Feature Extraction for Semi-Supervised Learning using Convolutional Neural Networks

The Unit Porcupines

Omid Askari
omid55@cs.ucsb.edu

Haraldur Tómas Hallgrímsson
hth@cs.ucsb.edu

Adam Ibrahim
ai@cs.ucsb.edu

Anastasiya Lazareva
anastasiya@cs.ucsb.edu

***Abstract*—We investigate how well a trained state of the art convolutional neural network (CNN) can be used for feature extraction on novel datasets and tasks. We quantitatively measure the goodness of the extracted features with respect to different tasks and by depth of the output layer. We use Google's Inception-v3, a CNN trained on the 2012 ImageNet object dataset containing a thousand classes of objects, and use the extracted features on tasks distinct in nature, such as recognizing textures or emotions evoked by an image, on various labeled datasets.**

We extract features from various layers of the CNN and show that, for tasks similar to the one the deep learning model was trained on, deeper layers generalize better. For more disparate tasks, the generalizability of the layers is shown to be not so straightforward.

***Index Terms*—Feature extraction, Convolutional Neural Networks, semi-supervised learning.**

I. Introduction

The mass availability of data and increased computational power over the last years has led to a sudden surge of interest in the field of neural networks and deep learning. Very accurate classifiers can be trained efficiently which seem to generalize extremely well by virtue of having been trained on more observed data than any human could reasonably be expected to digest.

However, not all tasks have enough labeled data to justify training a deep network on. Recent work has examined the question of *feature transferability* and recycling learned models to use the glut of training data in one field to achieve better results in a related field [1]. The idea stems from the observation that the first several layers of neurons in a convolutional neural network are often similar to very generalized feature extractors such as edge detectors. Obviously the output of the very last layer of a trained neural network is highly specialized to only identify which of the trained classes is most similar to a given input, so there is some gradient from the first to last layers from identifying general features to identifying specialized features.

Recent work in understanding what normalized inputs lead to high activations of given neurons in a neural network has led to the insight that deep layers can capture very semantically rich details in the input which are then used to ultimately classify the input [2]. Recycling these highly trained neurons and the structure of the network leading to them should then lessen the requirement of labeled data to train on when moving to new but related fields.

In this project, we recycle a state-of-the-art deep learning model as a feature extractor to identify the structure and relationships within new datasets with similar inputs but labeled with respect to very different tasks than what the model was trained on. This method allows for an unsupervised discovery of the structure of a novel dataset or a semi-supervised labeling of the dataset by propagating the labels of a small subset of inputs to identify probable labels of an unlabeled subset. For the purposes of the experiments to follow, we require the novel datasets to be labeled to measure the effectiveness of the method but this is not a requirement of the method.

This is a two step approach where first a high performing model is trained using a glut of information available in a specific field, such as various image classification contests such as ImageNet, to then use as a feature extractor in a domain with less or even no information to train on.

The advantage of this approach is that we can use trained, highly performing classifiers that work well on general features of the dataset, e.g. best performing classifiers on the ImageNet dataset, to then cluster new datasets containing classes that may not overlap with those it was trained on.

Specifically, we use Inception-v3 [3], which was trained on the 2012 ImageNet Large Visual Recognition Challenge. Inception-v3 has a 3.46% top-5 error rate - the best so far - and is implemented in Google's TensorFlow. The schematic diagram of the architecture of this model is given in Fig. 1. In order to evaluate how effective we are at identifying the structure of the novel dataset we require that it be labeled, but this is not a requirement of the method. We use both a simple synthetic dataset of shapes as well as datasets that have been labeled with respect to tasks that are similar or very different from the object recognition one in the ImageNet competition, such as recognizing handwritten numbers or emotions evoked by an image.

We seek to provide answers to the following questions:

1) Can trained CNNs detect similar features in classes that were not included in the original training set
2) Which layers of the CNN produce features that lead to better clustering of novel data.

