

## Appendix: Supplementary material

### A. Proof of Theorem 1

*Proof.* We assume in the statement of the theorem that

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \geq -(M-1)(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T} \quad (8)$$

for some constant  $M > 1$ . This assumption is justified if the accumulated cost obtained by the online algorithm is larger than the cost obtained by the optimal batch algorithm for most of the training examples. We argue that if this assumption is violated then the online algorithm is revealed to be a very good performer, and it therefore makes little sense to judge its performance by the deviation from another algorithm (the ‘‘optimal’’ batch algorithm) whose performance is worse for a significant fraction of the training sample.

Using the definition of  $\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}$  and  $\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}$ , we rewrite the regret (6) with a single loss function

$$\begin{aligned} \mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) &= \sum_{t=1}^T (\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) + \psi^l(\hat{\mathbf{W}}_t^l)) - \\ &\quad \left[ \sum_{t=1}^T (\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1}) + \psi^l(\hat{\mathbf{W}}_*^l)) \right] \end{aligned}$$

From the convexity of  $\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}$  it follows that

$$\begin{aligned} \sum_{t=1}^T (\mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) - \mathbf{L}_{\mathbf{t}, \mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1})) \\ \leq \sum_{t=1}^T \langle U_t^l, \hat{\mathbf{W}}_t^l - \hat{\mathbf{W}}_*^l - \mathbf{W}_*^{l-1} + \mathbf{W}_t^{l-1} \rangle \end{aligned}$$

Under the conditions of the theorem, it is shown in (corollary 2a (Xiao, 2010)) that

$$\sum_{t=1}^T \langle U_t, \mathbf{W}_t - \mathbf{W}_* \rangle + \psi(\mathbf{W}_t) - \psi(\mathbf{W}_*) \leq \Delta_T$$

where  $\Delta_T = (\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T}$ . Using this result and the sublinearity of the inner product, we get

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + \sum_{t=1}^T \langle U_t^l, \mathbf{W}_t^{l-1} - \mathbf{W}_*^{l-1} \rangle \quad (9)$$

From the definition of  $\mathbf{W}_t^{l-1} = \sum_{k=1}^{l-1} \hat{\mathbf{W}}_t^k$  and  $\mathbf{W}_*^{l-1} =$

$\sum_{k=1}^{l-1} \hat{\mathbf{W}}_*^k$  and the Cauchy-Schwarz inequality we get

$$\begin{aligned} \mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) &\leq \Delta_T + \sum_{k=1}^{l-1} \sum_{t=1}^T \langle U_t^l, \hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k \rangle \\ &\leq \Delta_T + \sum_{k=1}^{l-1} \sum_{t=1}^T \|U_t^l\| \|\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k\| \quad (10) \end{aligned}$$

We use the bound on  $\|\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k\|$  from (theorem 1b (Xiao, 2010)) and assumption (8) to obtain

$$\begin{aligned} \sum_{t=1}^T \|\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k\| &\leq \sum_{t=1}^T \sqrt{\frac{2M\Delta_t}{\sigma t + \gamma\sqrt{t}}} \\ &\leq Q \sum_{t=1}^T t^{-\frac{1}{4}} \leq Q \frac{4}{3} (T+1)^{\frac{3}{4}} \end{aligned}$$

where  $Q = \sqrt{\frac{2M}{\sigma}(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})}$ . Inserting this last inequality into (10), and since  $\forall \mathbf{t}, l \|U_t^l\| < \mathbf{G}$ , we obtain

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + (l-1)\mathbf{G}Q\frac{4}{3}(T+1)^{\frac{3}{4}} \quad (11)$$

from which (7) immediately follows.  $\square$

### B. Synthetic Data Experiments

The synthetic data is created in the following manner: we define a binary tree with  $k$  leaves. Each leaf in the tree represents a single binary classification task. Each node in the tree corresponds to a single binary feature  $f \in \{-1, 1\}$ . For a single task we divide the features into two groups: task-specific and task-irrelevant. Task-irrelevant features have equal probability of having the value 1 or  $-1$  for all examples in the task. Task-specific features are assigned the value 1 for all positive examples of the task. All negative examples of the task must have at least one feature from the task-specific feature set with a value of  $-1$ .

