Toward Correlating and Solving Abstract Tasks Using Convolutional Neural Networks

Kuan-Chuan Peng
Cornell University
kp388@cornell.edu

Tsuhan Chen
Cornell University
tsuhan@cornell.edu

Abstract

Most works using convolutional neural networks (CNN) show the efficacy of their methods in standard object recognition tasks, but not in abstract tasks such as emotion classification and memorability prediction, which are a subject of increasing importance (especially as machines become more autonomous, there is a need for semantic understanding). To verify whether CNN-based methods are effective in abstract tasks, we select 8 different abstract tasks in computer vision, evaluating the performance of 5 different CNN-based training approaches in these tasks. We show that CNN-based approaches outperform the state-of-the-art results in all the 8 tasks. Furthermore, we show that concatenating CNN features learned from different tasks can enhance the performance in each task. We also show that concatenating the CNN features learned from all the tasks under experiment does not perform the best, which is different from what is usually shown in previous works. Using CNN as a tool to correlate different tasks, we suggest which CNN features researchers should use in each task.

1. Introduction

Convolutional Neural Networks (CNN) have demonstrated better performance than non CNN-based approaches for various datasets in recent research [9, 13, 37, 38]. This trend of applying CNN-based approaches to computer vision problems started with Krizhevsky et al. [25] using CNNs to achieve significant improvements in ImageNet [6] classification results. Further, Donahue et al. [9] show that the network used in [25] can serve as a feature extractor which they use to outperform the state-of-the-art results on several generic tasks. Since then, more researchers have started studying and applying CNN-based approaches to different standard benchmark datasets. Summarizing several standard datasets and the recent CNN-related works using them in Table 1, we find that one unifying attribute of the datasets used in most CNN-related works is the fact that the associated tasks of these datasets are standard classical object/scene classification tasks (concrete tasks) instead of abstract tasks.

dataset(s)	CNN-related works
ImageNet [6]	[9, 13, 15, 25, 37]
PASCAL [11]	[1, 15]
SUN [34]	[1, 9, 13]
Caltech-101 [26]	[9, 15, 37]
Caltech-UCSD Birds 200 [33]	[9, 38]
8 **abstract tasks** w/ 6 datasets (Table 2)	**this work**

Table 1. Several standard datasets and the recent CNN-related works using them. In contrast to the concrete tasks associated with these datasets, we focus on 8 abstract tasks (associated with 6 datasets) which interest computer vision researchers recently.

Adopting the definition from Merriam-Webster dictionary [31], we use the term "abstract task" to refer to the task relating to or involving general ideas or qualities rather than specific people, objects, or actions. As opposed to an "abstract task", the term "concrete task" refers to a task that is not abstract. According to this definition, the tasks adopted in most CNN-related works are concrete tasks. Relatively few CNN-related works concentrate on the abstract tasks. The abstract tasks have recently received a great deal of focus, including classification tasks (emotions [28, 32], architectural styles [35], aesthetic qualities [27, 29], fashion styles [23], artist and artistic styles [20]) and regression tasks (memorability [4, 16, 17, 22, 24] and interestingness [12, 14]). Most of the mentioned references tackle the abstract tasks without using CNN. Figure 1 displays some example images from a concrete task and an abstract task along with their labels, showing that the labels are one major difference of the images from these two tasks.

In this work, we are interested in the performance of CNN in abstract tasks because it is not yet well studied in current literature. We are curious whether a CNN can outperform the current state-of-the-art results in abstract tasks. We would like to know, for a given network archi-

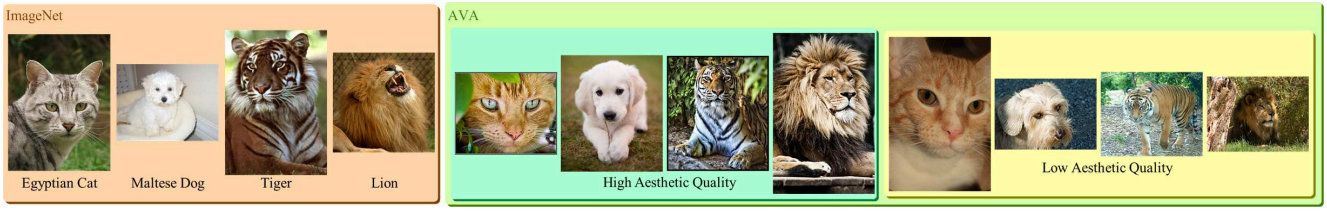

Figure 1. Example images from a concrete task (ImageNet [6] classification) and an abstract task (AVA [29] aesthetic classification). The left 4 images are from ImageNet [6], and the corresponding labels are shown at the bottom of the images. The right 8 images are from AVA [29]. In these 8 images, the left (right) 4 images are labeled as images with high (low) aesthetic quality.

