Group 4: Object Recognition

Varsha Nagarajan
North Carolina State University
vnagara2@ncsu.edu

Kevin Alvarez
North Carolina State University
kalvare3@ncsu.edu

Darshil Patel
North Carolina State University
dpatel14@ncsu.edu

Ashwin Risbood
North Carolina State University
arisboo@ncsu.edu

1 Introduction

Computer vision and image processing together form a widely researched and applied domain. In this paper, we focus on object recognition, which is a subset of this domain, tasked with identifying which object category is the best match for an image. With novel algorithms and striking results on large datasets, the problem of object recognition continues to be a challenging one as even the best algorithms cannot holistically sum up the complexities involved in this task. Computational bottleneck and dearth of large amounts of labeled data is what makes this problem all the more challenging. We seek to find solutions to this problem by proceeding with devising both supervised and unsupervised learning methods that could effectively approximate the solution and provide improved image classification.

1.1 Supervised Object Recognition

Deep convolutional neural networks (CNN) have led to an immense breakthrough in the field of object recognition. These deep network architectures integrate both high and low level features in a multi-layer fashion, with each layer enriching the features learned by the previous layer. What makes them so successful in the field of computer vision is their explicit assumption that all inputs have image-like structures. This allows them to encode certain properties into the architecture by taking advantage of the local spatial coherence of data that suits most for image inputs. Recent works [1, 2] reveal that the depth of these networks is crucial and many promising results on challenging datasets [1, 2, 3, 4] have exploited these deep network architectures. With advancements in the computational strength and prevalence of sophisticated systems and more optimized algorithms, it is now possible to train deeper networks on larger datasets.

One question that the authors of the paper [5] seek to answer is whether learning better networks have direct correlation to stacking more layers. With deeper networks, there arises a new problem of degrading accuracy [5] with increase in network depth. This essentially means that, with increasing depth, the deeper layers find it difficult to approximate the desired mapping, resulting in increase in training error. The paper proposes the idea of skip connections based on the premise that it is easier for layers to learn an identity mapping than learning a new function altogether. These shortcuts provide the later layers with residual mapping that they have to approximate. Thus, if added layers can be constructed as identity mappings, then the training accuracy of a deeper network cannot be worse than a shallow network. In other words, these extra layers can either learn something informative or, in the worst case, will approximate the identity mapping provided by the skip connection, thus solving the degrading problem.

Though we solved the degradation problem, the issue of CNNs not scaling well with increase in the number of categories remains open. As datasets become bigger and the number of object categories increases, the visual separability between different object categories is highly uneven. Hence, one other challenge to the task of object recognition is the identification of these fine distinctions between

similar object categories. For example, it is easy to distinguish between an apple and a car, but distinguishing a car from a truck is potentially difficult. While this can be a trivial task for humans, training a neural network to make such classifications is a very challenging task. This makes us question whether a flat structure is adequate for distinguishing all the difficult categories. Now, from the above example, we can say that *apple* belongs to the coarse category of *fruits* and *bus* and *car* belong to the coarse category of vehicles. An intuitive solution would be to embed this hierarchy into the network and train different classifiers to identify such coarse and fine categories in the data. For achieving this, we implement a hierarchical deep CNN model which is discussed later in the paper.

For the proposed supervised models in this paper, we perform a comparative study of different model architectures designed by employing ideas and intuitions gained from the literature discussed above.

1.2 Unsupervised Object Recognition: Clustering

Cluster analysis is a technique that seeks to sort data into similar groups, without using knowledge of true group membership [6]. Clustering has found important applications in a wide range of fields, including engineering, computer science, life and medical sciences, earth sciences, social sciences and economics [7].

