Active Sparse Feature Selection Using Deep Convolutional Features for Image Retrieval

Devin Conathan* Urvashi Oswal† Robert Nowak‡

1 Introduction

1.1 Content-based Image Retrieval Content-based image retrieval (CBIR) is an area of computer vision dealing with the task of finding desired images within a large corpus. For example, someone could be searching for similar images to a certain "seed" image, or searching for images with some specific content (e.g. a product or animal). CBIR refers to algorithms that approach this task using only the data within the image itself, as opposed to a more general image retrieval approach which could include metadata or natural-language descriptions in its search.

1.2 Active Learning Active learning is an area of machine learning applicable to CBIR. Active learning algorithms automatically select the most informative unlabeled training examples for human labeling so that their time is not wasted labeling easy or trivial examples. In the context of CBIR, we can apply active learning to find the most "positive" images for whatever category we are looking for within a large corpus while labeling as few images as possible.

Active learning is a computationally-intensive procedure. Generally it requires training and updating a model continuously as new labels are provided by users. Ideally, this process can happen in real time; the user should have not have to wait an excessive amount of time between each iteration, and the model should be retrained immediately upon receiving a new label to take advantage of them and yield the best performance.

The computationally-intensive nature of active learning is compounded by applying it to CBIR, since computer vision generally deals with very high-dimensional features.

1.3 Our Contribution This paper aims to tackle this problem of high-dimensional features for active learning. In general, we propose an active learning routine to find images based on a single "seed" image depicted at a high-level in Figure 1. We explore various feature-selection routines to alleviate the high-dimensionality problem by selecting a small subset of relevant features.

This setting is also closely related to stochastic generalized linear bandits, where a learner makes successive decisions to maximize cumulative reward (by finding positive examples). Mathematically, at time t given a set of arms $\mathcal{X}_t \subseteq \mathbb{R}^p$, the learner chooses an arm $x_t \in \mathcal{X}_t$ and receives a reward $y_t = \mu(x_t^T \beta^*) + \eta_t$, where $\beta^* \in \mathbb{R}^p$ is unknown and possibly sparse, $\mu : \mathbb{R} \to \mathbb{R}$ is a known non-linear mapping, η_t is zero-mean noise.

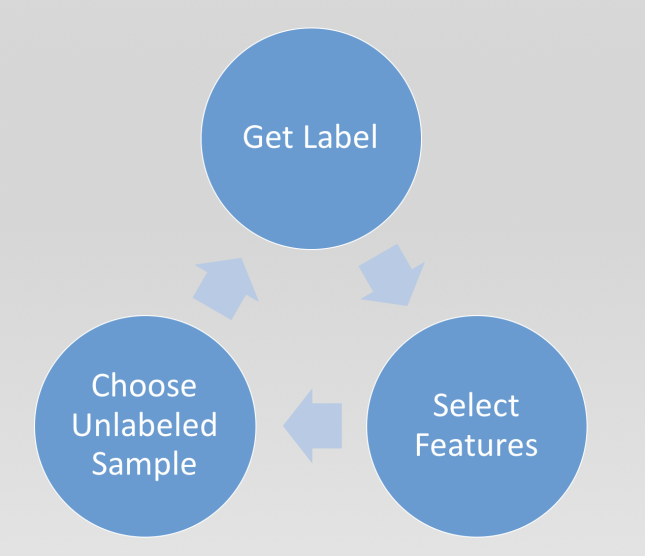

Figure 1: A high level view of the iterative active learning process.

1.4 Applications to Industry The general industry applications of CBIR are numerous. For this paper, we will focus on one pattern that is common in our experience. Often, a data scientist can easily acquire or train an image classification model for some general-purpose task, either by using a publicly available dataset or using a pretrained model. This model may not solve their exact problem, but may solve a very similar or more general version of the problem. Generally in these situations, the practitioner may employ a technique like transfer learning [8] to take this more general model and make it applicable to their domain.

However, transfer learning assumes you have a labeled dataset specific to your domain, which is not always the case. While our techniques can be applied to CBIR in general, this is the scenario we consider for this paper. Our techniques

*American Family Insurance
†University of Wisconsin-Madison
‡University of Wisconsin-Madison

help the practitioner take this general purpose model and leverage it using active learning to efficiently annotate data for their domain.

