Do deep features retrieve X?: A tool for quick inspection of deep visual similarities

Julieta Martinez
University of British Columbia

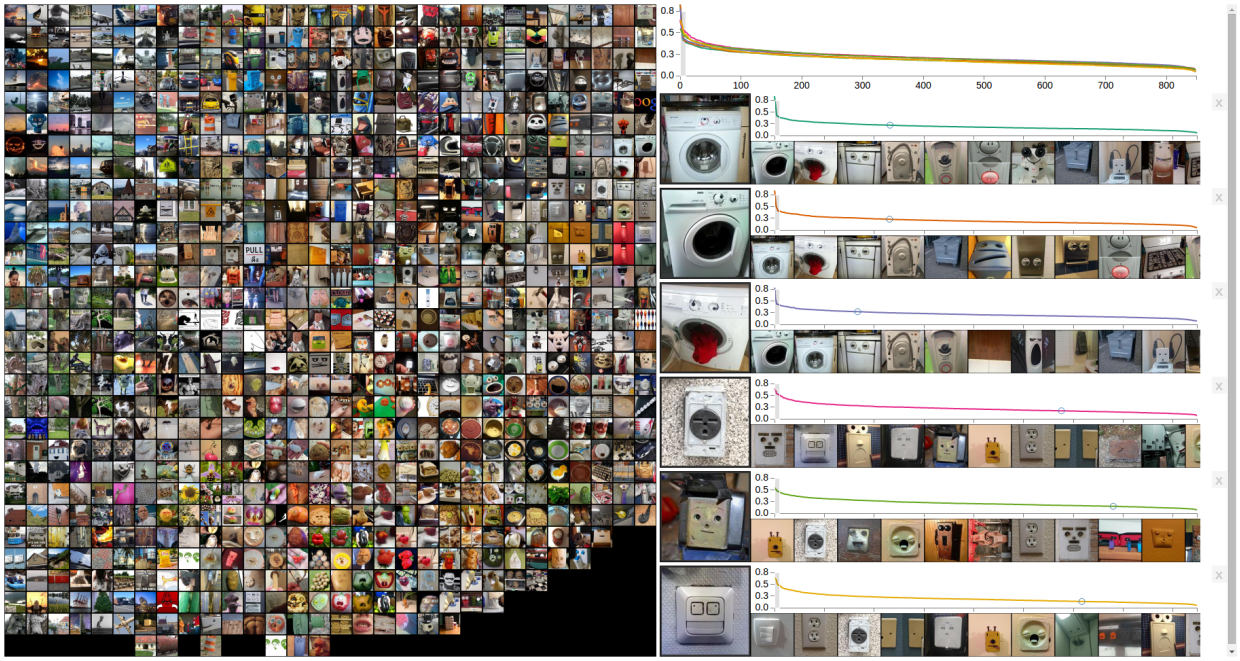

Fig. 1. A screenshot of our system, deepviz, being used to inspect the similarity of deep features on a dataset of "faces in things".

Abstract— Deep learning has become, over the last few years, the technique that achieves the best results in a series of challenging machine learning tasks such as speech and object recognition. However, arguably no other field has seen greater progress derived from deep learning as computer vision, where deep convolutional neural networks have quickly become the standard for computational vision recognition. While computer vision is often associated with computational object recognition; other problems in computer vision have extensively benefited from the use of features extracted from deep networks. One such common problem in visual recognition is visual retrieval, often also called "image-based" image search. The state of the art in this problem is heavily driven by benchmarks of object instance retrieval, and is defined by the use of features extracted from deep neural networks. In this work, we argue that a sense of visual similarity is often loosely defined, and in practice is not limited to exact object instance matching. The goal of this project is to create a tool that allows researchers to quickly verify whether a technique for image retrieval works well for other tasks. We introduce *deepviz*, a tool that given an image datset allows researchers to quickly answer the question: do deep features work for retrieving X?.

Index Terms—Computer vision, deep learning, convolutional neural networks, image retrieval

1 Introduction

Computer vision is a subfield of artificial intelligence focused on developing algorithms that can infer concepts from images in a way that mimics the human understanding of the visual world. Common tasks in computer vision include object recognition, image classification, object localization and image parsing (labelling every pixel in an image). Developing algorithms that can perform such tasks at or above human performance is a problem that has eluded researchers for over half a century, and is considered by many a fundamental challenge towards a full understanding of intelligence. Having fully-functional computer vision systems promises to bring, for example, better autonomous driving systems (potentially saving millions of lives), improved medical diagnoses, more robust security systems and easier ways to develop more engaging entertainment.

While research in the early years of computer vision was heavily focused on developing computational models that reflected the physical behaviour of light in the real world [26], during the last decade research in the field has shifted towards data-drive methods. With the advent of the Internet and ubiquitous, cheap consumer-grade cameras, it become easier to compile datasets with millions of images, such as Imagenet [13] and Microsoft's COCO [18]. Together with deep convolutional networks trained on consumer-grade GPUs [17], object classification has dramatically improved so much over the past years that it is now virtually considered a solved problem; for instance, the Imagenet competition for 2015 is not considering an image classification task anymore.

