged by fold. Table 4 (resp. Table 5) reports the results for architectures trained on the $\mathbf{X}$ (resp. $\mathbf{Y}$) and asked to reconstruct their inputs w.r.t these RVs. Recall that for this task CAT, CAT-dense and ACT provide the same decoding, therefore in the following comparison we refer only to ACT (using MPE leaf decoding) against ACT-full (using the L_1 distance in Eq.1 in this Appendix).

Other results for MLC

JACCARD, HAMMING and EXACT MATCH measures In this Section we report the additional results for the JACCARD and HAMMING measures in Table 6 and Table 7 respectively. Figures 2 and 3 report the resilience of the decoding scheme for missing at random embedding components for the JACCARD and HAMMING measures, respectively. We employ a euclidean 5-nearest neighbor classifier to perform the decoding step on all our models. These results are reported in the table last rows.

References

Antonucci, A.; Corani, G.; Mauá, D. D.; and Gabaglio, S. 2013. An ensemble of bayesian networks for multilabel classification. In *Proceedings of IJCAI*, 1220–1225.

Cano, A.; Luna, J. M.; Gibaja, E. L.; and Ventura, S. 2016. LAIM discretization for multi-label data. *Information Sciences* 330:370–384.

Dembczyński, K.; Waegeman, W.; Cheng, W.; and Hüllermeier, E. 2012. On label dependence and loss minimization in multi-label classification. *MLJ* 88(1):5–45.

Di Mauro, N.; Vergari, A.; and Esposito, F. 2016. Multi-label classification with cutset networks. In *PGM*.

Germain, M.; Gregor, K.; Murray, I.; and Larochelle, H. 2015. MADE: masked autoencoder for distribution estimation. *arXiv* 1502.03509.

Kong, X.; Ng, M. K.; and Zhou, Z. 2013. Transductive multilabel learning via label set propagation. *IEEE Trans. Knowl. Data Eng.* 25(3):704–719.

Larochelle, H.; Erhan, D.; Courville, A.; Bergstra, J.; and Bengio, Y. 2007. An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation. In *Proceedings of the ICML 2007*, 473–480.

Marlin, B. M.; Swersky, K.; Chen, B.; and Freitas, N. D. 2010. Inductive Principles for Restricted Boltzmann Machine Learning. In *AISTATS*, 509–516.

Molina, A.; Natarajan, S.; and Kersting, K. 2017. Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In *AAAI*.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2015. Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning. In *ECML-PKDD*, 343–358.

Wicker, J.; Tyukin, A.; and Kramer, S. 2016. A nonlinear label compression and transformation method for multi-label classification using autoencoders. In *PAKDD*, 328–340.

Table 4: Average train and test JACcard, HAMming and EXAct match scores for the reconstruction of the original $\mathbf{X}$ representations through our SPN models, turned into MPNs, on each dataset.

		score	Arts	Business	Cal	Emotions	Flags	Health	Human	Plant	Scene	Yeast	Average
train	ACT-full	JAC	99.34	79.76	99.94	99.43	100.00	99.53	99.26	99.52	99.44	99.75	97.60
		HAM	99.98	99.65	99.97	99.74	100.00	99.99	99.83	99.93	99.86	99.86	99.88
		EXA	93.95	77.50	98.35	83.47	100.00	96.92	52.39	75.89	56.69	87.67	82.28
	ACT	JAC	39.94	49.18	95.03	81.40	68.09	52.11	60.61	54.96	73.49	89.47	66.43
		HAM	99.08	99.35	97.62	90.32	89.74	99.45	89.85	92.70	87.74	93.79	93.96
		EXA	13.84	28.03	97.37	01.56	08.50	26.90	00.00	00.00	00.00	00.57	17.77
test	ACT-full	JAC	99.41	99.72	99.95	99.48	100.00	99.65	99.33	99.60	99.44	99.78	99.64
		HAM	99.98	99.99	99.98	99.76	100.00	99.99	99.85	99.94	99.76	99.87	99.91
		EXA	94.64	97.19	99.00	83.81	100.00	97.61	55.11	78.62	56.37	88.20	85.06
	ACT	JAC	37.97	48.03	94.56	79.35	66.88	51.08	59.01	99.44	71.74	88.98	69.70
		HAM	99.02	99.31	97.37	89.09	89.34	99.42	89.31	99.76	86.74	93.49	94.29
		EXA	13.20	27.84	29.08	01.85	07.76	26.31	00.00	56.37	00.00	00.49	16.29