dataset	Artphoto	Painting-91	Painting-91	AVA	HipsterWars	arcDataset	Memorability	Memorability
task	emotion classification	artist classification	artistic style classification	aesthetic classification	fashion style classification	architectural style classification	memorability prediction	interestingness prediction
reference	[28]	[20]	[20]	[29]	[23]	[35]	[17]	dataset: [17]; task: [14]
task ID	EMO	AST	ART	AVA	FAS	ARC	MEM	INT
# classes	8	91	13	2	5	10 / 25	regression task	regression task
# images	806	4266	2338	>250k	1893	2043 / 4786	2222	2222
image type	deviantart [7]	painting	painting	dpchallenge [10]	outfit	architecture	general	general
class labels	fear, sad, etc.	Rubens, Picasso, etc.	Baroque, Cubbism, etc.	high / low aesthetic quality	Bohemian, Goth, etc.	Georgian, Gothic, etc.	memorability	interestingness
# training images	∼645	2275	1250	∼233k	853	300 / 750	1111	1982
# testing images	∼160	1991	1088	19930	92	1743 / 4036	1111	240
data split	random	specified [20]	specified [20]	specified [29]	random	random	specified [17]	random
# fold(s)	5	1	1	1	100	10	25	10
evaluation metric	1-vs-all accuracy	accuracy	accuracy	accuracy	accuracy	accuracy	ρ	ρ
reference of above setting	[32]	[20]	[20]	[29]	[23]	[35]	[17]	[14]

Table 2. The datasets and associated abstract tasks used in this work along with their properties. In the entire paper, we refer to each task by the corresponding task ID listed under each task. The experimental setting for each task is provided at the bottom of the table, where ρ is Spearman rank correlation between the prediction and the ground truth.

tecture, which of the many approaches to training the network performs the best for abstract tasks. In addition, we are also curious about the following questions which are rarely discussed in current literature where different abstract tasks are treated in a relatively independent fashion. We would like to know whether the CNN features learned from an abstract tasks can improve the performance in another abstract task. Furthermore, we want to identify for a given abstract task, concatenating which of the many learned CNN features will perform the best. These questions are related to the transferability mentioned by Yosinski et al. [36]. However, Yosinski et al. [36] study the transferability only in the ImageNet [6] classification task. Our work is related to the transferability across different tasks and datasets. Our experiments include 8 abstract tasks from 6 datasets, covering the abstract tasks mentioned previously. The datasets and tasks used in this work are summarized in Table 2, where each task is assigned a task ID. In this paper, we use the task ID to refer to each abstract task.

Most works using CNN-related features, including the references in Table 1, adopt the CNN proposed in [25] and pre-train it with supervision for ImageNet [6] classification. As a strong and novel departure from the previous works, this paper uses not only the CNN features pre-trained with the supervision for ImageNet [6] but also the CNN features pre-trained with the supervision for AVA [29] under the same CNN architecture [25]. We compare the performance of these two sets of CNN features in all 8 abstract tasks in Table 2, identifying which set of features achieves better performance in each abstract task.

When using CNN-based approaches or features for abstract tasks, most existing works limit themselves to one specific domain instead of deriving cross-task insights and leveraging the CNN features learned from different domains of abstract tasks. For instance, Bar et al. [2] use CNN-based features to perform artistic style classification of the images in WikiArt dataset [3]. Peng et al. [30] predict emotion distributions with their proposed Emotion6 dataset. Karayev et al. [19] work on image style classification with their proposed datasets, but Lu et al. [27] are interested in classifying both aesthetic qualities and image styles using AVA dataset [29]. Though our datasets do not completely overlap, 3 of our 8 selected tasks in Table 2 (EMO, ART, and AVA) cover similar abstract tasks to theirs. Khosla et al. [21] adopt CNN-based features to predict popularity of Flickr images. Deza and Parikh [8] incorporate CNN-based

features to analyze image virality. We plan to include their abstract tasks as our future work. The novelty of this paper is in applying CNN-based features to 8 abstract tasks from 8 different domains, correlating different abstract tasks, and providing cross-task findings based on the empirical conclusions of the experimental results. The following subsection summarizes our findings.

1.1. Summary of Findings

Superior performance of CNN-based approaches in abstract tasks. Testing the performance of 5 different training approaches with the CNN in [25], we find that at least one of the five CNN-based approaches outperforms the current state-of-the-art performance in the 8 abstract tasks in Table 2.

Concatenating CNN features learned from different tasks can enhance the performance in each task. Unlike previous works [14, 17, 20, 23, 28, 29, 35] that only use standard or handcrafted features without using the features specifically trained for other abstract tasks, we argue that the performance of a given abstract task can benefit from the features learned from other existing abstract tasks. More precisely, our results show that for each of the abstract tasks in Table 2, concatenating the CNN features learned from some other task(s) which are different from the task of interest can outperform using only the CNN features learned from the task of interest. This finding supports that when the computer vision community keeps identifying and proposing brand new tasks, researchers should leverage the knowledge learned from other related tasks.

Concatenating CNN features learned from all the tasks does not perform the best. To identify which CNN features to concatenate will perform the best in each task, we evaluate different settings of concatenating CNN features. Our results show that concatenating the CNN features learned from all the tasks in our experiment does not perform the best, which is surprising and inconsistent with what is usually shown in previous works [16, 17, 20, 29] where combining all the features achieves the best performance. We also show that in some abstract tasks, using only the CNN features learned from the task of interest outperforms concatenating the CNN features learned from all the tasks. In addition, for each task, we identify the concatenating setting which results in the best performance in our experiment.