When humans attempt to group objects into similar or meaningful groups, they must first describe these objects in terms of their features. As one might expect, a good set of features for describing an object is crucial to being able to group them meaningfully, and this is no different with cluster analysis. Ideal features are ones that are of use in distinguishing patterns that can be used to separate input data into meaningful clusters, and are resilient to noise in the data [7]. This second point is especially important in the context of image clustering, since images are inherently very noisy. In fact, one of the largest obstacles in object recognition tasks is maintaining an invariance to pose, lighting and surrounding clutter [8]. Another inherent property of image data is high dimensionality. An effective method of image clustering must also deal with this, as most clustering algorithms are not sufficient for analyzing high-dimensional data [7]. Recently, two methods of extracting a lower dimensional set of meaningful and discriminative features have been shown to work well with images; namely the Convolutional Auto-Encoder and t-Distributed Stochastic Neighbor Embedding.

In this paper, we apply each of these approaches to the Caltech256 dataset in order to evaluate and compare the effectiveness of each in the task of learning object categories without supervision.

2 Background

2.1 Deep Convolutional Neural Network

The main challenge in constructing an effective deep ConvNet is to identify the optimal depth of the network and to enumerate the objective of the intermediate layers. We start off by first defining the most important layers of our architecture along with the problem we intend to solve using those layers.

Neuron Activation Function Earlier, the standard way to model a nueron's output was using a tanh activation function $f(x) = (1 + e^{-x})^{-1}$. It has been studied [1] that when it comes to the training time with gradient descent, saturating non-linearities like tanh functions are much slower than non-saturating non-linearities like ReLU [$f(x) = max(0, x)$]. Hence, in our settings, we will be working with ReLU non-linearity function.

Pooling Layer These layers generally follow the convolutional layers for performing downsampling. The function of such layers is to progressively reduce the spatial size of the representation in order to reduce the amount of parameters and computational expense in the network, and thus also control overfitting. The most popular form is a pooling layer with filters/kernels of size 2x2 applied with a stride of 2 which results in discarding 75% of the activations.

Dropout One approach to avoid overfitting is to use dropout layers which set certain outputs of the hidden layers to zero as dictated by the dropout rate. These dropped out neurons do not contribute to forward or backward propagation. Now that a neuron cannot completely depend on the presence of other particular neurons, each of them is forced to learn more robust features.

2.2 Convolutional Auto-Encoder (CAE)

A Convolutional Auto-Encoder (CAE) is a hierarchical unserpervised feature extractor that scales well to high-dimensional inputs, while preserving only essential aspects of the data to form discriminative representations [9]. A general Auto-Encoder (AE) is a neural network which takes an input and transforms it into, typically, a lower dimensional representation, and then attempts to predict the original input from this representation. Through this process, the AE is learning a new representation of a chosen size for the input, which is what makes it useful for feature extraction and dimensionality reduction. Additional hidden layers can be added to this model to better model highly non-linear dependencies in the input [9]. Since CNNs are one of the more successful models for supervised image classification [10, 11], it makes intuitive sense that a CAE would perform well at feature extraction for images, since they utilize the same methodologies that make CNNs so successful. Similar to how hidden layers can be added to a conventional AE, the same can be done to a CAE in order to form a deeper hierarchy for extracting hierarchical features. As suggested in prior literature [9], after training this deep CAE, the top level activations can be used as feature vectors for classifiers. In this paper we instead use these extracted feature vectors as input into unsupervised clustering algorithms.

2.3 t-Distributed Stochastic Neighbor Embedding

The t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm, a variation of Stochastic Neighbor Embedding, is a technique for dimensionality reduction of high-dimensional data. The technique creates a single map that reveals the complete structure at multiple different scales and hence is useful for high dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects of multiple classes seen from multiple viewpoints. [12] t-SNE uses a symmetrized version of the SNE cost function with comparatively simpler gradients and uses a t-distribution rather than a Guassian distribution to calculate the similarity between two points in the low dimensional space. t-SNE learns to reduce the dimensionality of the data in such a way that the local structure of the data is preserved as well as possible in the latent space [13].

