Low-shot Visual Recognition

Latha Pemula

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Dhruv Batra, Chair
Devi Parikh
A. Lynn Abbott

Sep 22, 2016
Blacksburg, Virginia

Keywords: Computer Vision, Machine Learning, Low-shot, Visual Recognition

Low-Shot Visual Recognition

Latha Pemula

ABSTRACT

Many real world datasets are characterized by having a long tailed distribution, with several samples for some classes and only a few samples for other classes. While many Deep Learning based solutions exist for object recognition when hundreds of samples are available, there are not many solutions for the case when there are only a few samples available per class. Recognition in the regime where the number of training samples available for each class are low, ranging from 1 to couple of tens of examples is called Lowshot Recognition. In this work, we attempt to solve this problem. Our framework is similar to [1]. We use a related dataset with sufficient number (a couple of hundred) of samples per class to learn representations using a Convolutional Neural Network (CNN). This CNN is used to extract features of the lowshot samples and learn a classifier . During representation learning, we enforce the learnt representations to obey certain property by using a custom loss function. We believe that when the lowshot sample obey this property the classification step becomes easier. We show that the proposed solution performs better than the softmax classifier by a good margin.

Low-Shot Visual Recognition

Latha Pemula

GENERAL AUDIENCE ABSTRACT

Deep learning, a branch of Artificial Intelligence(AI) is revolutionizing the way computers can learn and perform artificial intelligence tasks. The power of Deep Learning comes from being able to model very complex functions using huge amounts of data. For this reason, deep learning is criticized as being data hungry. Although AI systems are able to beat humans in many tasks, unlike humans, they still lack the ability to learn from less data. In this work, we address the problem of teaching AI systems with only a few examples, formally called the "low-shot learning". We focus on low-shot visual recognition where the AI systems are taught to recognize different objects from images using very few examples. Solving the low-shot recognition problem will enable us to apply AI based methods to many real world tasks. Particularly in the cases where we cannot afford to collect huge number of images because it is either costly or it is impossible. We propose a novel technique to solve this problem. We show that our solution performs better at low-shot recognition than the regular image classification solution, the softmax classifier.

To my family, because "Thank You" is not enough.

Acknowledgments

I am grateful to Dr. Dhruv Batra for being a constant source of motivation and guidance to me throughout my Master's program. I wish to express my sincere thanks to Dr. Devi Parikh for the encouragement and guidance she has provided. I enjoyed the discussions with you both regarding my work and your insightful comments have taught me many the nitty-gritty details of Deep Learning.

I would also like to express my sincere gratitude to Dr. Lei Zhang and Microsoft Research for providing the opportunity to work on this interesting problem during the summer. I would like to thank the entire CVMLP lab for sharing their passion, knowledge and experiences with me. I thank all my friends who made this journey enjoyable and supporting me.

Part of this work was done during my internship at Microsoft Research. This work was partially supported by a National Science Foundation CAREER award, an Army Research Office YIP Award, Office of Naval Research grant N00014-14-1-0679, and GPU donations by NVIDIA, all awarded to DB. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the U.S. Government or any sponsor.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government or any sponsor.

Contents

List of Figures

List of Tables

Chapter 1

Related Work

Low-shot visual recognition is more difficult than any other form of low-shot learning.The visual input is very high dimensional and the best performing models are CNNs with huge number of parameters. Due to the difficulty of this problem, most of the past works have worked with toy like datasets with very low dimensional data. These works used MNIST [2] and Omniglot[3] datasets. [1] is the first to expand it to a much complex dataset, the ImageNet[4] dataset. We describe below, a few past works in the one-shot and low-shot learning. All these works leveraged a related or similar dataset with enough samples per class. We will call this generic dataset. Since our focus is on low-shot learning on celebrity face images, below we discuss some previous approaches on face recognition, one-shot and lowshot learning.

1.1 Oneshot Visual Recognition

There have been several works on one-shot and zero-shot recognition which are based on intermediate level labels:"attributes". Since we do not have or use any such intermediate labels, we do not compare against those approaches.