E-mail: julm@cs.ubc.ca

Notably, the progress brought about by convolutional networks did not stop at the task of image classification. Crucially, in May of 2014 Razavian et al. [20] conducted 11 experiments on datasets focused on diverse visual tasks such as scene classification, fine-grained categorization, instance retrieval and attribute prediction, and demonstrated that using features extracted from deep convolutional networks achieved performance either competitive with, or better than the state of the art at the time. This astounding result showed that there was a relatively straightforward way to taking advantage of deep networks to improve *all* visual recognition tasks.

The correct way to test whether deep features can improve a certain task is to do what Razavian et al. did: (1) get the data, (2) obtain labels, (3) put together a recognition pipeline, and (4) measure qualitative results such as classification accuracy and precision-recall curves or mean average precision – this has proven to be a solid way to quantify progress in the field. Following all these steps, however, requires a lot of work from the researcher. Often, it is very easy to get the images and the features, and it is not hard for researchers to come up with ideas of diverse visual tasks. For example, we have found ourselves often wondering: *would it make sense to use deep features to predict chess positions from images of boards?* or *how well would deep features perform on the task of finding faces in things?* In both cases, getting the data and the features can be done in a matter of days, but annotating the data and putting together a full-fledged experimental setup would take anywhere from a few weeks to a few months. Is there a way in which we can help researchers get a feeling on what deep features can do while avoiding them all the pain of putting together a complete classification pipeline? We believe so, and this is precisely the goal of this project.

We introduce *deepviz*, an online visualization tool that allows researchers to quickly assess the similarities captured by deep features on a particular dataset. Given only the data and the features, our system combines a low-dimensional, space-filling embedding of the images in 2d that preserves neighbourhood similarities, and allows the user to query for total orders of the dataset with respect to particular images. The queries can then be compared to each other and examined in further detail *at different points of the image ranking* – not just for the top matches, as it has traditionally been done. Please refer to Figure 1 for a screenshot of our system.

Our tool can also be used to improve the way retrieval results for new methods are shown to the public. Typically, computer vision papers will present qualitative results by showing a query image and the top-retrieved neighbours in the dataset, often comparing 2 or more methods on the same query (see Fig. 2). In contrast, our tool can be used to explore different parts of the similarity distribution with respect to the entire dataset, and to compare multiple query image results and distributions side by side.

Fig. 2. Image reproduced from *Aggregating deep convolutional features for image retrieval* by Babenko and Lempitsky [7], which illustrates a typical qualitative comparison of image retrieval in computer vision papers: queries are shown at the left, and two retrieval methods are compared by showing the top retrieved neighbours from the image dataset. Our tool aims to provide a visualization to compare not only the top retrieved neighbours, but any part of the retrieval curve.

2 Previous Work

The goal of this work is to provide a tool to quickly inspect the output of an image retrieval system; however, a secondary goals is to allow users to explore the whole dataset such that it facilitates the exploration of image arrangements relative to a query. To compare visual similarities, we must allow the users to explore the distributions resulting from using several images as queries and thus, through several examples, get an idea of which visual concepts are retrieved. For the first part, we review work that focuses on image tiling for large visual datasets, and for the second part we focus on work that addresses focus+context exploration in large image collections. Finally, we make use of a dimensionality-reduction technique as a shorthand to explore the neighbourhood of each image – we quickly review t-SNE, our technique of choice as well.

2.1 Image tiling

There is a large amount of previous work focused displaying multiple images with different types of arrangement and tiling. Google images[1] shows a set of relevance-sorted images in vertically-stacked horizontal layouts, controlling for height and preserving the image aspect ratio. While it is not clear whether this visualization is optimal, it is probably the one that people are most familiar with, and has been subsequently been adopted by competing image search engines such as Bing and Flickr. In 2009, Google introduced Image Swirl [1], which clustered similar images in stacks along the z-dimension, and allowed users to expand clusters for closer inspection. However, the service was only available as beta, and seems to have been discontinued.

Recently, Brivo et al. [10] proposed using a Voronoi-based partition of the space to show multiple images in space-filling manner. After a force-directed layout is computed from the query to the rest of the dataset, a Voronoi diagram is used to resize and crop the images relative to their distance to the focus image. The authors further propose a method to smoothly change the diagram when new images are added or removed from the collection. The advantage of this method is that it is easy to implement, and achieves space-filling in an unconventional way. The disadvantage is that the cropping of Voronoi cells may confuse some users, who are mostly used to deal with rectangular crops. While the method is novel, the authors did not make a user study comparing to a standard rectangular arrangement and cropping.

Wang et al. [25] propose a tiling of images around a central query that uses size to encode importance. The query image is shown in the center with maximum size, and the most relevant results are shown around the image in a spiral layout, halving the size with each full wrap around the query. The advantage of this method is that it conveys the idea that retrieval importance is non-linear and rapidly decreases as one explores images further down in the ranking. The downside is that there is not an easy way to allow users to focus on images far from the query, and that the spiral layouts makes it hard to compare the relative

[1] https://images.google.com/

ranking of any two retrieved images with respect to the query.

2.2 Focus+context in image collections

The second task that we want to support is allowing users to explore the image collection to use new images as queries, with the output of a previous query as context. Previous work on focus+context in images is rather scarce. The only work that we are aware of is that of Chen et al. [12] contrast and study sliding (where neighbours are simply pushed away) and expanding (where neighbours are re-arranged according to a Voronoi diagram to better use the space) approaches to focus+context exploration. They found that users prefer the expanding (space-filling) approach in terms of "ease to use", "efficiency" and "fun". This suggests that we should prefer visualizations that densely populate the space in the focus+context exploration part of our tool.