Table 5: Average train and test JACcard, HAMming and EXAct match scores for the reconstruction of the original $\mathbf{Y}$ representations through our SPN models, turned into MPNs, on each dataset.

		score	Arts	Business	Cal	Emotions	Flags	Health	Human	Plant	Scene	Yeast	Average
train	ACT-full	JAC	99.94	99.88	98.72	100.00	100.00	99.96	100.00	100.00	100.00	99.95	99.85
		HAM	99.99	99.98	99.80	100.00	100.00	99.99	100.00	100.00	100.00	99.98	99.97
		EXA	99.80	99.67	72.75	100.00	100.00	99.87	100.00	100.00	100.00	99.73	97.18
	ACT	JAC	88.76	92.19	55.94	78.52	70.47	93.25	90.35	89.44	96.31	95.29	85.05
		HAM	98.75	99.34	92.41	91.52	81.82	99.41	98.55	98.42	98.76	98.32	95.73
		EXA	75.77	82.44	00.00	53.41	23.96	84.06	82.30	85.91	92.64	80.89	66.14
test	ACT-full	JAC	99.93	99.86	98.89	100.00	100.00	99.97	100.00	100.00	100.00	99.93	99.86
		HAM	99.99	99.98	99.82	100.00	100.00	99.99	100.00	100.00	100.00	99.97	99.98
		EXA	99.75	99.62	76.50	100.00	100.00	99.89	100.00	100.00	100.00	99.62	97.54
	ACT	JAC	88.42	92.13	51.98	77.89	70.56	93.11	90.39	89.32	99.95	94.81	84.86
		HAM	98.69	99.33	91.52	91.15	81.90	99.39	98.55	98.41	99.98	98.12	95.70
		EXA	75.46	82.34	00.00	52.12	23.90	83.87	82.38	85.79	99.73	79.39	66.50

Table 6: Average test set JACCARD scores. For each setting, best result for a dataset in bold and average ranks in the last column. Results for the 5-NN decoding are shown in the last two row groups.