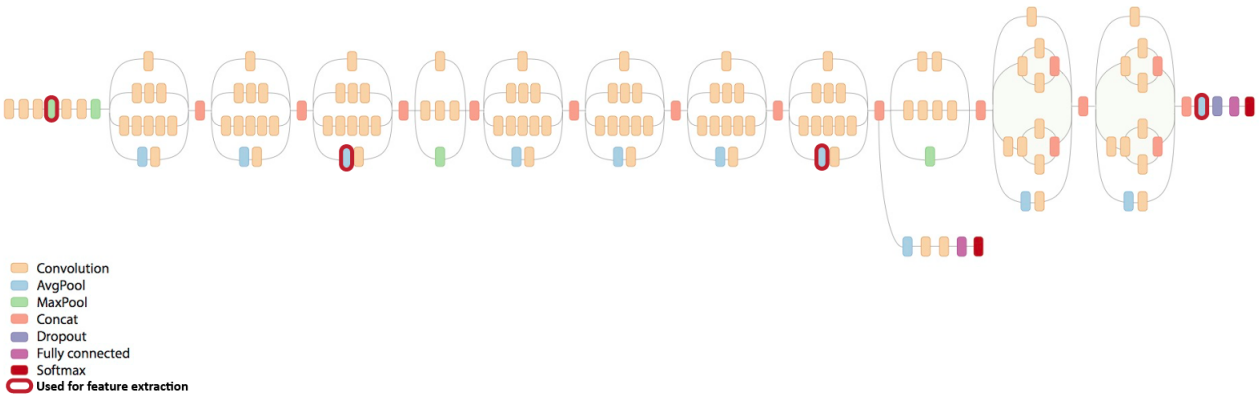

Fig. 1. Schematic visualisation of the Inception v3 neural network with pooling layers used as feature extractors circled in red

Name	Link	Samples	Classes
ImageNet	image-net.org/	14M	1000
Synthetic	N/A	800	4
MNIST	http://yann.lecun.com/exdb/mnist/	60k	10
Affective Image Classification	imageemotion.org	1429	8
Kylberg Texture Dataset 1.0 [4]	http://www.cb.uu.se/ gustaf/texture/	4480	28

TABLE I
THE CNN WAS TRAINED ON IMAGENET. CLUSTERING IS PERFORMED ON THE OTHER DATASETS IN THE TABLE.

Fig. 2. Synthetic objects

II. DATASETS

Four datasets labeled with respect to different tasks were used to compare the effectiveness of using various layers of the trained CNN as a feature extractor. These tasks ranged from similar to the task the CNN was trained on, such as identifying handwritten numbers or simple synthetic shapes, to recognizing textures or emotions evoked by an image.

A. Synthetic Dataset

A synthetic dataset was generated with four simple objects shown in Fig. 2 with 200 images per class. The rotation and size of each object was randomized in each image. The generated images were 299x299 pixels in size and were input in JPEG format.

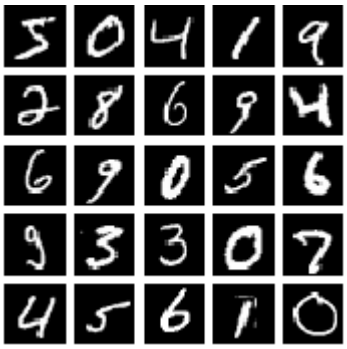

Fig. 3. Examples of MNIST images

B. MNIST

The MNIST database [5] consists of 60 000 handwritten digits and 10 classes extracted from NIST's Special Databases 1 and 3 as black and white images that were normalised to fit 20x20 pixel boxes while preserving the aspect ratio. The images are translated so that the center of mass of the pixels is centered in a 28x28 pixel box. The anti-aliasing in the normalisation algorithm introduces grey levels, making the MNIST images greyscale. We used a subset of 10 000 images (converted to JPEG format).

C. Affective Image Classification

This dataset is comprised of three smaller datasets [6]: the International Affective Picture System (IAPS) which contains 394 natural color photos and depicts scenes ranging from portraits to accidents; 807 artistic photographs each conveying an emotion through a specific choice of image composition, colours, lighting, and so on; and 228 abstract paintings which do not depict any object in particular. The point of the last dataset lies in the fact that while certain objects evoke specific emotions, emotions may arise in textures, or from the use of certain colours, so that emotions should not simply be attributes of objects. The ground truth labels of these

