On the Behavior of Convolutional Nets for Feature Extraction

Garcia-Gasulla, D.[1], Parés, F.[1], Vilalta, A.[1], Moreno, J.[1]
Ayguadé, E.[1,2], Labarta, J.[1,2], Cortés, U.[1,2] & Suzumura, T.[1,3]

[1] Barcelona Supercomputing Center (BSC)
[2] Universitat Politècnica de Catalunya - BarcelonaTECH
[3] IBM T.J. Watson

{dario.garcia, ferran.pares, armand.vilalta, jonatan.moreno}@bsc.es

March 13, 2017

Abstract

Convolutional neural networks (CNN) are representation learning techniques that achieve state-of-the-art performance on almost every image-related, machine learning task. Applying the representation languages build by these models to tasks beyond the one they were originally trained for is a field of interest known as transfer learning for feature extraction. Through this approach, one can apply the image descriptors learnt by a CNN after processing millions of images to any dataset, without an expensive training phase. Contributions to this field have so far focused on extracting CNN features from layers close to the output (*e.g.*, fully connected layers), particularly because they work better when used out-of-the-box to feed a classifier. Nevertheless, the rest of CNN features is known to encode a wide variety of visual information, which could be potentially exploited on knowledge representation and reasoning tasks. In this paper we analyze the behavior of each feature individually, exploring their intra/inter class activations for all classes of three different datasets. From this study we learn that low and middle level features behave very differently to high level features, the former being more descriptive and the latter being more discriminant. We show how low and middle level features can be used for knowledge representation purposes both by their presence or by their absence. We also study how much noise these features may encode, and propose a thresholding approach to discard most of it. Finally, we discuss the potential implications of these results in the context of knowledge representation using features extracted from a CNN.

1 Introduction

Image classification is among the most popular applications of deep learning. The performance of deep learning networks on challenges like the ImageNet Large Scale Visual Recognition Competition (ILSVRC) is based on the capabilities of these models at building an exceptionally rich representation language for a given dataset. A language that can be built and used for problems like classification or detection, achieving state-of-the-art performance [6]. Coherently, deep learning models are frequently defined as representation learning techniques [7]. Unfortunately, to build this representation language, deep networks require a lot of data and a lot of computational effort, which reduces the number of problems for which these models can be directly applied to.

Since the popularization of deep models, a lot of effort has been dedicated at trying to reuse the representation language learnt for a given task `t0` to solve a different problem `t1`. A field of research commonly known as *transfer learning*. Transfer learning is most frequently used to initialize a deep network using the features learnt for `t0`, so that the same network can be optimized later on through a fine-tuning process for the `t1` task. Significantly, this approach has been shown to produce better results than training a network for `t1` from scratch (*i.e.*, using random initialization) [17]. An alternative application of transfer learning is to use a neural network

trained for t0 as a feature extractor of t1, in order to use other machine learning methods on top of the resulting t1 representations. By doing so one is representing the t1 data in a language learnt for the t0 dataset, enabling the use of pre-trained deep network representations (not deep network models by themselves) on datasets which lack the size required to train these methods [1, 14]. This approach to transfer learning can be used to tackle unsupervised learning problems such as clustering [5].

In the context of convolutional neural networks (CNNs), most attempts at transfer learning for feature extraction have focused on using layers close to the output. When compared to lower-level layers, these high-level layers provide the best results when used out-of-the-box on top of classifiers or clustering algorithms [1, 14, 5, 3]. Regardless of these results, it is known that CNN convolutional layers encode a large amount of visual knowledge of varying complexity [18, 4], knowledge which has not been successfully exploited so far. Since the general purpose of transfer learning for feature extraction is to maximize representativeness, we may hypothesize that the optimal representation in this context will include, to some degree, information from a larger variety of layers (*i.e.*, beyond the fully connected ones). It is therefore relevant, particularly for the knowledge representation and reasoning fields, to understand what differences are there between convolutional layers and fully connected layers, so that all that is learnt by a CNN can be properly exploited in the future.

For that purpose, in this paper we analyse the behaviour of all features within a deep CNN in the context of feature extraction. We use a very deep CNN (VGG19 [15]) pre-trained on a large dataset (*ImageNet*) to build image representations for alternative datasets (mit67, flowers102, cub200), and study the behavior of individual features within specific classes in the context of a dataset. §2 introduces previous contributions to the transfer learning field, focusing on feature extraction. §3 introduces the datasets and CNN model used in our experiments, as well as the image embedding we build in our feature extraction process. The basis of our study are statistic distance methods, which we review in §4. The core of our experiments is in §5, along which we obtain various insights on the feature extraction process, and on dataset particularities. Conclusions are summarized and discussed in §6.

2 Related Work

CNN models are defined by a large number of parameters, requiring lots of images for their optimization. Until the release of large visual datasets, hand-made features [10] consistently achieved the best results for vision tasks. Nowadays, CNNs can be trained using datasets such as *ImageNet* and *VOC*, learning strong and powerful visual descriptors, which allows them to outperform previously competitive solutions like Improved Fisher Vectors on many visual tasks [2].

One of the first contributions studying the behavior of convolution filters in a feature extraction process was introduced in [3], where authors study a CNN (AlexNet architecture composed by 5 convolutional layers and 3 fully connected layers) trained using the *ImageNet 2012* (as t0) for transfer learning. Qualitatively, authors observe how the first fully connected layer is better at separating the high levels of the WordNet hierarchy than lower layers. Quantitatively, authors evaluate the features extracted from the last convolutional layer and the two first fully connected layers on various datasets (*e.g.*, Caltech-102, t1). Their results indicate that by using features from the first fully connected layers to train a support vector machine (SVM) one can achieve state-of-the-art results on various related tasks.

The contribution of [14] goes in a similar direction, using the OverFeat network architecture pre-trained using the ImageNet 2012 dataset (t0). Authors focus mostly on the first fully connected layer, performing data augmentation to increase the quality of those features (cropping and rotating samples), and doing component-wise power transform. After applying a l2-normalization to the resultant vectors, an SVM is trained and applied to a wide variety of tasks (t1, image classification, fine grained recognition, attribute detection, visual instance retrieval), achieving competitive results on all of them. For one of those problems (image classification), features from various layers are evaluated separately using an SVM, with the first fully connected layer obtaining the best results.

A rather different approach is depicted in [17], where the goal is to study the transferability of features for the purpose of fine tuning. In that regard, authors find that the distance between the source and target tasks is strongly related with the depth of the optimal layer to use for the transfer learning process.

In [1] authors empirically evaluate several parameters that can affect the transfer learning process for feature extraction. Among the parameters they consider some are related with the architecture and training of the initial CNN (network depth and width, distribution of training data, optimization parameters), and some are related with the transfer learning process (fine-tuning, network layer to be extracted, spatial pooling and dimensionality reduction). All these parameters are evaluated on 17 visual recognition tasks, identifying a set of apparently optimal parameters depending on the distance between the original training task and the target transfer learning task. Regarding the representation layer (which layer is used to build the embedding) authors find that the first or second fully connected layer produce the best results on most tasks when using an SVM.

Deep residual networks (ResNets) are an evolution of traditional CNN which include branching of paths. Unlike CNNs, ResNets do not stack layers, implementing shortcut connections instead, which eases the convergence during the training process, allowing the training of networks with more layers (up to thousands). In [8] authors explore the use of ResNets for feature extraction, particularly to solve three image classification problems. Results indicate that ResNets are a competitive alternative to classic CNN architectures, and that deep convolutional ResNet layers provide competitive results.