	Arts	Busin.	Cal	Emot.	Flags	Health	Human	Plants	Scene	Yeast	*RANK*
$\mathbf{X} \overset{p}{\Rightarrow} \mathbf{Y}$											
p : LR	28.48	49.92	17.43	55.78	48.66	41.11	29.44	32.70	65.43	38.59	-
p : CRF$_{\text{SSVM}}$	33.61	73.86	19.98	54.48	56.40	62.10	28.96	31.34	66.15	45.47	-
$(\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\text{LR}}{\Rightarrow} \mathbf{Y}$											
r: RBM$_{h\in\{500,1000,5000\}}$	29.16	48.59	17.52	**58.11**	51.90	40.73	30.33	**33.53**	71.73	39.25	**2.7**
r: MADE$_{h\in\{500,1000\}}$	29.71	48.50	**17.95**	55.92	**54.03**	42.84	28.07	31.53	71.49	**40.79**	3.3
r: CAE$_{\gamma\in\{0.7,0.8,0.9\}}$	28.42	**51.63**	17.78	55.54	47.88	43.98	26.71	32.77	66.68	36.90	4.2
r: DAE$_{\gamma\in\{0.7,0.8,0.9\}}$	29.24	49.58	17.29	56.60	46.39	**44.38**	28.24	32.72	70.84	37.43	3.6
r: SPAE$_{\text{ACT}}$	**31.63**	53.29	17.02	56.84	45.24	43.88	**31.51**	32.23	**71.87**	39.70	**2.7**
r: SPAE$_{\text{CAT}}$	23.99	44.52	15.92	55.93	39.7	31.69	25.56	25.04	66.79	35.49	6.7
r: SPAE$_{\text{CAT-dense}}$	28.07	46.23	16.74	56.87	39.95	40.54	28.64	27.42	70.98	38.42	4.9
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
t: MADE$_{h\in\{200,500\}}, p$: RR	5.08	68.60	20.05	30.02	48.95	40.14	2.58	11.31	15.37	42.82	7.6
t: SAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	39.96	**73.43**	-	49.41	56.51	60.72	33.19	31.37	54.52	**49.35**	4.05
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	30.17	72.32	21.22	49.52	57.31	55.71	25.12	24.09	53.18	46.81	5.4
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	32.62	71.6	22.00	52.29	57.58	56.49	31.31	28.29	55.46	47.39	3.9
t: SPAE$_{\text{ACT-full}}, p$: RR	29.30	**73.43**	20.30	54.30	58.18	57.80	25.86	29.39	61.20	46.83	3.95
t: SPAE$_{\text{ACT}}, p$: RR	35.72	70.53	20.77	52.08	55.86	55.31	27.61	33.07	**69.60**	47.08	4.4
t: SPAE$_{\text{CAT}}, p$: LR	**42.69**	66.12	**23.20**	**54.03**	**55.99**	**61.93**	**37.56**	**38.49**	66.12	44.95	**2.8**
t: SPAE$_{\text{CAT-dense}}, p$: LR	41.58	66.68	21.99	53.06	55.51	60.92	33.92	37.71	62.36	41.41	3.9
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\},h_t\in\{200,500\}}, p$: RR	8.65	68.55	20.20	34.19	48.76	39.50	5.96	12.58	17.92	42.64	7.4
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	28.29	71.41	21.36	48.74	56.54	58.02	20.67	22.56	54.86	46.54	5.5
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	30.68	71.34	22.52	53.94	57.56	58.65	30.56	27.09	60.02	**47.82**	3.7
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT-full}}, p$: RR	33.47	**73.88**	19.52	**54.48**	**57.70**	60.20	28.67	29.37	63.64	46.50	3.6
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT}}, p$: RR	37.64	69.98	20.52	52.50	56.56	59.28	27.82	33.24	65.20	46.05	4.2
