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. Even though striking results have been reported on large datasets, the problem of object recognition continues to be a challenging one. 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 ask is whether learning better networks have direct correlation to stacking more layers. One significant problem such deep layers could pose is overfitting. Some of the other possible problems that could arise in such scenarios is the one of vanishing and exploding gradients which could lead to non-converging solutions. This problem is largely addressed by popular techniques like batch normalization, local response normalization and by initializing the model with normalized weights, which enable the stochastic gradient descent with back-propagation to yield converging solutions. But 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 which 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.

One other challenge to the task of object recognition is the huge variation in the location of the information in an image. Ideally, we should prefer a smaller kernel for information that have a more localized spatial distribution, and a larger kernel for information which are distributed globally. It is crucial for us to leverage all the information provided by the data, in the right way possible, given the dearth of labeled training data. One step towards addressing this issue is to identify multiple kernels, each learning some spatial distribution in the data and stacking all this information together into one volume [4].

In this paper, we seek to build a simple yet powerful CNN from scratch by employing ideas from the above proposed architectures along with other schemes to avoid overfitting and facilitating faster learning. Our proposed architecture will be employed to test classification accuracy on Caltech256 dataset [6].

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 [7]. 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 [8].

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 [8]. 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 [9]. 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 [8]. 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.

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 [10]. 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 [10]. Since CNNs are one of the more successful models for supervised image classification [11, 12], 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 [10], 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.

The t-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. [13] t-SNE uses a symmetrized version of the SNE cost function with comparatively simpler gradients and use 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 [14].

In this paper, we apply each of these aproaches to the Caltech 256 dataset in order to evaluate and compare the effectiveness of each in the task of learning object categories without supervision.

2 Methods

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.

2.1.1 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 in [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.

2.1.2 1 x 1 Convolution

This Network-in-Network idea was first explored by [5] where they used 1 x 1 convolutions to deal with bottleneck architectures. Such convolutions have dual benefit. Most importantly, they are used as dimensionality reduction modules that can significantly control the depth of the volumes, thus overcoming computational bottlenecks which could otherwise prove counterproductive for deeper networks. The power of such convolutions lie in their ability to control both the width and depth of the networks without incurring significant performance penalty. This allows us to design better and deeper network architectures. Such convolutions are then passed to ReLU non-linearity function, thereby serving dual purpose. We seek to exploit these benefits in our architecture.

2.1.3 Pooling Layer

These layers generally follow the convolutional layers. 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 of size 2x2 applied with a stride of 2 which results in discarding 75% of the activations. Overlapping pooling [1] can capture more spatial information and might result in better approximation of localized data. In our approach, we seek to experiment with Overlapping, Average, Max and Global Max pooling layers and will also try downsampling using CNN [5] with larger stride instead of pooling.

2.1.4 Local Response Normalization(LRN)/ Batch Normalization

With the introduction of Batch Normalization, LRN has fallen out of taste in the researching community. In our studies, we will experiment with LRN using the same constants as in [1] and compare it against the performances of Batch Normalization. By normalizing the weights of the previous activation layers, we can improve the stability and performance of the network by speeding up learning.

2.1.5 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. We will be tuning the dropout rate but do not have any data as of now.

2.1.6 Optimizer

We shall be primarily working with Adam optimizer, stochastic gradient descent and RMSprop with momentum. Hyperparameters like learning rate and momentum are yet to be tuned.

2.1.7 Architecture

Simple Classifier: Our design of a basic classifier is inspired by the ideas enumerated in [1, 5, 15]. This architecture consists of 10 layers; 8 conv layers and 2 fully connected layers, totally to 25,869,632 parameters. The first 2 conv layers have 64 filters, the third and fourth layer have 128 filters, fifth and sixth layer have 512 filters and the last two conv layers have 512 filters. The number of filters double after every downsampling done by max pooling layer. All our conv layers have 3x3 filters with 'same padding' and employ batch normalization [3] immediately after each conv layer and before each activation in order to accelerate learning. 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. Stochastic gradient descent is used as an optimizer with a learning rate of 0.01 and mini-batch size of 12. (tuning of hyperparameters is ongoing). As mentioned previously, our architecture use ReLU non-linearity in all the layers and ends with a 256-way fully-connected layer with softmax.

2.2 Clustering

2.2.1 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 [10]. 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 parameter choices and compare their efficacy using our validation data set. After choosing the model structure, we train this model and evaluate its performance at learning object categories through unsupervised clustering on the test set using [Placeholder] metric.

2.2.2 t-SNE (under construction)

3 Experiment & Results

3.1 Dataset

Caltech256 dataset [6] contains a total of 30607 variable resolution images across 256 different categories. We generated our train, validation and test set by generating random indices with probabilities of 0.6, 0.15 and 0.25 respectively. Post this, our train, validation and test set consisted of 17803, 4665 and 7312 images respectively. This dataset is small and might overfit when training deep networks. For this purpose, apart from methods to avoid overfitting suggested above, we also increase the size of the dataset by incorporating data augmentation by random rotations, vertical and horizontal flips, brightness range, shear intensity, zoom range and randomly shifted versions of the images.

Since the Caltech256 [6] dataset comprises of images of variable resolutions, we first resize all the images to a fixed resolution of 128x128. (We used 64x64 for current results). The only preprocessing we perform is feature wise standardization. We will experiment with the results after ZCA whitening of images but will restrict images to dimensions of 64x64 due to computational issues associated with large matrices.

3.2 Results

All experiments were performed on Nvidia GTX 1080 (8 GB) system. We present our results in terms of Top-1 and Top-5 accuracy rates, similar to the submissions for ImageNet Large-Scale Visual Recognition Challenge (ILSVRC).

Table 1: **Top-1** and **Top-5** accuracy rate of the **Simple Classifier** trained on 50 epochs

Dataset	Images	Top-1	Top-5
Caltech256	30607	27%	47%

4 Conclusion

The accuracy of the simple classifier relies heavily on the number of training samples available, which in our case is just 17803. In our future experiments, we intend to perform data augmentation and design better architectures for achieving better classification rate.

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

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

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[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.