3 Models and Datasets

As discussed in §1, one of the applications of transfer learning is to apply the visual representation language learnt by a CNN for a original task `t0` to solve a target task `t1`. This is particularly useful when the target task does not include enough data to build its own CNN representation language. In a transfer learning for feature extraction scenario, `t1` will be expressed in features crafted for `t0`, and the quality of the resultant embedding will strongly depend on how similar `t0` and `t1` are. If the language learnt for `t0` lacks the vocabulary to define `t1` particularities (*e.g.*, because `t0` is defined by black and white images, and `t1` includes colorful patterns), the resultant embedding will be of poor quality and any learning applied to it will be deficient. With that in mind, `t0` is typically chosen to capture a range of visual patterns as broad as possible, so that its language will be likely to include features relevant for many different `t1` tasks, and capable of characterizing a wide variety of image classes. A CNN trained for the *ImageNet 2012* dataset[12] is a perfect candidate, since its hierarchy of 1,000 categories includes a huge variety of visual patterns hardly found on any other dataset.

Beyond the dataset and training task, to completely specify `t0` it is necessary to define the network architecture. There are many popular CNN architectures, and various have been used for feature extraction (as discussed in §2). Since our goal is to explore the behavior of convolutional layers in the feature extraction process, we will use an architecture which follows the most simple convolutional scheme of layers (*i.e.*, conv/pool/.../fc). However, we wish to use a model capable of building a very rich representation language (*i.e.*, a very deep network). This combination of requirements leads us to use the VGG19 architecture as source of features [15]. VGG19 is composed by 16 convolutional layers (with 5 pooling layers) and 3 fully-connected layers (see Table 1 for details on the architecture). To sum up, we use the VGG19 model pre-trained on the *ImageNet 2012* dataset [15]. This model is publicly available at the authors web page [1].

Once we have defined the source task (*i.e.*, `t0`), let us introduce the three datasets we will consider as target (*i.e.*, `t1`) in our transfer learning process:

- MIT Indoor Scene Recognition dataset (*mit67*) [11]. An image classification problem.
- Caltech-UCSD Birds-200-2011 dataset (*cub200*) [16]. A fine-grained recognition problem.
- Oxford Flower dataset (*flowers102*) [9]. A fine-grained recognition problem.

[1] http://www.robots.ox.ac.uk/∼vgg/research/very_deep/

Table 1: Details on the VGG19 architecture. For each layer, number of filters, parameters and activations.

Layer name	#Filters	#Parameters	#Activations
input			150K
conv1_1	64	1.7K	3.2M
conv1_2	64	36K	3.2M
max pooling			802K
conv2_1	128	73K	1.6M
conv2_2	128	147K	1.6M
max pooling			401K
conv3_1	256	300K	802K
conv3_2	256	600K	802K
conv3_3	256	600K	802K
conv3_4	256	600K	802K
max pooling			200K
conv4_1	512	1.1M	401K
conv4_2	512	2.3M	401K
conv4_3	512	2.3M	401K
conv4_4	512	2.3M	401K
max pooling			100K
conv5_1	512	2.3M	100K
conv5_2	512	2.3M	100K
conv5_3	512	2.3M	100K
conv5_4	512	2.3M	100K
max pooling			25K
fc6		103M	4K
fc7		17M	4K
output		4M	1K

Dataset sizes, number of classes and train/test splits are specified in Table 2. In our experiments we will not train models using these datasets, which means we do not require the provided train and test splits. Instead, we will use the subsets that are most similar in size (train for *mit67* and *cub200*, test for *flowers102*), so that differences in our analysis are not caused by a differences in dataset size.

3.1 Embedding

Given the source task `t0` and the target task `t1`, there are several parameters than can modify the construction of the embedding space. Most of those parameters were explored in [1]. In our case, we will use two main parameters which we consider to be coherent with our study. First, each image representation will be built as a result of processing 10 crops of the image (4 corners and middle crop, mirrored) through the CNN and averaging the resulting activations. This is

Table 2: For each of the three target datasets, total number of images, number of classes, distribution of images between train and test splits, and number of images per class. In bold, split of the dataset used in our analysis.

Dataset	#Images	#Classes	Train set size	Test set size	#Images per class
mit67	6,700	67	**5,360**	1,340	78 - 82
cub200	11,788	200	**5,994**	5,794	29 - 30
flowers102	8,189	102	2,040[a]	**6,149**	20 - 238

[a] Size of train set plus validation set.

Table 3: Description of layers and features composing the embedding.

Layer Name	# Features
conv1_1	64
conv1_2	64
conv2_1	128
conv2_2	128
conv3_1	256
conv3_2	256
conv3_3	256
conv3_4	256
conv4_1	512
conv4_2	512
conv4_3	512
conv4_4	512
conv5_1	512
conv5_2	512
conv5_3	512
conv5_4	512
fc7	4,096
Total	9,600

a frequently used methodology for feature extraction [14, 1], which provides robustness to the resultant representations. Second, we perform a spatial average pooling of each convolutional layer to obtain a single value per filter. This transformation reduces the number of features in the embedding, as well as the relative spatial information (*i.e.*, each resulting feature will determine if a visual pattern is found or not in the image on average, regardless of its exact location), while maintaining most of its descriptive power (*i.e.*, each feature is still separately accounted for in the embedding). This spatial pooling methodology is also a recurrent solution in the field [14, 1].

Since we wish to explore in depth the behavior of convolutional layers, our final embedding will contain all the 16 convolutional layers available in VGG19 (from conv1_1 to conv5_4). For comparison purposes we will also extract the second fully connected layer (fc7), so that we can contrast the behavior of the convolutional layers with that of a fc layer. Notice the spatial pooling performed on the convolutional layers cannot be applied to the fc layers. The exact components of the resultant embedding, composed by 9,600 values, is shown in Table 3. For the remaining of the document, all mentions to *the embedding* will refer to this specific representation.

4 Statistic Methods on Distributions

Previous studies on the usefulness of convolutional layers for feature extraction transfer learning have been purely empirical, based on the performance of specific classifiers (most frequently, an SVM) using the features extracted from a single layer of a CNN (see §2). This approach has been shown to provide consistent results, but it is limited to classification, and strongly influenced by the choice of classifier (*e.g.*, some classifiers may perform better with a certain number of variables, or may be affected differently by noise).

In this paper we propose a different approach to evaluate the behavior of CNN features. Instead of evaluating the performance of a specific machine learning algorithm on the embedding, we measure the descriptive power of CNN features statistically, studying their behavior for the different classes composing the datasets. The goal is to learn about the descriptive nature of CNN features, so that other knowledge representation and reasoning methodologies can be adapted accordingly.