Suggestions of choosing CNN features to use in abstract tasks. We address that CNN can be used as a tool to correlate different abstract tasks. According to the performance of the CNN features learned from different abstract tasks, we are able to interpret a given abstract task using higher-level semantics instead of only using standard or handcrafted features like previous works [14, 20, 23, 28, 29]. For example, according to our results, artistic style features (F_ART) outperforms fashion style features (F_FAS) in the artist classification task (AST), where the notation "F_T" denotes CNN features learned from the task T (T is a task ID). For each task, we release the performance ranking of the CNN features learned from different tasks. The ranking is an indicator of which CNN features we should consider in each task. We hope this method of correlating different abstract tasks will encourage researchers to leverage the knowledge learned from existing tasks more when they solve their tasks of interest.

2. Experimental Setup

2.1. Datasets and Tasks

Performing 8 abstract tasks from 6 datasets, we summarize all the datasets and tasks used in this paper in Table 2 along with their properties and related statistics. We refer to each task by its task ID listed in Table 2, where the experimental setting associated with each task is also provided. In Table 2, "data split" indicates whether the training/testing splits are randomly generated or specified by the work proposing the dataset/task, and "# fold(s)" represents the number of different training/testing splits used for the task. The experimental setting of each task follows that of the corresponding reference listed at the bottom of Table 2. More detail experimental settings of the tasks EMO, AVA, and FAS are presented in the supplementary materials.

2.2. Network Architecture

For the 6 classification tasks in Table 2, we use the Caffe [18] implementation of the CNN introduced in [25] except that the number of the nodes in the output layer is set to the number of classes in each task. For the 2 regression tasks (MEM and INT), we also use the Caffe [18] implementation of the CNN introduced in [25] except that the number of the nodes in the output layer is changed to 1 to predict a real value and that the softmax loss layer is replaced with the Euclidean loss layer. When using the Caffe [18] implementation, we adopt its default training parameters for training the CNN for ImageNet [6] classification unless otherwise specified.

2.3. Training Approaches

Before training, we resize all the images to 256×256 which is the same size used to train the CNN for ImageNet [6] classification in Caffe [18] implementation. We directly adopt the Caffe reference model [18] (denoted as M_{ImageNet}) for ImageNet [6] classification. Using the same CNN architecture (except that the number of the nodes in the output layer is set to 2), we train an AVA [29] reference model (denoted as M_{AVA}) with ∼233k training images and training from scratch approach (randomly initialize all the CNN parameters and train with the training set).

training approach ID	description
pt ImageNet + ft	Pre-train with M_{ImageNet} and fine-tune all the CNN parameters using the training set.
pt ImageNet + ft-fc8	The same as "pt ImageNet + ft" except that only the CNN parameters associated with the edges directly connected to the output layer are allowed to be updated using the training set.
pt AVA + ft	Pre-train with M_{AVA} and fine-tune all the CNN parameters using the training set.
pt AVA + ft-fc8	The same as "pt AVA + ft" except that only the CNN parameters associated with the edges directly connected to the output layer are allowed to be updated using the training set.
train from scratch	Randomly initialize all the CNN parameters and train with the training set.

Table 3. The five different CNN training approaches used in this work. M_{ImageNet} is the Caffe [18] reference model trained for ImageNet [6] classification, and M_{AVA} is our trained reference model for AVA [29] classification. In this work, we refer to each training method by its training approach ID.

task ID evaluation metric	EMO 1-vs-all accuracy (%)	AST accuracy (%)	ART accuracy (%)	AVA accuracy (%)	FAS accuracy (%)	ARC accuracy (%)	MEM ρ	INT ρ
previous work	63.163 [32]	53.100 [20]	62.200 [20]	73.250 [27]	70.971 [23]	69.170 / 46.210 [35]	0.500 [22]	0.600 [14]
pt ImageNet + ft	60.127	**56.102**	**68.290**	n / a	**71.294**	**71.159 / 52.953**	**0.520**	**0.643**
pt ImageNet + ft-fc8	**64.724**	53.541	65.165	n / a	66.228	67.246 / 51.469	-0.140	0.339
pt AVA + ft	59.836	25.615	40.625	n / a	57.337	35.841 / 20.401	0.368	0.511
pt AVA + ft-fc8	60.644	4.671	18.015	n / a	27.554	18.233 / 8.290	0.080	-0.113
train from scratch	61.572	21.698	38.327	**74.436**	54.304	21.532 / 12.386	0.372	0.382

Table 4. The summary of the experimental results of the 8 abstract tasks listed in Table 2 using the five training approaches in Table 3. In this table, ρ is the Spearman rank correlation between the prediction and the ground truth. The **bold** numbers represent the best performance given the specified evaluation metric, and the underlined numbers indicate the performance better than that of "train from scratch".

We train a reference model for AVA [29] instead of other datasets in Table 2 because the number of training images in AVA [29] is large enough (>230k) such that training from scratch can achieve reasonable performance.