r: SPAE$_{\text{CAT}}, t$: SPAE$_{\text{CAT}}, p$: LR	36.04	62.95	22.46	51.1	52.16	57.68	32.15	33.17	66.03	40.69	5.25
r: SPAE$_{\text{CAT-dense}}, t$: SPAE$_{\text{CAT-dense}}, p$: LR	39.05	64.5	22.34	52.29	55.86	59.35	33.44	35.53	67.53	40.69	4.05
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{CAT}}, p$: LR	**42.45**	65.43	**23.39**	53.71	57.21	**62.16**	**38.79**	**39.87**	**70.09**	44.27	**2.3**
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\text{5-NN}} \mathbf{Y}$											
t: RBM$_{h\in\{200,500,1000\}}, p$: RR	17.59	51.20	21.85	**53.73**	**59.19**	38.14	**39.16**	**44.61**	71.07	44.41	4.1
t: MADE$_{h\in\{200,500\}}, p$: RR	37.36	69.04	22.07	47.55	56.79	56.90	32.47	28.66	65.52	45.93	5.6
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	43.66	**73.60**	21.96	52.1	56.15	62.34	38.06	41.03	60.34	47.89	3.3
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	36.39	72.42	20.49	49.57	57.62	53.45	33.01	35.41	65.51	47.07	5.6
t: SPAE$_{\text{ACT-full}}, p$: RR	**45.24**	73.51	21.09	52.96	54.08	61.56	39.05	38.40	**74.22**	48.07	**2.6**
t: SPAE$_{\text{ACT}}, p$: RR	43.11	72.86	20.96	50.79	51.13	59.44	35.50	33.76	73.47	**48.24**	4.5
t: SPAE$_{\text{CAT}}, p$: LR	44.16	71.56	21.03	50.32	51.12	**62.37**	38.04	36.56	70.37	43.94	4.7
t: SPAE$_{\text{CAT-dense}}, p$: LR	41.39	61.35	**24.56**	52.93	52.56	61.06	30.42	32.79	60.85	41.31	5.6
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\text{5-NN}} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\},h_t\in\{200,500\}}, p$: RR	37.04	67.57	21.93	49.24	55.15	58.43	32.27	27.70	68.27	45.63	5.4
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	42.66	73.03	21.44	51.25	55.75	63.64	37.61	**39.95**	60.32	46.64	3.2
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	37.69	71.35	20.5	51.45	**56.78**	56.7	30.26	33.59	67.92	45.87	4.8
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT-full}}, p$: RR	**46.38**	73.90	20.56	53.04	52.16	**63.81**	36.36	36.86	70.27	**47.90**	**2.2**
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT}}, p$: RR	44.57	**73.04**	20.28	50.94	50.84	62.29	34.34	33.28	69.37	47.69	4.0
r: SPAE$_{\text{CAT}}, t$: SPAE$_{\text{CAT}}, p$: LR	38.14	67.32	19.32	47.16	48.39	57.87	32.38	31.13	67.18	40.13	6.9
r: SPAE$_{\text{CAT-dense}}, t$: SPAE$_{\text{CAT-dense}}, p$: LR	38.55	59.69	**25.34**	**53.77**	51.96	58.10	29.67	29.16	66.83	40.61	5.6
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{CAT}}, p$: LR	44.2	70.64	20.06	50.93	52.00	63.2	**38.62**	37.01	**73.32**	44.27	3.9

Table 7: Average test set HAMMING scores. For each setting, best result for a dataset in bold and average ranks in the last column. Results for the 5-NN decoding are shown in the last two row groups.