In detail, our approach consist on evaluating how characteristic each embedding feature is for each of the target classes of the various datasets. CNN neurons do not have a crisp behavior *w.r.t.* classes (*i.e.*, neurons do not activate binarily depending on the class), not even for the original

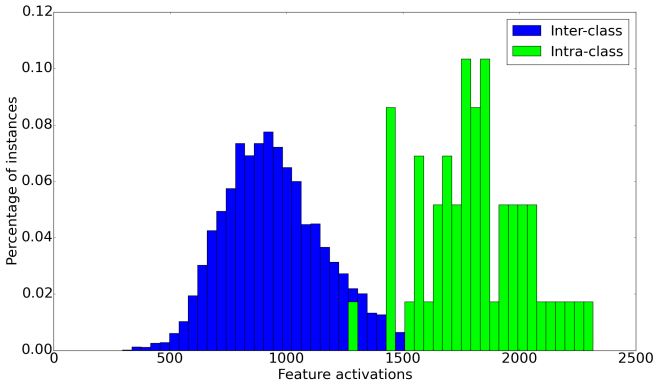

Figure 1: Intra/inter class density estimations for feature 221 from layer conv4_1, for class *gazania* (71) of the flowers102 dataset.

training task `t0`. Instead, each CNN neuron provides a fuzzy piece of information for each class, such that a richer distributed representation is built by considering all features. To contextualize the information provided by individual features we consider their activations within a target class (*intra-class behaviour*), comparing it with its activations for the rest of the dataset classes (*inter-class behaviour*). This will also give us some insight on how these features would perform on their own for knowledge representation purposes.

The intra/inter class behaviour can be visualized through two histograms of feature activations (see Figure 1). Statistically speaking, rescaled histograms are density estimations approximating a true probability density function (PDF). Although more sophisticated methods are available [13] we use the histogram for the sake of computational simplicity. To study the behavior of a given CNN feature for a given class, we compare the corresponding intra/inter density estimations. The first statistic we consider using for that purpose is the well-known Kullback-Leibler (KL) divergence.

The Kullback-Leibler (KL) divergence measures how much two PDF, P and Q, differ following the Equation 1, where i are the points in the domain.

$$D_{KL}(P,Q) = \sum_i P(i) ln \frac{P(i)}{Q(i)} \tag{1}$$

Although histograms are only approximations of PDFs, it is possible to fit a PDF (*e.g.*, normal distribution, uniform distribution, etc) to a histogram. This is, however, inconvenient since the histogram of different features may be fit by different PDFs, and there may be some features which are not properly fitted by a PDF.

The Bhattacharyya measure is an alternative to KL which can be applied to two discrete probability distributions. Analogously, it can be measured from two density estimations P & Q following the Equation 2, where i are the discrete points in domain X.

$$D_B(P,Q) = -ln\left(\sum_{i \in X} \sqrt{P(i)Q(i)}\right) \tag{2}$$

By comparing two density estimations, the Bhattacharyya measure can be used directly on the data, without having to choose a fitting PDF. However, its mathematical range is only positive ($[0, \infty)$), making Bhattacharyya measure unable to identify which density estimation is *above* and which is *below*. In our analysis it will be of interest to know if an intra-class behaviour is *higher* than the inter-class behavior or vice versa, since both situations may provide different insights.

The Kolmogorov-Smirnov statistic (D_{KS}) compares two empirical distribution functions (EDF) P and Q. For each point i in the domain X, D_{KS} measures the gap between $P(i)$ and $Q(i)$, and finds the maximum one. A signed version of the Kolmogorov-Smirnov statistic is formally defined in Equation 3. To reduce the computational cost of evaluating D_{KS}, we discretize the domain of each EDF into 100 bins, thus decreasing the domain resolution by 1%. Notice that, since EDF

Figure 2: 8 embedding features with very high D_{KS} (among top 10) for the mit67 dataset. Class producing that high D_{KS} is shown below each feature. Each feature corresponds to a specific neuron or filter from original CNN. To illustrate captured visual patterns of each feature, we show 9 cropped images from *ImageNet 2012* validation set producing highest activation values for this neuron or filter. Images are cropped to match the neuron receptive field.

is a cumulative distribution, D_{KS} is directly computed on a set of values (one pair of intra/inter class behaviours).

$$\begin{aligned} i' &= \arg\max_{i \in X} |P(i) - Q(i)| \\ D_{KS}(P, Q) &= P(i') - Q(i') \end{aligned} \tag{3}$$

In contrast with Bhattacharyya, D_{KS}'s mathematical range is $[-1, 1]$. $D_{KS} = 0$ means that both distributions are identical, while $D_{KS} = 1$ and $D_{KS} = -1$ means that both distributions do not intersect at some point. The sign indicates which EDF is above and which is below at the point where P and Q differ most. This mathematical range symmetry makes D_{KS} very suitable for our analysis, as it allows us to differentiate when intra class behavior is above inter class behavior and vice versa. On all following experiments we will use the Kolmogorov-Smirnov statistic (D_{KS}) as defined in Equation 3. This is a slightly modified version of the original Kolmogorov-Smirnov statistic, for the purpose of keeping the sign of the measure. Although D_{KS} is not formally a distance, for simplicity reasons in the remainder of this paper we may refer to it as such.

5 Statistic Analysis

Our statistic distance analysis is based on the intra/inter class D_{KS}. Simply put, a distance $D_{KS}(f, c) \simeq 0$ means that the distribution of activations of feature f for all the images belonging

to class c (*i.e.*, I_c) is almost identical to the distribution of values of feature f for all the images that do not belong to that class (*i.e.*, $I_{\neg C}$). If $D_{KS}(f, c) > 0$ then feature values for images I_c tend to be higher than for the rest of images $I_{\neg c}$, which implies that the visual elements represented by feature f are more commonly found in I_c images than in $I_{\neg C}$ images. Similarly, if $D_{KS}(f, c) < 0$, feature values are in general lower for I_C than for $I_{\neg c}$, which implies that elements represented by feature f are rare within I_c images when compared to the rest of the dataset.

To illustrate this behavior we explore which are the features with the highest D_{KS} values for the mit67 dataset. Figure 2 shows some of the top ten $D_{KS}(f, c)$ for various layers, indicating the class c in which the large D_{KS} value occurs. To provide insight into what a particular feature is encoding, we plot the 9 image crops from the *ImageNet 2012* validation set producing the highest activation value for that feature (images from this dataset will be better feature characterizations, since CNN features were originally trained for *ImageNet* classes). In this example, features having a high D_{KS} value for the *Greenhouse* class are either showing plants, grass or fields, regardless of the layer depth. The feature with a high D_{KS} value for the *Buffet* class identifies food on plates, while the feature with a high D_{KS} value for the *Cloister* identifies Gothic arches.

5.1 Negative D_{KS} values

Analogously to the study of positive D_{KS} values of Figure 2, we consider the lowest D_{KS} values (*i.e.*, closer to -1). Initially, one could expect that the features having the lowest D_{KS} for a given class c would be those identifying elements which never appear in the images of c. For example, a hypothetical class *whale* could be expected to have a extremely negative D_{KS} for a feature identifying a car. However, since the D_{KS} values are computed in the context of a dataset (*i.e.*, it indicates intra/inter class disparity) such assumption is incomplete. As a matter of fact, features having the lowest D_{KS} for a given class c are those identifying elements which appear in the images of c very rarely *when compared* with their frequency for the rest of images. For example, in a dataset composed only by the classes *whale* and *clownfish*, the features with the lowest D_{KS} values for the class *whale* would correspond to those having the highest D_{KS} values for the class