Given M_{ImageNet} and M_{AVA}, we list the 5 training approaches used in this work in Table 3, following the descriptions and setting of supervised pre-training and fine-tuning used in [1] unless otherwise specified. In Table 3, pre-training (pt) with M_{D} means using a data-rich auxiliary dataset D ($D \in \{\text{ImageNet}, \text{AVA}\}$) to initialize the CNN parameters. Fine-tuning (ft) means all the CNN parameters can be updated by continued training on the dataset of interest. "ft-fc8" is the same as "ft" except that only the CNN parameters associated with the edges directly connected to the output layer are allowed to be updated. In this paper, we use the training approach ID in Table 3 to refer to each training approach. Applying all the 5 training approaches listed in Table 3 to all the 8 tasks in Table 2, we report the experimental results and compare them with the corresponding state-of-the-art performance in Sec. 3. Our trained AVA [29] reference model, M_{AVA}, will be released upon publication. More details about the parameters used for training are shown in the supplementary material.

3. CNN Performance in Abstract Tasks

We summarize the experimental results of the selected 8 abstract tasks in Table 4, where more detailed results are available in the supplementary materials if there are multiple experimental settings for that task. Table 4 shows that in all the 8 tasks, at least one of the five training approaches in Table 3 outperforms the state-of-the-art methods. Table 4 also shows that for most of the 8 tasks, the training approaches involving pre-training and fine-tuning usually outperform training from scratch. For AST, ART, and ARC, the complete results are shown in Table 4, where the results of ARC are displayed in the form: 10-way / 25-way classification accuracy. The results of the other tasks are explained in detail in the supplementary materials.

Reviewing all the experimental results in Table 4 and the supplementary materials, we summarize the findings in the form of Q&A in Table 5, extract cross-task conclusions based on these findings, and provide probable reasons for some of these findings in the following paragraphs. In Table 5, Answer 2 shows that for all the listed abstract tasks except EMO and AVA, "pt ImageNet + ft" performs the best out of all the 5 training methods in Table 3. The reason why "pt ImageNet + ft-fc8" outperforms "pt ImageNet + ft" in EMO may result from the small amount of training data provided in Artphoto dataset [28]. The small amount of training data provide limited information on solving the task EMO, so updating all the CNN parameters according to the training set which is not representative enough is worse than keeping most CNN parameters the same as the initialized values learned from >1.2 million images in ImageNet [6]. For AVA, "pt AVA + ft-fc8" performs the best because the dataset used for pre-training is suitable for the task, and keeping most CNN parameters the same as the initialized values learned from >230k images in AVA [6] is better than

task ID	EMO 8-way classification	EMO 1-vs-all	AST	ART	AVA content-based testing set	AVA generic testing set	FAS	ARC	MEM	INT
Question 1	Can CNN-based methods outperform the state-of-the-art methods in the specified task?									
Answer 1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Question 2	Which training method in Table 3 results in the best performance?									
Answer 2	pt ImageNet ft-fc8	pt ImageNet ft-fc8	pt ImageNet ft	pt ImageNet ft	pt AVA ft-fc8	pt AVA ft-fc8	pt ImageNet ft	pt ImageNet ft	pt ImageNet ft	pt ImageNet ft
Question 3	Given "pt ImageNet," which one performs better, "ft" or "ft-fc8" ?									
Answer 3	ft-fc8	ft-fc8	ft	ft	ft	Tie	ft	ft	ft	ft
Question 4	Given "pt AVA," which one performs better, "ft" or "ft-fc8" ?									
Answer 4	ft	Tie	ft	ft	ft-fc8	ft-fc8	ft	ft	ft	ft
Question 5	Given "ft," which one performs better, "pt ImageNet" or "pt AVA" ?									
Answer 5	pt ImageNet	pt AVA	pt ImageNet	pt ImageNet	pt AVA	pt AVA	pt ImageNet	pt ImageNet	pt ImageNet	pt ImageNet
Question 6	Given "ft-fc8," which one performs better, "pt ImageNet" or "pt AVA" ?									
Answer 6	pt ImageNet	pt ImageNet	pt ImageNet	pt ImageNet	pt AVA	pt AVA	pt ImageNet	pt ImageNet	pt AVA	pt ImageNet
Question 7	Does the specified training method outperform "train from scratch" ?									
pt ImageNet + ft	Y	N	Y	Y	N	N	Y	Y	Y	Y
pt ImageNet + ft-fc8	Y	Y	Y	Y	N	N	Y	Y	N	N
pt AVA + ft	Y	N	Y	Y	Y	Tie	Y	Y	N	Y
pt AVA + ft-fc8	N	N	N	N	Y	Y	N	N	N	N

Table 5. The summary of the findings in Q&A form based on the experimental results (Table 4 and the supplementary materials) of all the abstract tasks in Table 2. We use the task ID in Table 2 and the training approach ID in Table 3 to refer to each task and each training method respectively.

updating CNN parameters according to a small training set. Answer 5 and 6 in Table 5 show that for most abstract tasks, pre-training from ImageNet [6] is better than pre-training from AVA [29]. The main exception is the task AVA, which is more relevant to AVA [29] than to ImageNet [6].