The task-specific feature set is the set of features corresponding to all nodes on the path from the leaf to the root, while all other nodes define the task-irrelevant feature set. For each group of tasks, their shared features are those corresponding to common ancestors of the corresponding leaves. An illustration of the binary tree and an example of a sample set of a specific task is given in Fig. 4.

*Structure discovery:* Feature weights learnt at different learning steps of the cascade for an experiment with 100 synthetic tasks, 199 features and 20 positive and negative samples per task, are shown in Fig 5. As can be seen, the first learning stages,  $l = 1$  and  $l = 2$  capture shared information, while the last stages,

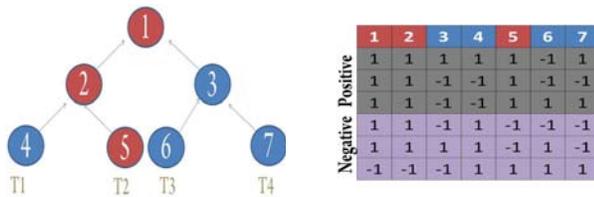


Figure 4. Synthetic data illustration. Left graph shows a tree of features corresponding to four tasks. The task specific features of task 2, 'T2', are highlighted in red. The task-irrelevant features are marked blue. The table on the right shows an example of a sample set sampled for task 'T2' given the task tree to the left. Each row denotes a sample. Each column denotes a feature. 'Positive' and 'Negative' denote the positive and negative sample sets, respectively.

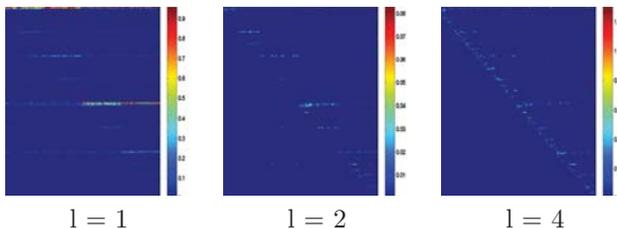


Figure 5. Synthetic experiment learnt parameters  $\mathbf{W}^l$ . Each plot corresponds to a single matrix learnt at stage  $l$  of the cascade. The rows correspond to features and each column corresponds to a single task.

e.g. stage  $l = 4$  capture task specific features. The hierarchical shared structure of features is discovered in the sense that higher levels in the cascade share the same set of features and as the cascade progresses the chosen features are shared by less tasks. The pattern of learnt feature weights fits the synthetic data pattern of feature generation.

**Parameter Robustness** Fig. 6 shows the robustness of the hierarchical approach with respect to the regularization parameter  $\phi$ . For all three regularizing approaches we varied the value of  $\phi$  in the range [0.01-2] with 0.01 jumps. For the hierarchical approach we set  $\phi^1 = \phi$  and  $\phi^l = \frac{\phi^{l-1}}{2}$  for all  $l \in [2..L]$ .

We also found the method to be quite robust to the parameter  $L$  determining the number of levels in the cascade. Varying the value of  $L$  between 3 to 7 on the synthetic data with 100 tasks gave close results in the range 94.5% to 95.5%.

**Single Level Comparison** Fig. 7 we show a comparison of the online cascade to a group of single level

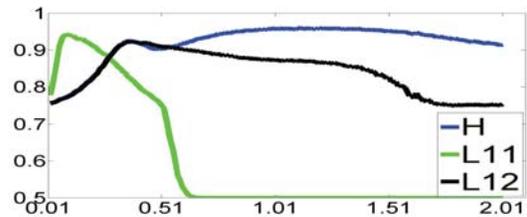


Figure 6. Performance as a function of the regularization parameter  $\phi$ . Synthetic data with 100 examples per task. The 'Y' axis corresponds to the average accuracy of all tasks on 10 repetitions of the experiment. The 'X' axis corresponds to the value of  $\phi$ . Note that the max values of each method are: 96.02, 94.23 and 92.30 for 'H', 'L11' and 'L12' respectively.

Table 4. Varying the number of levels  $L$

$L = 2$	$L = 3$	$L = 4$	$L = 5$	$L = 6$	$L = 7$	$L = 8$
92.42	95.47	95.73	94.74	95.21	94.48	93.52

regularization schemes, using the same set of  $\lambda$  values we used in the cascade. Clearly no single-level regularization achieves as good performance as the hierarchical method.