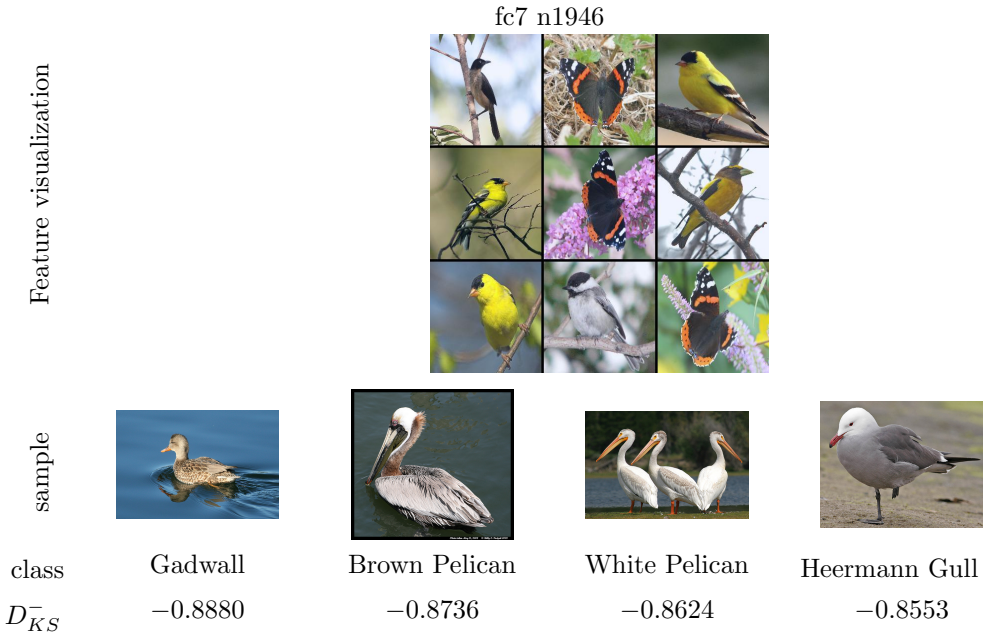

Figure 3: Example of filter having an extremely negative D_{KS} value for four different classes of the *cub200* dataset (the four values shown in forth row are among the top 10 lowest in this dataset). First row illustrates the visual pattern being captured by the feature, using the same method as in Figure 2. The second and third row contain sample images from the *cub200* dataset for the four different classes, and the name of those classes.

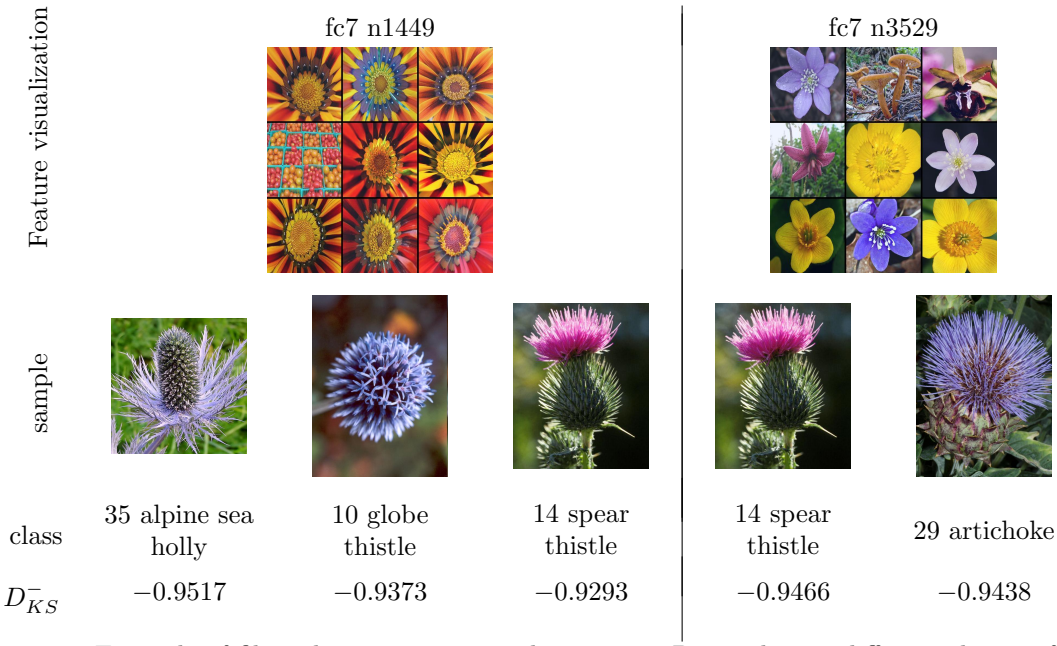

Figure 4: Example of filters having an extremely negative D_{KS} value on different classes of the *flowers102* dataset. First row illustrates the visual pattern being captured by each feature, using the same method as in Figure 2. The second and third row contain sample images from the *flowers102* dataset for the different classes, and the name of those classes.

clownfish, most likely features identifying orange related patterns.

To illustrate the behavior of extremely negative D_{KS} values, Figure 3 shows a feature which has extremely negative D_{KS} values (among the top 10 lowest) for four different classes of the cub200 dataset (which contains 200 classes of birds). This particular feature (the #1946 of the fc7 layer) is trained to recognize flying animals, as shown by the set of images from the *ImageNet* validation set which produce a maximum feature activation (see first row of Figure 3). However this feature produces top negative values for several bird classes. The explanation behind this lies in the particularities of the classes for which the feature produces extremely negative D_{KS} values: the four classes correspond to birds which live in a water or coastal environment, and which have dull colors (see second row of Figure 3). The feature, on the other hand, is specialized on identifying colorful flying animals lying on branches. In this case, the extremely negative D_{KS} values for this neuron identifies *flying animals of dull colors* through the absence of visual features. A consistent behavior happening in the flowers102 dataset is shown in Figure 4. Again, two features which produce top 10 negative D_{KS} values seem to be very representative of the whole dataset (classes of flowers), but not so for a few specific classes. One feature encodes the visual patterns corresponding to round, radial pistils of orange tones (feature #1449 of fc7), while the other focuses on wide petals (feature #3529 of fc7). Clearly, the classes of flowers with highly negative D_{KS} values do not have these properties. Hence, through the abnormal absence of both these features (*e.g.*, the *spear thristle* class shown in Figure 4), we are identifying *flowers without round pistils and wide petals*.

These two examples illustrate how the lack of feature activations can convey relevant information. Notice how the behavior of negative D_{KS} values depend on the context provided by the dataset, extremely negative values on some classes will only happen for features which have a consistently high value on the rest of the dataset. Statistically, the extremely negative values of a feature will only happen for a small set of classes, since, if the set of classes grew, the intra/inter class disparity would decrease, making D_{KS} closer to zero. This capability of extracting knowledge from the lack of data is particular of feature extraction problems where features are not designed for the target task. In this setting, both the presence and absence of visual patterns can provide relevant information for the characterization of images.

Let us also discuss the relevance of this behavior for fine-grained datasets, those containing

classes belonging to a small, rather similar family. Since extremely negative D_{KS} values identify infrequently low feature activations, it is needed for that feature to be frequent on most of the dataset (*e.g.*, *flying animals of bright colors that live on trees* is a frequent feature of birds). This may happen often in fine-grained datasets, where there are many common features in the data. However, in broad datasets which include a wide variety of classes (*e.g.*, *ImageNet*, *mit67*, there are much fewer features which are frequent on most classes and infrequent in a few. Hence, it will be much harder to obtain extremely negative D_{KS} values. This behavior is coherent with the bottom left of Figure 6, where the amount of extremely negative D_{KS} values for a dataset seems to be correlated with its specificity (*flowers102* > *cub200* > *mit67*).