According to the findings in Table 5, we summarize the following tips for future researchers applying CNN-based approaches to abstract tasks. If there is no prior knowledge about the abstract task of interest, one reasonable way is applying all the 5 training approaches in Table 3 and selecting the one with the best performance on the validation set. However, if we have the prior knowledge of which dataset (out of all the possible datasets at hand which can be used for pre-training) is more relevant to the abstract task of interest, we can directly use that dataset for pre-training instead of trying all of them. Even if we have no prior knowledge, from our empirical findings in Table 5, "pt ImageNet + ft" usually performs well for abstract tasks.

4. Correlate Abstract Tasks

4.1. Experimental Setting

To find out whether the features learned from one task can enhance the performance in another task, we select a total of 9 tasks (the 8 abstract tasks in Table 2 and Caltech-101 [26] object classification task) in our additional experiment. We include Caltech-101 [26] object classification task (we use CAL as its task ID) because we are also curious about whether the features learned from a concrete task can improve the performance in abstract tasks. For the task CAL, following the same setting in [37] (30 training images per class), we achieve comparable accuracy as that reported in [37] by using the network architecture in Sec. 2.2 and the training approach "pt ImageNet + ft."

For each of the 9 tasks in the experiment, we train the corresponding CNN with the network architecture in Sec. 2.2 and the training approach "pt ImageNet + ft." We treat each of the 9 trained CNN as a feature extractor which takes an image as input and outputs a 4096-d feature vector from its topmost fully connected layer. We use "F_T" to represent the 4096-d feature vector output from the CNN trained with the task T where we call F_T "self feature." For example, F_EMO and F_AST are learned CNN features corresponding to emotion and artist classification respectively. In the task EMO, F_EMO is self feature, but F_AST is not.

With the 9 trained CNN feature extractors, we illustrate the framework of our experiment in Figure 2. Our goal is to evaluate the performance in each task under different settings of concatenating learned CNN features.

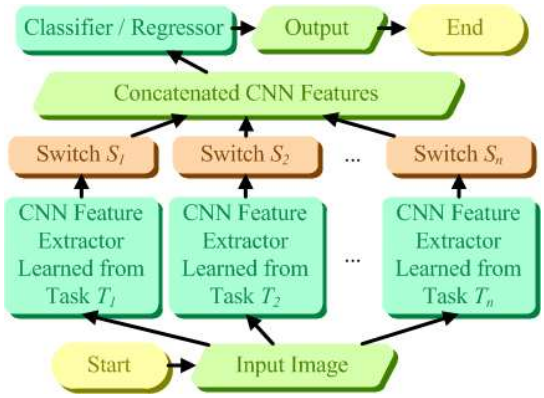

Figure 2. The framework of concatenating the CNN features learned from different tasks. We experiment on 9 tasks ($n = 9$), including the 8 abstract tasks in Table 2 and Caltech-101 [26] object classification task. The switch S_i associated with each task T_i ($i \in \{1, 2, \cdots, 9\}$) controls whether the CNN features learned from task T_i are concatenated in the final feature vector.

We generate the concatenated CNN features as follows. First, given an input image, we extract all the F_T_is where $i \in \{1, 2, \cdots, 9\}$ (T_i is one of the abstract task in Table 2 or CAL; $T_i \neq T_j$ if $i \neq j$). Second, we decide whether to concatenate F_T_i by the binary switch S_i associate with F_T_i. If S_i is set (reset), F_T_i will (will not) be part of the features concatenated to form the final feature vector. In other words, formed by concatenating all the F_T_i with set S_i, the final concatenated CNN features are a $4096 \times n_{set}$-d vector, where n_{set} is the total number of set S_i ($i \in \{1, 2, \cdots, 9\}$).

For each of the 9 tasks, we evaluate the performance under a total of 264 different settings of the 9 switches. These settings include: 1) 9 different combinations of S_i such that $n_{set} = 1$. 2) $2^8 - 1$ different combinations of S_i such that $n_{set} > 1$ and self feature is concatenated. For each task, we train a classifier or regressor for each of the 264 settings using the concatenated CNN features of the training dataset, and we test on the concatenated CNN features of the testing dataset using the trained classifier or regressor. In this experiment, we use support vector machine (SVM) or support vector regression (SVR) provided in LIBSVM [5], linear kernel, and the LIBSVM [5] default parameters to train all the classifiers and regressors.

Considering the efficiency of the experiment, we choose to do the first 5 folds of training/testing splits in the tasks FAS, ARC, MEM, and INT where more than 5 folds are provided. In the task EMO, we perform 8-way classification instead of 1-vs-all setting. For AVA dataset [29] associated with the task AVA, we use the generic training set with 2495 images to shorten the training time. In the task ARC, we do the 25-way classification task specified in Table 2. In the task CAL, we follow the setting in [37] (30 training images per class). Other experimental settings are consistent with Table 2 unless otherwise specified.

