Low-shot learning with large-scale diffusion

Matthijs Douze[†], Arthur Szlam[†], Bharath Hariharan[†*], Hervé Jégou[†]
[†]Facebook AI Research
[*]Cornell University

Abstract

This paper considers the problem of inferring image labels from images when only a few annotated examples are available at training time. This setup is often referred to as low-shot learning, where a standard approach is to retrain the last few layers of a convolutional neural network learned on separate classes for which training examples are abundant. We consider a semi-supervised setting based on a large collection of images to support label propagation. This is possible by leveraging the recent advances on large-scale similarity graph construction.

We show that despite its conceptual simplicity, scaling label propagation up to hundred millions of images leads to state of the art accuracy in the low-shot learning regime.

1. Introduction

Large, diverse collections of images are now commonplace; these often contain a "long tail" of visual concepts. Some concepts like "person" or "cat" appear in many images, but the vast majority of the visual classes do not occur frequently. Even though the total number of images may be large, it is hard to collect enough labeled data for most of the visual concepts. Thus if we want to learn them, we must do so with few labeled examples. This task is named *low-shot learning* in the literature.

In order to learn new classes with little supervision, a standard approach is to leverage classifiers already learned for the most frequent classes, employing a so-called *transfer learning* strategy. For instance, for new classes with few labels, only the few last layers of a convolutional neural network are re-trained. This limits the number of parameters that need to be learned and limits over-fitting.

In this paper, we consider the low-shot learning problem described above, where the goal is to learn to detect new visual classes with only a few annotated images per class, but we also assume that we have many unlabelled images. This is called semi-supervised learning [39, 37] (con-
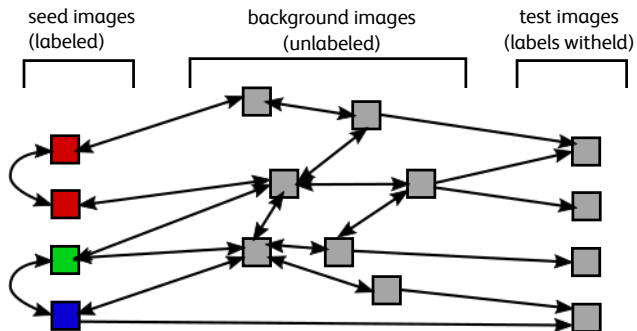
Figure 1. The diffusion setup. The arrows indicate the direction of diffusion. There is no diffusion performed from the test images. For the rest of the graph, the edges are bidirectional (*i.e.*, the graph matrix is symmetric). Except when mentioned otherwise, the edges have no weights.

sidered, *e.g.*, for face annotation [14]). The motivation of this work is threefold. First we want to show that with modern computational tools, classical semi-supervised learning methods scale gracefully to hundreds of millions of unlabeled points. A limiting factor in previous evaluations was that constructing the similarity graph supporting the diffusion was slow. This is no longer a bottleneck: thanks to advances both in computing architectures and algorithms, one can routinely compute the similarity graph for 100 millions images in a few hours [21]. Second, we want to answer the question: *Does a very large number of images help for semi-supervised learning?* Finally, by comparing the results of these methods on Imagenet and the YFCC100M dataset [33], we highlight how these methods exhibit some artificial aspects of Imagenet that can influence the performance of low shot learning algorithms.

In summary, the contribution of our paper is a study of semi-supervised learning in the scenario where we have a very large number of unlabeled images. Our main results are that in this setting, semi-supervised learning leads to state of the art low-shot learning performance. In more detail, we make the following contributions:

- We carry out a large-scale evaluation for diffusion methods for semi-supervised learning and compare

it to recent low-shot learning papers. Our experiments are all carried out on the public benchmark Imagenet [11] and the YFC100M dataset [33].

- We show that our approach is efficient and that the diffusion process scales up to hundreds of millions of images, which is order(s) of magnitude larger than what we are aware in the literature on image-based diffusion [19, 18]. This is made possible by leveraging the recent state of the art for efficient k-nearest neighbor graph construction [21].

- We evaluate several variants and hypotheses involved in diffusion methods, such as using class frequency priors [38]. This scenario is realistic in situations where this statistic is known a priori. We propose a simple way to estimate it without this prior knowledge, and extend this assumption to a multiclass setting by introducing a probabilistic projection step derived from Sinkhorn-Knopp algorithm.

- Our experimental study shows that a simple propagation process significantly outperforms some state-of-the-art approaches in low-shot visual learning when (i) the number of annotated images per class is small and when (ii) the number of unlabeled images is large or the unlabeled images come form the same domain as the test images.

This paper is organized as follows. Section 2 reviews related works and Section 3 describes the label propagation methods. The experimental study is presented in Section 4. Our conclusion in section 5 summarizes our findings.

2. Related work

Low-shot learning Recently there has been a renewed interest for low-shot learning, *i.e.*, learning with few examples thanks to prior statistics on other classes. Such works include metric learning [26], learning kNN [35], regularization and feature hallucination [16] or predicting parameters of the network [5]. Ravi and Larochelle introduce a meta-learner to learn the optimization parameters invovled in the low-shot learning regime [30]. Most of the works consider small datasets like Omniglot, CIFAR, or a small subset of Imagenet. In our paper we will focus solely on large datasets, in particular the Imagenet collection [31] associated with the ILSVRC challenge.

Diffusion methods We refer the reader to [3, 12] for a review of diffusion processes and matrix normalization options. Such methods are an efficient way of clustering images given a matrix of input similarity, or a kNN graph, and have been successfully used in a semi-supervised discovery

setup [14]. They share some connections with spectral clustering [6]. In [29], a kNN graph is clustered with spectral clustering, which amounts to computing the k eigenvectors associated with the k largest eigenvalues of the graph, and clustering these eigenvectors. Since the eigenvalues are obtained via Lanczos iterations [15, Chapter 10], the basic operation is similar to a diffusion process. This is also related to power iteration clustering [25], as in the work of Cho *et al.* [8] to find clusters.

Semi-supervised learning The kNN graph can be used for transductive and semi-supervised learning (see *e.g.*[3, 39] for an introduction). In transductive learning, a relatively small number of labels are used to augment a large set of unlabeled data and the goal is to extend the labeling to the unlabeled data (which is given at train time). Semi-supervised learning is similar, except there may be a separate set of test points that are not seen at train time. In our work, we consider the simple proposal of Zhu *et al.* [38], where powers of the (normalized) kNN graph are used to find smooth functions on the kNN graph with desired values at the labeled points. There exist many variations on the algorithms, *e.g.*, Zhou *et al.* [37] weight the edges based on distances and introduce a loss trading a classification fitting constraint and a smoothness term enforcing consistency of neighboring nodes.