In Section 5.1 we simulate this scenario by taking a model trained on Imagenet [2] and use it to find pictures of cats in the CIFAR dataset [6]. In Section 5.3 we apply the same technique to a large corpus of roof images obtained from inspections as part of the process of obtaining home insurance. The roof images are labeled generally as "good condition" and "bad condition", and our goal is to leverage these labels to generate a dataset with labels indicating more specific defects, such as "missing shingle(s)".

2 General Procedure

We formalize the scenario described in Section 1.4 as follows:

You begin with a large corpus $\mathcal{D}$ of data, labeled for some "general" (e.g. non-domain specific) task, $\mathcal{T}_g$. This might come in the form of a publicly available labeled dataset. You have another corpus $\mathcal{D}'$ (which may be the same as $\mathcal{D}$), and you wish to acquire labels for a subset of $\mathcal{D}'$ for some more specific task, $\mathcal{T}_s$.

First, we train a convolutional neural network (CNN) model using $\mathcal{D}$ and $\mathcal{T}_g$ labels. We then treat this model as a feature extraction model, mapping $\mathcal{D} \to \mathbb{R}^p$, by treating some set of the hidden layers as outputs. Using CNN models as feature maps in this way has proven very effective for classification tasks [9] [1]] [11].

Now we can build a set of features X for our set $\mathcal{D}'$. Because of how the model was trained, these features should encode information about the data relevant to $\mathcal{T}_g$. The hope is that $\mathcal{T}_g$ is related enough to $\mathcal{T}_s$ that the features also can encode information useful to $\mathcal{T}_s$. We use active learning using features X to obtain labels for $\mathcal{T}_s$.

We detail an application of this procedure in Section 5.3.

3 Feature Sparsity

As introduced in Section 1.2 one major issue of this approach (and motivation for our techniques) is the high-dimensional features commonplace in computer vision problems, which are particularly problematic for active learning, where quick model execution times are important, and we are dealing with extremely small sets of labeled data. In our examples, we use the VGG16 architecture [7], which has two fully connected layers with 4096 neurons each. They probably contain redundant and unnecessary information.

To alleviate the problem posed by these high-dimensional features, we introduce a feature selection subroutine for each query. The goal is to select a small subset of features that are most relevant to our search to increase the effectiveness of our image selection algorithm. By selecting a smaller number of features, a bias is created towards simpler models. The simple models lead to more interpretable solutions.

For example, we have a large dataset of roof images labeled by inspectors as good condition or poor condition. In general, these labels are very high-level and lead to less interpretable models. What we actually want are models that can pick out specific defects that are easy to interpret and take actions on, like "missing roof shingle".

To achieve this, we apply two approaches for feature selection. The first method uses a logistic regression model with a sparsity regularizer called Lasso [10] that is trained on images labeled by the annotator so far. The second method is based on the simple idea of Marginal Regression [3, 4] described in the next section.

4 Algorithms

Algorithms 1 and 2 show two feature selection routines. Algorithms 3 and 4 show two sample selection routines. In practice the algorithms can be mixed and matched.

Algorithm 1: LASSO feature selection

Data: features $\boldsymbol{X}$ and labels $\boldsymbol{y}$
Result: $\widehat{S}$
$\widehat{\beta} = \arg\min_\beta L(\boldsymbol{y}, \boldsymbol{X}\beta) + \lambda\|\beta\|_1$ where $\lambda > 0$ and $L(\cdot)$ is a logistic loss
$\widehat{S} = \text{supp}(\widehat{\beta}) = \{j | \widehat{\beta}_j \neq 0, 1 \leq j \leq p\}$

Algorithm 2: Marginal regression feature selection

Data: features $\boldsymbol{X}$ and labels $\boldsymbol{y}$
Result: $\widehat{S}$
$\widehat{\alpha} = \boldsymbol{X}^T\boldsymbol{y}$
Choose threshold τ
$\widehat{S} = \{j : |\widehat{\alpha}_j| \geq \tau, 1 \leq j \leq p\}$

Algorithm 3: Nearest neighbors with feature selection

Data: features $\boldsymbol{X}$ and seed positive y_i
Result: labels $\boldsymbol{y}$
$\widehat{S}_0 =$ all features
for $t = 1, 2, \ldots, n$ **do**
  Choose random x^* with positive label
  Choose $x_t = \arg\min_{x \in X} ||s(x) - s(x^*)||$, where $s(x)$ is x with support $\widehat{S}_{t-1}$
  Play x_t and observe reward y_t
  Update $\widehat{S}_t$ according to feature selection algorithm
end