5.2 Statistic Distance Distribution

After exploring the meaning and behavior of positive and negative D_{KS}, we now consider the overall distribution of D_{KS} values per dataset. Figure 5 shows this distribution for all classes, aggregated by layer. Positive and negative D_{KS} values are shown to compose a bimodal distribution, clearly separating positive (D^+_{KS}) from negative (D^-_{KS}). Each of the modes is similar to a log-norm distribution, being the negative part (D^-_{KS}) a mirror of the positive (see Figure 6). To represent this complex distribution for all layers and datasets in a single plot, each distribution is represented by the two modes and two corresponding error bars.

In Figure 5 we see that most features are characteristic of some classes (either positively or negatively), regardless of the layer depth. Notice however how the separation between D^+_{KS} and D^-_{KS} decreases on deeper layers. This speaks of the specificity of layers. To further understand the difference of behavior between low-level layers and high-level layers, Figure 6 shows the same distributions aggregated in two sets: one for features belonging to layers conv1_1 to conv5_3 and another one for features belonging to layers conv5_4 and fc7.

D_{KS} values close to zero imply that a feature provides little information *w.r.t.* a class. As shown in Figure 6, for layers conv1_1 to conv5_3 there are very few D_{KS} values close to zero, making these features either infrequently found or infrequently not found in most classes. This implies that almost every low-level layer feature is descriptive for most classes of all datasets. In contrast many high level layers' features are producing D_{KS} closer to zero, mainly on the D^-_{KS} side where the variance is also smaller. Thus, features from high-level layers are either infrequently found in a class ($D_{KS} > 0$) or irrelevant. This is the result of highly specific features crafted for

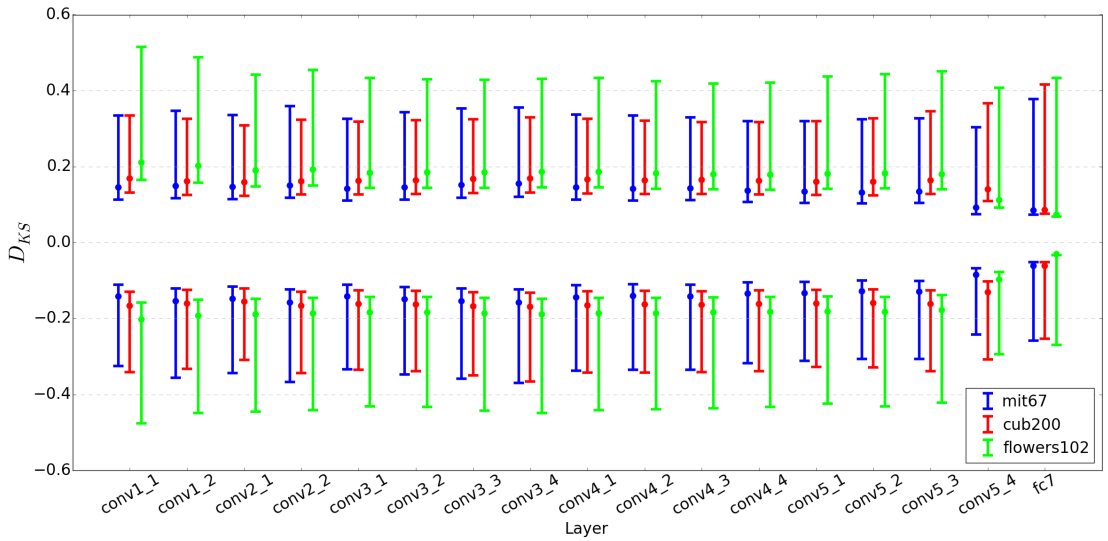

Figure 5: Intra/inter class D_{KS} distribution per layer for 3 different datasets on the embedding. Since the distribution of D^+_{KS} and D^-_{KS} is similar to a log-normal distribution we represent it by the mode at the central point and two error bars enclosing 68% of accumulated probability on each side (equivalent to $\mu \pm \sigma$ for normal distribution).

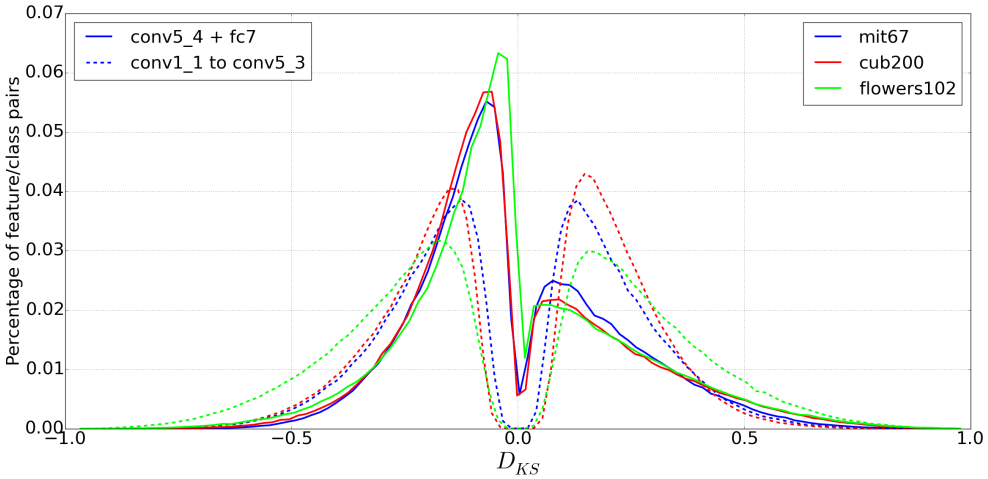

Figure 6: D_{KS} distance distribution for layers conv1_1 to conv5_3 versus conv5_4 and fc7 for 3 different datasets on the embedding. The x axis indicates the D_{KS} values per feature and class, while the y axis indicates the occurrence as a percentage of the corresponding total features/class pairs.

the source task `t0`, that may be of little relevance for specific classes of the target task `t1`.

These results are coherent with the state-of-the-art, indicating that features from high-level layer are more specific and discriminant, thus providing better results when used for to train classifiers for supervised learning [1]. On the other hand, our results indicate that features from low-level layers are more general and descriptive, which opens the door to using them for tasks more dependant on rich knowledge representations (*e.g.*, unsupervised learning).

Another particularity of Figure 5 is the different behavior among datasets at low-level layers (conv1_1 to conv5_3). flowers102 clearly has higher D^+_{KS} and lower D^-_{KS} extreme values than cub200, which has higher and lower values than mit67. Notice that the lower curve of flowers102 in the range between $[-0.25, 0.25]$ is caused by its higher curve on the ranges $[-1, -0.25]$ and $[0.25, 1]$, since the area of each curve is normalized. As discussed in §5.1, the more extreme values of flower102 on D^-_{KS} are caused by the level of fine-granularity of the dataset, which helps at producing highly characteristic features by their absence.

Another property of fine-grained datasets is that, since classes have many common features, simple characteristics (*e.g.*, plain textures) can be of special relevance for their description. The analysis of Figure 5 reveals that the dataset flowers has unusually high D^+_{KS} values for the first and second layers (conv1_1 and conv1_2) compared to the rest of convolutions. This particularity is not present on the other datasets. These low level layers learn filters similar to Gabor filters and color blobs [17] which turn out to be exceptionally descriptive for the flowers102 dataset. As an example, Figure 7 shows some of the features from layers conv1_1 and conv1_2 that produce very high D^+_{KS} values for a specific class of the flowers102 dataset. These particular features correspond to horizontal gradients, vertical gradients and edge detectors, features which appear infrequently often in images of this class when compared to the rest of the flowers in the dataset. These results show the potential of low-level layer features as descriptors of fine-grained datasets and how the proposed analysis can be used to identify meaningful features across the embedding.