Label propagation is a transductive method. In order to evaluate on new data, we need to extend the smooth functions out of the training data. While deep networks have been used before for out of sample extension, *e.g.*, in [7] and [20], in the speech domain, in this work, we use a weighted sum of nearest neighbors from the (perhaps unlabeled) training data [4].

Efficient kNN-graph construction The diffusion methods use a matrix as input containing the similarity between all images of the dataset. Considering N images, *e.g.*, $N = 10^8$, it is not possible to store a matrix of size N^2. However most of the image pairs are not related and have a similarity close to 0. Therefore diffusion methods are usually implemented with sparse matrices. This means that we compute a graph connecting each image to its neighbors, as determined by the similarity metric between image representations. In particular, we consider the k-nearest neighbor graph (kNN-graph) over a set of vectors. Several approximate algorithms [10, 23, 1, 17] have been proposed to efficiently produce the kNN graph used as input of iterative/diffusion methods, since this operation is of quadratic complexity in the number of images. In this paper, we employ the Faiss library,which was shown capable to construct a graph connecting up to 1 billion vectors [21].

3. Propagating labels

This section describes the initial stage of our proposal, which estimates the class of the unlabelled images with a diffusion process. It includes an image description step, the construction of a kNN graph connecting similar images, and a label diffusion algorithm.

3.1. Image description

A meaningful semantic image representation and an associated metric is required to match instances of classes that have not been seen beforehand. While early works on semi-supervised labelling [14] were using ad-hoc semantic global descriptors like GIST [27], we extract activation maps from a CNN trained on images from a set of *base* classes that are independent from the *novel* classes on which the evaluation is performed. See the experimental section for more details about the training process for descriptors.

The mean class classifier introduced for low-shot learning [26] is another way to perform dimensionality reduction while improving accuracy thanks to a better comparison metric. We do not consider this approach since it can be seen as part of the descriptor learning.

3.2. Affinity matrix: approximate kNN graph

As discussed in the related work, most diffusion processes use as input the kNN graph representing the $N \times N$ sparse similarity matrix, denoted by $\mathbf{W}$, which connects the N images of the collection. We build this graph using approximate k-nearest neighbor search. Thanks to recent advances in efficient similarity search [10, 21], trading some accuracy against efficiency drastically improves the graph construction time. As an example, with the FAISS library [21], building the graph associated with 600k images takes 2 minutes on 1 GPU. In our preliminary experiments, the approximation in the knn-graph does not induce any sub-optimality, possibly because the diffusion process compensates the artifacts induced by the approximation.

Different strategies exist to set the weights of the affinity matrix $\mathbf{W}$. We choose to search the k nearest neighbors of each image, and set a 1 for each of the neighbors in the corresponding row of a sparse matrix $\mathbf{W}_0$. Then we symmetrize the matrix by adding it to its transpose. We subsequently ℓ_1-normalize the rows to produce a sparse stochastic matrix: $\mathbf{W} = D^{-1}(\mathbf{W}_0^\top + \mathbf{W}_0)$, with D the diagonal matrix of row sums.

The handling for the test points is different: test points do not participate in label propagation because we classify each of them independently of the others. Therefore, there are no outgoing edges on test points; they only get incoming edges from their k nearest neighbors.

3.3. Label propagation

We now give details about the diffusion process itself, which is summarized in Figure 1. We build on the straightforward label propagation algorithm of [38]. The set of images on which we perform diffusion is composed of n_L labelled *seed* images and n_B unlabelled *background* images ($N = n_\mathrm{L} + n_\mathrm{B}$). Define the $N \times C$ matrix $\mathbf{L}$, where C is the number of classes for which we want to diffuse the labels, *i.e.*, the new classes not seen in the training set. Each row l_i in $\mathbf{L}$ is associated with a given image, and represents the probabilities of each class for that image. A given column corresponds to a given class, and gives its probabilities for each image. The method initializes l_i to a one-hot vector for the seeds. Background images are initialized with 0 probabilities for all classes. Diffusing from the known labels, the method iterates as $\mathbf{L}_{t+1} = \mathbf{W}\mathbf{L}_t$.

We can optionally reset the $\mathbf{L}$ rows corresponding to seeds to their 1-hot ground-truth at each iteration. When iterating to convergence, all l_i would eventually converge to the eigenvector of $\mathbf{W}$ with largest eigenvalue (when not resetting), or to the harmonic function with respect to $\mathbf{W}$ with boundary conditions given by the seeds (when resetting). Empirically, for low-shot learning, we observe that resetting is detrimental to accuracy. Early stopping performs better in both cases, so we cross-validate the number of diffusion iterations.

Classification decision & combination with logistic regression We predict the class of a test example i as the the column that maximizes the score l_i. Similar to Zhou *et al.* [37], we have also optimized a loss balancing the fitting constraint with the diffusion smoothing term. However we found that a simple late fusion (weighted mean of log-probabilities, parametrized by a single cross-validated coefficient) of the scores produced by diffusion and logistic regression achieves better results.

3.4. Variations

Using priors The label propagation can take into account several priors depending on the assumptions of the problem, which are integrated by defining a normalization operator η and by modifying the update equation as

$$\mathbf{L}_{t+1} = \eta(\mathbf{W}\mathbf{L}_t). \tag{1}$$

Multiclass assumption. For instance, in the ILSVRC challenge built upon the Imagenet dataset [31], there is only one label per class, therefore we can define η as a function that ℓ_1-normalizes each row to provide a distribution over labels (by convention the normalization leaves all-0 vectors unchanged).

Class frequency priors. Additionally, we point out that labels are evenly distributed in Imagenet. If we translate this

setup to our semi-unsupervised setting, it would mean that we may assume that the distribution of the unlabelled images is uniform over labels. This assumption can be taken into account by defining η as the function performing a ℓ_1 normalization of columns of $\mathbf{L}$.

While one could argue that this is not realistic in general, a more realistic scenario is to consider that we know the marginal distribution of the labels, as proposed by Zhu *et al.* [38], who show that the prior can be simply enforced (*i.e.*, apply column-wise normalization to $\mathbf{L}$ and multiply each column by the prior class probability). This arises in situations such as tag prediction, if we can empirically measure the relative probabilities of tags, possibly regularized for lowest values.