[5] Approached the one-shot recognition problem by modeling a Siemese Network architecture for image verification. The Siemiese network is trained to verify if a pair of images belong to the same class or not. The generic dataset is used to train the network on the verification task. During inference, the single training sample available for each oneshot class is presented along with the test image and verified if they belong to same class or not. Since it requires that the test image be verified against each of the training samples, this approach is not scalable for large number of classes or when the number of training samples available is more than one.

1.2 Face Recognition

Recognition of human faces is of particular interest to us for many applications ranging from surveillance to entertainment. The complexity for their recognition is more than MNIST dataset [2]and the Omniglot dataset [3], yet simpler than ImageNet dataset. While the general face detection is considered a solved problem, face identity recognition is still somewhat challenging. Discerning one entity from the other is a fine grained recognition task which requires the representations to capture the details.

DeepID2 [6] used supplementary tasks such as face verication to improve the quality of the representations learned. Both face identication and verication signals are used as supervisory signals. The face identication task increases the inter-personal variations by drawing features from different identities apart, while the face verication task reduces the intra-personal variations by pulling features from the same identity together.

FaceNet [7] finds a unified embedding for faces in a space where Euclidean distance between two points correspond to a similarity measure between the two corresponding faces. They use a deep convolutional Neural Netwok trained on Triplet loss to achieve this. In a space where the Euclidean distance corresponds to similarity of faces, many tasks such as classification, verification are as straight forward as computing the distances between the face embeddings and thresholding.To mitigate the convergence problems while training with triplet loss, they perform a careful triplet selection. The triplet selection is done online(per batch) to mine hard positives and hard negatives. To obtain a good selection of triplets, they used a batch size of 4096 samples.

1.3 Lowshot Visual Recognition

The Low-shot visual object recognition[1] is perhaps the most relevant work to ours. Though there have been several previous works on "**One-shot**" and "**Zero-shot**" recognition problems, this work is the first to introduce the "**Low-shot**" recognition problem. They also propose a method of evaluation for this task which we follow to evaluate our approach too.

As with many other approaches for the oneshot recognition they use a generic dataset to learn a feature extractor(ϕ) using Convolutional Neural Network.The features of the low-shot classes extracted using this model are then used to learn a classifier. They point out that, since the feature extractor is optimized for the generic dataset but not the low-shot samples, this model can perform well on the generic dataset but not the low-shot classes. For a model to perform well on both the generic data and the low-shot data, the optimal solution given all samples W*, should be same as or close to the solution obtained by the model using the generic dataset $\hat{W}$. For this solution to generalize well, the gradient at $\hat{W}$ should be low even for the low-shot samples. So they explicitly minimize the gradient of the loss function by adding it to the objective function. For a point x with a probability of p_k

for k^{th} class, label y and C classes, this results in minimizing the following objective function

$$W*, \phi* = \underset{w,\phi}{\operatorname{argmin}} \sum_{i=1}^{N} L(W, \phi(x_i), y_i) + \lambda ||\nabla_W L(W, \phi(x_i), y_i)||^2$$

The gradient norm for Cross Entropy loss takes the following form.

$$\nabla_W L(W, \phi(x_i), y_i) = [\nabla_{w_1} L(W, \phi(x_i), y_i), ..., \nabla_{w_{|C_{base}|}} L(W, \phi(x_i), y_i)]$$

$$\nabla_{W_k} L(W, \phi(x_i), y_i) = (p_{ik} - I(y_i == k))\phi(x_i)$$

where I() is the indicator function.

This loss takes a simple form of weighted norm of the feature activations. Where the weights are different for each sample and a wrongly classified sample is penalized more than the correctly classified sample. They call this the Squared Gradient Magnitude(SGM) loss which we will refer to as SGM from now on. In the case when each sample is weighted equally, this degenerates to the L2 regularization of the feature norm. They experimented with form of Loss function as well and found it to perform close to SGM.

Chapter 2

Approach

We follow the low-shot frame work similar to that proposed in [1]. We use a generic dataset for learning a feature extractor. This feature extractor is then applied on the lowshot samples a classifier is learnt on the lowshot samples.

Deep Neural networks enable us to learn features. They also enable us to force these features to satisfy arbitrary properties given that it is amenable to learning. The way to communicate with a Deep Network is through the data and its loss function. So it is important that we convey clearly what we want the model to do. Our approach to solving this problem is to device a loss function for representation learning, which would favour low-shot recognition. We call this new loss function as "Dense Loss". We describe this loss in detail in the following section.