4.2. Experimental Results

We summarize the performance of concatenating learned CNN features in Table 6, where we compare the performance of concatenating all the 9 F_T_is with that of self feature in each task. The best performance out of 264 different settings is also listed in Table 6, and the corresponding best concatenating setting is identified in Table 7. In Table 6, the underlined numbers represent the performance better than that of using self feature. The performance of each of the 264 different settings in each task is provided in the supplementary materials. Table 6 supports the finding that concatenating the CNN features learned from different tasks can improve the performance in each task, regardless of whether the task is concrete or abstract. Table 6 also shows that in all the 9 tasks, concatenating all the learned CNN features is not the best concatenating setting, which is different from most previous works [16, 17, 20, 29] where combining all the features achieves the best performance. One possible reason is because the amount of training data in each task is not sufficiently large enough to perfectly train the 4096×9-d weight vector of SVM/SVR.

Table 6 shows that in the 4 tasks (AST, CAL, ARC, and FAS), concatenating all the learned CNN features performs even worse than using only self feature, which suggests that we should concatenate useful features instead of concatenating all the features. For each task, the best concatenating setting out of 264 different settings is identified in Table 7, which can serve as a guide to selecting useful features in each task. Table 7 also shows that F_AVA appears in the best concatenating setting in 6 out of 9 tasks, which supports that our trained model M_{AVA} we plan to release in Sec. 2.3 is beneficial to different tasks.

To be consistent with most previous works [14, 16, 17, 20, 23, 29] where the performance of each single feature is reported, we summarize the performance ranking in each task in Table 8, where the numbers are presented in the form "rank (performance)." In Table 8, the rank 1 (9) represents the best (worst) performance in each task. We also separately list the best and the worst performances out of the 9 F_T_is in each task. We want to address that even the worst performance in each task is much better than random guessing, which indicates that the non-best performance of concatenating all the features shown in Table 6 is not simply because of combining useless features together. In Table 8, using self feature (the diagonal performances) performs the best in all the 9 tasks except the two regression tasks MEM and INT. One possible reason is that SVR is not designed to maximize ρ, the evaluation metric specified by the works [14, 17] proposing these tasks.

task ID evaluation metric	EMO accuracy (%)	AST accuracy (%)	ART accuracy (%)	AVA accuracy (%)	FAS accuracy (%)	ARC accuracy (%)	MEM ρ	INT ρ	CAL accuracy (%)
self feature	36.228	55.148	67.555	69.423	76.957	54.440	0.398	0.573	88.217
best concatenating setting (Table 7)	39.082	57.509	71.048	69.980	77.609	55.382	0.507	0.630	88.394
concatenate all	36.971	54.596	69.210	69.458	74.348	53.489	0.504	0.629	85.969

Table 6. The summary of the experimental results using the framework in Figure 2. The task CAL and the 8 abstract tasks listed in Table 2 are included in this experiment. In this table, ρ is the Spearman rank correlation between the prediction and the ground truth. The underlined numbers indicate the performance better than that of using self feature. The best concatenating setting in each task is shown in Table 7. This table shows that concatenating the CNN features learned from different tasks can improve the performance. However, in our experiment, concatenating all the learned CNN features never performs the best in each task. In fact, in 4 out of 9 tasks, concatenating all the learned CNN features performs even worse than using self feature.

task ID	ART	AST	CAL	ARC	EMO	AVA	FAS	MEM	INT
F_ART	v			v		v	v	v	v
F_AST	v	v						v	v
F_CAL	v		v				v		v
F_ARC				v	v			v	v
F_EMO	v	v		v	v		v	v	
F_AVA	v		v	v		v		v	v
F_FAS		v	v	v			v	v	v
F_MEM	v				v	v		v	v
F_INT						v			v

Table 7. The best concatenating setting out of 264 different settings in each task. The corresponding performance is listed in Table 6. This table shows that the best performance of each task is not achieved by concatenating all the learned CNN features, but by concatenating a subset of them.

Given that self feature usually performs the best (out of 9 learned CNN features) as shown in Table 8, we are curious about the effect if we add an additional feature to the self feature. In Table 9, we analyze the performance ranking of using self feature plus the CNN features learned from another task. The format of the numbers is the same as that of Table 8 except that the underlined numbers represent the performance better than that of using only self feature. The best and the worst performances in each task out of the 8 different combinations of features are listed at the bottom of Table 9, where the performance of using only self feature is also provided.

Table 9 supports that most combinations of using self feature and another CNN features outperform using only self feature, which encourages us to use the CNN features learned from another task when solving the task of interest. However, there are some cases where combining features decreases the performance (as shown in Table 9). These cases address the importance of choosing useful features, and both Table 8 and Table 9 can serve as the indicators of choosing useful features. For instance, in the task ART, we should consider using F_AST to enhance the performance. Table 8 and Table 9 are also examples of using CNN as a tool to correlate different tasks. Leveraging the characteristic of CNN that the features learned from one task can be naturally bundled as a feature set (for example, F_ART), we are able to interpret a given task using higher-level semantics. For example, in fashion style classification (FAS), aesthetic features (F_AVA) are more useful (in terms of enhancing performance) than the features learned from artist classification (F_AST) according to both Table 8 and Table 9.

