Testing using Privileged Information by Adapting Features with Statistical Dependence – Supplemental

Kwang In Kim
UNIST

James Tompkin
Brown University

This supplemental material provides additional related work discussion (Sec. 1), additional background information to the Hilbert-Schmidt independence criterion used in our TUPI algorithm (Sec. 2.1) and in the visual attributes rank learning problem on which we evaluate our TUPI approach (Sec. 2.2), and a brief summary of our denoising algorithm with an algorithm description (Sec. 3). We also provide further experimental details and results including tests of statistical significance, parameter sensitivity, and simple failure cases (Sec. 4), plus additional future work ideas (Sec. 6). Lastly, we present details of our adaptations of related baselines (Sec. 5): of Kim et al.'s algorithm [6] which only applies to one-dimensional test time features (Sec. 5.1); and of the CoConut algorithm proposed by Khamis and Lampert [5] (Sec. 5.2) for TUPI in visual attribute ranking problems, and our approaches to combine our algorithm with CoConut to exploit these two complementary algorithms for increased performance (Sec. 5.3). Some contents from the main paper are reproduced so that this document is self-contained.

1. Additional related work details

In the main paper (Section 1), we discuss how the TUPI problem is related to—but different from—works in semi-supervised learning, in multi-task learning, and in predictor combination.

Domain adaptation is another related class of work to TUPI: Here, an estimator trained on a data domain $\mathcal{G}$ equipped with a probability distribution $\mathbb{P}_{\mathbf{g}}$ is tested on data generated from an updated probability distribution $\mathbb{P}'_{\mathbf{g}}$ on the same domain $\mathcal{G}$. This setting leads to new algorithms enabling adaptation of estimators in test-time using data sampled from $\mathbb{P}'_{\mathbf{g}}$ as auxiliary information [11]. However, these algorithms focus on modeling the change of distributions in the same domain. In contrast, in our TUPI setting, we use test time data sampled from multiple heterogeneous feature domains $\{\mathcal{H}_i\}$ and thus our contribution is complementary to domain adaptation as a problem.

1.1. Related HSIC applications

The Hilbert-Schmidt independence criterion (HSIC) has been successfully applied in clustering [13] and domain adaptation [15]. Particularly relevant works are Song et al.'s feature selection algorithm [12], which receives a set of features and task-specific labels as random variables and outputs the most statistically relevant features measured by HSIC, plus Gevaert et al.'s kernel learning framework that tunes the kernel parameters to maximize the dependence between the task labels and features (kernels) [2]. For regression, instead of maximizing the dependence between the prediction and features, Mooij et al. minimize the dependence between the features and the residuals—the deviations between the estimated function values and the observed ground truths—to enable regression independently of the unknown noise generation process [8]. Our approach is inspired by these algorithms; however, as they are designed for use in training, they cannot be straightforwardly applied to the TUPI scenario without non-trivial adaptation.

2. Additional background

2.1. The Hilbert-Schmidt independence criterion

Suppose we have two data spaces $\mathcal{V}$ and $\mathcal{W}$, equipped with joint probability distribution $\mathbb{P}_{\mathbf{vw}}$ and marginals $\mathbb{P}_{\mathbf{v}}$ and $\mathbb{P}_{\mathbf{w}}$, respectively. For $\mathcal{V}$, we define a separable reproducing kernel Hilbert space (RKHS) $\mathcal{K}_{\mathbf{v}}$ of functions characterized by the feature map $\phi : \mathcal{V} \to \mathcal{K}_{\mathbf{v}}$ and the positive definite kernel function $k_{\mathbf{v}}(\mathbf{v}, \mathbf{v}') := \langle \phi(\mathbf{v}), \phi(\mathbf{v}') \rangle$.[1] The RHKS $\mathcal{K}_{\mathbf{w}}$, and the corresponding kernel $k_{\mathbf{w}}$ and feature map ψ are similarly defined for $\mathcal{W}$. The cross-covariance operator associated with the joint probability distribution $\mathbb{P}_{\mathbf{vw}}$ is a linear operator $C_{\mathbf{vw}} : \mathcal{K}_{\mathbf{v}} \to \mathcal{K}_{\mathbf{w}}$ which generalizes the cross-covariance matrix in Euclidean spaces:

$$C_{\mathbf{vw}} = \mathbb{E}_{\mathbf{vw}} \left[(\phi(\mathbf{v}) - \mathbb{E}_{\mathbf{v}} \phi(\mathbf{v})) \otimes (\psi(\mathbf{w}) - \mathbb{E}_{\mathbf{w}} \psi(\mathbf{w})) \right],$$

where $\otimes$ is the tensor product. Given this operator, the Hilbert-Schmidt independence criterion (HSIC) associated with $\mathcal{K}_{\mathbf{v}}$, $\mathcal{K}_{\mathbf{w}}$, and $\mathbb{P}_{\mathbf{vw}}$ is defined as the Hilbert-Schmidt norm of $C_{\mathbf{vw}}$ which generalizes the Frobenius norm defined

[1] A separable Hilbert space has a countable orthonormal basis facilitating the introduction of Hilbert-Schmidt operators.

for matrices to operators [3]:

$$\begin{aligned}\text{HSIC}(\mathcal{K}_{\mathbf{v}},\mathcal{K}_{\mathbf{w}},\mathbb{P}_{\mathbf{vw}}) &= \|C_{\mathbf{vw}}\|^2_{HS} \\ &= \mathbb{E}_{\mathbf{vv'ww'}}[k_{\mathbf{v}}(\mathbf{v},\mathbf{v}')k_{\mathbf{w}}(\mathbf{w},\mathbf{w}')] \\ &+ \mathbb{E}_{\mathbf{vv'}}[k_{\mathbf{v}}(\mathbf{v},\mathbf{v}')]\mathbb{E}_{\mathbf{w},\mathbf{w}'}[k_{\mathbf{w}}(\mathbf{w},\mathbf{w}')] \\ &- 2\mathbb{E}_{\mathbf{vw}}\left[\mathbb{E}_{\mathbf{v}'}[k_{\mathbf{v}}(\mathbf{v},\mathbf{v}')]\mathbb{E}_{\mathbf{w}'}[k_{\mathbf{w}}(\mathbf{w},\mathbf{w}')]\right]. \quad (1)\end{aligned}$$