2.1 Dense Loss

The main idea of this loss is to project the features into such a space which makes classification easier for us. Our approach is based on the intuition that if we could force the model to place samples of each class very tightly close to each other in the feature space, this would make classification easier. We impose that all features belonging to same class to be closer to their mean feature(estimated as another parameter). This loss is proportional to the distance of the feature point from its mean. This idea is somewhat similar to the triplet loss which bring samples of same class closer and samples of different classes farther. But Triplet loss deals with only 3 points at a time. This can be thought of some local constraint where as the Dense Loss imposes that all features be closer to a particular mean feature. This acts as a global constraint. This can be used along with regular cross entropy loss. We hope this would result in denser clusters for each class and help with low-shot classification.

Another way of interpreting this is when different samples of a class are forced to be close to

each other in the feautre space, the model is building more invariant features. For example, we can think of this being all variations and poses and lightings are being mapped to close to a canonical representation. This could benefit low-shot learning greatly because it would be very difficult otherwise to learn pose invariant features using a CNN when very few examples are available. For samples x, their representations $f(x)$, a batch size of N, and the mean vector μ, the dense loss L is given by the following equation.

$$L = \frac{1}{N}\sum_{i=0}^{N}\left(f(x_i) - \mu_{y_i}\right)^2$$

This is optimized along with the Cross Entropy loss to result in the following final loss function

$$L = \frac{1}{N}\sum_{i=0}^{N}\frac{e^{f(x_i)}}{\sum_{k=0}^{C} e^{f(x_k)}} + \lambda\frac{1}{N}\sum_{i=0}^{N}\left(f(x_i) - \mu_{y_i}\right)^2$$

Below we give a brief description of the losses refer to in this document

Dense Loss: Dense loss pulls each sample closer to its class mean. The simplest form of Dense loss is defined as below.

$$L = \frac{1}{N}\sum_{i=0}^{N}\left(f(x_i) - \mu_{y_i}\right)^2$$

Cross Entropy Loss: Cross Entropy Loss is used for training the Softmax Classifier. The loss is defined as follows

$$L = \frac{1}{N}\sum_{i=0}^{N}\frac{e^{f(x_i)}}{\sum_{k=0}^{C} e^{f(x_k)}}$$

where C is the number of classes.

Squared Gradient Magnitude (SGM) Loss: SGM loss is specially designed loss for lowshot learning. The motivation behind the SGM loss is explained in detail in the Related work section.For a point x with a probability of p_k for k^{th} class, label y and C classes, SGM loss is defined as below

$$L = \frac{1}{N}\sum_{i=0}^{N}\alpha_i||(f(x_i)||^2$$

where I() is the indicator function and

$$\alpha_i = \sum_{k=0}^{C} p_{ik} - I(y_i == k)^2$$

SGM L2 Loss: SGM loss is a variation of the SGM loss where all training samples are equally weighed. This degenerates to the simple form of L2 normalization of the feature activations. SGM L2 loss is defined as below

$$L = \frac{1}{N} \sum_{i=0}^{N} ||f(x_i)||^2$$

Dropout: Dropout [8] is a very popular regularization technique used for reducing overfitting in Deep Neural Networks. Dropout, when applied to any layer in the DNN, causes stochastically dropping out activations of that layer with a probability p. This can be interpreted as model averaging of various possible architectures. This also prevents the co-adaptation of neurons in different layers and helps learn better representations.

DeCov Loss: The DeCov loss [9] is a form of regularization which has been shown to perform comparable to dropout. It promotes the representations learnt in Deep Neural Networks to be less redundant by minimizing the cross covariance of hidden activations. This loss can be applied to any layer. For a chosen layer, if $h^n \in \mathrm{R}^d$ represent the hidden activations,then for a batch size of N, the Covariance matrix C, DeCov loss is given by

$$L = \frac{1}{2}(||C||_F^2 - ||diag(C)||_2^2)$$

where diag(.) operator extracts the main diagonal of a matrix into a vector.