Combined Multiclass assumption and class frequency priors. We propose a variant way to use both a multiclass setting and prior class probabilities by enforcing the matrix $\mathbf{L}$ to jointly satisfy the following properties:

$$\mathbf{L}\mathbf{1}_C = \mathbf{1}_N \qquad \mathbf{1}_N^\top \mathbf{L} \propto \mathbf{p}_\mathcal{C} \qquad (2)$$

where $\mathbf{p}_\mathcal{C}$ is the prior distribution over labels. For this purpose, we adopt a strategy similar to that of Cuturi [9] in his work on optimal transport, in which he shows that the Sinkhorn-Knopp algorithm [32] provides an efficient and theoretically grounded way to project a matrix so that it satisfies such marginals. The Sinkhorn-Knopp algorithm iterates by alternately enforcing the marginal conditions, as

$$\mathbf{L} \leftarrow \mathbf{L}\,\mathrm{diag}(\mathbf{L}\mathbf{1}_C)^{-1}\mathrm{diag}(\mathbf{p}_\mathcal{C}) \qquad (3)$$
$$\mathbf{L} \leftarrow \mathrm{diag}(\mathbf{1}_N^\top \mathbf{L})^{-1}\mathbf{L} \qquad (4)$$

until convergence. Here we assume that the algorithm only operates on rows and columns whose sum is strictly positive. As discussed by Knight [24], the convergence of this algorithm is fast. Therefore we stop after 5 iterations. This projection is performed after each update by Eqn. 1. Note that Zhu *et al.* [38] solely considered the second constraint in Eqn. 2, which can be obtained by enforcing the prior, as discussed by Bengio *et al.*[3]. We evaluate both variants in the experimental section 4.

Non-linear updates. The Markov Clustering (MCL) [13] is another diffusion algorithm with nonlinear updates originally proposed for clustering. In contrast to the previous algorithm, MCL iterates directly over the similarity matrix as

$$\mathbf{W}'_t \leftarrow \mathbf{W}_t \cdot \mathbf{W}_t \qquad \mathbf{W}_{t+1} \leftarrow \Gamma_r(\mathbf{W}'_t), \qquad (5)$$

where Γ_r is an element-wise raising to power r of the matrix, followed by a column-wise normalization [13]. The power $r \in (1, 2]$ is a bandwidth parameter: when r is high, small edges quickly vanish along the iterations. A smaller

r preserves the edges longer. The clustering is performed by extracting connected components from the final matrix. In Section 4 we evaluate the role of the non-linear update of MCL by introducing the Γ_r non-linearity in the diffusion procedure. More precisely, we modify Equation 1 as $\mathbf{L}_{t+1} = \Gamma_r\left(\eta(\mathbf{W}\mathbf{L}_t)\right)$.

3.5. Complexity

For the complexity evaluation, we distinguish two stages. In the **off-line** stage, *(i)* the CNN is trained on the base classes, *(ii)* descriptors are extracted for the background images, and *(iii)* a knn-graph is computed for the background images. In the **on-line** stage, we receive training and test images from novel classes, *(i)* compute features for them, *(ii)* complement the knn-graph matrix to include the training and test images, and *(iii)* perform the diffusion iterations. Here we assume that the $N \times N$ graph matrix $\mathbf{W}_0$ is decomposed in four blocks

$$\mathbf{W}_0 = \left[\begin{array}{cc} \mathbf{W}_{LL} & \mathbf{W}_{LB} \\ \mathbf{W}_{BL} & \mathbf{W}_{BB} \end{array} \right] \in \{0,1\}^{(n_L+n_B)\times(n_L+n_B)} \quad (6)$$

The largest matrix $\mathbf{W}_{BB} \in \{0,1\}^{n_B \times n_B}$ is computed off-line. On-line we compute the three other matrices. We combine $\mathbf{W}_{BL}$ and $\mathbf{W}_{BB}$ by merging similarity search result lists, hence each row of $\mathbf{W}_0$ contains exactly k non-zero values, requiring to store the distances along with $\mathbf{W}_{BB}$.

We are mostly interested in the complexity of the on-line phase. Therefore we exclude the descriptor extraction, which is independent of the classification complexity, and the complexity of handling the test images, which is negligible compared to the training operations. We consider the logistic regression as a baseline for the complexity comparison:

Logistic regression the SGD training entails $\mathcal{O}(I_{\mathrm{logreg}} \times B \times C \times d)$ multiply-adds, with d denotes the descriptor dimensionality and C the number of classes. The number of iterations and batch size are I_{logreg} and B.

Diffusion the complexity is decomposed into: computing the matrices $\mathbf{W}_{LL}$, $\mathbf{W}_{LB}$ and $\mathbf{W}_{BL}$, which involves $\mathcal{O}(d \times n_L \times n_B)$ multiply-adds using brute-force distance computations; and performing I_{dif} iterations of sparse-dense matrix multiplications, which incurs $\mathcal{O}(k \times N \times C \times I_{\mathrm{dif}})$ multiply-adds (note, sparse matrix operations are more limited by irregular memory access patterns than arithmetic operations). Therefore the diffusion complexity is linear in the number of background images n_B. See the supplemental for more details.

Memory usage. One important bottleneck of the algorithm is its memory usage. The sparse matrix $\mathbf{W}_0$ occupies $8Nk$ bytes in RAM, and $\mathbf{W}$ almost twice this amount,

because most nearest neighbors are not reciprocal; the $\mathbf{L}$ matrix is $4CN$ bytes. Fortunately, the iterations can be performed one column of $\mathbf{L}$ at a time, reducing this to $2 \times 4N$ bytes for $\mathbf{L}_t$ and $\mathbf{L}_{t+1}$ (in practice, when memory is an issue, we group columns by batches of size $C' < C$).

4. Experiments

4.1. Datasets and evaluation protocol

We use **Imagenet** 2012 [11] and follow a recent setup [16] previously introduced for low-shot learning. The 1000 Imagenet classes are split randomly into two groups, each containing base and novel classes. Group 1 (193 base and 300 novel classes) is used for hyper-parameter tuning and group 2 (196+311 classes) for testing with fixed hyper-parameters. We assume the full Imagenet training data is available for the base classes. For the novel classes, only n images per class are available for training. Similar to [16] the subset of n images is drawn randomly and the random selection is performed 5 times with different random seeds.

As a large source of unlabelled images, we use the **YFCC100M** dataset [33]. It consists of 99 million representative images from the Flickr photo sharing site[1]. Note that some works have used this dataset with tags or GPS metadata as weak supervision [22].

Learning the image descriptors. We use the 50-layer Resnet trained by Hariharan *et al.* [16] on all base classes (group 1 + group 2), to ensure that the description calculation has never seen any image of the novel classes. We run the CNN on all images, and extract a 2048-dim vector from the 49th layer, just before the last fully connected layer. This descriptor is used directly as input for the logistic regression. For the diffusion, we PCA-reduce the feature vector to 256 dimensions and L2-normalize it, which is standard in prior works on unsupervised image matching with pre-learned image representations [2, 34].

Performance measure and baseline In a given group (1 or 2), we classify the Imagenet validation images from *both* the base and novel classes, and measure the top-5 accuracy. Therefore the class distribution is heavily unbalanced. Since the seed images are drawn randomly, we repeat the random draws 5 times with different random seeds and average the obtained top-5 accuracy (the $\pm$xx notation gives the standard deviation).