A somewhat related work (although definitely closest in terms of implementation) in this vein is the NYT Fashion Week front row visualization by Michael Bostock [2]. The online system lets users explore several front shots of models wearing designer clothes. The images are aligned horizontally, and a 1-dimensional fisheye zoom focused on the user's mouse is used to expand the images for closer focus+context inspection. This is a simpler problem compared to ours, as (a) there is already a precompiled visual structure and similarity in the pictures – notice that a central slit is shown in the cropped images, and this slit manages to give an idea of the attire –, and (b) fisheye zooming is only done in one dimension, as horizontal scrolling is used to show the work of different designers. In contrast, our work has to deal with changing (computed on-the-fly) visual similarities, and also an arrangement in 2 dimensions.

2.3 t-SNE

t-SNE [24] is a common technique for dimensionality reduction that, as opposed to visualization-agnostic techniques such as principal component analysis, is particularly designed to yield good visualizations in low dimensions of high-dimensional data points. The algorithm assigns t-distributed probabilities for similarities in the high and low dimensional spaces, and uses stochastic gradient descent to minimize the KL-divergence between the probabilities in both spaces. Since t-SNE heavily penalizes large distances in low dimensions for data points that are close in the high-dimensional space, the results often greatly preserve local similarities. We use this embedding as a proxy to neighbourhood exploration in our tool.

3 Data and tasks

In information visualization, the data is *what* we want to visualize, and the task is the *why* we want to do the visualization [19]. Moreover, the *domain* analysis is made in terms that require specific knwoledge about the context of the task and the data, while the *abstraction* analysis is made in made in terms of generic visualization objectives and idioms, without requiring domain knowledge about the task or the data. We explicitly separete both analyses below.

3.1 Domain data

The main input data for deepviz is an image collection of up to $\sim 1\,000$ images. In our case, we obtained our dataset downloading the images from a the twitter account @FacesPics[2]; under the tagline *admit it, you see a face*, the account broadcasts images of things where people tends to see a face. As of December of 2015, the account had 505 000 followers, and had tweeted 850 pictures. We downloaded the images using Tweepy [3], a python library that allows for easily dowloading all the tweets of an account; we later parsed the resulting csv for urls ending in .jpg and .png. In spite its name and tagline, the account often tweets images that do not necessarily have a face, but will often tweet images that are otherwise suggestive of other human, and sometimes animal, body parts; we estimate that these images make up around 10% of the dataset. Figure 3 shows a sample of the images obtained from the FacesPics twitter account.

[2] https://twitter.com/facespics

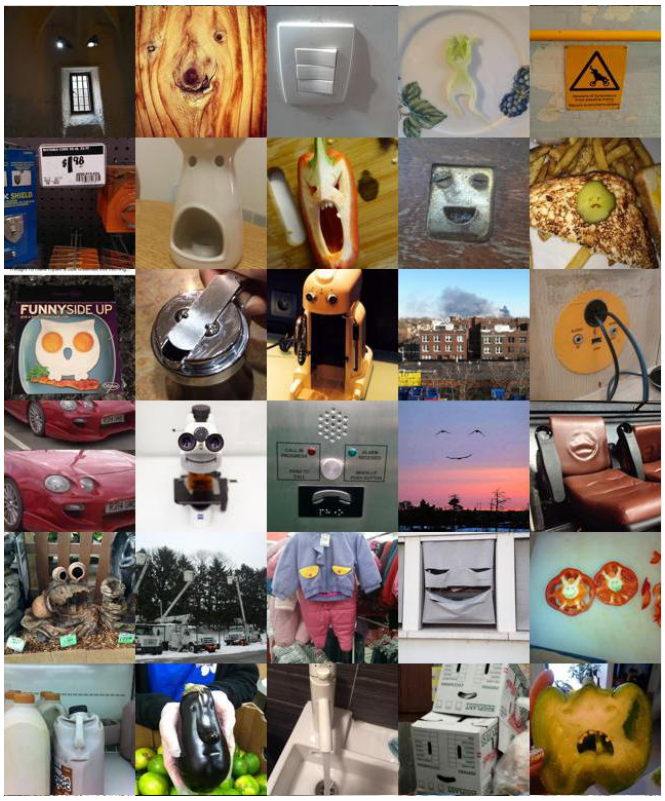

Fig. 3. Typical images from our dataset, as seen after cropping a central square such that the smallest dimension is fully preserved. While in most images it is fairly easy to see a face, it is not uncommon to find other images that depict, for example, animals. For example, the upper right image is a sign where one would see an elephant, and the image to the left seems to have food that looks like a frog. There are also some images with captions, such as the fried eggs that look like an owl at the middle left. Finally, the second image in the next-to-last row depicts a building, but it is not immediately clear what makes the image particularly unusual or interesting. These images are good examples of the diversity found in our dataset. A complete tile of the images that we downloaded can be seen on http://jltmtz.github.io/deepviz/imgs/facespics_128/bigtile.jpg

Preprocessing. In order to extract deep features, the images had to be resized to a resolution of 224-by-224 pixels. The resizing was done such that the smallest dimension was fully preserved. This is also the representation that we used in our final visualization, as it accurately reflects what the network "sees", and therefore avoids misleading the users into thinking that the entire image was taken into account.