2.2 Dataset

We show results on two different datasets: (1) Celebrity 10k dataset from Microsoft Research (2) Imagenet dataset

2.2.1 Celebrity 10k dataset from Microsoft Research

Celebrity 10k dataset from Microsoft Research is a subset of 1 Million Celebrities dataset from Microsoft Research. As described in the introduction, comprises of face images collected from the internet. Hence they form complex "In the wild images". The data distribution among the categories forms a long tail distribution with several hundred images available

for a few celebrities to a handful of samples for many others. To achieve good classification accuracy on this data requires one to tackle this low-shot recognition problem.

The celebrity 10K dataset is a cleaner subset of the 1 Million Celebrities dataset. We use this dataset instead of the 1 Million Celebrities dataset to eliminate the challenges presented by scalability and noisy samples and focus on low-shot learning. The celebrity 10K dataset contains face images of 10000 different celebrities. The data distribution within this celebrity 10k dataset is not uniform but not as extremely long tailed as the Celebrity 1 Million dataset This has an average of 200 images per class. We create an artificial low-shot learning scenario as described below.

Celebrity 10k Low-shot dataset We sample 1000 classes randomly and consider these as our novel classes. The remaining 9000 classes are considered as base classes. 80% of the base classes data is split into training data and the rest as validation set. The Novel classes data is also split into two partitions, a training pool set and a validation set. The Training pool data is used to samples the training data. The low-shot training data for the novel classes is generated by sampling 1,2,5,10,20 samples from each class from the training pool dataset. The feature extractor is learnt on the base class data. The classifier is learnt on the base class training data and the sampled novel class training data. The final evaluation is done a validation set created from the novel and base classes validation data. The evaluation is described in detail in the next section.

2.2.2 ImageNet-1k dataset

The ImageNet-1k challenge dataset consists of 1000 object categories. The ImageNet dataset is split into 389 base categories and 611 novel categories following the setting used by [1]. The training data from ImageNet training data for the base classes are split into training and validation sets for representation learning. The training data from ImageNet training data for the novel classes are split into training pool set and validation set. The low-shot training data for the novel classes is generated by sampling $1, 20$ sampling from each class from the training pool dataset. For learning the classifier, we use the 50% of the base class data(to be comparable with [1]) along with the novel class data sampled as described above. For the detail list of base class and novel classes, please refer to Appendix.

2.3 Evaluation

The evaluation for low-shot learning is slightly different from the traditional one-shot case. In one-shot evaluation, we are only interested in the model's performance in one-shot categories. So the training and test classes are exclusive. The one-shot testing set includes only the novel classes with one image per class. In the low-shot learning scenario, we are interested in our models performance on both the novel classes and the base classes. We would like our model

to perform well on the novel classes and we also want to make sure the performance on base classes is not affected. So we would be evaluating our final performance on both novel classes and the base classes. Performance is measured in terms of top-1 accuracy for the celebrity dataset and top-5 accuracy for the ImageNet dataset because top-5 is more stable than top-1 for Imagenet dataset.Final performance for the Celebrity datset is reported to on val dataset formed by combining both base class and novel class validation set and keeping uniform number samples for all classes and discarding the rest. Note that we have not tuned any hyper parameters using the validation set. For ImageNet dataset, we report the results on ImageNet Validation set.

2.4 Preprocessing

Each of the faces is cropped from a full image using a generic face detector. The faces are transformed to a canonical shape using similarity transformation, i.e. rotation, scale, and translation. All the images are cropped and resized to 224x224.

2.5 Architecture

We use the following two architectures for the Convolutional Neural Networks for the two different datasets.

2.6 CaffeNet net

CaffeNet is modified version of the AlexNet [10] architecture. It is release by Berkely Vision group. We use the CaffeNet architecture for the Celebrity 10K dataset. The Caffenet consists of 5 convolutional and 3 fully connected layers. The architecture of CaffeNet is as shown in figure Fig. 2.1.

2.7 ResNet

The ResNet[12] architecture stands as the state-of-the-art architecture for Image Recognition and won the ImageNet 2015 Recognition and Detection challenges. We use this architecture for all our experiments on ImageNet dataset. It has been shown in the past [13] [14] that deeper architectures result in superior performance. However training them successfully is challenging. The ResNet architecture makes certain assumptions about the function to model that makes training very deep networks possible. It tries to find a solution close to

identity mapping. At each stage it aims to model the residual by using stacked networks. The identity mapping is modelled by shortcut connections from the input to the output of each stage.