The baseline is a logistic regression applied on the labelled points. We employ a per-class image sampling strategy to circumvent the unbalanced number of examples per class. We optimize the learning rate, batch size and L2 regularization factor of the logistic regression on the group 1

---

[1]Of the 100M original files, some are videos and some are not available anymore. We replace them with uniform white images.

background	none	F1M	Imagenet
edge weighting			
constant	62.7±0.68	65.4±0.55	73.3±0.72
Gaussian weighting*	62.7±0.66	65.4±0.58	73.6±0.71
meaningful neighbors*	62.7±0.68	40.0±0.20	73.6±0.62
η operator			
none	40.6±0.18	41.1±0.10	42.3±0.19
Sinkhorn	61.1±0.69	56.8±0.50	72.3±0.72
column-wise	62.7±0.68	65.4±0.55	73.3±0.72
non-linear transform* Γ_r	62.7±0.68	65.4±0.55	73.3±0.72
class frequency prior*	62.7±0.66	65.4±0.60	73.3±0.65

Table 1. Variations on weighting for edges and normalization steps on iterates of $\mathbf{L}$. The tests are performed for $n = 2$ and $k = 30$, with 5 runs on the group 1 validation images. Variants that require a parameter (*e.g.*, the σ of the Gaussian weighting) are indicated with a "*". In this case we report only the best result, see the supplementary material for full results. In the rest of the paper, we use the variants indicated **in bold**, since they are simple and do not add any parameter.

images. It is worth noticing that our baseline outperforms the reported state of the art in this setting.

Background images for diffusion We consider the following sets of background images:

1. *None:* the diffusion is directly from the seed images to the test images;

2. *In-domain setting:* the background images are the Imagenet training image from the novel classes, but without labels. This corresponds to a use case where all images are known to belong to a set of classes, but only a subset of them have been labelled;

3. *Out-of-domain setting:* the n_B background images are taken from YFCC100M. We denote this setting by F100k, F1M, F10M or F100M, depending on the number of images we use (*e.g.*, we note F1M for $n_B = 10^6$). This corresponds to a more challenging setting where we have no prior knowledge about the image used in the diffusion.

4.2. Parameters of diffusion

We compare a few settings of the diffusion algorithm as discussed in section 3.4. In all cases, we set the number of nearest neighbors to $k = 30$ and evaluate with $n = 2$. The nearest neighbors are computed with Faiss [21], using the IVFFlat index. It computes exact distances but occasionally misses a few neighbors.

Graph edge weighting. We experimented with different weightings for $\mathbf{W}_0$, that were proposed in the literature. We compared a constant weight, a Gaussian weighting [25, 3],
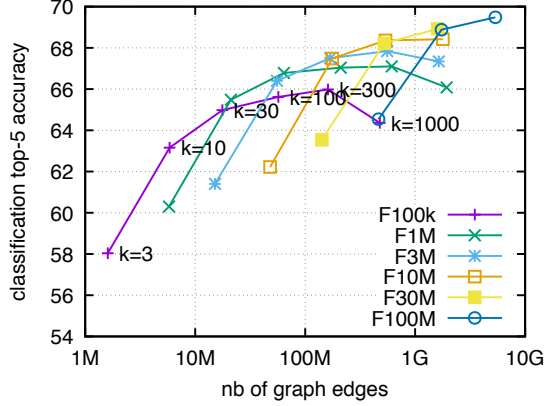
Figure 2. Classification performance with $n = 2$, with various settings of k and n_B, ordered by total number of edges (average of 5 test runs, with cross-validated number of iterations).

n	Hariharan et al. [16]	logistic regression	in-domain diffusion	diffusion + logistic
1	63.6	60.4±0.78	69.7±0.86	69.76±0.88
2	71.5	68.8±0.82	75.4±0.64	75.60±0.69
5	80.0	79.1±0.35	79.9±0.17	81.35±0.22
10	83.3	83.4±0.16	82.1±0.14	84.56±0.12
20	85.2	86.0±0.15	83.6±0.12	86.72±0.09

Table 2. **In-domain diffusion on Imagenet**: We compare against logistic regression and a recent low-shot learning technique [16] on this benchmark. Results are reported with $k = 30$ for diffusion.

(with σ a hyper-parameter), and a weighting based on the "meaningful neighbors" proposal [28].

Table 1 shows that results are remarkably independent of the weighting choice, which is why we set it to 1^2. The best normalization that can be applied to the $\mathbf{L}$ matrix is a simple column-wise L1 normalization. Thanks to the linear iteration formula, it can be applied at the end of the iterations.

4.3. Large-scale diffusion

Figure 2 reports experiments by varying the number of background images n_B and the number k of neighbors, for $n = 2$. All the curves have an optimal point in terms of accuracy *vs* computational cost at $k=30$. This may be a intrinsic property of the descriptor manifold. An additional number: before starting the diffusion iterations, with $k=1000$ and no background images (the best setting) we obtain an accuracy of 60.5%. This is a knn-classifier and this is the fastest setting because the knn-graph does not need to be constructed nor stored.

4.4. Comparison with low-shot classifiers

We compare the performance of diffusion against the logistic baseline classifiers and a recent method of the state of the art [16], using the same features.

In-domain scenario. For low-shot learning ($n \leq 5$), the in-domain diffusion outperforms the other methods by a large margin, see Table 2. The combination with logistic regression is not very effective.

Out-of-domain diffusion. Table 3 shows that the performance of diffusion is competitive only when 1 or 2 images are available per class. As stated in Section 3.2, we do not include the test points in the diffusion, which is standard for a classification setting. However, if we allow this, as in a fully transductive setting, we obtain a top-5 accuracy of **69.6%**±0.68 with $n = 2$ with diffusion over F1M, *i.e.*, on par with diffusion over F100M.

Classifier combination. We experimented with a very simple late fusion: to combine the scores of the two classifiers, we simply take a weighted average of their predictions (log-probabilities), and cross validate the weight factor. Both in the in-domain (Table 2) and out-of-domain (Table 3) cases, the results are significantly above the best of the two input classifiers. This shows that the logistic regression classifier and the diffusion classifier access different aspects of image collection. We also experimented with more complicated combination methods, like using the graph edges as a regularizer during the logistic regression, which did not improve this result.

Comparison with the state of the art. With the in-domain diffusion, we notice that our method outperforms the state-of-the-art result of [16] and which, itself, outperforms or is closely competitive with [35, 36] in this setting. In the out-of-domain setting, out results are better only for n=1. However, their method is a complementary combination of a specific loss and a learned data augmentation procedure that is specifically tailored to the experimental setup with base and novel classes. In contrast, our diffusion procedure is generic and has only two parameters (n_B and k). Note that the out-of-domain setting is comparable with the standard low-shot setting, because the unlabeled images from F100M are generic, and have nothing to do with Imagenet; and because the neighbor construction and diffusion are efficient enough to be run on a single workstation.