	Arts	Busin.	Cal	Emot.	Flags	Health	Human	Plants	Scene	Yeast	*RANK*
$\mathbf{X} \overset{p}{\Rightarrow} \mathbf{Y}$											
p : LR	86.67	92.13	65.25	76.70	68.41	92.24	84.72	86.95	87.69	65.17	-
p : CRF$_{\text{SSVM}}$	94.93	97.67	84.28	80.91	71.59	96.96	92.13	91.53	91.09	79.22	-
$(\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\text{LR}}{\Rightarrow} \mathbf{Y}$											
r: RBM$_{h\in\{500,1000,5000\}}$	85.90	92.31	64.95	**78.27**	**68.34**	91.21	85.64	86.72	89.92	65.51	3.1
r: MADE$_{h\in\{500,1000\}}$	84.82	93.00	**65.04**	77.37	68.16	91.48	83.68	85.56	90.14	**70.52**	3.6
r: CAE$_{\gamma\in\{0.7,0.8,0.9\}}$	84.84	93.39	64.62	76.48	65.45	91.62	85.06	87.36	87.95	64.5	4.25
r: DAE$_{\gamma\in\{0.7,0.8,0.9\}}$	84.90	93.74	64.49	77.44	67.46	91.71	84.87	**87.63**	89.34	64.59	3.4
r: SPAE$_{\text{ACT}}$	**86.10**	**94.12**	62.25	77.60	66.87	**91.76**	**87.39**	86.92	**90.20**	67.54	**2.2**
r: SPAE$_{\text{CAT}}$	77.87	89.8	56.93	76.48	59.56	87.12	84.68	80.48	87.93	62.87	6.85
r: SPAE$_{\text{CAT-dense}}$	83.49	92.12	58.55	77.69	62.76	90.67	86.28	83.58	90.12	65.95	4.6
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
t: MADE$_{h\in\{200,500\}}, p$: RR	93.80	97.17	86.14	74.08	67.11	95.82	91.54	91.09	82.86	77.98	5.8
t: SAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	94.27	97.51	-	78.55	70.06	96.64	89.95	88.68	85.88	77.54	5.35
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	94.68	97.51	86.19	79.83	72.10	96.61	91.97	91.53	86.63	79.1	2.75
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	94.61	97.47	86.20	80.22	71.66	96.53	91.91	91.48	89.36	79.05	3.1
t: SPAE$_{\text{ACT-full}}, p$: RR	**94.80**	**97.62**	**86.25**	**80.69**	**73.20**	**96.81**	**92.09**	**91.69**	**91.14**	**79.34**	**1.0**
t: SPAE$_{\text{ACT}}, p$: RR	92.26	97.28	85.62	77.71	70.35	95.78	89.44	89.35	89.67	74.47	6.0
t: SPAE$_{\text{CAT}}, p$: LR	93.45	96.57	83.36	77.86	70.27	96.6	90.39	89.72	88.52	74.56	5.3
t: SPAE$_{\text{CAT-dense}}, p$: LR	93.17	96.39	79.24	77.21	70.22	96.37	89.9	89.46	87.25	71.59	6.7
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\},h_t\in\{200,500\}}, p$: RR	93.86	97.18	86.15	74.39	66.73	95.87	91.53	91.17	83.02	77.89	5.2
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	94.69	97.44	**86.29**	78.78	71.67	96.81	**91.82**	**91.52**	86.45	78.82	2.55
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	94.6	97.46	86.03	**80.25**	72.27	96.68	91.81	91.11	**90.24**	78.82	2.25
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT-full}}, p$: RR	**94.93**	**97.68**	86.01	79.99	**73.36**	**96.96**	91.16	91.00	89.77	**78.94**	**2.2**
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT}}, p$: RR	92.78	97.21	85.27	77.66	70.94	96.27	89.34	89.18	88.20	74.16	5.6
r: SPAE$_{\text{CAT}}, t$: SPAE$_{\text{CAT}}, p$: LR	92.29	96.19	82.75	76.76	67.89	96.16	89.55	88.73	88.48	73.05	6.95
r: SPAE$_{\text{CAT-dense}}, t$: SPAE$_{\text{CAT-dense}}, p$: LR	92.76	96.12	78.81	76.93	70.48	96.16	89.81	89.02	89.02	71.21	6.65
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{CAT}}, p$: LR	93.4	96.49	83.12	77.72	71.46	96.62	90.59	89.93	89.87	74.65	4.6
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\text{5-NN}} \mathbf{Y}$											
t: RBM$_{h\in\{200,500,1000\}}, p$: RR	91.34	95.38	84.83	78.22	**73.13**	94.67	**90.64**	**90.45**	90.15	78.40	4.3
t: MADE$_{h\in\{200,500\}}, p$: RR	92.84	97.05	**85.50**	76.78	71.16	96.23	89.31	87.51	88.17	77.43	5.5
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	93.93	97.59	85.49	78.53	70.2	96.77	90.54	89.58	85.11	78.5	3.1
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	93.34	97.5	85.71	77.04	71.9	95.87	88.44	88.27	86.44	78.27	4.7
t: SPAE$_{\text{ACT-full}}, p$: RR	**94.04**	**97.60**	84.96	**79.42**	69.22	**96.78**	90.56	89.38	**91.15**	**78.88**	**2.1**
t: SPAE$_{\text{ACT}}, p$: RR	93.62	97.52	85.01	78.69	66.28	96.49	89.82	88.36	90.75	78.09	4.1
t: SPAE$_{\text{CAT}}, p$: LR	93.74	97.35	82.25	76.2	64.04	96.75	89.94	88.96	89.8	73.77	5.3
t: SPAE$_{\text{CAT-dense}}, p$: LR	92.66	95.82	82.16	76.9	66.47	96.3	87.69	88.05	86.04	71.38	6.9
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\text{5-NN}} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\},h_t\in\{200,500\}}, p$: RR	92.78	96.94	85.35	77.03	70.82	96.37	89.29	87.51	89.17	76.84	5.1
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	93.77	97.54	85.39	78.39	70.41	96.86	**90.26**	**89.37**	85.29	78.24	2.7
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	93.45	97.44	**85.66**	77.66	**72.77**	96.14	88.25	88.44	87.59	78.24	4.25
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT-full}}, p$: RR	**94.16**	**97.64**	84.81	**79.28**	67.08	**96.95**	89.90	89.06	89.66	**78.54**	**2.0**
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{ACT}}, p$: RR	93.82	97.53	84.78	78.39	66.58	96.75	89.39	88.33	89.25	77.66	3.75
r: SPAE$_{\text{CAT}}, t$: SPAE$_{\text{CAT}}, p$: LR	92.89	96.98	81.67	74.32	63.15	96.26	89.18	87.8	88.78	71.42	6.6
r: SPAE$_{\text{CAT-dense}}, t$: SPAE$_{\text{CAT-dense}}, p$: LR	91.87	95.64	81.85	77.18	66.51	95.92	87.62	87.11	88.26	71.12	7.2
r: SPAE$_{\text{ACT}}, t$: SPAE$_{\text{CAT}}, p$: LR	93.74	97.18	81.94	76.34	65.24	96.8	90.03	88.96	**90.88**	73.98	4.4