The ResNet architecture that won the ImageNet challenge is an ensemble of ResNet architectures with layers varying from 32 to 152 layers. We use a 10 layer ResNet architecture as shown in figure Fig. 2.2. For the identity mapping to work, the input and output dimensions need to be the same. Wherever the dimension are different, the dimensions are increased by using projection layers.

2.8 Training

For all the experiments on Celebrity 10k dataset, the training details are as follows. For representation learning, a base learning rate of 0.01 is used and the learning rate is decreased 10 times after every 50K iteration and the training is run for 200K iterations. We used stochastic gradient descent with momentum 0.9 is used. The weight decay is set to 0.0001.

For learning the low-shot classifier on Celebrity dataset, a base learning rate of 0.01 is used and the learning rate is decreased 10 times after every 30K iteration and the training is run for 150 iterations.Stochastic gradient descent with momentum 0.9 is used. The weight decay is set to 0.0005.

For all the experiments on ImageNet dataset, the training details are as follows. A base learning rate of 0.1 is used and the learning rate is decreased 10 times after every 20 epochs and the training is run for 60 epochs. Stochastic gradient descent with momentum 0.9 is used. The weight decay is set to 0.0001

For learning the low-shot classifier on ImageNet dataset, a base learning rate of 0.1 is used and the learning rate is decreased 10 times after every 50K iteration and the training is run for 100K iterations.Stochastic gradient descent with momentum 0.9 is used. The weight decay is set to 0.001.

2.8.1 Mean Estimation

There are two ways to incorporate dense loss into a network. One can compute the mean feature offline, (referred to as Offline mean computation) or estimate it as a parameter (referred to as online mean computation). In the offline computation, one can use a network trained with cross entropy loss, compute the mean of all $fc7$ features for each class and then enforce that all features be close to their means. Notice that the mean is not a parameter here.

In the online method, we would treat the means as parameters and let the network learn the

mean parameters. Notice that the mean using offline and online are different and need not be comparable.

2.8.2 Training Difficulties

When estimating the mean as a parameter, care has to be taken to avoid degenerate solutions. When the means are initialized with a small value, all the means degenerate to the 0 vector. This is because the feature vectors are also close to the 0 vector and causes the affinity towards 0 vector. This results in the dense loss degenerating to L2 regularization of features. Note that the SGM L2 loss is given by the following equation

$$L = \frac{1}{N} \sum_{i=0}^{N} ||f(x_i)||^2$$

This can be solved by using a combination of offline estimation followed by online finetuning of the means. We take an almost trained network trained with cross entropy loss. The means are now either estimated offline or can be estimated as parameters in CNN with all the other layers frozen. Once the mean parameters are learnt, the features are finetuned with Dense Loss. We do not want the means to change much but want all the feature points to move closer to the mean. We also want the means to adapt to the new features. This would make optimization hard when the features are changing to be closer to the mean and the means themselves are changing. To mitigate this effect, the learning rate of the mean parameters is reduced to be one order less than the rest of the network.

2.8.3 Training a network with Dense Loss

Celebrity dataset: We train a network with cross Entropy loss for 200K iterations. We then freeze the entire network except the mean parameters and let the mean parameters to be learnt. This could have been done using offline estimation as well. We then unfreeze all the parameters and reduce the learning rate for the mean parameters by an order less. We now add the Dense loss as additional loss and use a loss weight of 10^{-6} and train the network for 100K iterations. We make sure that the performance gain is not due to the additional training by comparing the Network trained for 200K iterations with one trained for 300K iterations to make sure it is saturated at 200K iterations.

ImageNet dataset: We follow the same strategy as above. We train with cross entropy loss for 180K iterations, let the means get updated and then train with dense loss for 40K iterations with a loss weight of 0.003.