**Adding Tasks** We examined the effect of the number of tasks in the multitask setting. Ideally adding more tasks should never reduce performance, while in most cases leading to improved performance. We tested two scenarios - adding tasks which are similarly related to the existing group of tasks Fig. 8a, and adding tasks which are loosely related to all other tasks but strongly related among themselves Fig. 8b.

With additional tasks of the same degree of relatedness, we increase the amount of information available for sharing. As expected, we see in Fig. 8a that performance improves with increasing number of tasks, both for the hierarchical algorithm and for 'L12Reg'. 'L1Reg' is not intended for information sharing, and

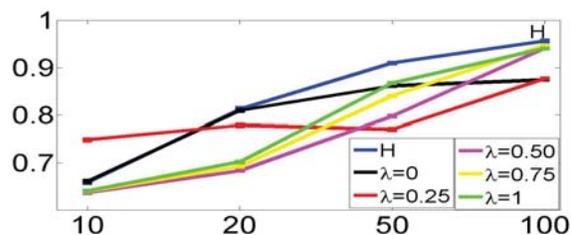


Figure 7. A comparison of our online cascade approach 'H' to variants corresponding to intermediate levels in the cascade, defined by the value  $\lambda$ . 'Y' axis measures the average accuracy over all tasks, and the 'X' axis the sample size.

therefore it is not affected by increasing the number of tasks. When adding loosely related tasks, we see in Fig. 8b that the performance of the hierarchical algorithm increases as we add more tasks; for the 'L12Reg' method, on the other hand, we see a significant drop in performance. This is because in this case the overall relatedness among all tasks decreases as additional tasks are added; the 'L12Reg' method still tries to share information among all tasks, and its performance therefore decreases.

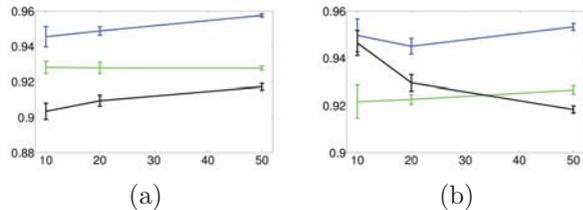


Figure 8. Adding tasks results. Plots correspond to the average 1-vs-rest accuracy as a function the number of tasks, when (a) adding similarly related tasks and (b) adding loosely related tasks. Blue denotes our hierarchical Algorithm 1, green the 'L1Reg' baseline and black the 'L12Reg' baseline method.

**Data Rotation** Next, we wish to isolate the two factors of sparsity and shared information. The synthetic data was constructed so that there is an increased level of shared information between classes as a function of the distance between their respective leaves in the defining tree of features. The shared features are also sparse. In order to maintain the shared information and eliminate sparseness, we rotate the vector of features; when the rotation is applied to more features, the amount of sparseness decreases respectively.

Table 5 shows the comparative results both for the case when all features are rotated and when only half are rotated (in which case the features being rotated are chosen randomly). As expected, the regularization-free method - 'NoReg' - is not effected by any of this. The performance of 'L1Reg', which assumes sparsity, drops as expected, reaching baseline with full rotation, presumably because during cross-validation a very low value for the regularization parameter is chosen. The two methods which exploit shared information, our hierarchical algorithm and the 'L12Reg' baseline method, perform better than baseline even with no sparseness (full rotation), showing the advantage of being able to share information.

Table 5. Performance comparison for the different methods applied to the Synthetic data. 'T 100 S 20' denotes the multi-task setting with 100 tasks and 20 samples each, 'half rotated' - the same setting as 'T 100 S 20' with a random rotation of half of the features, and 'full rotation' - a random rotation of all the features.

	T 100 S 20	half rotation	full rotation
H	95.40 ± 0.17	90.37 ± 0.61	78.49 ± 0.16
L1Reg	92.54 ± 0.17	86.70 ± 0.59	73.01 ± 0.09
L12Reg	91.49 ± 0.2	85.56 ± 0.62	78.49 ± 0.16
NoReg	72.88 ± 0.19	72.81 ± 0.12	73.03 ± 0.10

## C. Real Data

### C.1. ILSVRC2010 Baseline Comparison

In Fig 9 we show the error rates for the Top-1 scenario, considering a single classification. We show results when training using all the data Fig 9-left, and when using only 100 examples per each task Fig 9-right. The results are shown as a function of the number of repetitions of the algorithm over all the data.