5. Future Work

Encouraged by the performance improvement from our proposed framework in Figure 2, we plan to replace the switches (S_is) with real-valued weights and design a new framework to automatically learn these weights from the task of interest such that the performance can be improved the most. In addition, this work combines the CNN features learned from different tasks by concatenating them, but there are other ways to combine different features. One of the goals of our ongoing project is to design an algorithm to better fuse different features. Furthermore, we will continue studying the relationship between different abstract tasks such that the knowledge learned from one task can be utilized in other tasks.

6. Conclusion

In this work, we apply 5 different CNN-based training approaches to the selected 8 abstract tasks receiving great attention in computer vision recently, showing that our results outperform the state-of-the-art results in all the 8 tasks. Unlike previous researchers who use standard or handcrafted features to solve abstract tasks, we propose a framework to leverage the CNN features learned from different tasks. By evaluating the performance of concatenating features in different settings, we show that using the CNN features leaned from one task can enhance the performance in another task. We also show that concatenating all the learned CNN features in this work is not the best option. Instead, we should identify the useful features in each task to achieve the best performance.

To identify the useful features in each task, we not only show the best concatenating setting but also use CNN as a

task ID	ART	AST	CAL	ARC	EMO	AVA	FAS	MEM	INT
F_ART	1 (67.555)	2 (48.569)	4 (81.080)	4 (48.845)	7 (31.138)	5 (63.181)	4 (68.478)	3 (0.454)	5 (0.520)
F_AST	1 (67.555)	1 (55.148)	5 (81.066)	6 (48.320)	6 (31.139)	7 (62.915)	6 (66.739)	6 (0.442)	6 (0.508)
F_CAL	3 (61.121)	5 (45.907)	1 (88.217)	3 (50.411)	5 (31.886)	3 (63.297)	7 (66.739)	1 (0.464)	8 (0.493)
F_ARC	4 (60.938)	4 (46.760)	2 (83.667)	1 (54.440)	2 (33.868)	4 (63.277)	2 (71.739)	7 (0.434)	9 (0.487)
F_EMO	4 (60.938)	3 (47.564)	8 (70.771)	2 (50.629)	1 (36.228)	2 (63.392)	5 (68.043)	2 (0.459)	7 (0.497)
F_AVA	7 (56.985)	7 (41.487)	6 (80.475)	7 (46.824)	4 (33.372)	1 (69.423)	3 (69.565)	5 (0.445)	1 (0.575)
F_FAS	6 (57.813)	6 (44.751)	3 (81.514)	5 (48.697)	3 (33.748)	6 (63.006)	1 (76.957)	4 (0.450)	4 (0.542)
F_MEM	9 (51.838)	9 (37.167)	9 (65.643)	9 (40.986)	9 (27.170)	9 (61.004)	9 (60.870)	8 (0.398)	3 (0.560)
F_INT	8 (55.790)	8 (40.532)	7 (74.316)	8 (43.925)	8 (30.029)	8 (61.666)	8 (66.087)	9 (0.346)	2 (0.573)
evaluation metric	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	ρ	ρ
best performance	67.555	55.148	88.217	54.440	36.228	69.423	76.957	0.464	0.575
worst performance	51.838	37.167	65.643	40.986	27.170	61.004	60.870	0.346	0.487

Table 8. The performance ranking in each task by using the CNN features learned from a single task. The numbers are presented in the form "rank (performance)." The rank 1 (9) represents the best (worst) performance in each task. The table shows that self feature usually performs the best if we use the CNN features learned from one task. The best and the worst performances out of the 9 different features in each task are listed at the bottom of the table, which shows that even the worst performance is non-trivial (much better than random guessing).

task ID	ART	AST	CAL	ARC	EMO	AVA	FAS	MEM	INT
self feature + F_ART	n / a	3 (56.404)	4 (88.231)	2 (55.124)	1 (37.095)	6 (69.498)	2 (76.957)	7 (0.459)	6 (0.598)
self feature + F_AST	1 (70.129)	n / a	5 (87.823)	6 (54.574)	3 (36.560)	3 (69.528)	7 (74.348)	3 (0.466)	4 (0.599)
self feature + F_CAL	2 (68.658)	2 (56.605)	n / a	5 (54.747)	5 (36.352)	8 (69.323)	4 (76.522)	2 (0.468)	5 (0.598)
self feature + F_ARC	2 (68.658)	7 (55.098)	3 (88.244)	n / a	7 (35.855)	5 (69.503)	5 (75.652)	5 (0.463)	7 (0.598)
self feature + F_EMO	4 (68.382)	1 (57.308)	6 (87.728)	1 (55.149)	n / a	7 (69.473)	6 (75.435)	1 (0.470)	3 (0.599)
self feature + F_AVA	6 (67.739)	6 (55.349)	1 (88.319)	4 (54.990)	2 (36.601)	n / a	1 (77.391)	4 (0.465)	1 (0.610)
self feature + F_FAS	7 (67.647)	4 (56.103)	2 (88.299)	3 (55.064)	6 (36.105)	1 (69.729)	n / a	6 (0.461)	2 (0.603)
self feature + F_MEM	5 (67.831)	5 (55.550)	7 (86.975)	7 (53.271)	4 (36.479)	3 (69.528)	2 (76.957)	n / a	8 (0.585)
self feature + F_INT	8 (67.096)	8 (53.541)	8 (85.834)	8 (52.359)	8 (35.361)	2 (69.594)	7 (74.348)	8 (0.417)	n / a
evaluation metric	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	accuracy (%)	ρ	ρ
self feature only	67.555	55.148	88.217	54.440	36.228	69.423	76.957	0.398	0.573
best performance	70.129	57.308	88.319	55.149	37.095	69.729	77.391	0.470	0.610
worst performance	67.096	53.541	85.834	52.359	35.361	69.323	74.348	0.417	0.585