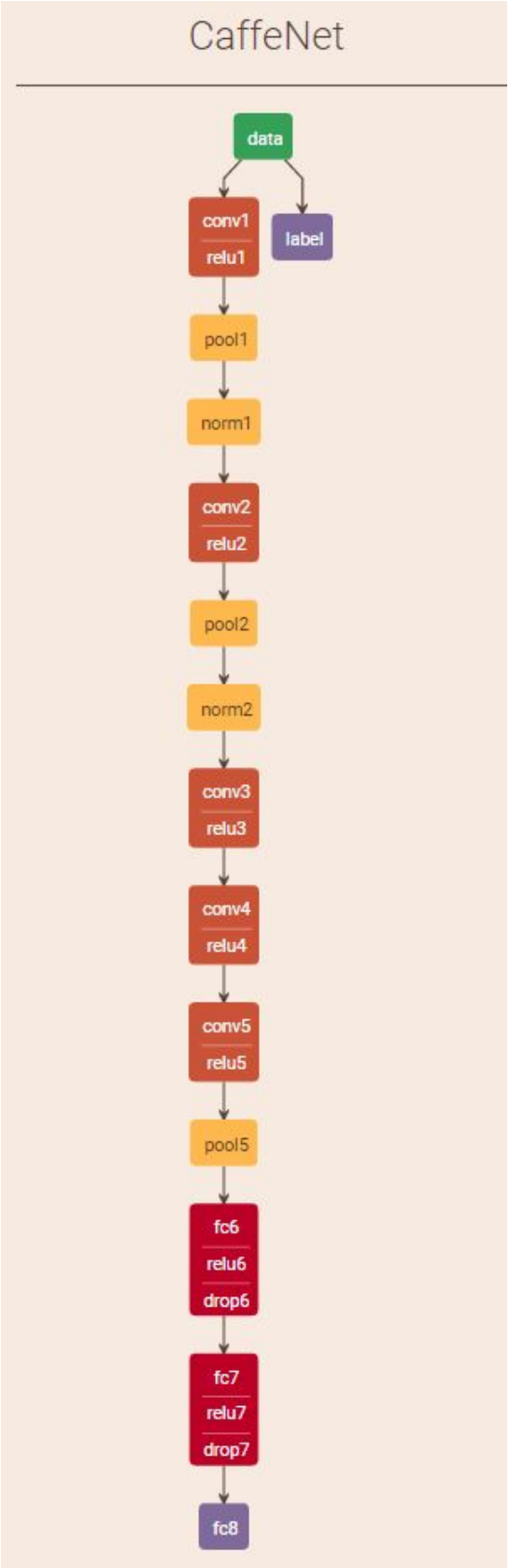

Figure 2.1: CaffeNet Architecture.convx: Convolutional layers,relux:ReLu, poolx:Pooling, fcx:Fully Connected.dropx:Dropout layers. Visualization is generated using ethereon, http://ethereon.github.io/netscope/#/editor/