Table 8: Average test set EXACT MATCH. For each setting, best result for a dataset in bold and average ranks in the last column. Results for the 5-NN decoding are shown in the last two row groups.

	Arts	Busin.	Cal	Emot.	Flags	Health	Human	Plants	Scene	Yeast	*RANK*
$\mathbf{X} \overset{p}{\Rightarrow} \mathbf{Y}$											
p : LR	7.00	27.31	0.00	23.78	9.81	14.14	10.11	19.23	46.36	7.20	-
p : CRF$_{\mathsf{SSVM}}$	25.33	58.68	0.00	30.18	14.98	49.40	22.54	24.34	60.74	10.72	-
$(\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{\mathsf{LR}}{\Rightarrow} \mathbf{Y}$											
r: RBM$_{h\in\{500,1000,5000\}}$	6.52	24.59	0.00	**27.65**	8.76	13.26	11.01	17.90	56.00	6.87	4.0
r: MADE$_{h\in\{500,1000\}}$	7.79	24.37	0.00	24.45	9.76	17.24	8.53	16.35	**59.78**	**8.06**	3.8
r: CAE$_{\gamma\in\{0.7,0.8,0.9\}}$	6.15	**36.15**	0.00	22.77	**9.78**	19.24	9.41	**20.76**	49.4	5.63	4.2
r: DAE$_{\gamma\in\{0.7,0.8,0.9\}}$	7.67	29.41	0.00	21.59	8.77	**20.24**	9.24	18.71	53.39	6.71	4.0
r: SPAE$_{\mathsf{ACT}}$	**10.37**	30.03	0.00	24.62	8.70	19.88	**15.03**	18.41	56.95	6.95	**2.7**
r: SPAE$_{\mathsf{CAT}}$	7.77	25.02	0.00	26.98	6.24	8.35	9.57	10.64	53.48	6.67	4.7
r: SPAE$_{\mathsf{CAT\text{-}dense}}$	6.51	25.65	0.00	25.47	5.17	14.8	11.85	10.53	57.3	6.54	4.6
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
t: MADE$_{h\in\{200,500\}}, p$: RR	3.30	53.25	0.00	10.28	3.63	28.78	1.96	6.55	11.30	4.10	7.45
t: SAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	25.70	56.51	-	22.11	14.51	45.23	23.08	24.75	45.37	12.41	4.25
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	22.15	57.00	0.00	26.81	10.82	42.64	19.64	19.24	45.29	12.09	5.55
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	25.41	56.37	0.00	25.98	13.96	44.22	24.25	26.29	52.89	14.11	3.95
t: SPAE$_{\mathsf{ACT\text{-}full}}, p$: RR	22.45	**58.32**	0.00	**29.51**	**15.46**	46.27	21.34	23.72	56.54	12.04	3.7
t: SPAE$_{\mathsf{ACT}}, p$: RR	25.18	54.50	0.00	25.97	13.44	38.79	23.66	31.29	**66.51**	12.04	4.65
t: SPAE$_{\mathsf{CAT}}, p$: LR	**34.45**	52.05	0.00	29.36	13.95	**49.27**	**33.42**	**36.71**	62.98	**16.01**	**2.45**
t: SPAE$_{\mathsf{CAT\text{-}dense}}, p$: LR	31.37	50.71	0.00	27.5	13.44	45.78	29.24	35.99	59.29	11.75	4.0
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{g_t} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\}, h_t\in\{200,500\}}, p$: RR	5.17	53.29	0.00	9.94	3.63	28.09	3.28	5.22	10.84	3.93	7.35
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	22.31	56.1	0.00	25.13	14.48	45.14	17.2	19.13	45.26	10.76	5.45
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	25.03	55.96	0.00	**28.84**	10.33	46.22	22.19	25.99	55.8	13.95	4.45
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{ACT\text{-}full}}, p$: RR	**35.98**	**57.79**	0.00	25.46	7.24	**50.13**	28.75	34.25	63.60	14.81	2.85