5.3 Distance Distribution on *ImageNet*

To contextualize the results of §5.2, we now apply the same methodology to the *ImageNet* dataset. In particular we apply our embedding on 500 classes from *ImageNet*, using images from the validation set (50 images per class). Our embedding originates from a CNN which was trained on the complete *ImageNet* dataset (as discussed in §3), which means the embedding features are optimized for recognizing these 500 classes. By analyzing the distribution of D_{KS} for *ImageNet* we can gain insight on the behavior of the transfer learning process. For that purpose we show

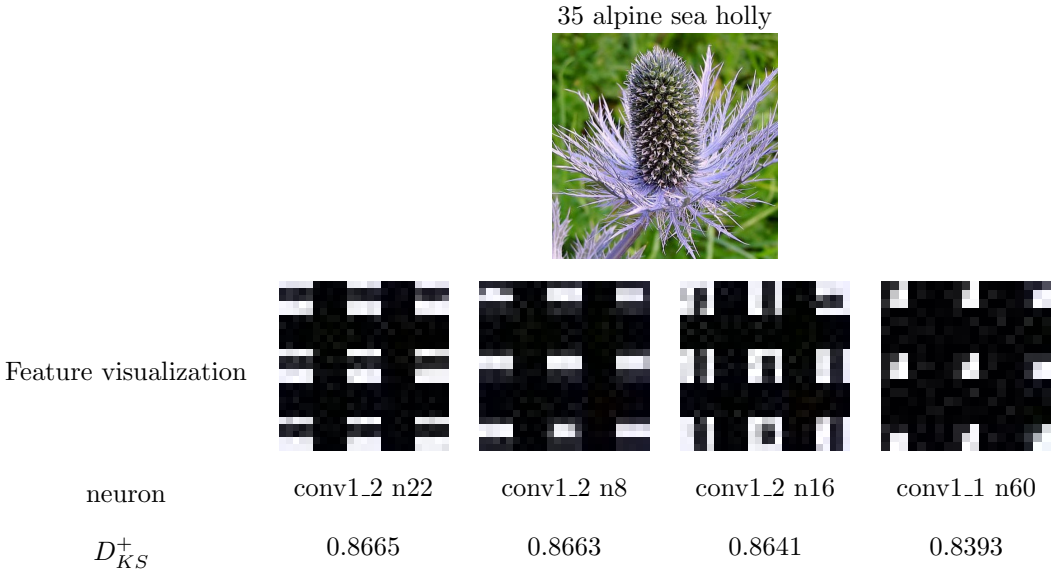

Figure 7: Example of low-level features (from layers conv1_1 and conv1_2) with a high D^+_{KS} value (among the top 10) for a given class of the flowers102 dataset. The first row contains a sample image of the corresponding class. The second row contain feature visualizations from *ImageNet*, obtained with the same process as in Figure 2.

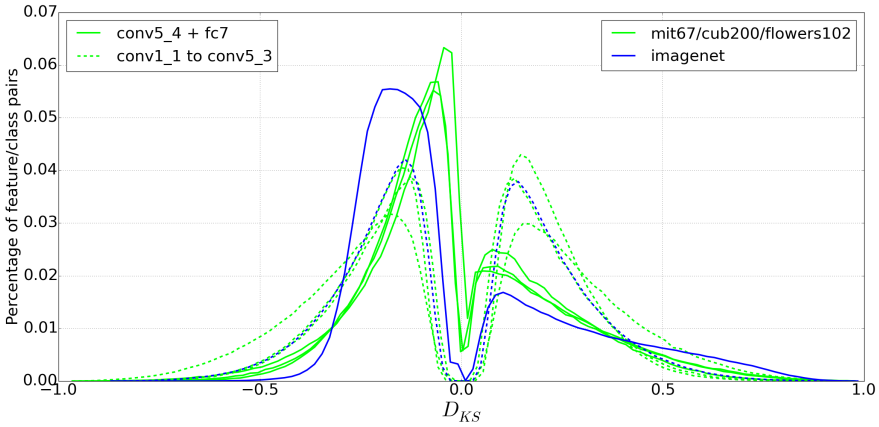

Figure 8: D_{KS} distance distribution for layers conv1_1 to conv5_3 versus conv5_4 and fc7. Distribution for the *ImageNet* dataset is shown in blue and distributions for mit67, cub200 and flowers102 in green. The x axis indicates the D_{KS} values per feature and class, while the y axis indicates the occurrence as a percentage of the corresponding total features/class pairs.

Figure 8, which is analogous to Figure 6 with the added curve of the *ImageNet* dataset. Since the differences between the other datasets (*i.e.*, mit67, cub200, flowers102) were already explored in Figure 6, now we only plot the *ImageNet* embedding in a different color.

The first particularity of Figure 8 we discuss is how, rather surprisingly, low-level layers features (from conv1_1 to conv5_3) behave very similarly for all four datasets. In fact the curve for *ImageNet* on this set of layers is almost identical to the one for mit67. Even though these features were also trained to optimize the classification of *ImageNet* classes, it seems that they are as descriptive for *ImageNet* as they are for the other datasets. This provides further evidence on why transfer learning for fine tuning produces such good results [17], but also indicates that features from these layers could be used almost ubiquitously for knowledge representation. The only dataset behaving differently at the extremes values of low-level layers features is the flowers102 dataset. As discussed in §5.2 this is explained by its particularities.

For the high-level layers (conv5_4 and fc7) Figure 8 shows a very different behavior on the negative side of the curve for the *ImageNet* dataset. This is due to two different factors. First, high-level features are highly optimized for the *ImageNet* classes, which implies that there will be no irrelevant features in its embedding. As said before, irrelevancy in this context is defined by $D_{KS} \simeq 0$, which moves the *ImageNet* spike on the negative side away from the 0 on the x axis. This is further supported by the fact that, while for three alternative datasets there are features with $D_{KS} = 0$ (the curve does not reach 0 on the y axis at that 0.0 D_{KS} distance), that does not happen for the *ImageNet* dataset.

The second factor modifying the negative side of the *ImageNet* curve explains why there are no values on the range $[-1, -0.5]$, thus causing a more steeper slope for low negative values (in the range $[-0,5, -0.2]$). As discussed in §5.1, extremely negative values of D^{-}_{KS} correspond to features largely common for most of the dataset, which are lacking in a particular class (*e.g.*, the *flying animals of dull colors which do not live on trees* case). This situation, which is already rare among high-level features due to their higher specificity, never happens in the *ImageNet* dataset because features so close to the CNN output are penalized for being common during training.

Finally, let us discuss the specific behavior of the *ImageNet* curve of Figure 8 for the positive values on high-level layers (conv5_4 and fc7). Clearly, the *ImageNet* dataset achieves higher D^{+}_{KS} values for most of the $[0.5, 1]$ range of the x axis. Since many of those high-level features are optimized to discriminate *ImageNet* classes during the CNN training process, the number of features which can be found infrequently often in a given class will be much higher. This particularly is therefore an effect of the specialization of high-level features for the *ImageNet* data. In this context, we argue that D^{+}_{KS} values within the $[0.5, 1]$ range of high-level layers provide enough evidence as to describe the similarity between a source task `t0` and a target task `t1`.