Feature extraction. After resizing, features were obtained by passing the images through the 16-layer "very deep" network provided by Simonyan et al. [21], commonly referred to as VGG-net. The network consists of 16 convolutional layers with filters of size 3-by-3, and 5 rectified linear units, as well as two fully-connected layers that produce a $d = 4096$-dimensional feature. The network obtained the second place on the 2014 Imagenet large-scale image classification competition; while the top performance in the competition was achieved by Goog-le-net [22], a network by the Google team, it has been repeatedly shown VGG-net obtains better performance on transfer-learning tasks. Moreover, since VGG-net became publicly available shortly after the competition results were announce in the summer of 2014, the network has become the de-facto standard to extract deep features for vision tasks. The feature extraction process was used using an intel i7 CPU and took slightly above 2 hours for the 850 images in our dataset – for a larger-scale dataset, it would be necessary to run this part of the pipeline in a GPU.

We want to note that, even though we used the output of the last layer in our experiment, it is also common to use the output of previous layers for tasks other than image classification; in particular, it has been noticed that the output of the last convolutional layer obtains state-of-the-art performance on object instance retrieval [8, 7]

Distance computation. During training, the images are classified in 1 of 1 000 categories using a multi-class soft-max logistic regression. While logistic regression has the advantage of easily providing a gradient for back-propagation, it is not immediately clear how the features can be used for a retrieval task. Experimentally, it has been shown that given 2 features $\mathbf{x}_1$ and $\mathbf{x}_2$, the Euclidean distance

$$d = \|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sum_{i=1}^{d} \left(\mathbf{x}_1^i - \mathbf{x}_2^i\right)^2 \quad (1)$$

yields good results for retrieval. Unfortunately, the Euclidean distance is not upper-bounded (i.e., it can have any value between zero and infinity), which can make it hard to compare different similarity distributions. Chatfield et al. [11] have shown, however, that a minimal loss in performance can be achieved by pre-normalizing the features: $\hat{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|_2$ and using the dot-product similarity

$$d = \langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle = \sum_{i=1}^{d} \hat{\mathbf{x}}_1^i \times \hat{\mathbf{x}}_2^i, \quad (2)$$

which is then equivalent to the cosine similarity used in linear support vector machines. We opt for the dot-product similarity because it also has the advantage of providing distances in the range $[-1, 1]$, which we consider to be easier for users to compare, as well as easier to plot. Moreover, as opposed to Euclidean distance, the dot-product similarity can benefit from approximate distance computation techniques based on multi-codebook quantization that use non-orthogonal codebooks and define the state of the art in very large-scale image retrieval [6, 27] – while our dataset was small enough to not require such techniques, some type of compression will be indispensable in a larger-scale version of this project.

3.2 Domain task

The task that we want to support is to help researchers understand the similarities capured by deep features. The goal is to show researchers an ordering of the similarities computed to all the images in the dataset. Crucially, in our visualization task our data consists of features, which define a raking computed by the machine, and images, form which humans can judge similarity at a glance. Therefore, we want to show our users both statistics about retrieval *and the retrieved images themselves*. To facilitate the comparison of machine-computed vs. human-computed image similarities is the main goal of deepviz.

3.3 Abstract data

Our data can be abstracted at two different levels: on one hand, our images can be seen as generic tensors which happen to have a trivial way to visualize. This "abstraction", is not very useful by itself, but it does suggest that displaying the images should not be a too-difficult task. On the other hand, we can focus on the features, which provide a mapping from the image space to a high-dimensional Hilbert space $\mathscr{H}$. In such space, similarities are trivial, though potentially computationally expensive, to compute via dot-product or Euclidean distance, which provides with yet more derived data – for each image, one may compute a total order of the rest of the images in the dataset, resulting in a (n-1) real numbers that may be treated as a generic 1-dimensional distribution.

3.4 Abstract task

We identified two main tasks exploration of the image dataset, and comparison of queries. The exploration of the entire datasets allows the user to familiarize themselves with the images in the dataset; plus, if the exploration is done in a neighbourhood-preserving embedding, the user can also get a sense of the highest matches for each image in the dataset. The comparison has actually 3 dimensions to it:

1. **Computed vs. perceived similarity.** This means the user should be able to compare the number that the retrieval system gives for the similarity between a pair and images vs. how similar the user perceives the images to be.
2. **Computed similarity for many queries.** The user should be able to compare two similarity distribution using different images as queries.
3. **Perceived similarity for many queries.** The user should be able to compare two perceived similarity from a query to the rest of the dataset.

A summary of this what-why-how analysis is presented in Table 1.