At convergence we see an improvement of 1.67% in accuracy when using the cascade with 7 levels, 28.96% compared to 27.29%. (Zhao et al., 2011) obtained an improvement of 1.8% in accuracy when comparing their approach to their own baseline, 22.1% vs. 20.3%. We obtained a similar rate of improvement using much less information (not knowing the hierarchy) for a higher range of accuracies.

We note that our baseline approaches converge after 20 repetitions when using all the data, (for clarity we show only up to 15 repetitions in the left plot of Fig 9). This effectively means the same runtime, as the cascade runtime is linear in the number of levels where each level has the same complexity of the baseline approaches. On the other hand the online cascade algorithm 2 can be trivially parallelized where as the repetitions over the data for a single baseline cannot. Thus, in a parallel setting the gain in runtime would be linear in the number of levels of the cascade. A trivial parallelization can be implemented by running each level of the online cascade on a time stamp shifted by  $l$  thus the first level of the cascade will see at time  $t$  the  $t$  sample while level  $l$  will see sample  $t - l$ .

## D. Knowledge Transfer

**Batch Method** The batch knowledge-transfer method is described below in Algorithm 3. The projection matrix  $\mathbf{P}^l$  is defined by the first  $z$  columns of the orthonormal matrix  $\mathbf{U}^l$ , where  $svd(\mathbf{W}^l) = \mathbf{U}^l \Sigma \mathbf{V}^l$ .

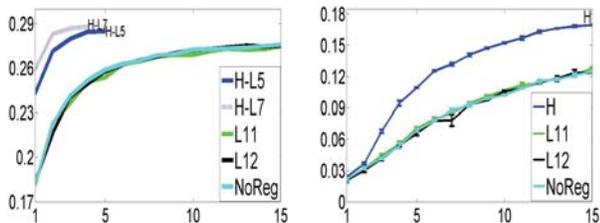


Figure 9. Real data, showing performance of Top-1 classification on the ILSVRC(2010) challenge (Berg et al., 2010) using all examples (left plot) or only 100 examples per each category (right plot). Here the 'X'-axis corresponds to repetitions over the training data. In the left plot 'H-L5' and 'H-L7' denote our hierarchical algorithm with 5 and 7 levels respectively. In the right plot 'H' corresponds to 5 levels in our hierarchical algorithm. The error bars in correspond to the standard error given 3 different choices of 100 Examples.

**Online Method** The online knowledge-transfer algorithm is described below in Algorithm 4; it succeeds  $\mathbf{t}_{old}$  iterations of Algorithm 2. The input to the algorithm is the same set of parameters used for Algorithm 2 and its intermediate calculations - the cascade  $\{\mathbf{W}_{old}^l\}_{l=1}^L$  and the set of final average subgradients  $\{\bar{\mathbf{U}}_{old}^l\}_{l=1}^L$ . These are used to approximate future subgradients of the already learnt tasks, since Algorithm 4 receives no additional data-points for these tasks. The parameters of Algorithm 2 are used because cross-validation for parameter estimation is not possible with small sample.

Below we denote the vector corresponding to the mean value of each feature as  $mean(\mathbf{W}_{old}^l)$ . We denote the concatenation of columns by  $\circ$ . In order to account for the difference between the old time step  $\mathbf{t}_{old}$  to the new time step  $\mathbf{t}$  we consider  $h(\mathbf{W})$  to be the squared  $l_2$  norm applied to each column separately. We calculate the inner product of the resulting vector with  $\frac{\gamma}{\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}}$  in step 1(a).(iv);  $\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}$  denotes a vector derived by the concatenation of  $\mathbf{k}$  times  $\mathbf{t}_{old}$  with  $\mathbf{t}$ .

## D.1. Experiments

In this section we evaluate our algorithms for knowledge transfer in small sample scenarios. We start by comparing the different methods on controlled synthetic data in Section D.1.1. We then test the performance of our method using several real data datasets employing different image representation schemes in two different settings: *medium size*, with several tens of classes and a dimensionality of 1000 features as im-

---

**Algorithm 3** Knowledge-Transfer with shared features projections

---

Input :

$L$  number of levels

$\{\mathbf{P}^l\}_{l=1}^L$  Set of projections matrices learnt from the  $k$  pre-trained tasks

---

Output :