HSIC is defined as long as the kernels $k_{\mathbf{v}}$ and $k_{\mathbf{w}}$ are bounded, and it is always non-negative [3]. Furthermore, when $k_{\mathbf{v}}$ and $k_{\mathbf{w}}$ are *universal* [14], such as when they are Gaussian, HSIC is zero only when the two distributions $\mathbb{P}_{\mathbf{v}}$ and $\mathbb{P}_{\mathbf{w}}$ are independent: HSIC is the maximum mean discrepancy (MMD) between the joint probability measure $\mathbb{P}_{\mathbf{vw}}$ and the product of marginals $\mathbb{P}_{\mathbf{v}}\mathbb{P}_{\mathbf{w}}$ computed with the product kernel $k_{\mathbf{vw}} = k_{\mathbf{v}} \otimes k_{\mathbf{w}}$ [9]:

$$\begin{aligned}\text{HSIC}(\mathcal{K}_{\mathbf{v}},\mathcal{K}_{\mathbf{w}}) &= \text{MMD}^2(\mathbb{P}_{\mathbf{vw}},\mathbb{P}_{\mathbf{v}}\mathbb{P}_{\mathbf{w}}) \\ &= \|\mu_k[\mathbb{P}_{\mathbf{vw}}] - \mu_k[\mathbb{P}_{\mathbf{v}}\mathbb{P}_{\mathbf{w}}]\|_k,\end{aligned}$$

where $\|\cdot\|_k$ is the RKHS norm of $\mathcal{K}_k$ and $\mu_k[\mathbb{P}]$ is the *kernel mean embedding* of $\mathbb{P}$ based on k [9]. If the kernels $k_{\mathbf{v}}$ and $k_{\mathbf{w}}$ are universal, the MMD becomes a proper distance measure of probability distributions (i.e., $\text{MMD}(\mathbb{P}_A,\mathbb{P}_B) = 0$ only when $\mathbb{P}_A$ and $\mathbb{P}_B$ are identical), which applied to the distance between the joint and marginal distributions corresponds to the condition of independence.

In practice, we do not have access to the underlying probability distributions but only a sample $\{\mathbf{x}_i,\mathbf{y}_i\}_{i=1}^n$ drawn from $\mathbb{P}_{\mathbf{vw}}$. We construct a sample-based HSIC estimate:

$$\widehat{\text{HSIC}} = \text{tr}[\mathbf{K}_{\mathbf{v}}\mathbf{C}\mathbf{K}_{\mathbf{w}}\mathbf{C}],$$

where $[\mathbf{K}_{\mathbf{v}}]_{ij} = k_{\mathbf{v}}(\mathbf{v}_i,\mathbf{v}_j)$, $[\mathbf{K}_{\mathbf{w}}]_{ij} = k_{\mathbf{w}}(\mathbf{w}_i,\mathbf{w}_j)$, and $\mathbf{C} = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ with $\mathbf{1} = [1,\ldots,1]^\top$. The estimate $\widehat{\text{HSIC}}$ converges to the true HSIC with $O(1/\sqrt{n})$ [3].

2.2. The visual attribute rank learning problem

Binary object labels describing the *presence or absence* of attributes might be useful for automatic search [7, 17]. However, binary descriptors are insufficient for many attributes. Imagine shopping for shoes: there is no clear boundary between 'sporty' and 'non sporty' shoes. However, it is easy as a human to state that one shoe is 'sportier' than another. Thus, the Parikh and Grauman approach of measuring *relative attributes* (RA) [10] has broadened attribute-based data analysis to abstract and non-categorical labels.

To facilitate the training of automatic attribute predictors, users rank pairs of data points to describe their relationships: object $\mathbf{x}_i$ exhibits a stronger/weaker presence of attribute A than $\mathbf{x}_j$. This technique can be thought of as implicitly introducing a global *ranking function* to a dataset for a given attribute: There is a function f^* such that $f^*(\mathbf{x}_i) > f^*(\mathbf{x}_j)$ implies that the rank of $\mathbf{x}_i$ is higher than that of $\mathbf{x}_j$.

Suppose we have a set of input features $G^{tr} = \{\mathbf{g}_1^{tr},\ldots,\mathbf{g}_l^{tr}\} \subset \mathcal{G}$ representing the underlying objects $X^{tr} = \{\mathbf{x}_1^{tr},\ldots,\mathbf{x}_l^{tr}\} \subset \mathcal{X}$ via a feature extractor $g : \mathcal{X} \mapsto \mathcal{G}$ ($g(\mathbf{x}_i^{tr}) = \mathbf{g}_i^{tr}$ for $1 \le i \le l$) and the corresponding pairwise rank labels $R = \{(i(r),j(r))\}_r \subset \{1,\ldots,l\}\times\{1,\ldots,l\}$: $(i,j) \in R$ implies that the rank of $\mathbf{x}_i$ is higher than $\mathbf{x}_j$. An estimate f of f^* can be constructed by minimizing the average rank loss:

$$\begin{aligned}L(f) &= \sum_{(i,j)\in R} l((\mathbf{x}_i,\mathbf{x}_j);f), \\ l((\mathbf{x}_i,\mathbf{x}_j);f) &= \max\left(0, 1-(f(\mathbf{g}_i)-f(\mathbf{g}_j))\right)^2.\end{aligned}$$

Once an estimate f is constructed, we can apply it to unseen test data points $G = \{\mathbf{g}_1,\ldots,\mathbf{g}_n\}$ to construct the prediction $\mathbf{f}^I := f|_G = [f(\mathbf{g}_1),\ldots,f(\mathbf{g}_n)]^\top$.

3. Summary of our denoising algorithm for TUPI

Suppose we are given an initial predictor $\mathbf{f}(0) = \mathbf{f}^I$ and a set of test time features $\{H^i\}_{i=1}^m$. Our algorithm improves $\mathbf{f}(t)$ by embedding the predictor and the test time features into a manifold $\widehat{M}$ of (centered and scaled) kernel matrices, and performing manifold denoising therein:

$$\begin{aligned}\mathbf{f}(t) \to \widetilde{\mathbf{K}}_{\mathbf{f}}(t) &:= \frac{\mathbf{K}_{\mathbf{f}}(t)\mathbf{C}}{\|\mathbf{C}\mathbf{K}_{\mathbf{f}}(t)\mathbf{C}\|_F} \\ H^i \to \widetilde{\mathbf{K}}_i &:= \frac{\mathbf{K}_i\mathbf{C}}{\|\mathbf{C}\mathbf{K}_i\mathbf{C}\|_F},\end{aligned}$$

where $\|A\|_F$ is the Frobenius norm of A, and $\mathbf{K}_{\mathbf{f}}$ and $\mathbf{K}_i$ are the kernel matrices constructed from $\mathbf{f}$ and H^i, respectively:

$$\begin{aligned}[\mathbf{K}_{\mathbf{f}}]_{kl} &= k_f([\mathbf{f}]_k,[\mathbf{f}]_l) = \exp\left(-\frac{\|[\mathbf{f}]_k-[\mathbf{f}]_l\|^2}{\sigma_f^2}\right) \quad (2) \\ [\mathbf{K}_i]_{kl} &= k_i(\mathbf{h}_k^i,\mathbf{h}_l^i) = \exp\left(-\frac{\|\mathbf{h}_k^i-\mathbf{h}_l^i\|^2}{\sigma_i^2}\right)\end{aligned}$$

with scale hyperparameters σ_f^2 and $\{\sigma_i^2\}_{i=1}^m$. Here, $\mathbf{h}_k^i$ denotes the k-th element of the feature set $H^i = \{\mathbf{h}_1^i,\ldots,\mathbf{h}_n^i\}$. $\{\sigma_f^2\}$ is set to be twice the standard deviations of pairwise distances of elements of $\mathbf{f}$; $\{\sigma_i^2\}_{i=1}^m$ are tuned similarly.

This process is instantiated as iterative minimization of an energy functional $\mathcal{O}(\cdot;t)$

$$\begin{aligned}\mathcal{O}(\mathbf{f};t) &= d^2_{\widehat{\mathcal{M}}}(\widetilde{\mathbf{K}}_{\mathbf{f}},\widetilde{\mathbf{K}}_{\mathbf{f}}(t)) \\ &+ \lambda\sum_{i=1}^m\left(\frac{w^i(t)}{\sum_{j=1}^m w^j(t)}\right)d^2_{\widehat{\mathcal{M}}}(\widetilde{\mathbf{K}}_{\mathbf{f}},\widetilde{\mathbf{K}}_i) \quad (3) \\ w^i(t) &= \exp\left(-\frac{d^2_{\widehat{\mathcal{M}}}(\widetilde{\mathbf{K}}_{\mathbf{f}}(t),\widetilde{\mathbf{K}}_i)}{\sigma_w^2}\right), \quad (4)\end{aligned}$$

Algorithm 1 TUPI algorithm

Input: Initial predictor evaluations $\mathbf{f}^I$; class of test time features $\{H^i\}_{i=1}^m$; hyperparameters λ and σ_w^2 (Eq. 3); (maximum iteration number T);
Output: Denoised evaluations $\mathbf{f}^O$;
$t = 0$;
$\mathbf{f}(t) = \mathbf{f}^I$;
repeat
  Calculate weights $\{w^i(t)\}$ based on Eq. 4;
  Update $\mathbf{f}(t)$ by minimizing $\mathcal{O}$ (Eq. 3);
  $t = t+1$;
until termination condition is met (e.g. if $t \geq T$);

where $d^2_{\widehat{\mathcal{M}}}(\widetilde{\mathbf{K}}_A, \widetilde{\mathbf{K}}_B) = 1 - \mathrm{tr}[\widetilde{\mathbf{K}}_A \widetilde{\mathbf{K}}_B]$.

The number of iterations is a hyperparameter. In the experiments, we set the maximum iteration number T at 50 and monitored the progress of ftion accuracy: we finish the iteration immediately whenever the validation accuracy did not increase from the previous iteration. Algorithm 1 summarizes the TUPI denoising process.

3.1. Large scale problems

For tasks where the time ($O(mn^3)$) and memory ($O(mn^2)$) complexities of optimizing $\mathcal{O}$ are limiting, we adopt the Nyström approximation of kernel matrix $\mathbf{K}_\mathbf{f}$:[2]

$$\mathbf{K}_\mathbf{f} \approx \mathbf{K}_{\mathbf{f}B} \mathbf{K}_{BB}^{-1} \mathbf{K}_{\mathbf{f}B}^\top, \tag{5}$$

where $[\mathbf{K}_{\mathbf{f}B}]_{kl} = k_f(b_k, b_l)$ for the basis set $B = \{b_1, \ldots, b_K\}$ and $[\mathbf{K}_{\mathbf{f}B}]_{kl} = k_f([\mathbf{f}]_k, b_l)$. The rank K of the approximation is based on computational and memory capacity limits. Similarly, each $\mathbf{K}_i$ is approximated based on the corresponding basis set ($\mathbf{K}_i \approx \mathbf{K}_{iB}[\mathbf{K}_{BB}^i]^{-1}\mathbf{K}_{iB}^\top$). For example, the second (unnormalized) trace term in Eq. 3 and its derivative with respect to $\mathbf{f}$ are written as:

$$\begin{aligned}\mathrm{tr}[\mathbf{K}_\mathbf{f}\mathbf{C}\mathbf{K}_i\mathbf{C}] &\approx \mathcal{C}(\mathbf{f}) \\ &= \mathrm{tr}[\mathbf{K}_{\mathbf{f}B}\mathbf{K}_{BB}^{-1}\mathbf{K}_{\mathbf{f}B}^\top\mathbf{C}\mathbf{K}_{iB}[\mathbf{K}_{BB}^i]^{-1}\mathbf{K}_{iB}^\top\mathbf{C}] \quad (6)\\ \frac{\partial \mathcal{C}(\mathbf{f})}{\partial [\mathbf{f}]_k} &= 2[\partial \mathbf{K}_{\mathbf{f}B}]_{(k,:)} \cdot \\ &\quad \left[\mathbf{K}_{BB}^{-1}\mathbf{K}_{\mathbf{f}B}^\top\mathbf{C}\mathbf{K}_{iB}[\mathbf{K}_{BB}^i]^{-1}\mathbf{K}_{iB}^\top\mathbf{C}\right]_{(:,k)}, \quad (7)\end{aligned}$$

where $[A]_{(k,:)}$ denotes the k-th row of A and $[\partial \mathbf{K}_{\mathbf{f}B}]_{kl}$ corresponds to the derivative of $k_f([\mathbf{f}]_k, b_l) = \exp(-\|[\mathbf{f}]_k - b_l\|^2/\sigma_f^2)$. The main computational bottleneck in the gradient evaluation is the multiplication $\mathbf{K}_{\mathbf{f}B}^\top\mathbf{C}\mathbf{K}_{iB}$ for each $i = 1, \ldots, m$, which takes total $O(m \times n \times K^2)$ time. As such, the complexity is linear in the number of data points n and the number of test time features m.

[2]Readers are referred to [18] for other sparse approximations including random Fourier features and block-averaged statistic.

3.2. TUPI parameter smoothness

As our algorithm is unsupervised, in practical applications, we expect users to evaluate different hyperparameter combinations. Figure 1 shows that this approach is feasible as the rank accuracy surface with respect to the two parameters is smooth, enabling practical sampling approaches.