	Do deep features retrieve X?
What: Data	A dataset D of images $D = \{x_1, x_2, \ldots, x_n\}$ and a set of image queries $Q = \{q_1, q_2, \ldots, q_m\}$. Potentially, $Q \subseteq D$.
What: Derived	n d-dimensional features representing the database, and m d-dimensional features for the queries. A similarity measure for the derived features $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. Typically, for deep features $S(f_1, f_2) = \|f_1 - f_2\|_2^2$ or $S(f_1, f_2) = \langle \hat{f}_1, \hat{f}_2 \rangle$, where $\hat{f} = f/\|f\|_2$.
Why: Explore	The user must be able to see all the images in the dataset.
Why: Compare	The user can see the results of different queries, and is able to compare them. There are 3 comparisons to be done: (1) the computed similarity vs. the perceived similarity, (2) computed similarities for different images, and (3) perceived similarities for different data points.
How: dim. reduction	The data is preprocessed to lie in a negihbourhood-preseving 2-dimensional space. We use t-SNE on the derived features.
How: Encode	Each datapoint is displayed as a minified version of its central crop. It has been shown that a size of 32×32 is sufficient to recognize objects in images [23], so we use this encoding size by default for exploration. We also use a size of 128×128 to show the queries, and 64×64 to show the retrieved images.
How: Facet	A direct display of the low-dimensional embedding might occlude the images, so a gridified version of the embedding is computed as well. Tiling is trivial because all the images are squares of the same size.
How: Small multiples	Each query results in a distribution of similarities across the entire dataset. We show multiple queries side by side in a pane next to the embedding. Synchronized navigation and highlight across the images and the distributions are also provided.
Scale	$\sim$ 1 000 images, assuming a display of 1 920 $\times$ 1 080 pixels.)

Table 1. What-why-how analysis of our system, deepviz.

4 Solution

We had a 2-step process to reach our final solution design. While our first iteration was deemed a failure, we think it is worthwile to share and analyze the rational behind and the failure reasons for this first iteration. We then show the eventual design of deepviz, and analyze the strenghts that it has with respect to our first iteration.

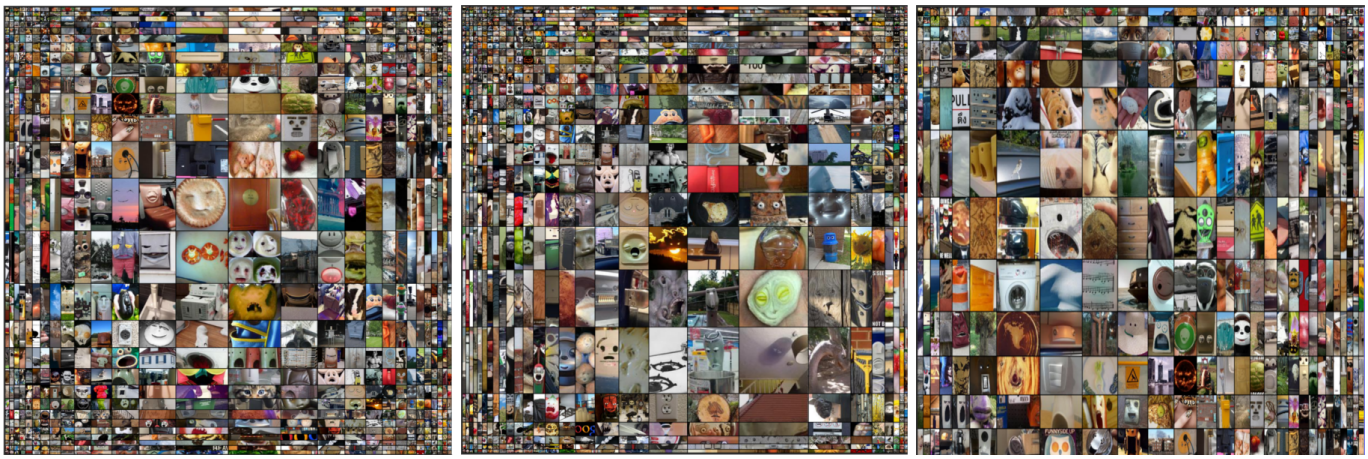

Fig. 4. A first iteration to visualize image retrieval. In the three images, the query is at the center, and the most similar images wrap around it in a spiral. The left-most image shows a moderate fisheye distortion focused at the centre, the middle image shows a more pronounced distortion and the last image shows a more modest distortion, both focusing on different parts of the retrieval set. It can be appreciated that it is very hard to distinguish the objects in the most compressed images, even for a modest distortion level. In practice, the animation was also reported to be "dizzying" by two independent users.

4.1 First iteration

The first iteration of our design consisted of the merge of 2 ideas:

1. **A grid arrangement** to explore query similarities such as that proposed in visdb [16]. In visdb, the query is shown at the center of the image, and the most similar matches are shown around the query in a spiral.

2. **A fisheye distortion** for focus+context exploration of the spiral arrangement. This was inspired by the Fashion week visualization of Michael Bostock for the New York Times [2]. the image that end up with less space due to the distortion are not actually resized, but cropped, which visually helps preserve proportions and, in the case of fashion pictures, still conveys the attire being worn.

The result of this first implementation is shown in Figure 4. After using this implementation for a while, and showing it to two colleagues in the lab, it became apparent that the visualization was a failure. On one hand, the images are already quite small at 32×32 pixels, and further reducing their size (or cropping them) due to Cartesian fisheye distortion makes most images barely recognizable. This is a failure because the objective of focus+context is, precisely, to give context around the focus: in this case, the context is distorted beyond recognition, and rather becomes noise.

The second reason for failure, as reported by our colleagues and ourselves, was that moving the mouse around and recomputing the distortion results in too much motion in the visual field of the user. This results in a sensation of dizziness. This result was surprising and unexpected, but we believe that it confirms the rule of thumb "no unjustified 2d" [19], and is in line with previous findings that have shown that fisheye distortion can be disorienting.

4.2 Second iteration

Our second iteration built upon the lessons learned of our first iteration, and opted for a more conservative approach. The solution consists of 2 panes: a left pane that shows a gridified low-dimensional embedding of the images – which primarily supports the exploration task –, and a right pane that displays queries and their distributions – for the comparison tasks.