Table 9. The performance ranking in each task by using self feature and the CNN features learned from another task. The format of the numbers is the same as that of Table 8 except that the underlined numbers represent the performance better than that of using only self feature. The best and the worst performances in each task out of the 8 different combinations of features are listed at the bottom of the table, where the performance of using only self feature is also provided. The table shows that most of the listed feature combinations outperform using only self feature in each task.

tool to correlate different tasks. By presenting the performance ranking of the CNN features learned from different tasks, we provide suggestions of which CNN features to use in each task. We hope that the results of this work will encourage researchers proposing new tasks or interested in existing tasks to cooperate instead of only focusing on the task of interest without utilizing the knowledge learned from existing tasks.

References

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, pages 329–344, 2014.

[2] Y. Bar, N. Levy, and L. Wolf. Classification of artistic styles using binarized features derived from a deep neural network, 2014.

[3] I. Ben-Shalom, N. Levy, L. Wolf, N. Dershowitz, A. Ben-Shalom, R. Shweka, Y. Choueka, T. Hazan, and Y. Bar. Congruency-based reranking. In *CVPR*, pages 2107–2114, 2014.

[4] B. Celikkale, A. Erdem, and E. Erdem. Visual attention-driven spatial pooling for image memorability. In *CVPRW*, pages 976–983, 2013.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[7] deviantart. `http://www.deviantart.com`.

[8] A. Deza and D. Parikh. Understanding image virality. In *CVPR*, 2015.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[10] dpchallenge. `http://www.dpchallenge.com/`.

[11] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88:303–338, 2010.

[12] Y. Fu, T. M. Hospedales, T. Xiang, S. Gong, and Y. Yao. Interestingness prediction by robust learning to rank. In *ECCV*, pages 488–503, 2014.

[13] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, pages 392–407, 2014.

[14] M. Gygli, H. Grabner, H. Riemenschneider, F. Nater, and L. V. Gool. The interestingness of images. In *ICCV*, pages 1633–1640, 2013.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, pages 346–361, 2014.

[16] P. Isola, J. Xiao, D. Parikh, A. Torralba, and A. Oliva. What makes a photograph memorable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1469–1482, 2014.

[17] P. Isola, J. Xiao, A. Torralba, and A. Oliva. What makes an image memorable? In *CVPR*, pages 145–152, 2011.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[19] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. In *BMVC*, 2014.

[20] F. S. Khan, S. Beigpour, J. V. D. Weijer, and M. Felsberg. Painting-91: a large scale database for computational painting categorization. *Machine Vision and Applications*, 25:1385–1397, 2014.

[21] A. Khosla, A. D. Sarma, and R. Hamid. What makes an image popular? In *WWW*, pages 867–876, 2014.

[22] A. Khosla, J. Xiao, A. Torralba, and A. Oliva. Memorability of image regions. In *NIPS*, pages 296–304, 2012.

[23] M. H. Kiapour, K. Yamaguchi, A. C. Berg, and T. L. Berg. Hipster wars: Discovering elements of fashion styles. In *ECCV*, pages 472–488, 2014.

[24] J. Kim, S. Yoon, and V. Pavlovic. Relative spatial features for image memorability. In *ACMMM*, pages 761–764, 2013.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[26] F.-F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW*, 2004.

[27] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang. RAPID: rating pictorial aesthetics using deep learning. In *ACMMM*, 2014.

[28] J. Machajdik and A. Hanbury. Affective image classification using features inspired by psychology and art theory. In *Proceedings of the International Conference on Multimedia*, pages 83–92, 2010.

[29] N. Murray, L. Marchesotti, and F. Perronnin. AVA: A large-scale database for aesthetic visual analysis. In *CVPR*, pages 2408–2415, 2012.

[30] K.-C. Peng, A. Sadovnik, A. Gallagher, and T. Chen. A mixed bag of emotions: Model, predict, and transfer emotion distributions. In *CVPR*, 2015.

[31] Merriam-Webster Online: Dictionary and Thesaurus. `http://www.merriam-webster.com/`.

[32] X. Wang, J. Jia, J. Yin, and L. Cai. Interpretable aesthetic features for affective image classification. In *IEEE International Conference on Image Processing*, pages 3230–3234, 2013.

[33] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[34] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010.

[35] Z. Xu, D. Tao, Y. Zhang, J. Wu, and A. C. Tsoi. Architectural style classification using multinomial latent logistic regression. In *ECCV*, pages 600–615, 2014.

[36] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[38] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based R-CNNs for fine-grained category detection. In *ECCV*, pages 834–849, 2014.