4. Additional experimental details

4.1. Dataset details

Multiple Features dataset (*MFeat*). This contains 6 different feature representations of 2,000 handwritten digits: Each input digit is represented by F1) 76 Fourier coefficients, F2) 216 profile correlations, F3) 64 Karhunen-Loève coefficients, F4) 240 local color averages, F5) 47 Zernike moments, and F6) 6 morphological features [1]. The target rank outputs are obtained based on digit class labels. In the main paper, we use each single feature set F1–F6 as the baseline features $\mathbf{g}$, while the remaining features are used as test time features $\{H^i\}_{i=1}^5$.

Public Figure Faces (*PubFig*), *Shoes*, and Outdoor Scene Recognition (*OSR*) datasets. *PubFig* contains 772 images of 8 people with 11 attributes [10]. The goal is to estimate rankings on each of the target attributes: *Masculine-looking*, *White*, *Young*, *Smiling*, *Chubby*, *Visible-forehead*, *Bushy-eyebrows*, *Narrow-eyes*, *Pointy-nose*, *Big-lips*, *Round-face*. The labels are provided as category-wise comparisons, i.e., each category (person for *PubFig*) has a stronger or weaker presence of certain attributes than other categories. *Shoes* dataset contains 14,658 images of 10 attributes and 10 categories [7]: *pointy-at-the-front*, *open*, *bright-in-color*, *covered-with-ornaments*, *shiny*, *high-at-the-heel*, *long-on-the-leg*, *formal*, *sporty*, and *feminine*. *OSR* contains 2,688 images of 6 attributes from 8 categories: *natural*, *open*, *perspective*, *large-objects*, *diagonal-plane*, and *close-depth*. Similar to *PubFig*, the rank labels for *Shoes* and *OSR* are constructed from pairwise category-wise comparisons.

For *PubFig* and *Shoes*, we use GIST features and color histograms provided by Parikh and Grauman [10] as input features to construct the initial rankings $\mathbf{f}^I$. For *OSR*, we construct $\mathbf{f}^I$ with GIST features from the authors of [7].

For each target attribute, the remaining attributes are used as the source of test time features. However, our preliminary experiments showed that all target output attributes are strongly correlated, and so applying TUPI with other target attributes as test time features leads to almost perfect results. As such, instead of directly using these ground-truth attributes, we use the outputs of the corresponding individually trained rankers. This corresponds to a practical application scenario where the estimated rankers are denoised by using the other rank estimates as test time information.

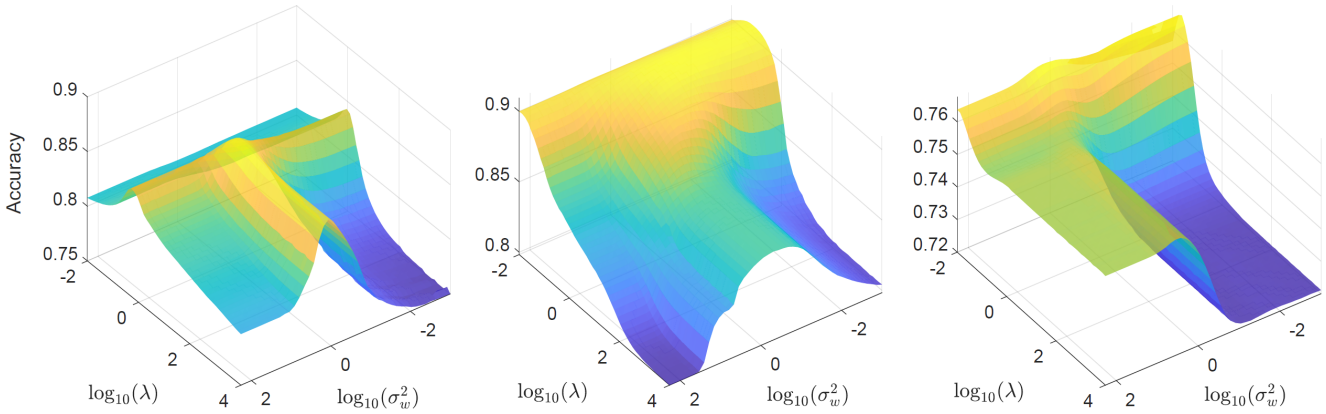

Figure 1. Accuracy of our algorithm on *PubFig* (left; attribute 2), *OSR* (center; attribute 2), and *Shoes* (right; attribute 5) datasets with respect to hyperparameters λ (Eq. 3) and σ_w^2 (Eq. 4) varying over logarithmic intervals.

4.2. Results with absolute accuracies and statistical significance tests

In the main paper, we show results relative to f^I for easier interpretation of the bar charts. Figure 2 shows the absolute accuracy results. Further, Table 2 shows tests of statistical significance of result differences of different algorithms:

- Our algorithm is better than initial predictions $\mathbf{f}^I$ in 87.18% of cases, and it is not worse in any cases. This shows the effectiveness of exploiting test time features.
- Our algorithm is statistically significantly better than Kim et al.'s algorithm in 62.96% of cases, and it is not worse in any cases.
- Our algorithm is statistically significantly better than SSL in 66.67% of cases and worse in 33.33% of cases. The worse cases occurred only on *Zap50K*, where test time features are powerful enough for SSL to be better.
- Our algorithm is statistically significantly better than retraining on test time features in 50% of cases and worse in 33% of cases. Again, worse cases occurred only on *Zap50K*, where test time features are powerful enough for SSL to be better.
- In comparison with CoConut, our algorithm is better in 64.10% of cases and worse in 2.56% of cases.

Overall, our approach is more useful as an approach in our setting as it provides better performance on average.

4.3. A simple failure case

We assume that the additional features $\{H^i\}$ are useful if they exhibit strong statistical dependence to the underlying ground-truth predictor $\mathbf{f}^*$. However, since we do not have access to $\mathbf{f}^*$, we instead measure and enforce statistical dependence to the main predictor $\mathbf{f}(t)$ that is being denoised through iterative optimization (Algorithm 1 and Eq. 3). While our experiments on real-world problems have demonstrated the effectiveness of this approach, simple failure cases exist. For example, if all additional features are identical to the initial predictions, trivially there is no gain. Table 1 shows slightly more involved cases: Here, on the *MFeat* dataset, we gradually increase the number of copies of initial predictors $\mathbf{f}^I$ in the original reference sets (each containing 5 features). When there is only a single copy of $\mathbf{f}^I$ included in the reference set, for all features except for F4, the performance is roughly on par with the case where the original references are used ($\mathbf{f}^O$). However, as the number of $\mathbf{f}^I$ copies increases, the accuracy decreased rather rapidly. Future work should explore the possibilities of automatically identifying such features.