The separation of panes allows a separation of the two tasks. As opposed to the first iteration, triggering a query does not result in a rearrangement of all the images. Rather, what the user has explored remains unchanged and query exploration happens in a different pane. This has the advantage that, once the user has found an interesting neighbourhood, they can keep exploring the queries withouth having to restart the search for a particular image. For example in Figure 1, the user explores a cluster of laundry machine and then moves to explore the presumably similar cluster of power outlets.

The right pane is dedicated to query exploration and has 2 main components: a big chart for direct query distribution comparison, and a tab to explore each query individually. When the user clicks on an image in the left pane, this triggers a query – that is, a similarity to every other image in the dataset is computer and sorted from high-to-low. This produces a monotonically decreasing curve that we show together with a canvas to explore the images that this similarity corresponds to. Because we show both the computer similarity curve *and* the retrieved images, our design allows users to compare the computed vs. perceived image similarities. Moreover, since multiple queries can be shown side-by-side, both computed and perceived similarities can be visually compared for different queries. Finally, the top-most pane allows for a more direct comparison the similarity curves by plotting the curves on the same chart.

5 Implementation

We implemented deepviz using primarily javascript and d3 [9]. For our first iteration, we used the Cartesian fisheye plugin provided by Michael Bostock [4], and followed the tutorial by Irene Ros [5] to learn how to draw images on a canvas. From browsing the code of the New York Times' fashion week visualization, we borrowed the idea of using a large image tile to reduce the number of http requests: this had a big impact in performance compared to our first implementations.

5.1 t-SNE gridification

For the gridification process of t-SNE, we used the static kd-tree javascript library by Mikola Lysenko [**?**]. Despite it's name, it actually uses a Voronoi diagram to accelerate nearest neighbour search in 2-dimensional spaces. This allowed us to reduce the time for gridification from ¿20 seconds to ¡300 milliseconds to render the gridified t-SNE. While this is not implemented in our current syste, this means that we could potentially dynamically adjust to the grid to different pane sizes while maintaining performance close to real-time.

The gridification of t-SNE was made in a greedy manner. For each image in the dataset, the top k nearest neighbours in the grid are found, and the image is assigned to the first empty spot found; the process is repeated until all the images have a place in the grid. We found that for $k = 50$ this process runs in under 300 milliseconds and takes less than five iterations to find a place for all the images.

Fig. 5. The gridified t-sne embedding produced for our dataset. This supports the task of dataset exploration, while giving a sense of (close) similarities for each image.

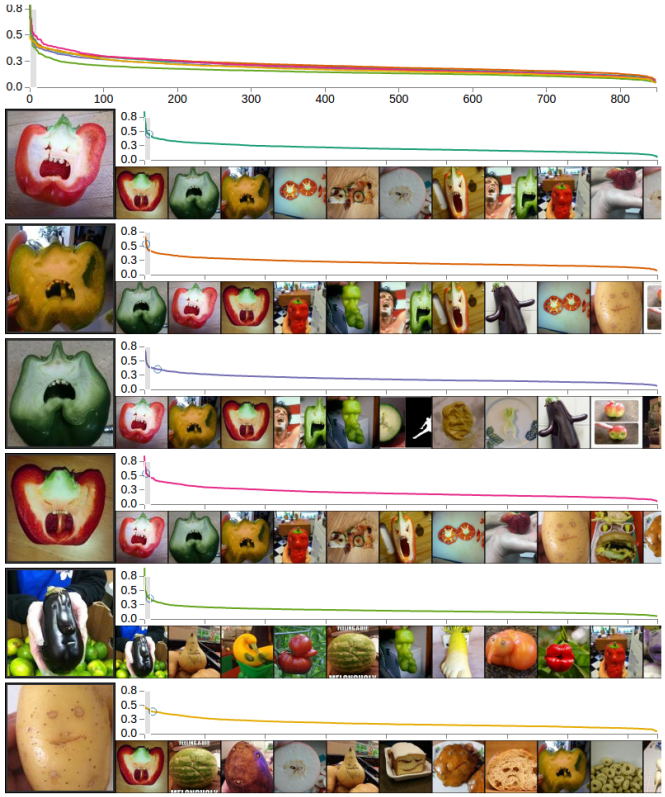

Fig. 6. The queries as shown right after the user clicks on an image.

5.2 Query distribution

We implemented the similarity search by brute-force: thanks to the relatively small size of the dataset, there was not scalability problem on this side. All the data was pre-processed in Python, using the standard image library of Python 3 for cropping and resizing. For feature extraction, we used the VGG-net implementation provided by Karen Simonyan together with Caffe [15] and its python bindings. The datasets were all saved to disk using HDF5, and exported to json files using Python's native json library.

The similarity distribution plots were implemented using svg panels with d3, and for their colors we used Cynthia Brewer's color palette [14]. The joint highlighting through the canvases and the query plots was achieved by maintaining global arrays that store the brush position and infer the position given the mouse coordinates over the canvas. Where possible, closures were preferred over global variables, but the observer pattern (i.e., the use of global "listener" variables) was implemented for all the events that span more than one html element.

6 Results

A global screenshot of our system is shown on Figure 1. The two main components are the gridified t-sne pane and the query pane. Each merit their own individual exposition.