**W**

1.  $\mathbf{W}^0 = 0$
  2. for  $l = 1$  to  $L$ 
    - (a) Projection:
      - i.  $\hat{\mathbf{x}} = \mathbf{P}^l * \mathbf{x}, \forall i \in [1..k]$  and  $\forall \mathbf{x} \in \mathbf{S}^i$
      - ii.  $\hat{\mathbf{w}}^{l-1} = \mathbf{P}^l * \mathbf{w}^{l-1}$
    - (b)  $\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \mathbf{L}(\{\hat{\mathbf{S}}^i\}_{i=1}^k, \mathbf{W} + \hat{\mathbf{W}}^{l-1})$
    - (c) Backprojection:  $\mathbf{w} = \mathbf{P}^l * \hat{\mathbf{w}}$
    - (d)  $\mathbf{w}^l = \mathbf{w}^{l-1} + \mathbf{w}$
  3.  $\mathbf{w} = \mathbf{w}^L$
- 

age representation in Section D.1.2; and *large size*, with hundreds of classes and an image representation of 21000 features in Section D.1.3. We also compared the performance in a '1-vs-rest' setting and in a setting with a common negative class (as in clutter).

The methods used for comparison are the following:

Batch methods

- *KT-Batch-H*: corresponds to Algorithm 3 where knowledge-transfer is based on the projection matrices extracted from the batch cascade learning.
- *KT-Batch-NoReg*: here knowledge transfer corresponds to Algorithm 3 with  $L = 1$  and  $\phi = 0$ , where information is transferred from the previously learnt models which were learnt in a single level with no regularization and no incentive to share information.

Online methods

- *KT-On-H*: corresponds to Algorithm 4 where we transfer information given the full cascade of regularization functions.
- *KT-On-L12*: corresponds to Algorithm 4 with  $L = 1$  and  $\lambda = 0$ , where a single level of informa-

---

**Algorithm 4** Online Knowledge-Transfer learning cascade

---

Input :

$L, \{\lambda^l\}_{l=1}^L, \{\phi^l\}_{l=1}^L, \gamma$  set of parameters as in Algorithm 2

$\{\bar{\mathbf{U}}_{old}^l\}_{l=1}^L$  the average subgradient of the last iteration of Algorithm 2

$\{\mathbf{W}_{old}^l\}_{l=1}^L$  the set of parameters learnt by Algorithm 2

$t_{old}$  number of temporal iterations of Algorithm 2

Initialization:

$$\hat{\mathbf{W}}_0^l = \mathbf{W}_{old}^l \circ \text{mean}(\mathbf{W}_{old}^l), \bar{\mathbf{U}}_0^l = \bar{\mathbf{U}}_{old}^l \circ 0 \quad \forall l \in \{1..L\}$$

$$\mathbf{W}_t^0 = 0 \quad \forall t$$

1. for  $t = 1, 2, 3, \dots$  do
    - (a) for  $l = 1$  to  $L$ 
      - i. Given the function  $\mathbf{L}_{t, \mathbf{W}_t^{l-1}}$ , compute a subgradient  $\mathbf{U}_{t, new}^l \in \partial \mathbf{L}_{t, \mathbf{W}_t^{l-1}}$
      - ii.  $\bar{\mathbf{U}}_{t, new}^l = \frac{t-1}{t} \bar{\mathbf{U}}_{t-1, new}^l + \frac{1}{t} \mathbf{U}_{t, new}^l$
      - iii.  $\bar{\mathbf{U}}_t^l = \bar{\mathbf{U}}_{old}^l \circ \bar{\mathbf{U}}_{t, new}^l$
      - iv.  $\hat{\mathbf{W}}_t^l = \underset{\mathbf{W}}{\text{argmin}} \bar{\mathbf{U}}_t^l \mathbf{W} + \psi^l(\mathbf{W}) + \frac{\gamma}{\sqrt{t_{old} \circ t}}, h(\mathbf{W}) >$
      - v.  $\mathbf{W}_t^l = \mathbf{W}_{old}^l \circ (\mathbf{W}_{t, new}^{l-1} + \hat{\mathbf{W}}_{t, new}^l)$
    - (b)  $\mathbf{W}_t = \mathbf{W}_t^L$
- 

tion transfer is based on models trained to share information between all tasks equally.

Baseline methods, without Knowledge Transfer:

- *NoKT-Batch*: corresponds to the multi-task batch Algorithm 1 with  $L = 1$  and  $\phi = 0$ .
- *NoKT-On-NoReg*: corresponds to the multi-task online Algorithm 2 with  $L = 1$  and  $\phi = 0$ .