5. TUPI adaptation of Kim et al.'s algorithm and CoConut

5.1. Adapting Kim et al. [6]

This method forms predictive *distributions* from reference tasks, then penalizes their pairwise Kullback-Leibler divergence from the target distribution. This implies metric comparison: If a test time feature is the negative of the perfect prediction, then the KL-divergence will be large and Kim et al. would penalize it; however, the feature is *still statistically dependent*, and our approach would exploit this.

To adapt their method to our setting, if we let their reference task predictions be new features $\{H^i\}$, then this approach works when the (probability) space of each feature coincides with the space of predictions $\mathcal{Y}$. This makes their algorithm applicable only to datasets with one-dimensional test time features as the predictions made for potentially-related tasks (e.g., the *PubFig*, *Shoes* and, *OSR* datasets in the experiments). Figure 2 demonstrates that our general multi-dimensional test time feature algorithm is a strong alternative to Kim et al.'s approach even in this special setting.

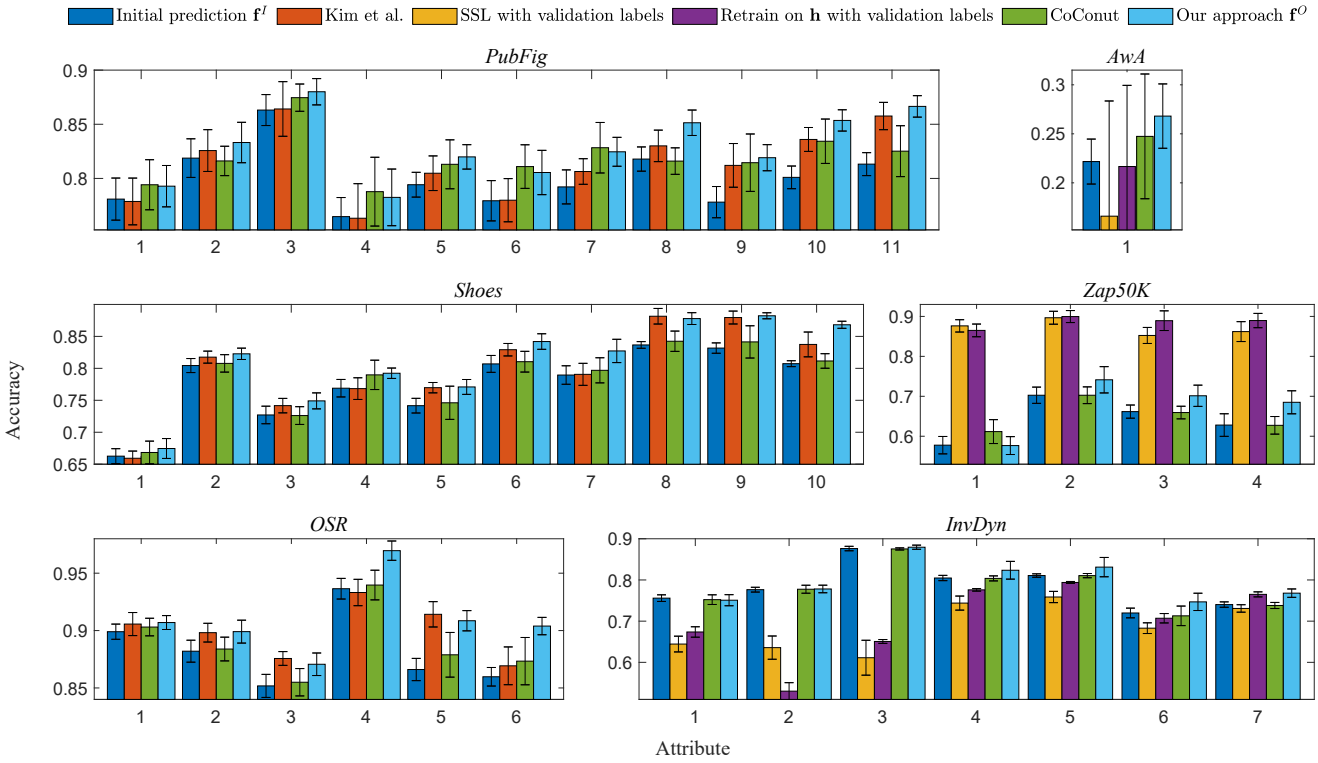

Figure 2. Absolute mean accuracies for *PubFig*, *AwA*, *Shoes*, *Zap50K*, *OSR*, and *InvDyn* datasets (higher is better; error bars are standard deviations). The main paper shows performance relative to $\mathbf{f}^I$ for easier viewing.

Table 1. *MFeat* dataset. Ranking algorithm mean accuracy percent, plus standard deviation in parenthesis, given the F1–F6 features.
$\mathbf{f}^I$: The initial predictions.
$\mathbf{f}^O$: TUPI with other F-feature sets as test time information.
$\mathbf{f}^{F_i}$: $\mathbf{f}^O$ with other F-feature sets plus i copies of $\mathbf{f}^I$ as test time information.

	F1	F2	F3	F4	F5	F6
$\mathbf{f}^I$	77.85 (2.26)	79.28 (1.23)	75.70 (2.38)	70.88 (1.23)	76.05 (2.66)	77.10 (1.60)
$\mathbf{f}^O$	81.97 (2.95)	81.45 (1.69)	78.31 (3.13)	74.33 (5.00)	78.19 (3.29)	82.25 (2.09)
$\mathbf{f}^{F_1}$	82.81 (2.35)	81.92 (1.84),	79.22 (2.95)	71.86 (2.54)	79.65 (3.20)	82.44 (1.90)
$\mathbf{f}^{F_3}$	80.50 (2.16)	81.34 (1.43)	77.93 (2.75)	71.12 (1.64)	78.20 (2.83)	79.83 (1.65)
$\mathbf{f}^{F_5}$	79.54 (2.23)	80.80 (1.34)	77.17 (2.54)	70.95 (1.47)	77.54 (2.76)	78.76 (1.53)
$\mathbf{f}^{F_7}$	79.24 (2.23)	80.48 (1.29)	76.88 (2.47)	70.92 (1.48)	77.30 (2.71)	78.37 (1.49)
$\mathbf{f}^{F_9}$	79.06 (2.23)	80.31 (1.27)	76.73 (2.43)	70.90 (1.46)	77.15 (2.79)	78.16 (1.47)
$\mathbf{f}^{F_{11}}$	78.96 (2.23)	80.19 (1.25)	76.65 (2.41)	70.89 (1.46)	77.07 (2.75)	78.05 (1.47)

5.2. Adapting CoConut [5]

The CoConut framework was developed for *co-classification* problems where multiple data instances are jointly classified. The authors propose to apply a graph Laplacian-type regularizer via adopting the *Cluster Assumption* [16] to improve during testing the classifications once predicted. In general, a graph Laplacian-type regularizer is defined based on the pairwise similarities of predictions weighted by the corresponding input feature similarities. In CoConut, the new regularizer can also be constructed based on additional features that are available at test time, facilitating a TUPI-like scenario. Please note that we adopt the mathematical notations from our main paper which differ from the notation of the original CoConut paper [5].