6.1 Gridified t-SNE

When the user first interacts with our system, they are presented solely with the low-dimensional gridified and minified embedding of their data, as show in Figure 5. We confirmed that the a resolution of 32×32 is sufficient to recognize the objects in the images. Since the images are heavily curated, and the objects tend to be centered and well-focused, using a central crop to represent the image does not result in significant drawbacks – but it makes tiling trivial and the arrangement more organized. Clicking on an image results in a query being added to the right pane of our system.

6.2 Query pane

A closer look at the queries pane is shown in Figure 6. In this case, the user has decided to explore a cluster of peppers followed by a squash and a potato. For each query, the similarity distribution is shown at the top and the corresponding images are rendered below. In this example, we can see that the queries follow moslty an inverse Gaussian distribution, but have subtle differences between them. While the peppers form a tight cluster (as evidenced by their proximity in the t-SNE embedding and the fact that they retrieve each other at the top), the first, second and fourth pepper retrieve an image ofr an angry red pepper with a wooden background. The third pepper does not show this image among its top matches, but manages to retrieve another pepper that is compared to the face of Sylvester Stallone. This suggests, for example, that the queries are complementary, and that deep features are not retrieving all the pepper faces for all the queries; however, this could be alleviated with query expansion techniques, as the neighbours of the top matches provide more diversity in the visual search.

Exploring similarity distributions The interface also allows the user to explore different parts of the similarity distribution. In Figure 7, the user explores the tail of the similarity distribution. Looking at the images, it is revealed that the top strawberries, which feature shapes that are reminiscent of a chicken and a baby elephant, have a similar tail; they, for example, share an image featuring jeans and several indoors and vehicle pictures. The last picture also shares a motorcycle image with the first query, and a round object with the second query. This suggests that the similarity distributino is not affected by the zoomorphic shape of the first two strawberries, but is shared by all strawberries in the dataset.

Another example of this distribution exploration is show in Figure 8. In this case, the user explores the middle of the distribution, and finds that the space is not consistent across the queries. Some queries feature signs and animals, while others have coffee mugs and washing machines. Overall, there is no apparent structure at the middle of the distribution. This is further confirmed by the synchronized ball along the curves: the same image appears at very different places in the distribution, showing that the same image is given very different similarities for different queries. This suggests that, for deep features, similarity distributions are consistent at the beginning and at the end, but not in the middle.

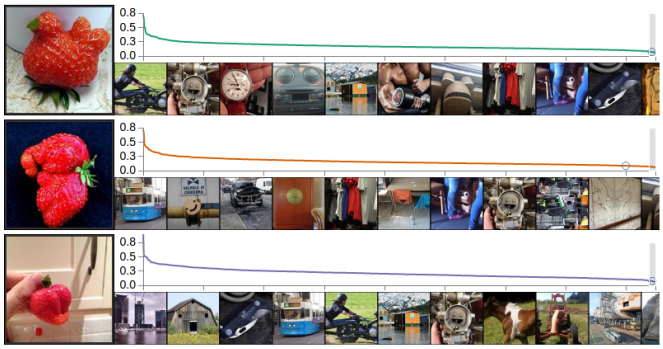

Fig. 7. Queries of strawberries with a focus on the end of the distribution tail. The fact that the first two strawberries look like animals does not result in distinctive dissimilarities compared to the less interestingly shaped strawberry at the bottom.

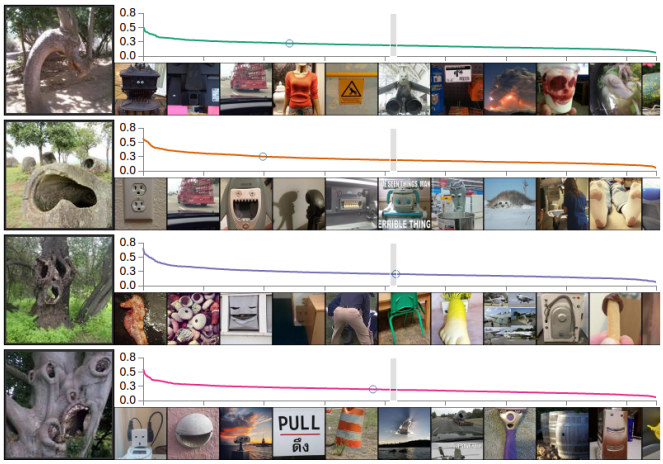

Fig. 8. Queries of trees with a focus on the middle end of the distribution. Unlike previous examples focused at the top and the end, there is no clear structure in the images retrieved by these examples.

Query-to-query comparison Finally, we show an exmple where two queries are compared against each other in Figure 9. The users queries the system to compute the similarity of two music players that have big eyes and an even bigger smile. While one would expect them to have similar query distributions, a closer look at the plot reveals that their similarities are, in fact, quite different from one another: regarding the computed similarity, the bottom image produces a much higher ranking from beginning to end, and regarding the perceived similarity, we can see that the bottom image and is primarily match to household items, such as electric plugs, a vacuum cleaner and even some tolet paper. In contrast, the top image is matched against other music players and car music systems. This example further confirms that deep features probably do not retrieve faces in things (as in this case they would retrieve other big-eyed, big-smile items), but rather simply retrieve similar objects.