3 Methods

3.1 Plain Classifier

As our first method, we design a basic classifier inspired by the ideas enumerated in [1, 5, 14] that follow one simple rule: if the feature map size is doubled, the number of filters shall be halved and vice-versa. Our proposed architecture consists of 7 convolutional layers and 2 fully connected layers, totaling to 25,869,632 parameters. In order to progressively work with lower dimensionality, we included a couple of max-pooling layers. For accelerating learning and to avoid the problem of vanishing/exploding gradients, we employ batch normalization [3] immediately after each convolutional layer before passing it through the activation function. Extending the same philosophy as [5], if the feature map size in a layer is halved, we compensate it by doubling the number of filters, thus ensuring that same computational time is required for each layer. While dealing with deeper networks, we might come across the problem of potential overfitting and hence a dropout fraction was added to handle such cases. To determine the best working model, we build various models with different filter sizes, layers and dropouts along with various optimizers with differing configurations, thus tuning the required hyper-parameters and then use top-1 and top-5 validation accuracy to select the optimal model. As mentioned previously, our architecture uses ReLU non-linearity as the activation function in all layers and ends with a 256-way fully-connected layer with softmax.

3.2 OurResNet50

Fine tuned ResNet50 model trained on ImageNet did not give satisfactory results for our dataset. So we proposed a new architecture, very similar to the original ResNet50 model [5], with one small modification. In the original model, the shortcut or skip connections skip 2 convolutional layers, whereas in our proposed model, the residual mappings are obtained by skipping over 3 layers. Our intuition behind this change was to contain the degradation problem by feeding the residuals deeper into the network.

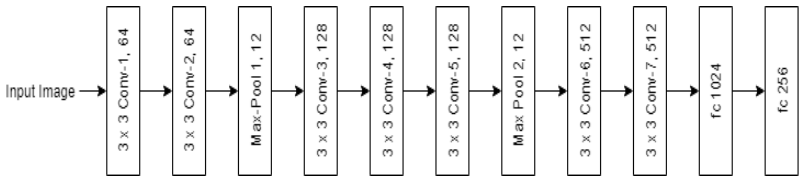

Figure 1: Plain Classifier Architecture

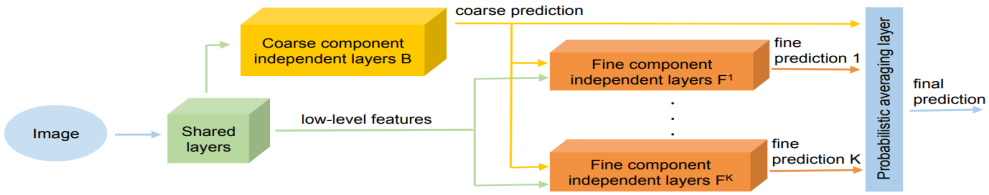

Figure 2: Hierarchical Deep Convolutional Neural Network (HD-CNN) architecture (Source: Image is taken from the paper [15] that first proposed this architecture).

3.3 Hierarchical Deep CNN (HD-CNN) Model

As mentioned previously, the task of identifying fine distinctions among similar categories is very challenging and one natural intuition to solve this could be to leverage the inherent hierarchy in the data and embed this into our model. For this purpose, we implement a HD-CNN that consists of shared layers, a single coarse category component B, multiple fine category components F and a single probabilistic averaging layer. HD-CNNs are viewed to be more scalable in comparison to traditional CNNs as the individual fine classifiers can be trained in parallel, hence reducing the overall training time. Additionally, when a test image is input to the network, only certain portions of the network are conditionally executed thus giving us performance gains.

Spectral Clustering While CIFAR-100 dataset comes with predefined coarse category labels against images, such is not the case with Caltech256 dataset. In order to identify coarse (high-level category aggregation) categories in this dataset, we use spectral clustering which makes use of the spectrum (eigenvalues) of the similarity matrix of the data and performs Laplacian Eigenmap dimensionality reduction before clustering in fewer dimensions. The plain classifier detailed in Figure 1 was used to learn the hierarchy in our data. The intuition behind designing the distance or similarity matrix as shown in Figure 4 was to identify categories that were consistently predicted as some other category. This indicates that these commonly misclassified categories must be very similar thus making it very difficult for traditional CNNs to distinguish between them. We cluster all such categories into one coarse category using spectral clustering.