Algorithm 4: Linear bandits with feature selection

Data: features $\boldsymbol{X}$ and seed positive y_i
Result: labels $\boldsymbol{y}$
$\widehat{S}_0 =$ all features
for $t = 1, 2, \ldots, n$ **do**
  if *Lasso* **then**
    $\widehat{\beta}^t := \arg\min_\beta L(\boldsymbol{y}_t, \boldsymbol{X}_t\beta) + \lambda\|\beta\|_1$
  else
    $\widehat{\beta}^t := \arg\min_\beta L(\boldsymbol{y}_t, s(\boldsymbol{X}_t)\beta) + \lambda\|\beta\|_2$
  end
  where $\lambda > 0$, $L(\cdot)$ is logistic loss, and $s(\boldsymbol{X})$ is $\boldsymbol{X}$ with support $\widehat{S}_{t-1}$
  $x_t = \arg\max_{x \in \mathcal{X}_t} \langle s(x), \widehat{\beta}^t \rangle$
  Play x_t and observe reward y_t
  Update $\widehat{S}_t$ according to feature selection algorithm
end

4.1 LASSO The lasso [10] constrains the solutions to be sparse, meaning only a few variables are selected. This is done by using the ℓ_1 penalty which is a convex proxy to the ℓ_0 norm representing cardinality of the coefficient vector. In practice, λ is chosen via cross-validation.

4.2 Marginal Regression The marginal regression based algorithm offers a faster alternative to the Lasso regularization since it regresses the label vector separately on each feature. The marginal regression estimates for feature selection are computed using the following coefficients. Assuming the features are standardized,

$$\widehat{\alpha} := X^T y.$$

Using the tuning parameter, $\tau > 0$, we estimate the subset of relevant features by

$$\widehat{S} := \{j : |\widehat{\alpha}_j| \geq \tau\}$$

Roughly, this amounts to picking the top features that are most correlated with the reward or label vector. In practice, we choose the top $n/\log(n)$ $\hat{\alpha}$ with the largest magnitude instead of specifying a τ. We experimented with tuning τ through cross-validation but the performance difference was negligible.

4.3 Computational Complexity Performance is key for active learning algorithms, which need to be executed in near real-time to maintain a good user experience for the human labelers. One major motivation of our feature selection step is to reduce the computational complexity of the algorithm.

Marginal regression is the cheaper feature-selection algorithm since it only requires one matrix-vector multiplication: it is $O(nd)$. The LASSO requires solving a logistic regression problem and is at least $O(nd^2)$. In practice, the cost is increased by the need to tune the λ parameter via cross-validation. Our experimental results in 5 are in line with this theory.

Similarly, the linear bandit sample selection model has to fit a logistic regression model. If $\tilde{n}$ is the number of unlabeled examples and $\tilde{d}$ are the number of selected features, once the the weights are determined, an $O(\tilde{n}\tilde{d})$ matrix multiplication is necessary. The nearest neighbor complexity is $O(\tilde{n}\tilde{d})$ as well since it requires $\tilde{n}$ subtractions of size $\tilde{d}$ vectors. (all sample selection algorithms need to do a sort, so we ignore that cost here).

Note that $\tilde{n} \gg n$, so reducing the size of $\tilde{d}$ for the sample selection phase will have significant improvements to the overall computational complexity.

5 Experiments

We use the crowdsourcing, active learning platform NEXT [5] to implement and evaluate our algorithms. Linear bandit models are indicated by LB and nearest neighbor models are indicated by NN. Feature selection algorithms are indicated in parantheses.

5.1 Simulations using CIFAR In this first experiment section, we simulate the active learning process to evaluate the algorithms. Here, our goal is to find pictures in the CIFAR dataset with the label "cat". To more closely emulate our real-world scenarios, we downsampled the "cat" category to include only 500 examples (about 1% of the total images), so the dataset used for the simulations had 45500 images.

To generate the features, we used the VGG16 architecture trained on the Imagenet corpus, and ran each image in the CIFAR dataset through the network and concatenated the last two fully-connected layers to generate an 8192-dimensional embedding for each CIFAR image.

This experiment is supposed to emulate the real-world scenario, since Imagenet is trained to recognize thousands of categories, and represents the "general-purpose" task described in Section 2. Since "cat" (among other animals) is one of the categories in Imagenet, the embeddings should encode discrimative features about cats, though we would expect there to be much unecessary information in the features related to discriminating between any of the other categories.