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{ACT}}, p$: RR	31.93	56.03	0.00	23.44	6.25	47.58	25.94	30.06	61.36	13.16	4.65
r: SPAE$_{\mathsf{CAT}}, t$: SPAE$_{\mathsf{CAT}}, p$: LR	28.35	49.31	0.00	26.65	9.28	44.40	28.47	31.50	62.94	12.70	4.75
r: SPAE$_{\mathsf{CAT\text{-}dense}}, t$: SPAE$_{\mathsf{CAT\text{-}dense}}, p$: LR	29.03	48.64	0.00	26.49	12.94	43.67	28.76	33.96	64.32	11.30	4.35
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{CAT}}, p$: LR	34.52	51.37	0.00	28.68	**15.00**	48.98	**34.55**	**38.15**	**66.89**	**15.69**	**2.15**
$(\mathbf{X} \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\mathsf{5\text{-}NN}} \mathbf{Y}$											
t: RBM$_{h\in\{200,500,1000\}}, p$: RR	10.06	20.11	0.00	**27.48**	**17.00**	24.74	33.03	**42.53**	67.96	11.34	4.05
t: MADE$_{h\in\{200,500\}}, p$: RR	25.08	50.82	0.00	17.36	15.98	43.55	24.50	26.08	60.24	8.68	5.85
t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	31.43	**58.20**	0.00	26.31	14.42	48.39	32.3	37.64	52.98	14.19	3.05
t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	23.08	54.74	0.00	18.38	12.86	30.54	20.13	28.34	53.77	8.94	6.05
t: SPAE$_{\mathsf{ACT\text{-}full}}, p$: RR	34.46	57.49	0.00	25.46	8.31	47.78	**33.16**	35.49	**69.75**	14.52	**2.85**
t: SPAE$_{\mathsf{ACT}}, p$: RR	29.79	56.02	0.00	22.93	5.71	43.78	28.94	30.17	67.59	12.90	4.75
t: SPAE$_{\mathsf{CAT}}, p$: LR	**34.99**	56.29	0.00	25.32	10.82	**50.20**	31.46	32.51	67.14	**14.94**	3.15
t: SPAE$_{\mathsf{CAT\text{-}dense}}, p$: LR	25.21	41.00	0.00	24.63	9.85	43.03	18.10	26.09	47.99	11.18	6.25
$((\mathbf{X} \xrightarrow{f_r} \mathbf{E_X}) \overset{p}{\Rightarrow} (\mathbf{Y} \xrightarrow{f_t} \mathbf{E_Y})) \xrightarrow{\mathsf{5\text{-}NN}} \mathbf{Y}$											
r, t: MADE$_{h_r\in\{500,1000\}, h_t\in\{200,500\}}, p$: RR	25.53	49.79	0.00	20.23	**15.92**	44.87	24.53	25.69	63.23	10.21	5.55
r, t: CAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	30.91	57.72	0.00	**27.83**	10.8	49.24	30.72	**36.20**	52.98	13.86	3.25
r, t: DAE$_{\gamma\in\{0.7,0.8,0.9\}}, p$: RR	25.25	54.24	0.00	18.56	12.37	35.53	18.45	28.76	57.01	8.36	6.05
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{ACT\text{-}full}}, p$: RR	**35.98**	**57.79**	0.00	25.46	7.24	50.13	28.75	34.25	63.60	14.81	2.85
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{ACT}}, p$: RR	31.93	56.03	0.00	23.44	6.25	47.58	25.94	30.06	61.36	13.16	4.45
r: SPAE$_{\mathsf{CAT}}, t$: SPAE$_{\mathsf{CAT}}, p$: LR	29.59	52.80	0.00	22.27	6.20	45.70	27.15	27.30	63.86	12.58	5.15
r: SPAE$_{\mathsf{CAT\text{-}dense}}, t$: SPAE$_{\mathsf{CAT\text{-}dense}}, p$: LR	22.29	39.59	0.00	27.32	11.37	38.05	17.55	21.68	55.13	10.60	6.15
r: SPAE$_{\mathsf{ACT}}, t$: SPAE$_{\mathsf{CAT}}, p$: LR	34.88	54.98	0.00	25.98	9.25	**50.28**	**32.10**	32.72	**69.97**	**15.93**	**2.25**