The main tools for spectral clustering are graph Laplacian matrices. For our method, we used an unnormalized graph Laplacian matrix defined as:

$$L = D - W \tag{1}$$

where D is the distance matrix and W is the weight matrix.

Using the confusion matrix (T) obtained from the classification performed by plain classifier in Section 3.1, we compute the distance matrix as:

$$D = \frac{1}{2} \times [(I - T)(I - T)^T] \tag{2}$$

Overlapping Coarse Categories: If we identify just disjoint coarse categories, the overall classification will heavily depend on the correctness of coarse category classifier and the clusters produced by spectral clustering. Now, if an image is fed to an incorrect fine category classifier, then the mistake can not be corrected as the probability of ground truth label is implicitly set to zero there. Hence, we add more fine categories to these coarse classifiers using the same metrics as defined in the paper [15].

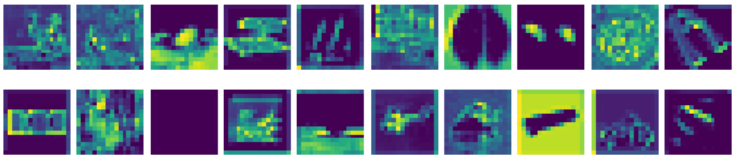

Figure 3: Random subset of feature maps from the encoded representation of the entire dataset. These are the types of features that are used as input to k-means.

Shared Layers Both coarse category component and fine category components share common layers. This is because preceding layers in deep networks learn low-level features like corners and edges. Since low-level features are useful for both coarse and fine classification tasks, the preceding layers are shared by both coarse and fine components.

Coarse Category Classifier A coarse category classifier reduces the fine predictions into coarse predictions using a mapping $P : [1, C] \rightarrow [1, K]$, where C is the number of fine classifiers and K is the number of coarse classifiers.

Fine Category Classifier Every coarse category classifier will have a corresponding fine-tuned fine category classifier which will be trained on all images belonging to the fine categories identified within that coarse category. As each fine component only excels in classifying a small set of categories, they produce a fine prediction over a partial set of categories. The probabilities of other fine categories absent in the partial set are implicitly set to zero.

Probabilistic Averaging Layer The probabilistic averaging layer receives both the fine and coarse category predictions and produces a final prediction based on weighted average. The formula is:

$$p(x_i) = \frac{\sum_{k=1}^{K} B_{ik} p_k(x_i)}{\sum_{k=1}^{K} B_{ik}} \tag{3}$$

where B_{ik} is the probability of coarse category k for the image x_i predicted by the coarse category component B. $p_k(x_i)$ is the fine category prediction made by the fine category component F^k [15].

3.4 Convolutional Auto-Encoder

As with any deep learning based approach, careful consideration must be given in choosing an appropriate network architecture. We choose this architecture by pulling from prior literature, and from additional empirical experimentation. First, we decide to include max-pooling layers in our model in order to distill translation-invariant representations [16]. In fact, for CAEs, interesting and biologically plausible filters have been found to only emerge when the network structure contains these max-pooling layers [9]. As in our supervised CNN, we use the rectified linear unit (ReLU) activation function. To determine the depth of the model and other hyper-parameters such as the size of the convolutional kernels, we build several models with various hyper-parameter choices and compare the mean squared error (MSE) between the input images and their reconstructions for each model. After choosing the model with the lowest MSE, we use this model to encode each image in our data set, as shown in Figure 3 and use k-means to cluster these encodings, using a k value of 256. We then evaluate the purity and entropy of the clustering results with respect to the ground truth category labels.

3.5 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Since t-SNE can require a long runtime when working with very high dimensional input, we preprocess the images using principle component analysis (PCA) to suppress some of the noise and speed up the computation process. This process reduces the number of features to 75 before applying t-SNE. We then use t-SNE to reduce the dimensionality of the images to 3 before using k-means to cluster the images. We again use purity and entropy to assess the results.