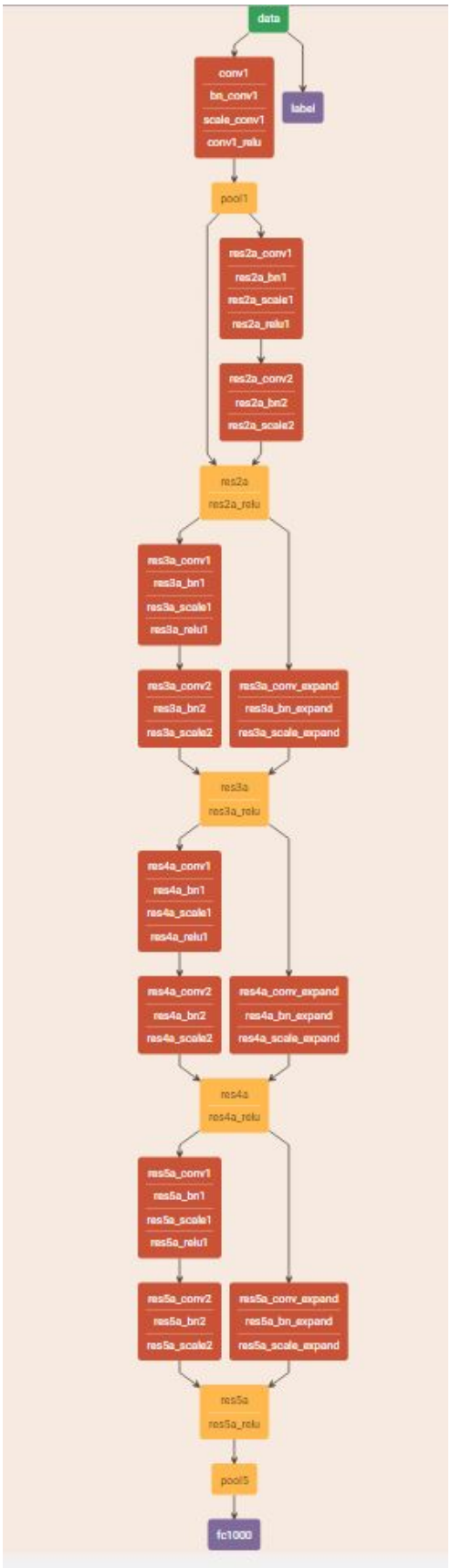

Figure 2.2: ResNet Architecture. _conv:Convolutional layers, _bn:Batch Normalization, _scale:Scaling, _expand:1x1 convolutions to expand the dimensions. Visualization is generated using ethereon, http://ethereon.github.io/netscope/#/editor/

Chapter 3

Implementation

We used Caffe, a deep learning framework developed by Berkeley Vision group for all our experiments. Caffe has support for many common functionalities required for building a Deep Learning model. It also allows us to incorporate custom code. I implemented the following Losses in Caffe. Below is a brief description of what they compute in forward and backward passes.

3.0.1 Dense Loss

Dense loss takes as input the feature activations, labels and produces the Dense Loss as output. For each sample x_i with feature activation $f(x_i)$, the mean parameter μ_i and batch size N, It computes the following function for forward propagation step

$$L = \frac{1}{N} \sum_{i=0}^{N} (f(x_i) - \mu_{y_i})^2$$

During backward propagation, the gradients with respect to the parameters and the inputs are computed as follows.

$$\frac{\partial L}{\partial \mu_{y_i}} = \frac{2}{N}(\mu_{y_i} - f(x_i))$$
$$\frac{\partial L}{\partial f(x_i)} = \frac{2}{N}(f(x_i) - \mu_{y_i})$$

3.0.2 Gradient Checking

Caffe allows to verify any layer with unit tests. Most important part of the unit test for any layer is the gradient checking. This enables us to verify that our backward propagation code is correct given the forward propagation. It does so by computing the numerical gradient using forward propagation and checking that the gradient computed by the backward propagation is within an error tolerance. Numerical gradient is computed as

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\delta) - f(x-\delta)}{2\delta}$$

I verified all the loss functions by writing unit tests to perform gradient checking.

Chapter 4

Results

4.1 Celebrity 10K dataset

Below we present the results for low-shot cases of $n = 1, 2, 10, 20$ samples on Celebrity 10K dataset. We compare our Dense loss approach against the following approaches. (I did not compare it against the [1] results due to the unavailability of the code and lack of complete details to reproduce the results). **Baseline** Our baseline apporach uses a Cross Entropy loss to learn the representation **DeCov Loss** It has been shown inn previous work[1] that many generic regularization techniques worked well for the Low-shot recognition. So we show our results using the DeCov Loss for representation learning.

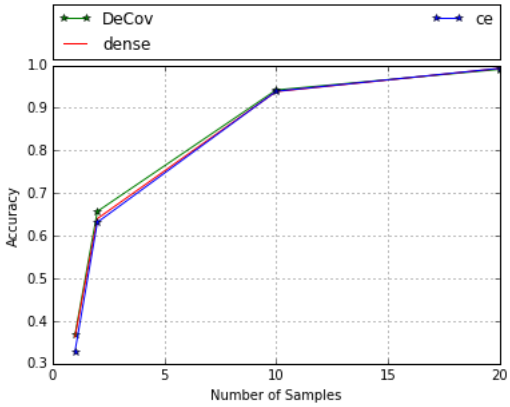

Figure 4.1: Plot of the Classification accuracy of Novel classes using various methods of Representation Learning on Celebrity 10K dataset

The results on novel classes are presented in Figure 4.1 and Table 4.1. We see that Dense Loss performs much better than the baseline when the available samples are very low(1,2). In as the number of available samples increase, the performances of these converge. We