Suppose that we have a set of test data features $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_n\} \subset \mathcal{G}$ and our goal is to predict a classification label vector $\mathbf{f} = [f_1, \ldots, f_n]^\top$ where the value of each element f_i is assigned from a label set $\mathcal{Y} = \{1, \ldots, L\}$. Further, we assume that a set of *base classifiers* $\{(f^I)^l : \mathcal{G} \to \mathbb{R}\}_{l=1}^L$ are constructed such that $(f^I)^l(\mathbf{g})$ provides a *confidence* that the sample point $\mathbf{g}$ belongs to class l. Based only on the base classifiers, the initial class label prediction y_i^I for the i-th test data point can be made as

$$y_i^I = \underset{l=1,\ldots,L}{\arg\max} (f^I)^l(\mathbf{g}_i). \tag{8}$$

CoConut improves this *initial predictions* $\mathbf{y}^I = [y_1, \ldots, y_n]^\top$ by minimizing the following energy[3]

$$\mathcal{O}'(\mathbf{f}) = -\sum_{i=1}^{n}\sum_{l=1}^{L} \mathbb{1}[f_i = l](f^I)^l(\mathbf{g}_i) + \lambda^C \sum_{i=1}^{n} \frac{1}{|N_i|} \sum_{\mathbf{g}_j \in N_i} w_{ij}\mathbb{1}[f_i \neq f_j], \quad (9)$$

where $\mathbb{1}[\cdot]$ is the indicator function, $\lambda^C \geq 0$ is the regularization hyperparameter, and N_i is the neighbors of $\mathbf{g}_i$ in $\mathcal{G}$ that Khamis and Lampert [5] defined as the k-nearest neighbors (NNs). The first term ensures that the final solution does not deviate significantly from the initial class assignments $\mathbf{y}^I$, while the second term enforces smoothness in the final solution measured in the pairwise similarities of output values weighted by $\{w_{ij}\}$. The weight w_{ij} is defined based on the pairwise similarity of the input features $\mathbf{g}_i$ and $\mathbf{g}_j$:

$$w_{ij}^C = \exp\left(-\frac{d^2_{\mathcal{G}}(\mathbf{g}_i - \mathbf{g}_j)}{(\sigma^C)^2}\right), \quad (10)$$

where $d^2_{\mathcal{G}}$ is a distance measure on $\mathcal{G}$ and σ^C is a hyperparameter. When an additional set of *test time* features $H = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\} \subset \mathcal{H}$ is provided, $d^2_{\mathcal{G}}$ can be replaced by $d^2_{\mathcal{G}} + d^2_{\mathcal{H}}$ taking both features into account. This corresponds to a TUPI-like usage of test time features. The discrete optimization problem of minimizing $\mathcal{O}'$ can be approximately solved based on convex relations. The authors proposed to tune hyperparameter λ^C based on the training set used for building the base classifiers $\{(f^I)^l\}_{l=1}^L$. This requires the training labels in testing; we will discuss this in the 'tuning hyperparameters' paragraph of the next subsection.

CoConut adaptation applied to Relative Attributes ranking. As the original CoConut optimization problem (Eq. 9) was designed for discrete classification problems, it needs to be *adapted* before it can be applied to Relative Attributes (RA) ranking problems where the outputs of the base predictors take continuous values. First, it should be noted that when $\{f_i\}$ and $\{(f^I)^l\}$ take continuous values, counting the occurrence of equal values via the indicator evaluations ($\mathbb{1}[\cdot]$) in Eq. 9 leads to zero values in the first term of Eq. 9 in general. Instead, we reinterpret the first term as the measure of deviation (per test instance) between the hypothesized solution f_i and the initial base prediction f_i^I: $f_i^I := \max_{l=1,\ldots,L}(f^I)^l(\mathbf{g}_i)$ is the confidence of predicting the label y_i^I for $\mathbf{g}_i$ [5] in the original classification setting (see Eq. 8). Instantiating this interpretation in the real-valued prediction case, we cast the first term in $\mathcal{O}'$ into

$$\mathcal{O}'_1(\mathbf{f}) = \sum_{i=1}^{n}(f_i^I - f_i)^2 \quad (11)$$

measuring the deviation between $\mathbf{f} = [f_1, \ldots, f_n]^\top$ and $\mathbf{f}^I = [f_1^I, \ldots, f_n^I]^\top$. Unlike the ($L$-class) classification problems initially considered by Khamis and Lampert [5], we do not have L different base predictors (one per class). Therefore, a single base predictor $\mathbf{f}^I$ is used.

Now, relaxing the equality constraints in the second regularization term of $\mathcal{O}'$ into a measure of continuous squared deviations, and adopting the k-NN structure for $\{N_i\}$ (with k^C neighbors) as used in the original CoConut setting, the second term can be restated as

$$\mathcal{O}'_2(\mathbf{f}) = \frac{\lambda^C}{k^C}\mathbf{f}^\top L\mathbf{f}, \quad (12)$$

where L is the graph Laplacian as $L = D^C - W^C$, and

$$W_{ij}^C = \begin{cases} w_{ij}^C & \text{if } \mathbf{g}_j \in N_i \text{ (see Eq. 10)} \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

and $[D^C]$ is a diagonal matrix of row sums of W^C: $[D^C]_{ii} = \sum_j [W^C]_{ij}$. $\mathcal{O}'_1$ ensures that the final solution does not deviate significantly from $\mathbf{f}^I$, and $\mathcal{O}'_2$ contributes to improving the final solution by enforcing its spatial smoothness measured via the Laplacian L.