---

[2]Note that our parametric experiments use the set of baseline image descriptors used in the arXiv version of [16], and the table compares all methods using those underlying features, so the results are not directly comparable with the rest of the paper.

n	out-of-domain diffusion				logistic regression	diffusion+logistic		Hariharan et al. [16]
	none	F1M	F10M	F100M		+F10M	+ F100M	
1	58.5±0.52	61.4±0.61	62.7±0.76	63.6±0.61	60.4±0.78	63.3±0.73	**64.0**±0.70	63.6
2	63.6±0.60	66.8±0.71	68.4±0.74	69.5±0.60	68.8±0.82	70.6±0.80	71.1±0.82	**71.5**
5	69.0±0.46	72.5±0.27	74.0±0.35	75.2±0.40	79.1±0.35	79.4±0.34	79.7±0.38	**80.0**
10	73.9±0.15	76.2±0.19	77.4±0.31	78.5±0.34	83.4±0.16	83.6±0.13	**83.9**±0.10	83.3
20	78.0±0.15	79.1±0.23	80.0±0.27	80.8±0.18	86.0±0.15	86.2±0.12	**86.3**±0.17	85.2

Table 3. **Out-of-domain diffusion:** Comparison of classifiers for different values of n, with $k = 30$ for the diffusion results. The "none" column indicates that the diffusion solely relies on the labelled images. The results of the rightmost column [16] are state-of-the-art on this benchmark to our knowledge, generally outperforming the results of matching networks and model regression [35, 36] in this setting.

background	none	F1M	F10M	F100M
optimal iteration	2	3	4	5
timing: graph completion	2m57s	8m36s	40m41s	4h08m
timing: diffusion	4.4s	19s	3m44s	54m

Table 4. Timings for the different steps on a 24-core 2.5GHz machine, for a varying number of unlabelled images from YFCC100M. Note, the timing of 4h08m for graph completion over F100M takes only 23m when executed on 8 GPUs.

4.5. Complexity: Runtime and memory

We measured the run-times of the different steps involved in diffusion process and report them in Table 4. The graph construction time is linear in n_B, thanks to the precomputation of the graph matrix for the background images (see Section 3.5). For comparison, training the logistic regression takes between 2m27s and 12m, depending on the cross-validated parameters.

In terms of memory usage, the biggest F100M experiments need to simultaneously keep in RAM a **W** matrix of 5.3 billion non-zero values (39.5 GiB), and $\mathbf{L}_t$ and $\mathbf{L}_{t+1}$ (35.8 GiB, using slices of $C' = 96$ columns). This is the main drawback of using diffusion. However Table 3 shows that restricting the diffusion to 10 million images already provides most of the gain, while dividing by an order of magnitude memory and computational complexity.

4.6. Analysis of the diffusion process

We discuss how fast **L** "fills up" (it is dense after a few iterations). We consider the rate of nodes reached by the diffusion process: we consider very large graphs, few seeds and a relatively small graph degree. Figure 3 measures the sparsity of the matrix **L** (on one run of validation), which indicates the rate of (label, image) tuples that have not been attained by the diffusion process at each diffusion step. While the graph is not necessarily fully connected, we observe that most images can be reached by all labels in practice.

The fraction of nodes reached by all labeled points grows rapidly and converges to a value close to 1 in a few iterations when $k \geq 10$. In order to relate this observation with the performance attained along iterations, it is interesting to compare what happens in this plot to the one on the
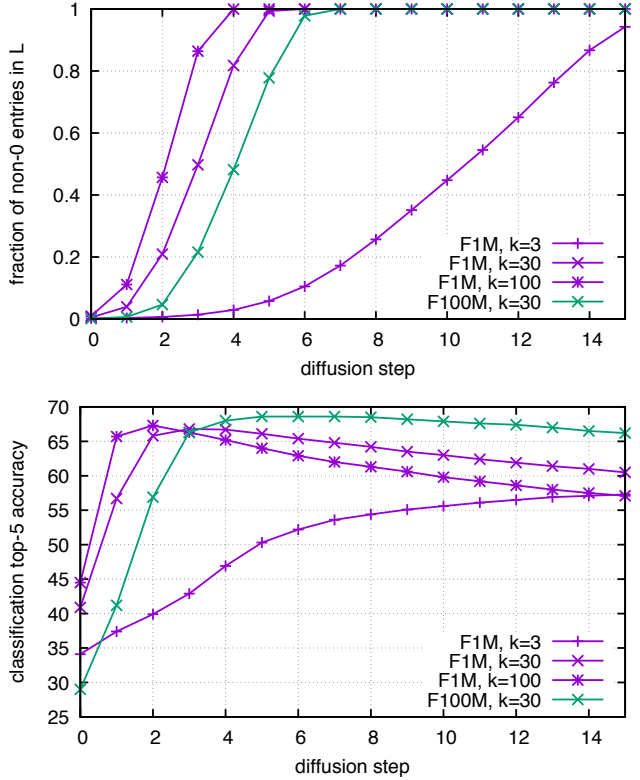

Figure 3. Statistics over iterations, for $n = 2$. *Top*: Rate of non-zero element in the matrix **L**. *Bottom*: corresponding accuracy.

right. The plot on the right shows that the iteration number at which the matrix close to 1 is similar to the iteration at which accuracy is maximal, as selected by cross-validation. The maximum occurs later if n_B is larger and when k is smaller. Note also that early stopping is important.