5.2 Simulation Results The plot of number of positives vs number of queries is shown in Figure 2. In the long run, the linear model using no feature selection is the clear winner. Table 1 shows the average number of features and execution times for each algorithm. We see that using all of the features is the most computationally expensive to execute. On average, it took about 15.8 seconds to train the linear

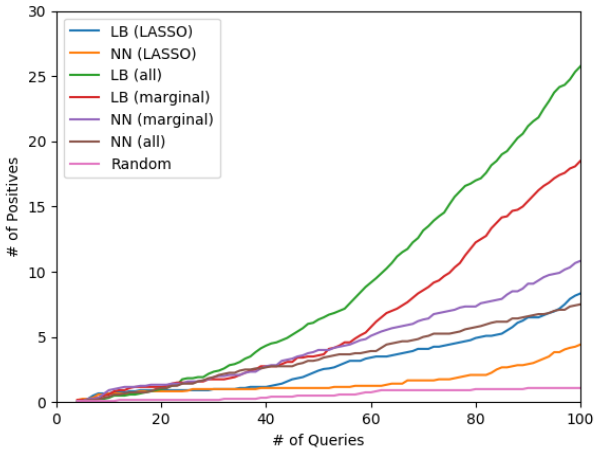

Figure 2: Number of positives vs number of queries for the CIFAR experiment (averaged over 10 simulations).

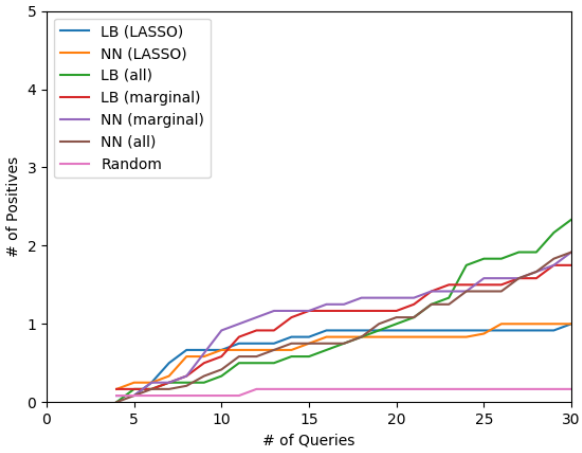

Figure 3: Detail of the first 30 queries for the CIFAR experiment (averaged over 10 simulations).

Model	d	executions	time/execution
NN (all)	8192	2.85	17.4
NN (LASSO)	9.1	23.0	0.85
NN (marginal)	29.3	**29.8**	**0.48**
LB (all)	8192	6.8	15.8
LB (marginal)	26.8	26.3	0.54
LB (LASSO)	134.5	14.5	3.9

Table 1: Number of features selected (d), number of iterations executed, and average execution time for the first 100 queries for each algorithm in the CIFAR experiment.

Figure 4: An example image from a roof survey that would get a positive label for the "missing shingle" label in our experiment.

model and select the next example. Since it is unreasonable to expect a user to wait that long to label each example, we maintain a queue of examples to choose (as ranked by the algorithm) and update the queue whenever the algorithm finishes. This means it takes on average 15.8 seconds for each new label to be reflected by the model. If we examine the results in more detail (as plotted in Figure 3), we see on average that the less computationally-expensive algorithms are in general more performant in the first 30 queries.

5.3 Missing Shingles Search Here we take the algorithms and apply them to solve a problem encountered in the insurance industry. American Family Insurance has a corpus of about 400,000 images of roofs obtained from roof surveys as part of the home inspection process. As part of the survey, the roofs are graded by experts on a scale from 1 to 4, indicating condition. Using this survey, we are able to train a deep learning model (using the VGG16 architecture) to predict which roofs have "good" vs. "poor" condition (we reduce it a classification problem because a regression on the survey results was deemed too difficult a problem).

We would like to leverage this general, high-level assessment to build a more granular corpus of images with more specific labels. In this experiment, we wish to find all the images that have the quality "missing shingle(s)". Figure 4 shows an example image with that quality. Since this condition is rare (in our experiments, about 2% of the images in the "poor" condition had a missing shingle), we treat this as a CBIR problem and use active learning to most efficiently use human labeling resources. As described in Section 2, we treat the model trained from the good/poor labels as a feature extractor, treating the last two fully-connected layers of the network as an embedding for the images. We then use these features for the active learning process.