Tuning hyperparameters. In the original CoConut setting, the authors proposed to tune the hyperparameter λ_C (Eq. 9) based on performance on the training set that is used to train the base predictors $\{(f^I)^l\}$. This requires access to training labels at test time. However, in our TUPI scenario, a large set of labeled training data points is not available: If such training labels are available, a better alternative to TUPI is often simply to re-train the baseline predictor f^I with the original G and the test time features H. Indeed, in our preliminary RA experiments using deep neural networks as baselines, this constantly led to better performance than our TUPI algorithm as well as our CoConut adaptation.

Furthermore, we also observed in preliminary experiments that for the problem of estimating continuous predictions in the RA setting, tuning the CoConut hyperparameters in this way suffered from ovefitting and it leaded to much worse results than the original predictors f^I.

Therefore, in our experiments provided in the main paper, we tune the hyperparameters based on separate validation sets. The hyperparameters include k^C for k-NNs, σ^C (Eq. 10), and λ^C. We adaptively decided σ^C as twice the mean distance within each N_i as suggested by Hein and Maier [4]. The other two parameters k^C and λ^C are tuned based on the validation accuracies. In the original CoConut

[3] In $\mathcal{O}'$, the initial predictions $\mathbf{y}^I$ are only indirectly taken account via $\{f_l(\mathbf{g}_i)\}$.

framework, the authors tuned only one parameter λ^C and the selection of k^C was not discussed. We observed that the best choice of k^C differs across different datasets, and the impact of varying this value on the final results is substantial. Thus, we concluded that k^C also needs to be tuned per dataset.

CoConut adaptation summary. We minimize the energy

$$\mathcal{O}'(\mathbf{f}) = \|\mathbf{f} - \mathbf{f}^I\|^2 + \frac{\lambda^C}{k^C}\mathbf{f}^\top L\mathbf{f}, \tag{14}$$

where L is the graph Laplacian calculated based on local k-nearest neighbors (with $k = k^C$) in the test time feature space. The first term in $\mathcal{O}'$ ensures that the final solution does not deviate significantly from $\mathbf{f}^I$ while the second term contributes to improving the final solution by enforcing its spatial smoothness measured via the Laplacian L. The two hyperparameters λ^C and k^C are tuned based on validation accuracies similarly to our algorithm.

5.3. Combining CoConut and our algorithm

We observed in the experiments that, overall, our algorithm provides higher accuracy than CoConut (Fig. 2). At the same time, the specific results on the *PubFig* dataset demonstrate that CoConut and our algorithm have complementary strengths: Our algorithm generates the best results in attributes 2, 3, 5, and 8–11 while CoConut is the best on attributes 1, 4, 6, and 7. For these attributes, we observe noticeable accuracy differences in the corresponding results of CoConut and our algorithm.

As such, we developed two new algorithms which combine the benefits of CoConut and our algorithm. Our first combination attempt 'CoConut+Ours$_1$' algorithmically combines the two. This algorithm minimizes a new energy $\mathcal{O}''$ which combines the energy functional of our algorithm with the graph Laplacian regularizer of CoConut:

$$\mathcal{O}''(\mathbf{f}; t) = \mathcal{O}(\mathbf{f}; t) + \mathcal{O}'_2(\mathbf{f}). \tag{15}$$

The resulting new algorithm leverages the global statistical dependence present among multiple features (via our denoising strategy) as well as the local spatial smoothness of the predictor variables (via the Coconut regularizer). Figure 3 shows the results: Indeed, combining these benefits, CoConut+Ours$_1$ often generates the best results (attributes 3–7, 9–11). More importantly, even when it is not the best, the corresponding accuracy is close to the best except for the third attribute where our original algorithm is clearly better. However, a drawback of this approach is that to obtain these results, it required tuning four hyperparameters (λ and σ_w^2 from our algorithm and λ^C and k^C from CoConut), which is often prohibitively expensive in practical applications.

Our second algorithm is computationally affordable: CoConut+Ours$_2$ *selects* either of the outputs of CoConut and our algorithm based the validation accuracy: For both algorithms, the predictions are independently generated and these predictions with higher validation accuracy are selected per prediction set as the final outputs. Figure 3 demonstrates that while this approach is overall worse than CoConut+Ours$_1$ and often even worse than either of the CoConut or our algorithm, it still delivers performance that does not deviate significantly from the best results among the CoConut and our algorithm per attribute. This shows that CoConut+Ours$_2$ facilitates trading the hyperparameter tuning complexity of CoConut+Ours$_1$ with the final prediction accuracies.

6. Additional future work

In our application scenario, we assumed no access to the underlying ranking function f and focused on evaluating f^O on a fixed set of points G. When an explicit functional form of f^O is required, e.g. when one wishes to apply f^O to new test points $\mathbf{g}_{\text{new}} \notin G$, two scenarios are possible: 1) If we remove the assumption that the parametric form of f is unknown, one could apply our algorithm to tune the parameter vector $\mathbf{w}$ of $f_\mathbf{w}$. Since our objective function $\mathcal{O}$ (Eq. 3) is smooth, this approach is straightforward when $f_\mathbf{w}$ is continuously differentiable with respect to $\mathbf{w}$ (which is the case for DNNs, RSVMs, and many other predictors); 2) If f form remains unavailable, one could train a smooth regressor f^O on the large set of inputs G using the corresponding non-parametric estimates $\mathbf{f}^O$ as labels.

References

[1] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. https://archive.ics.uci.edu/ml. 3

[2] C. M. Gevaert, C. Persello, and G. Vosselman. Optimizing multiple kernel learning for the classification of UAV data. *Remote Sensing*, 8(12):1025:1–22, 2016. 1

[3] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *ALT*, pages 63–77, 2005. 2

[4] M. Hein and M. Maier. Manifold denoising. In *NIPS*, pages 561–568, 2007. 6

[5] S. Khamis and C. Lampert. CoConut: Co-classification with output space regularization. In *BMVC*, pages 1–11, 2014. 1, 5, 6, 9

[6] K. I. Kim, J. Tompkin, and C. Richardt. Predictor combination at test time. In *ICCV*, pages 3553–3561, 2017. 1, 4, 9

[7] A. Kovashka, D. Parikh, and K. Grauman. Whittlesearch: Image search with relative attribute feedback. In *CVPR*, pages 2973–2980, 2012. 2, 3

[8] J. Mooij, D. Janzing, J. Peters, and B. Schölkopf. Regression by dependence minimization and its application to causal inference in additive noise models. In *ICML*, pages 745–752, 2009. 1