4 Experiment & Results

The code for all the proposed methods can be found on GitHub. [1]

4.1 Dataset

Caltech256 dataset [17] contains a total of 30607 variable resolution images across 256 different categories. The train, validation and test splits were done by generating random indices with probabilities of 0.6, 0.15 & 0.25 respectively. After this random split, the train, validation and test set consisted of 17803, 4665 and 7312 images respectively.

The CIFAR-10/100 datasets [18] are labeled subsets of 80 million tiny images consisting of 60000 32x32 colour images with 6000 and 100 images per class across 10 and 100 categories respectively. 20% images were used as test set and the experiments were carried out with a validation split of 0.25.

4.2 Data Preprocessing

Following the literature [1, 5], we perform featurewise centering and featurewise standardization on our dataset. All images were rescaled to a value between 0 and 1 for computational feasibility. In our experiments, we tried performing ZCA whitening on the images but did not notice any substantial gain. Considering it to be computationally expensive on large datasets such as ours, we decided to exclude pre-processing images by zca whitening.

Since we are dealing with deep networks and do not have sufficient data samples, we performed data augmentation to engineer more train, validation and test samples. Images at various rotations, zooming, width shift, height shift ranges, different brightness and shear intensities were considered. We note that data augmentation was only performed for dataset fed to the proposed supervised methods as we believe that such an addition will not provide substantial benefits for non-supervised approaches.

4.3 Results

4.3.1 Supervised Methods

For the final architecture of the Plain Classifier model, we find that an architecture consisting of 7 convolutional layers with 64, 64, 128, 128, 128, 512, 512 respectively, each with kernel size of 3x3 and incrementally followed by max-pooling layers as shown in Figure 1 and concluded by 2 fully connected layers produces best accuracy results when used with Stochastic gradient descent(SGD) optimizer with a learning rate of 0.01, momentum of 0.9, decay of 0.000001, and a mini-batch size of 32. We then use this same classifier model to generate the similarity matrix for identifying coarse categories using spectral clustering. Both the coarse and fine category label of images are now used to access the performance of classifier's predictions that are probabilistically averaged over the fine and coarse predictions. All experiments were performed on Nvidia GTX 1080 (8 GB) system and few of them took more than 1.5 days to run.

4.3.2 Unsupervised Methods

For the final network architecture of the CAE, we find that using 4 convolutional layers with 100, 250, 400, 600 filters respectively, each with a kernel size of 3x3, and each followed by max pooling with a pool size of 2 works best for the encoder portion of the model. This resulted in an encoded dimensionality of 8x8x600. For the decoder portion, we use 4 convolutional layers with 600, 400, 250, 100 filters respectively, each with a kernel size of 3x3, and each followed by upsampling, and a final convolutional layer with 3 filters and a 3x3 kernel for making the prediction. We evaluate the effectiveness of our two unsupervised approaches, and additionally compare them against a baseline approach of using raw pixel values with k-means in order to test the hypothesis that these feature extraction techniques will result in improved clustering results. As shown in Table 2, the CAE outperformes both the t-SNE and the baseline method with respect to both purity and entropy. The t-SNE approach has a lower entropy, but also a lower purity than the baseline approach.