5.4 Results We see similar results to our CIFAR example, though we get more variation because this is a more difficult problem. The number-of-positives plot is given in Figure 5 and the execution times are given in Table 2. It is notable that the computational costs are significantly more extreme

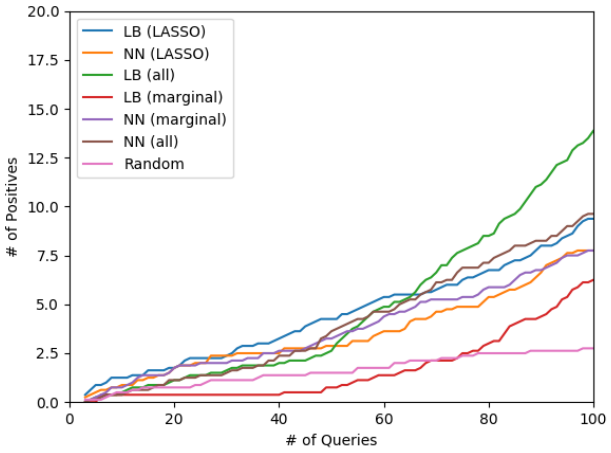

Figure 5: Number of positives vs number of queries for the live, roof images experiment (averaged over 8 experiments).

Model	d	executions	time/execution
NN (all)*	8192	21.4	4.1
NN (LASSO)	18.5	28.5	1.2
NN (marginal)	29.9	**40.4**	1.0
LB (all)	8192	5.3	148.9
LB (marginal)	27.2	38.0	**0.65**
LB (LASSO)	631.2	25.6	14.6

Table 2: Number of features selected (d), number of iterations executed, and average execution time for the first 100 queries for each algorithm in the live, roof images experiment. *Because of out-of-memory errors, we had to downsample the unlabeled pool each iteration, hence the quicker execution times.

in this example due to the size of the corpus. Again, in the long run the linear model performs the best. However, we see that approaches that use a smaller subset of features (the LASSO linear model in particular) can be more performant when the number of queries is small. For reasons unknown, the linear model with marginal regression-derived features was *worse* than random for the first 70 queries.

6 Conclusion and Future Work

We have introduced some feature-selection routines to be applied to active learning in the context of CBIR. Theory and experimentation show that the computational cost of active learning can be greatly reduced by dynamically selecting relevant subsets of features. We consider this preliminary work exploring the possibilities of sparse feature selection for high-dimensional active learning. The fact that the linear model with all features is still so performant even with the slow execution times suggests that more time should be spent in optimizing the execution of that model, or perhaps explore downsampling techniques. However, we believe there is merit in sparse techniques, particularly when the number of labels is small. The optimal algorithm is probably some hybrid algorithm that dynamically changes feature selection routines based on how many labels are present, but significantly more research is necessary before such an optimal algorithm could be determined.

Another unanswered question is why linear model with marginal features performed so poorly in our roof experiments, while it was the second most performant algorithm in the simulations. This is perhaps because our thresholding strategy was "overfitting" the simulation and does not generalize well. We would like to explore better methods for thresholding which features get selected via marginal regression.

References

[1] B. Athiwaratkun and K. Kang, *Feature representation in convolutional neural networks*, arXiv preprint arXiv:1507.02313, (2015).

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, in CVPR09, 2009.

[3] C. Genovese, J. Jin, and L. Wasserman, *Revisiting marginal regression*, arXiv preprint arXiv:0911.4080, (2009).

[4] C. R. Genovese, J. Jin, L. Wasserman, and Z. Yao, *A comparison of the lasso and marginal regression*, Journal of Machine Learning Research, 13 (2012), pp. 2107–2143.

[5] K. G. Jamieson, L. Jain, C. Fernandez, N. J. Glattard, and R. Nowak, *Next: A system for real-world development, evaluation, and application of active learning*, in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds., Curran Associates, Inc., 2015, pp. 2656–2664.

[6] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, Master's thesis, Department of Computer Science, University of Toronto, (2009).

[7] S. Liu and W. Deng, *Very deep convolutional neural network based image classification using small training sample size*, in 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Nov 2015, pp. 730–734.

[8] S. J. Pan and Q. Yang, *A Survey on Transfer Learning*, IEEE Transactions on Knowledge and Data Engineering, 22 (2010), pp. 1345–1359, https://doi.org/10.1109/TKDE.2009.191.

[9] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, *CNN features off-the-shelf: an astounding baseline for recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2014, pp. 806–813.

[10] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological), 58 (1996), pp. 267–288.

[11] L. ZHENG, Y. ZHAO, S. WANG, J. WANG, AND Q. TIAN, *Good practice in CNN feature transfer*, arXiv preprint arXiv:1604.00133, (2016).