7 Discussion and Future Work

To the best of our knowledge, our work is the first to provide a tool that allows users to explore image retrieval results at different similarity levels. The low-dimensional embedding is effective at displaying neighbourhood relations, but we also found interesting relations in queries at other parts of the similarity distribution.

One of the biggest limitations of our system is scale. For a fun dataset like our "faces in things" this is not much of a problem, but retrieval inevitably gets harder as more and more images are added to the database, since there is a higher potential for confusion. Typical datasets in academic benchmarks have around 100 000 images, and

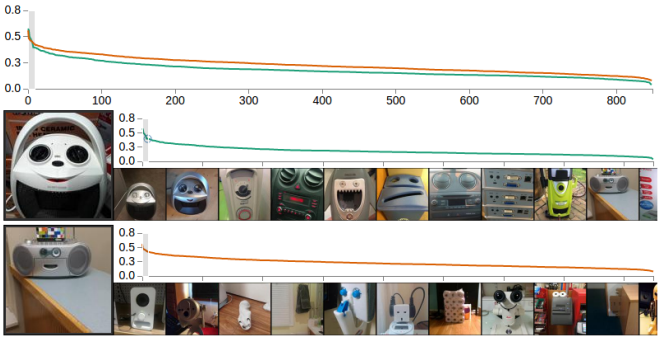

Fig. 9. Two queries of music players one next to the other.

our system is far from handling such numbers.

7.1 Lessons learned

Valuable lessons were learned from this project; among them, that

1. the "no unjustified 2d" rule of thumb by Munzner [19] proved especially true for our first iteration
2. fisheye lenses can be easily abused, and feel disorienting
3. moving images quickly around the field of view of the user can easily lead to a design that causes dizziness
4. pixels are a very precious resource: it is extremely easy to come up with designs that need more pixels than most users have. In our case, we programmed everything in a desktop computer with a big monitor, and did not realize that laptops tend to have smaller screens.

On the more technical side, we also learned that javascript is way faster than it was a few years ago: we were surprised that little optimization was necessary to achieve real-time performance for our system.

7.2 Future work

Future work should focus mainly on usability: we ran out of time to, for example, allow users to delete their queries, or to give them the option to choose the sizes of the images in the t-SNE embedding and in the queries. We would also like to add a resize option to the embedding panel, and synchronized highlighting in the canvases (both for queries and the embedding). Finally, we would also like to add a visualization that matches the images ranking in a pair of query similarity distributions explicitly, for example, by linking with lines two query lines – this would help users to explicitly see the correlation among queries in a way similar to parallel coordinates. It would also be helpful to remove duplicated images, of which we found around 5, as whe they are found they give the sense that the data has not been adequately preprocessed.

8 Conclusions

We have introduced deepviz, a system that allows user to explore the similarities captured by deep features in medium-sized image datasets. We have shown our implementation on a dataset of faces in things, and demonstrated multiple uses cases for query comparison, visual similarity comparison, and similarity exploration in a gridified low-dimensional neighbourhood-preserving embedding. Our system allows, for the first time, an easy exploration of diffent parts of the similarity distribution for image retrieval systems, in contrast to traditional visualizations where only the top matches are available.

Our implementation is open source, and is hosted publicly on https://github.com/jltmtz/deepviz. Similarly, a live demo with the faces in things datset can be accessed online at jltmtz.github.io/deepviz.

ACKNOWLEDGMENTS

The author wishes to thank Tamara Munzner for her faith in this project idea, even thought it ended up changing three times and did not have a clear visualization component at the beginning. We also thank our lab colleagues, Fred and Ankur for trying out the first iteration of this project and honestly expressing how uneasy their stomachs felt.

REFERENCES

[1] https://googleblog.blogspot.ca/2009/11/explore-images-with-google-image-swirl.html.
[2] http://www.nytimes.com/newsgraphics/2013/09/13/fashion-week-editors-picks/.
[3] https://github.com/tweepy/tweepy.
[4] https://github.com/d3/d3-plugins/tree/master/fisheye.
[5] https://bocoup.com/weblog/d3js-and-canvas.
[6] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, 2014.
[7] A. Babenko and V. Lempitsky. Aggregating deep convolutional features for image retrieval. *arXiv preprint arXiv:1510.07493*, 2015.
[8] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. *ECCV*, 2014.
[9] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.
[10] P. Brivio, M. Tarini, and P. Cignoni. Browsing large image datasets through voronoi diagrams. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1261–1270, 2010.
[11] K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using convnets and GPUs. *arXiv preprint arXiv:1407.4764*, 2014.
[12] J. Chen, Y. Xu, G. Turk, and J. Stasko. Easyzoom: Zoom-in-context views for exploring large collections of images (Technical report). 2013.
[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
[14] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.
[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ICM*, pages 675–678.
[16] D. Keim, H.-P. Kriegel, et al. Visdb: Database exploration using multidimensional visualization. *Computer Graphics and Applications, IEEE*, 14(5):40–49, 1994.
[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014.
[19] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2014.
[20] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.
[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.
[23] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *TPAMI*, 30(11), 2008.
[24] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
[25] C. Wang, J. P. Reese, H. Zhang, J. Tao, Y. Gu, J. Ma, and R. J. Nemiroff. Similarity-based visualization of large image collections. *Information Visualization*, 2013.
[26] R. J. Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *22nd Annual Technical Symposium*, pages 136–143. International Society for Optics and Photonics, 1979.
[27] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, 2014.