4.7. Qualitative results

Figure 4 shows paths between a seed image and test images, which gives a partial view of the diffusion. Given a class, we backtrack the path: for a given node (image) and iteration i, we look up the preceding node that contributed most to the weight in $\mathbf{L}_i$ that node at that iteration. At iteration 0, the backtracking process always ends in a source node. Each row of the figure is one such paths. For a test image (right), we show the path for the ground-truth class
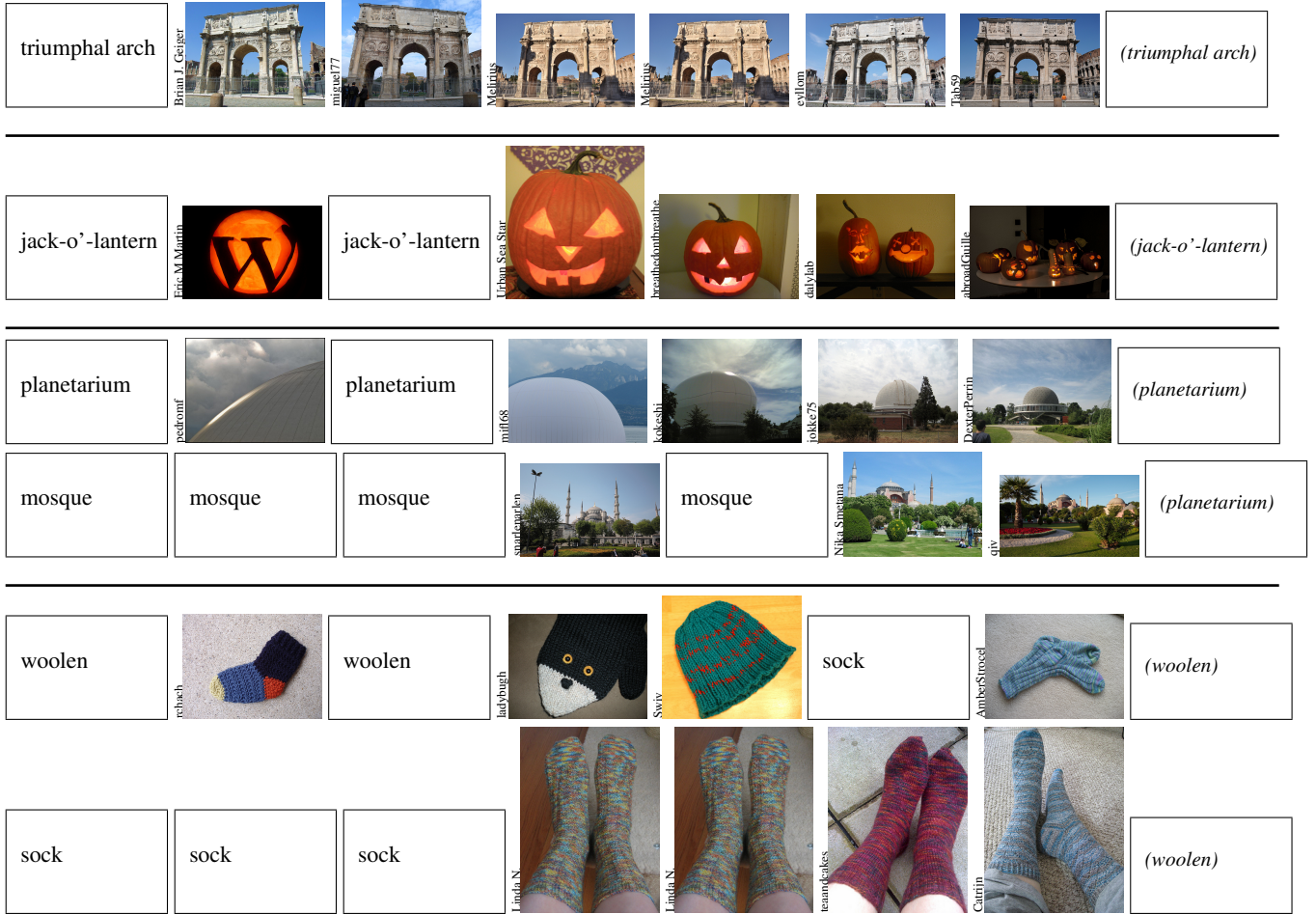
Figure 4. Images visited during the diffusion process from a seed (*left*) to the test image (*right*). We give ground-truth class for Imagenet images (test images marked by parentheses). The first two rows are classified correctly. The two bottom ones are failure cases. Imagenet images are not shown for copyright reasons, but the labels are shown. For YFCC100M images, we provide the Flickr id of their creators.

and that for the found class, or a single row for both when the image is classified correctly. Note that the preceding node can be the image itself, since the diagonal of the **W** matrix is not set to 0. Thanks to the size of the dataset, the paths are "smooth": they evolve through similar images.

5. Conclusion

We experimented on large-scale label propagation for low-shot learning. Unsurprisingly, we have found that performing diffusion over images from the same domain works much better than images from a different domain. We clearly observe that, as the number of images over which we diffuse grows, the accuracy steadily improve. The main performance factor is the total number of edges, which also reasonably reflects the complexity. We also report neutral results for most sophisticated variants, for instance we show that edge weights are not useful. Furthermore, labeled images should be included in the diffusion process and not just used as sources, *i.e.*, not enforced to keep their label.

The main outcome of our study is to show that diffusion over a large image set is superior to state-of-the-art methods for low-shot learning when very few labels are available. Interestingly, late-fusion with a standard classifier's result is effective. This shows the complementary of the approaches, and suggests that it could be combined with forthcoming methods for low-short learning.

When more labels are available, simple logistic regression becomes superior to the methods we describe (and to other state of the art low-shot learning methods). However, we note that there are many circumstances where even a few labels per class are more difficult to get than building (and then keeping) a graph over unlabeled data. For example, if there are a large number of "tail" classes which we will need to classify, a few examples per class can multiply to many labels. In these cases diffusion combined with logistic regression is the best method.

References

[1] Y. Avrithis, Y. Kalantidis, E. Anagnostopoulos, and I. Z. Emiris. Web-scale image clustering revisited. In *ICCV*, 2015.

[2] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *ECCV*, September 2014.

[3] Y. Bengio, O. Delalleau, and N. L. Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, chapter 11, pages 195–216. MIT Press, Boston, 2006.

[4] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS*, pages 177–184, 2003.

[5] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.

[6] R. R. C. Boaz Nadler, Stephane Lafon and I. G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. Technical report, Arxiv, 2008.

[7] T. Chin, L. Wang, K. Schindler, and D. Suter. Extrapolating learned manifolds for human activity recognition. In *ICIP*, pages 381–384, 2007.

[8] M. Cho and K. M. Lee. Mode-seeking on graphs via random walks. In *CVPR*, June 2012.

[9] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, pages 2292–2300, 2013.

[10] W. Dong, M. Charikar, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, March 2011.

[11] W. Dong, R. Socher, L. Li-Jia, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, June 2009.

[12] M. Donoser and H. Bischof. Diffusion processes for retrieval revisited. In *CVPR*, pages 1320–1327, 2013.

[13] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research*, 30(7), 2002.

[14] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, pages 522–530, 2009.

[15] G. H. Golub and C. V. Loan. *Matrix computations*. John Hopkinks University Press, 2013.

[16] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.

[17] B. Harwood and T. Drummond. Fanng: Fast approximate nearest neighbour graphs. In *CVPR*, 2016.

[18] A. Iscen, Y. Avrithis, G. Tolias, T. Furon, and O. Chum. Fast spectral ranking for similarity search. In *CVPR*, June 2017.

[19] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum. Efficient diffusion on region manifolds: Recovering small objects with compact CNN representations. In *CVPR*, June 2017.

[20] A. Jansen, G. Sell, and V. Lyzinski. Scalable out-of-sample extension of graph embeddings using deep neural networks. *CoRR*, abs/1508.04422, 2015.