[1] GitHub Link to the code can be found at CSC-522_Object_Recognition

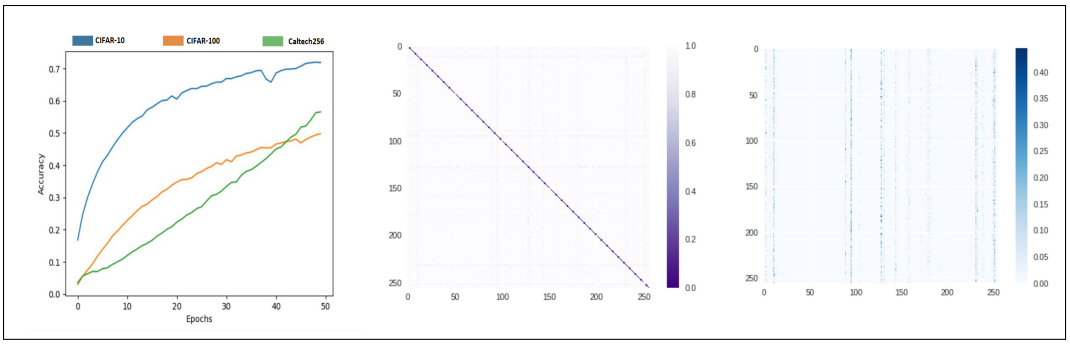

Figure 4: **Left:** Training accuracy against various epochs. **Middle:** Distance Matrix based on predictions made by Plain Classifier Model in Figure 1. **Right:** Confusion Matrix for the predictions made by Plain Classifier Model in Figure 1.

Model		Caltech256		CIFAR-10		CIFAR-100	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Plain Classifier	256	15.58%	30.79%	67.34%	96.98%	37.56%	68.40%
OurResNet50	256	8.50%	22.65%	68.91%	96.72%	26.66%	54.94%
HD-CNN	256	10.17%	-	-	-	42.84%	-
Random Guess	256	0.8%	1.25%	0.2%	1.04%	0.4%	0.98%

Table 1: Top-1 and Top-5 test accuracy of our proposed supervised models on benchmark datasets.

Approach	Purity	Entropy
CAE	**0.143**	**5.45**
t-SNE	0.119	5.61
Raw Pixels	0.125	5.66

Table 2: Comparison of K-means cluster results with our proposed unsupervised methods.

5 Conclusion

Drawing straight from the numbers shown in Table 1, we observe that even though our proposed models perform really well on CIFAR-10 and CIFAR-100, they do not seem to produce satisfactory results for Caltech256. This can be attributed to the quality and the sheer amount of noise prevalent in the images which make it difficult for traditional CNNs to learn meaningful features. Another shortcoming of the Caltech256 dataset was the lack of number of training samples. Even after sufficient data augmentation, the generated samples were not enough for training an architecture as deep and complex as OurResNet50 and HD-CNN. With the lack of sufficient training samples, both the coarse and fine classifiers of HD-CNN suffered from overfitting, with training accuracy of over 99%, which could not be controlled by any fraction of dropout. HD-CNN heavily relies on availability of large number of image samples to effectively train its coarse and fine classifiers. Working with datasets like CIFAR-100 and Caltech256, that do not have more than 100 images per category on an average, provides sound reasoning for the poor results in case of HD-CNN. Training these architectures on ImageNet [19] will surely give improved results but we could not train it due to computational limitations.

From experiments with the unsupervised approaches, we conclude that while a high dimensional representation caused computational complexity issues with k-means, it did not seem to influence the effectiveness of the clustering. This is evident from the fact that the best CAE architecture found only reduces the dimensionality to 38400 features, when models which output lower dimensional

encodings were considered. The main factor in the cluster results for our data seemed to be the quality of the image representation. This explains why the raw pixel approach has a higher purity than t-SNE, as we conclude that the 3 features found by t-SNE simply were not enough to effectively describe an image. It also explains why the CAE outperformed both other methods, since it extracts the same types of meaningful, high-level features used by traditional CNNs that work so well with images.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] D. S. Wilks, "Cluster analysis," in *International geophysics*, vol. 100, pp. 603–616, Elsevier, 2011.

[7] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[8] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. II–104, IEEE, 2004.

[9] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, pp. 52–59, Springer, 2011.

[10] 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.

[11] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[12] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[13] L. Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, pp. 384–391, 2009.

[14] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, p. 319, 1999.

[15] Z. Yan, V. Jagadeesh, D. DeCoste, W. Di, and R. Piramuthu, "HD-CNN: hierarchical deep convolutional neural network for image classification," *CoRR*, vol. abs/1410.0736, 2014.

[16] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks–ICANN 2010*, pp. 92–101, Springer, 2010.

[17] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Tech. Rep. 7694, California Institute of Technology, 2007.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[19] 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.