#### D.1.1. SYNTHETIC DATA

To test Algorithms 3 and 4 we trained 99 tasks using the multi-task batch and online algorithms with only 99 tasks, keeping the remaining task aside as the unknown novel task. Each known task was trained with 50 examples, with 10 repetitions over the data for the online Algorithm 2. After this multi-task pre-processing had finished, we trained the left out task us-

ing Algorithms 3 and 4 with either 1-10, 20 or 50 examples (with 100 repetitions in the online Algorithm 4). This was done 100 times, leaving out in turn each of the original tasks. In the batch knowledge-transfer we chose the rank of each projection matrix to keep 99.9% of the variance in the data. In the hierarchical knowledge-transfer this resulted in approximately 10 dimensions at the top level of the cascade, and 90 dimensions at the lowest level of the cascade.

As can be seen in Fig. 10a, our Knowledge-Transfer methods based on shared multi-task models achieve the best performance as compared to the alternative methods. The online knowledge-transfer method achieves the best results on this dataset. Note that with very small samples, the method which attempts to share information with all tasks equally - *KT-On-L12* - achieves the best performance. As the sample increases to 50, Algorithm 4 is able to perform better by exploiting the different levels of sharing given by the hierarchical approach *KT-On-H*. For both online Knowledge-Transfer options we see a significant improvement in performance as compared to the online with no knowledge transfer approach, *NoKT-On-NoReg*.

Looking at the batch method we see that Knowledge-Transfer based on sharing information between the original models, *KT-Batch-H*, outperforms significantly the knowledge-transfer based on no sharing of information in the original model training, *KT-Batch-NoReg*. The *KT-Batch-NoReg* actually performs no better than the no knowledge-transfer approach *NoKT-Batch*.

It is also interesting to note that the difference in average performance between the novel task to the pre-trained tasks is less than 0.5% for 50 training examples when using the hierarchical knowledge-transfer. This indicates that in this experiment our hierarchical knowledge-transfer method reaches the potential of sharing as in the multi-task method, which outperforms all other methods on this synthetic data.

#### D.1.2. MEDIUM SIZE

We tested our method with real data, starting with a moderate problem size and only 31 classes. For these experiments we used a subset of the ILSVRC(2010) challenge (Berg et al., 2010), which is an image dataset organized according to the WordNet hierarchy. From this huge dataset we chose 31 classes (synsets)<sup>4</sup> for the

---

<sup>4</sup>quail, partridge, hare, Angora rabbit, wood rabbit, indri, Madagascar cat, orangutan, chimpanzee, gorilla, fire engine, garbage truck, pickup truck, trailer truck, police wagon, recreational vehicle, half track, snowmobile, trac-

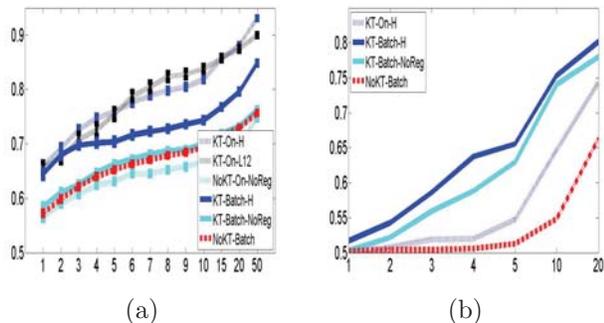


Figure 10. Accuracy comparison. In all plots the 'Y'-axis corresponds to the average accuracy over all tasks, and the 'X'-axis to the sample size. (a) Results for Synthetic data experiment. (b) results for the large size imagenet experiment.

set of pre-trained known classes with many training examples. This group of classes was chosen heuristically to contain varying levels of relatedness among classes, grouping together various terrestrial, aerial and sea vehicles, buildings, sea animals etc. For the novel classes with small sample we considered 30 randomly chosen classes from the remaining 969 classes in the dataset. The set of features used to describe images in this data set is based on the Sift features quantized into a code-book of 1000 words, which was tf-idf normalized.