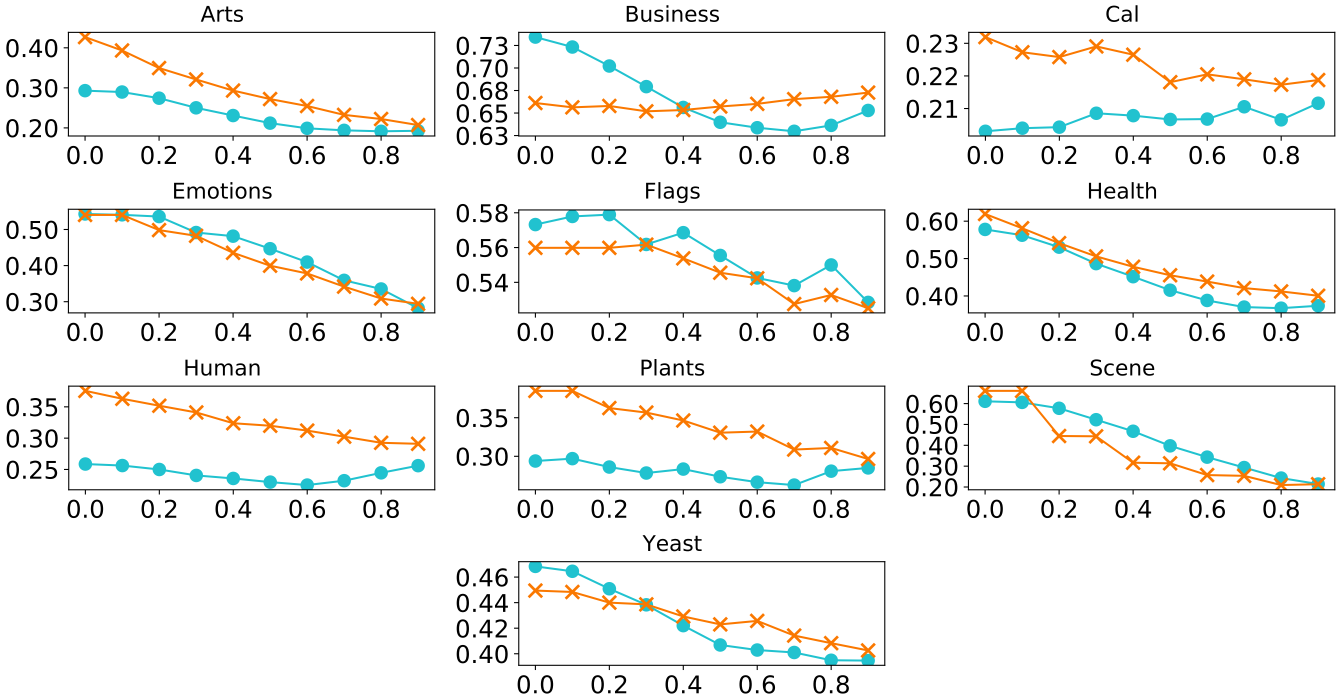

Figure 2: Average JACCARD scores (on the y axis) while imputing different percentages (on the x axis) of missing-at-random embedding components by employing ACT (blue circles) or CAT (orange crosses). Best viewed in colors.

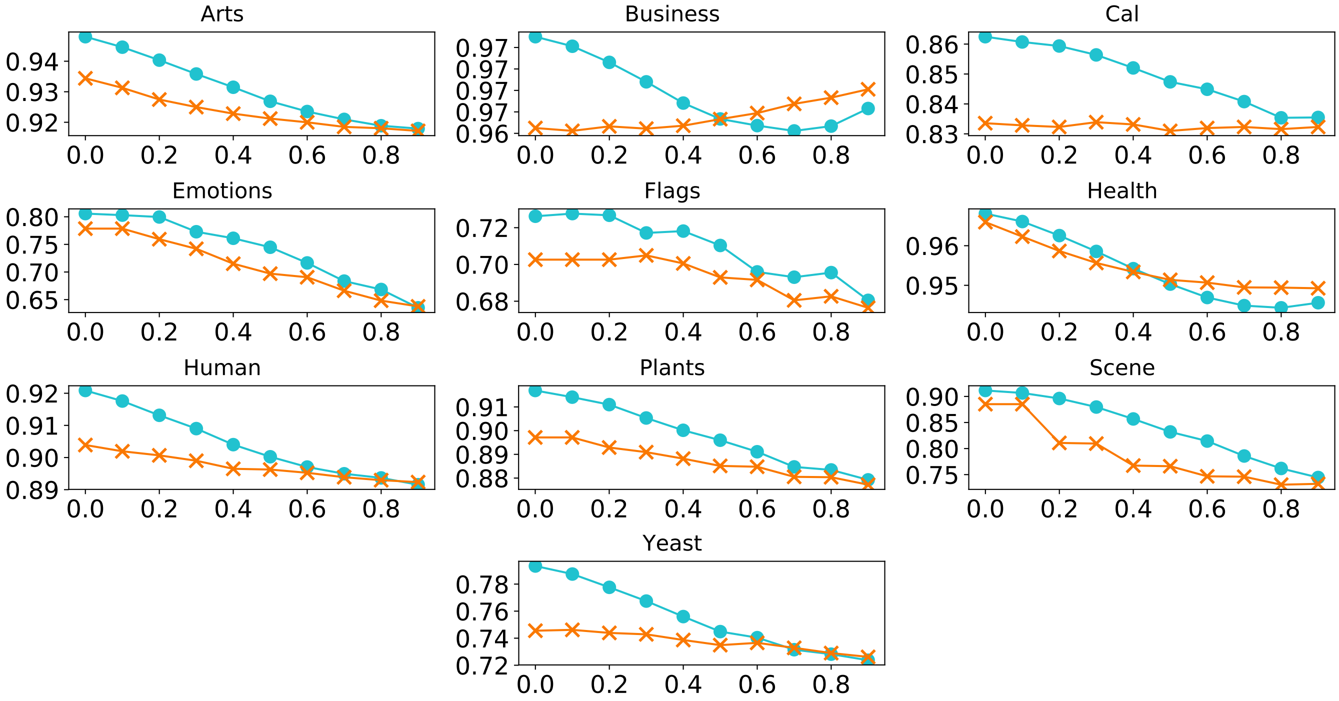

Figure 3: Average HAMMING scores (on the y axis) while imputing different percentages (on the x axis) of missing-at-random embedding components by employing ACT (blue circles) or CAT (orange crosses). Best viewed in colors.