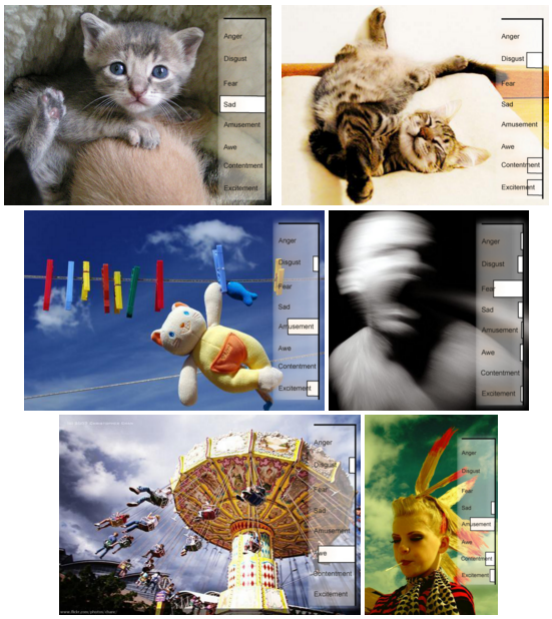

Fig. 4. Examples of images from the Affective Image dataset

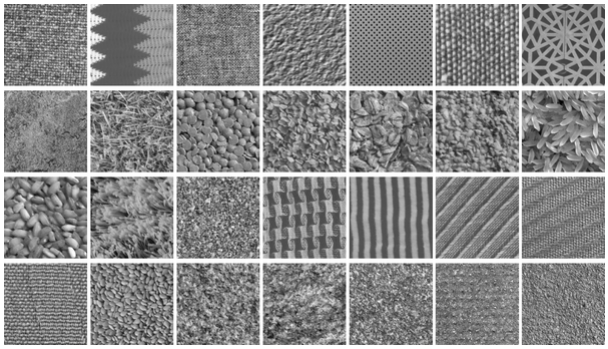

Fig. 5. Example patches from each of the classes of the Kylberg dataset

datasets were obtained from an independent study, by the artists themselves, or by a peer rated web-survey, respectively.

D. Kylberg Texture Dataset

We use version 1.0 of the Kylberg Texture Dataset [4] as a benchmark to evaluate how well the feature extractors generalize to the notoriously hard task of texture classification. The dataset contains 28 texture classes and 160 unique texture patches per class, with size 576x576 pixels. The images were converted to JPEG format.

III. Methodology

In order to test the performance of CNNs as feature extractors we perform the following experiment. Pooling layers from various depths of Inception v3 are selected as feature extractors as these layers by design minimize the size of feature representation. The layers selected can be seen in Fig. 1. The dimensionality of the features progressively decreases for deeper layers in the network. The first pooling layer produces

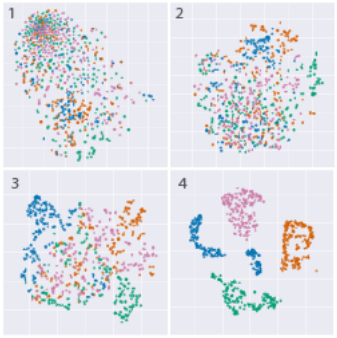

Fig. 6. t-SNE for the synthetic dataset for each of the four pooling layers

341 056 features, the second produces 83 232, pooling layer 3 produces 49 152, and the final layer produces 2 048 features.

Images that have been labeled according to different tasks as described in section II are used as inputs to these feature extractors which map each input to a high-dimensional space. We consider the feature extractor good if images that have been labeled similarly end up close to each other in this space as compared to images that do not have the same label. To this end, we compute pairwise distances between each input image for a given task (using the L^2 norm) and compare how close pairs of images with the same label are compared to pairs with different labels,

$$\frac{\text{Between-class distance}}{\text{Within-class distance}} \tag{1}$$

This measure will be greater than one for a good clustering of labeled images, otherwise less than or equal to one. There is a choice of measure to aggregate the distances for the two populations; we simply choose to compute the average distances for both.

To measure the significance of the clustering goodness measure with regards to the task we assume the null hypothesis that the labels were randomly distributed and apply a permutation test for each dataset and feature extraction layer.

We use the t-SNE method [7] to show extracted features in two dimensional space for qualitative assessment of clustering goodness. It is a nonlinear dimensionality reduction technique which minimizes the difference between the distance of samples in their original high dimensional space and a two or three dimensional space.