[21] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017.

[22] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016.

[23] Y. Kalantidis, L. Kennedy, H. Nguyen, C. Mellina, and D. A. Shamma. Loh and behold: Web-scale visual search, recommendation and clustering using locally optimized hashing. *arXiv preprint arXiv:1604.06480*, 2016.

[24] P. A. Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[25] F. Lin and W. W. Cohen. Power iteration clustering. In *ICML*, 2010.

[26] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, December 2012.

[27] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[28] D. Omercevic, O. Drbohlav, and A. Leonardis. High-dimensional feature matching: employing the concept of meaningful nearest neighbors. In *ICCV*, October 2007.

[29] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *Computer Vision, Graphics & Image Processing*, 2008.

[30] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, April 2017.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[32] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348, 1967.

[33] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[34] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*, 2016.

[35] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[36] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.

[37] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, pages 321–328, 2003.

[38] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, 2003.

[39] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

Appendix

We present several additional results and details to complement the paper. Section A reports another evaluation protocol, which restricts the evaluation to novel classes. Sections B and C are parametric evaluations. Section D gives some details about the graph computation.

A. Evaluation results on novel classes

In the main paper, we evaluated the search performance on all the test images from group 2. The performance restricted to only the novel classes is also reported in prior work [16] using a combination of classifiers. Table 5 shows the results in this setting.

As to be expected, the results reported in these tables are inferior to those obtained in the setup where all test images are classified. This is because the novel classes are harder to classify than the base classes. Otherwise the ordering of the methods is preserved and the conclusions identical. The diffusion is effective in the low-shot regime and is, by itself, better than the state of the art by a large margin when only one example is available. The combination with late fusion significantly outperforms the state of the art, even in the out-of-domain setup.

B. Details of the parametric evaluation

In the paper we reported results for the edge weighting and graph normalization with the best parameter setting. Here, we report results for all parameters[3]. We evaluate the following edge weightings (Figure 5, first row):

- Gaussian weighting. The edge weight is e^{-x^2/σ^2} with x the distance between the edge nodes. Note that $\sigma \to \infty$ corresponds to a constant weighting;

- Weighting based on the "meaningful neighbors" proposal [28]. It relies on an exponential fit of neighbor distances. For a given graph node, for the neighbor i of its list of results, the weight is $s(1 - e^{-\lambda s})^k$, where s is the distance, remapped linearly to $[0, 1]$ so that the first neighbor has $s = 1$ and the kth neighbor has $s = 0$. We vary parameter λ in the plot.

---

[3]Note that our parametric experiments use the set of baseline image descriptors used in the arXiv version of the paper by Barath *et al.* [16], and the figure compares all methods using those underlying features. Therefore the results are not directly comparable with the rest of the paper.

We also report results for different normalizations of the matrix **L**. In Figure 5 (second row), we compare:

- The non-linear Γ_r normalization, all elements of **L** are raised to a power r. We vary the parameter r, and $r = 1$ corresponds to the identity transform;

- We classify all images in a graph with a logistic regression classifier. We use the predicted frequency of each class over the whole graph, and raise it to some power (the parameter) to reduce or increase its peakiness. This choice is inspired by the Markov Clustering Algorithm [13]. This gives a normalization factor that we enforce for each column of **L**, instead of the default uniform distribution.

The conclusion of these experiments is that these variants do not improve over constant weights and a standard diffusion, most of them having a neutral effect. Therefore, we conclude that the diffusion process mostly depends on the topology of the graph.

C. Late fusion weights

Let denote by l_i^{logreg} and $l_i^{\text{dif}} \in [0, 1]^C$ the distributions over classes returned by the two classifiers for image i. We fuse the loglikelihood by a weighted average, which amounts to retrieving the top-5 class prediction as those maximizing

$$a_n \log(l_{ic}^{\text{logreg}}) + (1 - a_n) \log(l_{ic}^{\text{dif}}), \qquad (7)$$

where a_n is the optimal mixing coefficient for n seed points, as found by cross-validation.

Figure 6 shows these optimal mixing factors. Since the logistic regression is better at classifying with many training examples, the parameter a_n increases with n.

D. Computation of the $\mathbf{W}_0$ blocks

As stated in the paper, we need to compute the 4 blocks of the matrix $\mathbf{W}_0$:

$$\mathbf{W}_0 = \begin{bmatrix} \mathbf{W}_{\text{LL}} & \mathbf{W}_{\text{LB}} \\ \mathbf{W}_{\text{BL}} & \mathbf{W}_{\text{BB}} \end{bmatrix} \in \{0, 1\}^{(n_{\text{L}} + n_{\text{B}}) \times (n_{\text{L}} + n_{\text{B}})}, \tag{8}$$

where, usually, $n_{\text{L}} \ll n_{\text{B}}$. Each block requires to perform a k-nearest neighbor search. We employ the Faiss library[4] optimized for this task [21], and use it as follows:

- $\mathbf{W}_{\text{BB}} \in \{0, 1\}^{n_{\text{B}} \times n_{\text{B}}}$: we use a Faiss index referred to as "IVFFlat". The accuracy-speed compromise is controlled by a parameter giving the number of inverted lists visited at search time. We adopted a relatively high probe setting (256) to guarantee that most

---

[4]http://github.com/facebookresearch/faiss

	out-of-domain diffusion				in-domain	logistic	combined		best reported
n	none	F1M	F10M	F100M	(Imagenet)	regression	+F10M	+ F100M	results [16]
1	39.4±0.85	43.9±0.96	46.3±1.28	47.6±1.09	57.7±1.28	42.6±1.31	46.5±1.23	**47.9**±1.18	45.1
2	47.8±0.94	52.7±1.14	55.2±1.21	57.0±1.05	66.9±1.06	54.4±1.29	57.5±1.34	58.4±1.29	**58.8**
5	56.8±0.73	62.2±0.44	64.6±0.57	66.3±0.68	73.8±0.29	71.4±0.54	71.9±0.55	72.3±0.58	**72.7**
10	64.9±0.28	68.0±0.33	69.9±0.47	71.7±0.54	77.6±0.23	78.6±0.27	78.7±0.21	**79.2**±0.14	79.1
20	71.4±0.26	72.7±0.40	74.1±0.38	75.3±0.29	80.0±0.21	82.9±0.20	83.0±0.15	**83.2**±0.22	82.6