5.4 Positive Distances per Class

In §5.3, we discussed the distribution of D_{KS} values on the various datasets. However, the distribution does not show the behavior of D_{KS} values at a class level. Thus, it could happen that, even though the distribution shows that most features are relevant for most classes, there may be a few classes which are under-represented, such that there are no feature to properly characterize them, while others are over-represented. This could happen if the embedding does not contain features capable of describing a given class within the context of its dataset. To answer this, and similar questions in Figure 9 we plot an accumulated distribution of D^{+}_{KS} values per class. Each of the black lines represents a single class in the dataset. This graph is accumulative, showing how many features are greater than a D^{+}_{KS} value. So, at $D_{KS} = 0.2$ we are evaluating the number of features that meet $D_{KS} > 0.2$ for each class.

Figure 9 shows a significant variance among classes of the same dataset for any given D_{KS} value. This implies that some classes are more richly characterized by the embedding than others. Although no class reaches 0 on the y axis until around $D_{KS} = 0.4$, some reach 0 at $D_{KS} = 0.9$. To know if reaching 0 features at $D_{KS} = 0.4$ implies that the class is characterized by the embedding to a minimum degree, in the same figure we show the behavior of the same dataset with randomized labels. By randomizing the labels we observe the behavior of a purely noisy dataset. As shown for all three datasets in Figure 9, most randomized classes drop to 0 features between $D_{KS} = 0.1$ and $D_{KS} = 0.3$. On one hand this allows us to assert that all classes are represented by meaningful features to a minimum degree. On the other, it triggers the question of which portion of this curve could or should be pruned to maximize representativeness while minimizing noise. This is equivalent to ask which is the minimum D_{KS} value we consider to be relevant when choosing the features to characterize a class.

5.5 Distance Threshold

As said before, one of the main goals of this paper is to study the viability of using convolutional features for knowledge representation, when using a feature extraction process. Many convolutional features however, may correspond to noise, which would complicate any effort in that direction. To tackle this issue, we first considered the behavior of noisy data in §5.4, by analyzing

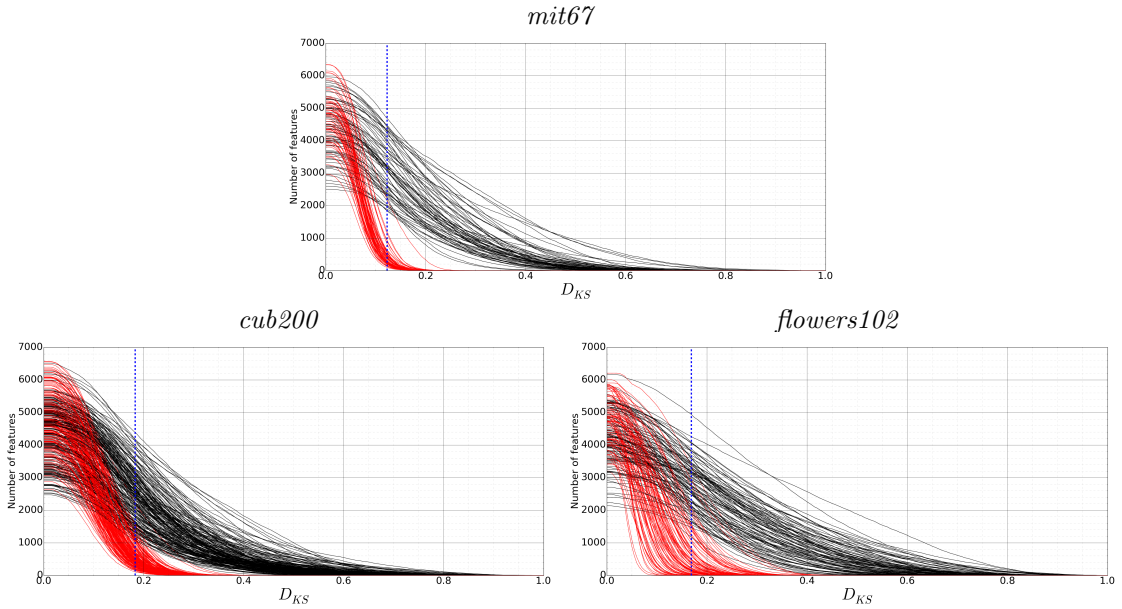

Figure 9: D_{KS} positive distance (D^+_{KS}) accumulated distribution for 3 different datasets (black) on the embedding. Each line corresponds to a different class of the dataset. Results for the same dataset with randomized labels are shown in red. Dashed line marks the t^+ threshold introduced in §5.5.

a randomized dataset. Indeed, the fact that a feature may appear slightly more frequently in a class than in a dataset (thus having a D_{KS} a little over 0) may be the result of a small data sample, of random variance or of class noise (*e.g.*, if by accident, in the cub200 dataset images a child appears in the background of a few seagull class images and nowhere else). Similarly, D_{KS} values a little *under* 0 can also correspond to this type of noise.

To continue with our analytical approach to CNN features behavior, and provide tools for building faithful representations, in this section we consider the definition of thresholds t^+ and t^- on D_{KS}, such that every $D^+_{KS} < t^+$ or $D^-_{KS} > t^-$ could be discarded for knowledge representation purposes. These thresholds should allow us to determine which features are likely to be relevant for each class, canceling out a significant amount of noise. In general, thresholds t^+ and t^- should tend towards zero to be as descriptive as possible, while also tend towards 1 and -1 respectively to reduce the amount of noise. It is also of interest to see how these thresholds behave on the various datasets and layer depths.

For t^+ (an analogous reasoning applies to t^-), our threshold analysis is based on the behavior of the randomized labels datasets shown in Figure 9, using it as an indicator of D^+_{KS} values which are likely to correspond to noise. Our threshold will be one which maximizes a measure of distance between the various datasets and their corresponding version with randomized labels (as shown in each subfigure of Figure 9). The t^+ threshold we propose is based on computing the average number of features having a $D_{KS} > x$ for all x in the range $[0, 1]$. This is analogous to compute, for every point along the x axis of one of the subfigures of Figure 9, the average y axis values for all black/red lines. The average of black lines will give us the behavior on the regular dataset, while the average of red lines will give us the behavior on its randomized version. By obtaining the difference between both values we obtain the *average distance* (d_{avg}) between a dataset and its randomized version. Formally, the average distance for a value $D_{KS} = x$ is:

$$d_{avg}(x) = \frac{\sum_{c \in c_{data}} |D_{KS}(f, c) > x|}{|C_{data}|} - \frac{\sum_{c' \in c_{rand}} |D_{KS}(f, c') > x|}{|C_{rand}|} \quad \forall f \in embedding \tag{4}$$

We compute both the t^+ and t^- thresholds for all three datasets as the maximum d_{avg} value

Table 4: t^+ and t^- thresholds as defined by the maximum average distance for each of the three datasets explored. D^+_{KS} and D^-_{KS} regions are computed separately. The third and fifth column shows the maximum d_{avg} distance between the dataset and its randomized version.