IV. Results

In three of the four datasets—the synthetic, MNIST digits, and texture datasets—we see the clustering goodness measure peaking at the last pooling layer selected as a feature extractor. For both the synthetic and texture datasets the measure increases monotonically with deeper layers while the MNIST digit dataset performs better at the earliest pooling layer compared to the second. In the case of the task most dissimilar to the object recognition task the deep learning model was trained on, the Affective Image Classification, we see more complex dynamics at play. This task had the earliest

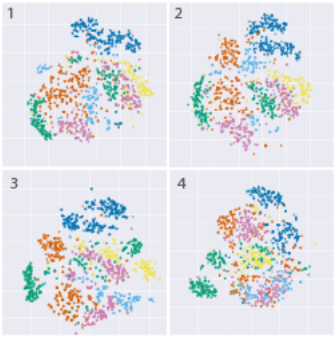

Fig. 7. t-SNE for the MNIST dataset for each of the four pooling layers

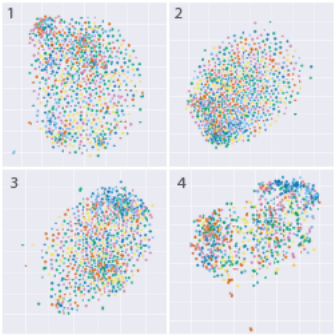

Fig. 8. t-SNE for the Emotions dataset for each of the four pooling layers

pooling layer generalizing the best, but with by far the lowest clustering goodness measure of the four datasets.

We measured the significance of the clustering for each dataset and for each layer via permutation testing. These results are shown in figures 10 through 13. In each case we measure the significance as $p \leq 0.001$ and reject the null hypothesis that the clusters are not meaningful with respect to the task-specific labels.

Qualitatively, the clustering goodness can be observed in figures figures 6, 7, 8, and 9. These figures projected the high dimensional featurs into 2D space using t-SNE.

V. Conclusion

We've shown the feasibility of recycling a state of the art deep learning model that has been trained on a very broad task

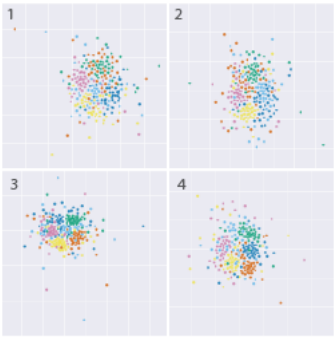

Fig. 9. t-SNE for the Textures dataset for each of the four pooling layers

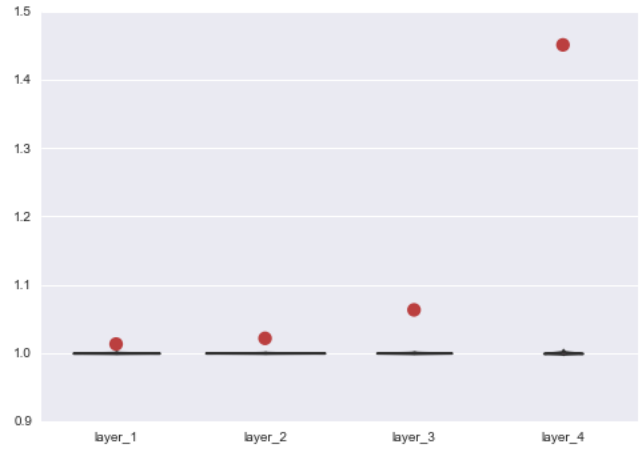

Fig. 10. Clustering goodness measure versus depth of different pooling layers in Synthetic dataset. Violin plots show distribution of values obtained via permutation testing, red dot shows value of true permutation of labels.

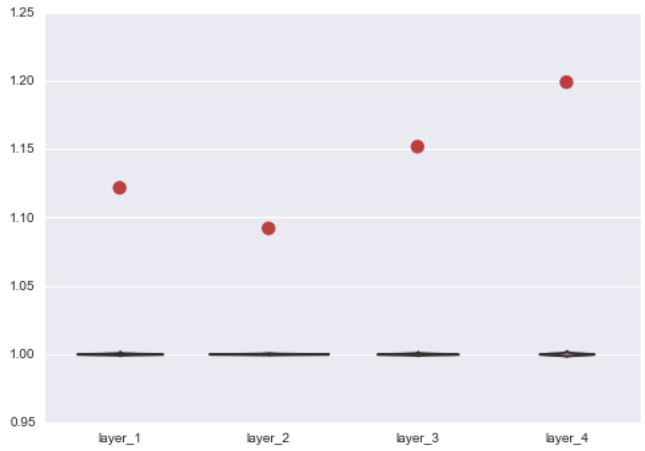

Fig. 11. Clustering goodness measure versus depth of different pooling layers in MNIST dataset. Violin plots show distribution of values obtained via permutation testing, red dot shows value of true permutation of labels.