[9] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions: a review

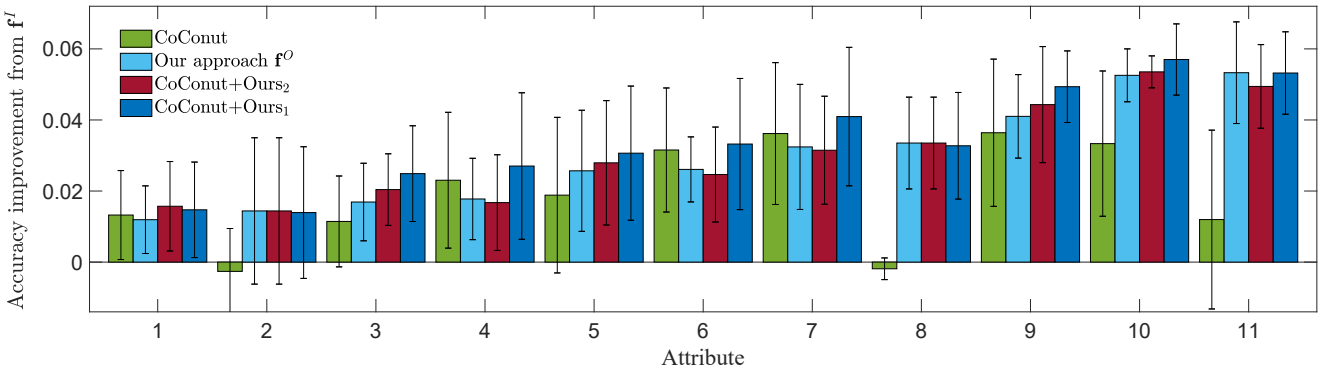

Figure 3. Accuracy improvements over $\mathbf{f}^I$ for *PubFig* dataset where CoConut and our algorithm demonstrate complementary strengths.

and beyond. *Foundations and Trends in Machine Learning*, 10(1–2):1–141, 2017. 2

[10] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, pages 503–510, 2011. 2, 3

[11] A. Royer and C. H. Lampert. Classifier adaptation at prediction time. In *CVPR*, pages 1401–1409, 2015. 1

[12] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt. Feature selection via dependence maximization. *JMLR*, 13:1393–1434, 2012. 1

[13] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt. A dependence maximization view of clustering. In *ICML*, pages 815–822, 2007. 1

[14] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *JMLR*, 2:67–93, 2001. 2

[15] D. Tuia and G. Camps-Valls. Kernel manifold alignment for domain adaptation. *PLOS ONE*, 11(2):e0148655, 2016. 1

[16] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. 5

[17] A. Yu and K. Grauman. Just noticeable differences in visual attributes. In *ICCV*, pages 2416–2424, 2015. 2

[18] Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, pages 1–18, 2017. 3

Table 2. Results of t-test with $\alpha = 0.95$ for relative accuracy differences of different algorithms. 1 and -1: statistically significantly positive and negative, respectively and 0: statistically insignificant.
$\mathbf{f}^I$: The initial predictions.
$\mathbf{f}^K$: Kim et al.'s algorithm [6].
$\mathbf{f}^S$: Semi-supervised learning on $\mathbf{h}$.
$\mathbf{f}^R$: Retrain on $\mathbf{h}$.
$\mathbf{f}^C$: CoConut [5].
$\mathbf{f}^O$: Our TUPI algorithm.

Dataset	Attr.	$\mathbf{f}^K - \mathbf{f}^I$	$\mathbf{f}^S - \mathbf{f}^I$	$\mathbf{f}^R - \mathbf{f}^I$	$\mathbf{f}^C - \mathbf{f}^I$	$\mathbf{f}^O - \mathbf{f}^I$	$\mathbf{f}^O - \mathbf{f}^K$	$\mathbf{f}^O - \mathbf{f}^S$	$\mathbf{f}^O - \mathbf{f}^R$	$\mathbf{f}^O - \mathbf{f}^C$
Pubfig	1	0	N/A	N/A	1	1	1	N/A	N/A	0
	2	0	N/A	N/A	0	0	0	N/A	N/A	1
	3	0	N/A	N/A	1	1	1	N/A	N/A	0
	4	0	N/A	N/A	1	1	1	N/A	N/A	0
	5	0	N/A	N/A	1	1	1	N/A	N/A	0
	6	0	N/A	N/A	1	1	1	N/A	N/A	0
	7	1	N/A	N/A	1	1	1	N/A	N/A	0
	8	1	N/A	N/A	0	1	1	N/A	N/A	1
	9	1	N/A	N/A	1	1	0	N/A	N/A	0
	10	1	N/A	N/A	1	1	1	N/A	N/A	1
	11	1	N/A	N/A	0	1	1	N/A	N/A	1
InvDyn	1	N/A	-1	-1	0	0	N/A	1	1	0
	2	N/A	-1	-1	0	0	N/A	1	1	0
	3	N/A	-1	-1	0	0	N/A	1	1	1
	4	N/A	-1	-1	0	1	N/A	1	1	1
	5	N/A	-1	-1	0	1	N/A	1	1	1
	6	N/A	-1	0	0	1	N/A	1	1	1
	7	N/A	-1	1	0	1	N/A	1	0	1
AwA	1	N/A	0	0	0	1	N/A	1	0	0
Shoes	1	0	N/A	N/A	0	1	1	N/A	N/A	0
	2	1	N/A	N/A	0	1	0	N/A	N/A	1
	3	1	N/A	N/A	0	1	1	N/A	N/A	1
	4	0	N/A	N/A	1	1	1	N/A	N/A	0
	5	1	N/A	N/A	0	1	0	N/A	N/A	1
	6	1	N/A	N/A	0	1	1	N/A	N/A	1
	7	0	N/A	N/A	0	1	1	N/A	N/A	1
	8	1	N/A	N/A	0	1	0	N/A	N/A	1
	9	1	N/A	N/A	0	1	0	N/A	N/A	1
	10	1	N/A	N/A	0	1	1	N/A	N/A	1
OSR	1	0	N/A	N/A	0	1	0	N/A	N/A	0
	2	1	N/A	N/A	0	1	0	N/A	N/A	1
	3	1	N/A	N/A	0	1	0	N/A	N/A	1
	4	0	N/A	N/A	0	1	1	N/A	N/A	1
	5	1	N/A	N/A	0	1	0	N/A	N/A	1
	6	0	N/A	N/A	1	1	1	N/A	N/A	1
Zap50K	1	N/A	1	1	1	0	N/A	-1	-1	-1
	2	N/A	1	1	0	1	N/A	-1	-1	1
	3	N/A	1	1	0	1	N/A	-1	-1	1
	4	N/A	1	1	0	1	N/A	-1	-1	1