Dataset	t^+	$d_{avg}(t^+)$	t^-	$d_{avg}(t^-)\|$
mit67	0.123	2,742	-0.114	3,027
cub200	0.183	1,833	-0.165	1,864
flowers102	0.169	2,264	-0.147	2,515

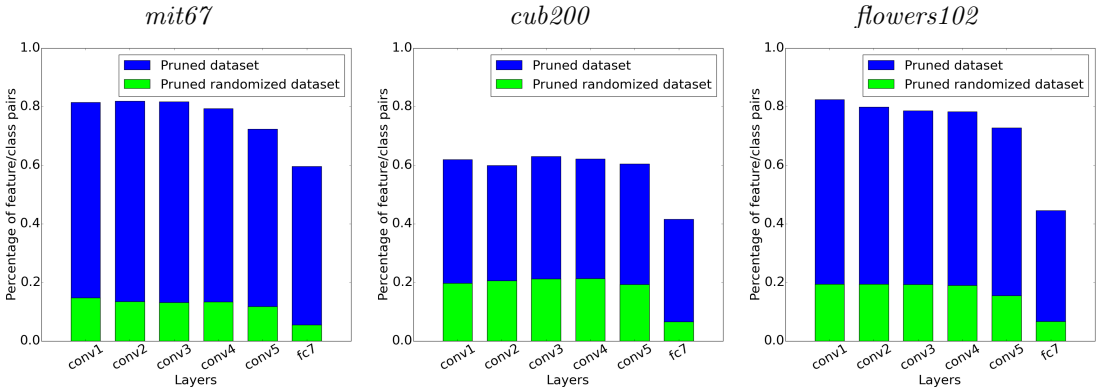

Figure 10: Percentage of feature/class pairs for layers on pruned datasets (blue) and pruned randomized datasets (green).

for the whole curve (see Table 4). Significantly, the thresholds are stable across datasets, being in range $[0.11, 0.19]$ in all cases. This implies that any D_{KS} value between 0 and 0.1 (or between -0.1 and 0 for the negative case) is likely to correspond to noise, could be treated as irrelevant, and be pruned with minimum reliability. The location of the t^+ threshold on each dataset is plotted as a dashed line in Figure 9, showing that all classes would be represented to a certain degree (having some features to describe them) after applying it.

To study the degree of noise layer-wise, we plot the percentage of $D_{KS}(f, c)$ values that are discarded by the t^+ and t^- thresholds on various sets of layers in Figure 10. While convolutional layers keep between 80% to 60% of their $D_{KS}(f, c)$ pairs (only 20% to 40% were categorized as noise), for the later layers that percentage drops significantly (specially for fc7). This is caused by the higher specificity of high level features, which are more frequently irrelevant for characterizing many other classes. Further evidence in that direction is the number of features that are discarded for all classes in a dataset after applying the threshold (*i.e.*, features with no $D_{KS}(f, c)$ values over the threshold throughout a dataset). If we considering both t^+ and t^- thresholds (thus accepting very positive or very negative D_{KS} values), for the mit67 dataset there are 38 irrelevant features (37 from fc7), for the cub200 there are 62 (all fc7) and for the flowers102 there are 108 (all fc7). Exactly the same results are obtained if we consider only the t^+ threshold (*i.e.*, if we are only accepting very positive D_{KS} values). Considering only the t^- threshold significantly increases the number of useless features (to the order of thousands), since there are very few features which are informative by their absence (see §5.1), particularly on high-level layers. As it was to be expected, the representation language built by a CNN is richer on the side of positive characterization (features which are descriptive by their presence) than on the side of negative characterization (features which are descriptive by their absence). Finally, let us remark that no low level feature was found to be useless using this methodology, which provides further evidence regarding their potential for knowledge representation purposes.

6 Conclusions

CNN feature extraction has been studied in the past through the performance of a classifier (most commonly, a SVM), measuring how each layer performs separately at discriminating classes.

Through these contributions we know that, when considered together, the features composing a fully connected layer provide the most discriminant of embedding spaces. In contrast with these contributions, the purpose of this paper was to analyze the behavior of each feature individually, to measure their applicability for knowledge representation. We do so by exploring the intra/inter class activations of each feature separately, for all classes of three different datasets. Some of the conclusions we draw from this study are coherent with the current state-of-the-art, some are new. Next we outline them all:

- Some convolutional features can be used to describe classes by their absence, thus providing a different type of information for knowledge representation purposes. This is particularly relevant for fine-grained datasets, where there may be more broadly common features, and could be exploited by knowledge representation or reasoning methods (§5.1).
- Features from the last convolutional layer and fully connected layers are highly specific, being either characteristic of a class or irrelevant for it. Features from the rest of convolutional layers convey more variate information, and can be characteristic of a class both by their presence or by their absence. This motivates the use of two distinct knowledge extraction approaches, depending on layer depth (§5.2).
- Low and middle level features have a very similar behavior for *ImageNet* (the dataset they were trained for) as for the rest of target datasets. This indicates that CNN features from these layers could be used for knowledge representation on a wide variety of datasets without fine-tuning. Our analysis also indicates that large, positive distances in high-level layers could provide enough evidence as to be used to measure similarity between tasks (§5.3).
- For all classes of the three evaluated datasets, characteristic features were found on all layers of the embedding. This means that, in a knowledge representation setting, no class would become indescribable, showcasing the richness of the representation language built by the CNN (§5.4).
- Through the behavior of randomized datasets we obtain an estimation of the intra/inter class distances that can be accounted for by noise. We find that a threshold can be defined across datasets, with little variance (§5.5).
- Context is key, both in the feature extraction and knowledge representation processes. The significance of some feature activations (or its lack of) depends on the dataset being used as reference. The level of specificity or of fine-grain of a dataset is also context dependant, defined as relative to a source task.
- Our proposed feature behavior analysis can be used to tune the feature extraction process for a given dataset. Through this study we can learn which are the most relevant features for the various layers (both by presence and absence), which layers provide an exceptional amount of information, how much fine-grained the target dataset is, and how to reduce the amount of noise when representing knowledge.

Acknowledgements

This work is partially supported by the Joint Study Agreement no. W156463 under the IBM/BSC Deep Learning Center agreement, by the Spanish Government through Programa Severo Ochoa (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project and by the Generalitat de Catalunya (contracts 2014-SGR-1051), and by the Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST).

References

[1] Hossein Azizpour et al. "Factors of transferability for a generic convnet representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9 (2016), pp. 1790–1802.

[2] Ken Chatfield et al. "Return of the devil in the details: Delving deep into convolutional nets". In: *arXiv preprint arXiv:1405.3531* (2014).

[3] Jeff Donahue et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition." In: *Icml*. Vol. 32. 2014, pp. 647–655.

[4] D. Garcia-Gasulla et al. "A visual embedding for the unsupervised extraction of abstract semantics". In: *Cognitive Systems Research* 42 (2017), pp. 73–81.

[5] Liangke Gui and Louis-Philippe Morency. "Learning and transferring deep ConvNet representations with group-sparse factorization". In: *Proc. IEEE International Conference on Computer Vision*. 2015.

[6] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[8] Ammar Mahmood et al. "ResFeats: Residual Network Based Features for Image Classification". In: *arXiv preprint arXiv:1611.06656* (2016).

[9] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes". In: *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*. IEEE. 2008, pp. 722–729.

[10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. "Improving the fisher kernel for large-scale image classification". In: *European conference on computer vision*. Springer. 2010, pp. 143–156.

[11] Ariadna Quattoni and Antonio Torralba. "Recognizing indoor scenes". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 413–420.

[12] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[13] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[14] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 806–813.

[15] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[16] C. Wah et al. *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.

[17] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[18] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579* (2015).