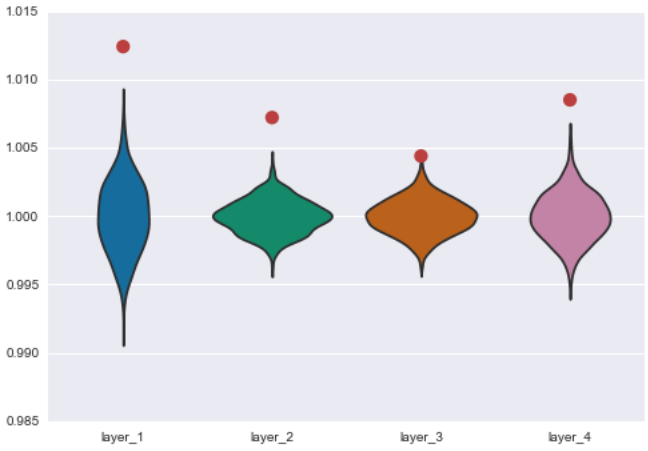

Fig. 12. Clustering goodness measure versus depth of different pooling layers in Affective Image dataset. Violin plots show distribution of values obtained via permutation testing, red dot shows value of true permutation of labels.

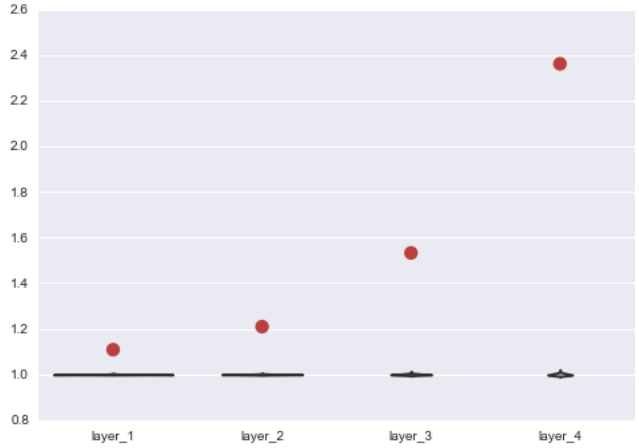

Fig. 13. Clustering goodness measure versus depth of different pooling layers in Texture dataset. Violin plots show distribution of values obtained via permutation testing, red dot shows value of true permutation of labels.

domain of recognizing and discriminating between a thousand different classes. This model can be re-purposed to extract features that are useful for tasks that have similar inputs but are otherwise highly different in nature to the original task, such as identifying the emotions evoked in a viewer of a given image. We've shown that for tasks that are similar to the one the model was trained on highly abstract features from deep in the model generalize the best, but for more dissimilar tasks there is need for careful choice of depth of the layer to act as a feature extractor.

There are many avenues still open in exploring the idea of feature transferability. Here, a state of the art deep learning model was used to extract features in novel datasets. Alternatively, the output of the encoding stage of an autoencoder could be used; an autoencoder being a neural network whose output is as similar as possible to its input composed of two stages: an encoder and a decoder. The encoder maps the input to a smaller size than the input which the decoder can then use to recreate the input as close as possible again. The output of the encoder is then a semantically rich and information dense representation of the input which generalizes well to the inputs give. Using the output of the encoder would allow for a fully unsupervised pipeline of the entire method and skips the need for a trained deep learning model

REFERENCES

[1] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," pp. 3320–3328, 2014.

[2] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015.

[4] G. Kylberg, "Kylberg texture dataset v. 1.0," 2011.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] J. Machajdik and A. Hanbury, "Affective image classification using features inspired by psychology and art theory," in *Proceedings of the international conference on Multimedia*, pp. 83–92, ACM, 2010.

[7] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.