Table 5. Comparison of classifiers for different values of n, with $k = 30$ for the diffusion results, evaluating *only on novel classes* using the two different sets of features that we consider.
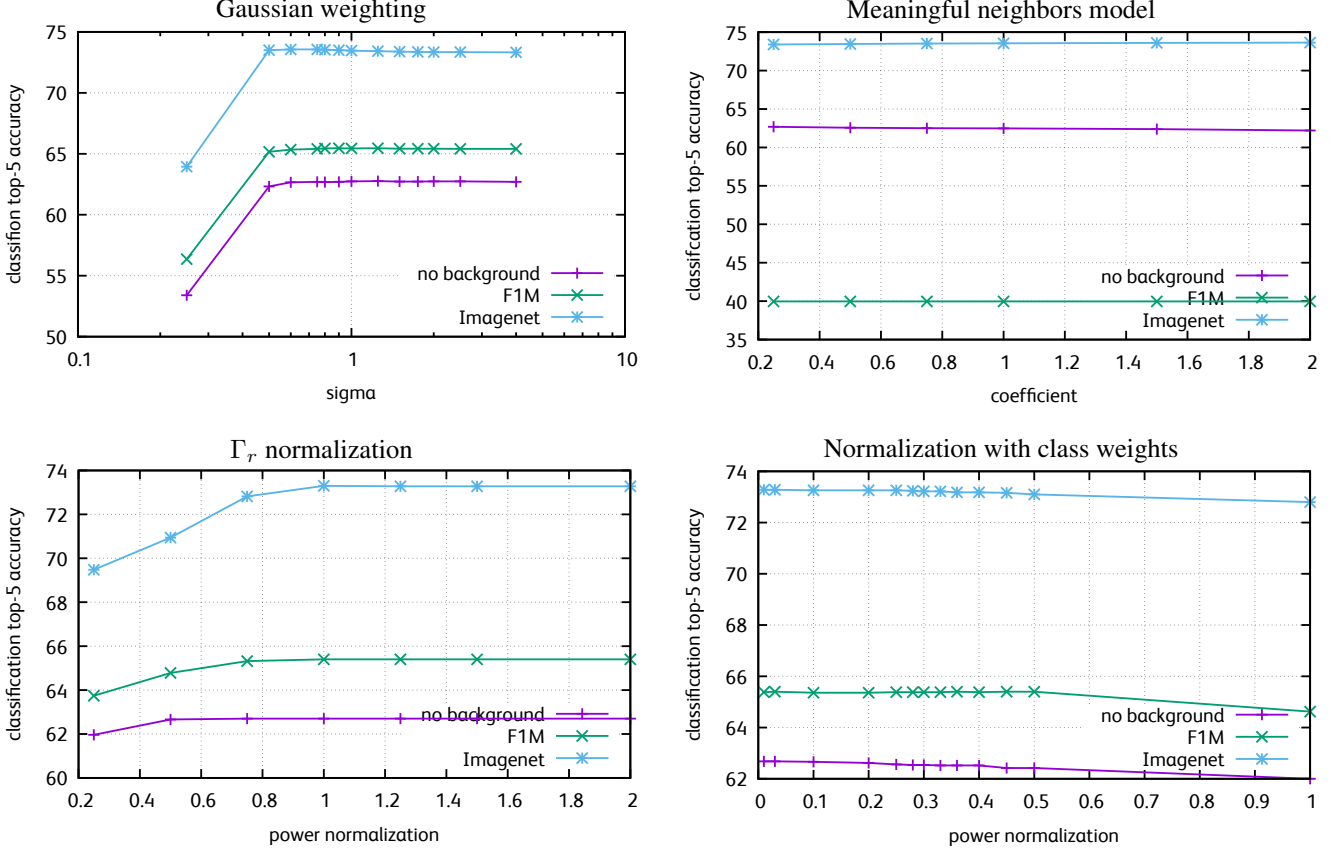

Figure 5. Evaluation of edge weighting (top) and matrix normalizations (bottom) used in the diffusion. The common settings are: $k = 30$, $n = 2$, evaluation is averaged over 5 runs on the validation set (group 1), and we select the best iteration.

of the actual neighbors are retrieved. With the recommended settings of Faiss, the complexity of one search is proportional to $d\sqrt{n_\text{B}}$, so the total complexity is $\mathcal{O}(dn_\text{B}^{1.5})$. This is super-linear with respect to n_B, but it is still relatively efficient (see our timings) and performed off-line;

- $\mathbf{W}_\text{LB} \in \{0,1\}^{n_\text{L} \times n_\text{B}}$: we re-use the same index to do n_L similarity search operations, this time using only $\mathcal{O}(dn_\text{L}\sqrt{n_\text{B}})$;

- $\mathbf{W}_\text{BL} \in \{0,1\}^{n_\text{B} \times n_\text{L}}$: we need to index on the seed image descriptors. We found that in practice, con-

structing an index on these images is at best $1.4\times$ faster than brute-force search. Therefore, we use brute-force search in this case, which if of order $\mathcal{O}(dn_\text{B}n_\text{L})$;

- $\mathbf{W}_\text{BB} \in \{0,1\}^{n_\text{L} \times n_\text{L}}$: it has a negligible complexity.

The fusion of the result lists $[\mathbf{W}_\text{LL} \quad \mathbf{W}_\text{LB}]$ and $[\mathbf{W}_\text{BL} \quad \mathbf{W}_\text{BB}]$ to get k results per row of $\mathbf{W}_0$ is done in a single pass and in a negligible amount of time. Therefore the dominant complexity is $\mathcal{O}(dn_\text{B}n_\text{L})$. A typical breakdown of the timings for F100M is (in seconds):
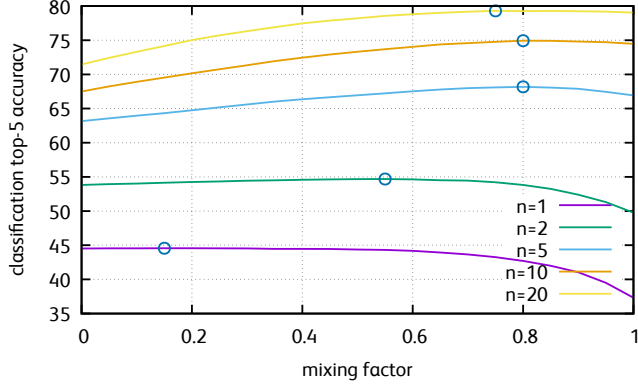
Figure 6. Performance as a function of the late fusion weight a_n on the validation (group 1) images, averaged over 5 runs. A weight of 0 is pure diffusion, 1 is pure logistic regression. The mixing factors that are selected for test are indicated with circles.

Timings (s)	$\mathbf{W}_{BB}$	$\mathbf{W}_{LB}$	$\mathbf{W}_{BL}$	$\mathbf{W}_{LL}$
on CPU	—	65+2783	12003	32
on 8 GPUs	25929	732+64	533	1

For $\mathbf{W}_{LB}$ we decompose the timing into: loading of the precomputed IVFFlat index (and moving it to GPU if appropriate) and the actual computation of the neighbors.