We considered binary learning tasks where each chosen class, either pre-trained or novel, is contrasted with a set of images (negative examples) chosen from a group of different classes. The negative set was constructed in two ways: In the *1-vs-rest* condition the negative set of classes, the *Rest*, was defined as the group of 31 classes from which knowledge is transferred. In the second condition the negative set included 31 different classes sampled randomly from the original dataset excluding the small sample classes and the set of pre-trained classes. In this second condition all classes, both pre-trained and novel, had the exact same set of negative examples. This condition resembles previous experiments with knowledge-transfer (Zweig & Weinsshall, 2007), where all tasks share the same negative set.

In both conditions the pre-trained models were trained using 480 positive examples and 480 negative examples. For the positive set of the small sample classes, we considered a sample size in the range 1-20. For the negative set we used all examples from each negative

tor, tricycle, fiddler crab, king crab, silver salmon, rainbow trout, striper, airliner, warplane, lifeboat, catamaran, boathouse and church building.

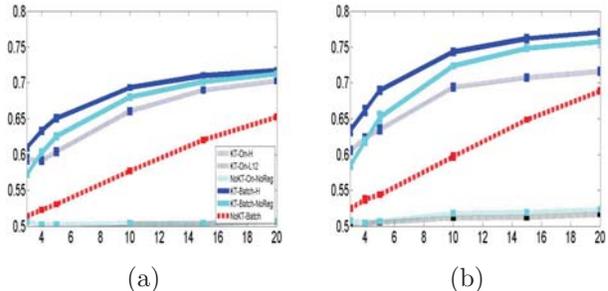


Figure 11. Mid-size experiment accuracy results, with (a) common negative set, and (b) 1-vs-rest. In all plots the 'Y'-axis corresponds to the average accuracy of all tasks, and the 'X'-axis to the sample size.

group (480 examples per class in the 1-vs-rest condition and 480 in total in the second condition). Examples were weighted according to sample size. For the pre-trained models we used a validation set of 60 examples per class; we used 100 examples per each class as its test set.

We considered each of the novel 30 classes separately. The experiment was repeated 8 times with different random splits of the data, for both the pre-trained and novel tasks. In the batch knowledge transfer methods we set the projection rank to maintain 99.9% of the variance in the original models learnt.

Results for the condition with shared negative set are shown in Fig. 11a. Results for the 1-vs-rest condition are shown in Fig. 11b. We see that knowledge-transfer methods achieve improved performance as compared to the alternative. In both conditions the best performer is the hierarchical batch approach for Knowledge-transfer, *KT-Batch-H*. The poor performance of the *KT-Online-L12* can be explained by the fact that the regularization coefficient  $\phi$  chosen by cross-validation during the multi-task pre-learning phase of the pre-trained models was chosen to be very low, indicating that a single level of sharing is not sufficient for this data.

### D.1.3. LARGE SIZE

As the ultimate knowledge transfer challenge, we tested our method with real data and large problem size. Thus we used all 1000 classes from the ILSVRC(2010) challenge (Berg et al., 2010). Each image was represented by a vector of 21000 dimensions, following the representation scheme used by (Zhao et al., 2011). 900 classes were chosen randomly as pre-trained classes, while the remaining 100 classes were used as the novel classes with small sample. We con-

sidered *1-vs-rest* tasks as explained above. Pre-trained tasks were trained using 500 examples from the positive class and 500 examples chosen randomly from the remaining 899 classes. We used the online Algorithm 2 with 2 repetitions over the training data to train pre-trained tasks.

During test, the set of negative examples in the *1-vs-rest* condition was chosen randomly from all of the dataset, total of 999 classes. We used the labeled test set provided by (Berg et al., 2010). As small sample we considered 1-20 examples per class. Due to the original large image representation, in the batch knowledge-transfer methods we fixed the projection rank to maintain only 80% of the variance in the original models learnt.

We note that once the pre-trained models are computed, each step of training with the projected batch approach is faster than each step of the online approach as the online Algorithm 4 needs at each step to consider all the pre-trained parameters in order to compute the regularization value, while the batch Algorithm 3 considers these parameters only once during the projection phase. Using a big image representation as we do the online methods becomes computationally expensive if repeating the experiment for each of the novel small samples classes separately.

Results are shown in Fig. 10b. Clearly all methods inducing information sharing outperformed significantly the batch and online learning with no sharing. The *NoKT-on-NoReg* method performed poorly similarly to *NoKT-batch* and was omitted for brevity. *KT-on-L12* also performed poorly due to the very low regularization parameter  $\phi$  automatically chosen.