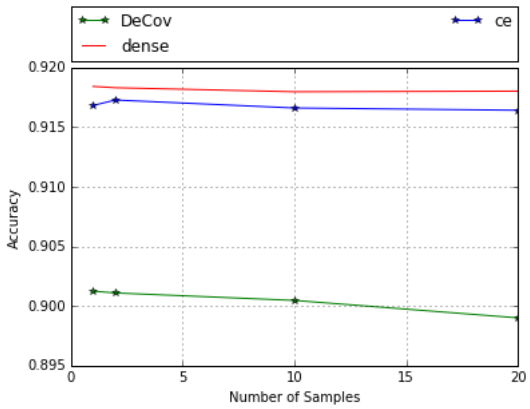

Figure 4.2: Plot of the Classification accuracy of Base classes using various methods of Representation Learning on Celebrity 10K dataset

	Novel Classes		
n	Baseline	DeCov	Dense
1	32.89%	36.76%	36.3%
2	63.16%	65.71%	64.01%
10	93.9%	94.21%	93.84%
20	99.29%	99.05%	99.28%

Table 4.1: Accuracy on Novel Classes of Celebrity 10K dataset with varying number of samples for Novel Classes

also observe that the DeCov loss performs marginally better than Dense Loss for the novel classes. However this comes at a cost of performance on base classes. This can be seen in the results corresponding to the base classes as shown in Figure 4.2 and Table 4.2. This is even more profound when we see the overall accuracy (Base + Novel classes)in Figure 4.3 and Table 4.3. Here we can see that Dense Loss has the best overall accuracy among these methods.T

Also we observe that the Dense loss has marginally better than the Cross Entropy loss for General Classification task. During the representation learning stage, we observe that the accuracy on the 9k classes is 91.49 where as Dense loss is marginally better with an acccuracy of 91.7. However the DeCov loss is only 89.35. All things considered, Dense loss proves that it results in superior representations which result in good classification accuracy in general case as well as Low-shot scenario.

The TSNE embeddings[16] for the novel class samples generated using the feature extractor trained with Cross Entropy Loss. The figure on left shows embeddings generated using the feature extractor trained with Dense loss. We can see that the samples trained using Dense loss form tighter clusters. It is easy to see that classification on these tighter clusters is much easier compared to the other clusters.

n	Base Classes		
	Baseline	DeCov	Dense
1	91.68%	90.12%	91.84%
2	91.72%	90.11%	91.83%
10	91.66%	90.05%	91.79%
20	91.64%	89.90%	91.8%

Table 4.2: Accuracy on Base Classes of Celebrity 10K dataset with varying number of samples for Novel Classes

n	Over all Accuracy		
	Baseline	DeCov	Dense
1	85.98%	84.78%	86.28%
2	88.87%	87.76%	89.05%
10	91.88%	90.46%	92.00%
20	92.4%	90.81%	92.56%

Table 4.3: Over all accuracy on Base Classes and Novel Classes combined of Celebrity 10K dataset with varying number of samples for Novel Classes

4.2 ImageNet Dataset

For Imagenet we use the following methods to compare our approach. Baseline is the simple SoftMax classifier with out dropout(the orginal ResNet model did not use any dropout [12]). **Dropout** uses the SoftMax classifier along with a dropout layer after the final pooling layer. **Dense** corresponds to our Dense loss approach. We use Dense loss along with dropout. Results on ImageNet Novel classes are shown in Table 4.4. Results on ImageNet Base classes are shown in Table 4.5.

We see that Dense loss does improve the performance on Imagenet. How over not by a huge margin. The ImageNet is overfitting regime (Test loss much higher than Training loss). Regularization must better help in this situation. This is apparent with the huge gain in accuracy using Dropout. It is worth exploring data augmentation using scale and color augmentation. Once the overfitting is remedied, Dense loss can provide good improvement both in Novel class accuracy and Base class accuracy.

We see in the figure 4.5 shows TSNE embeddings[16] for the novel class samples generated using the feature extractor trained with Cross Entropy loss and Dense loss. The figure on left shows embeddings generated using the feature extractor trained with Dense loss. The difference in in the quality of features is much more profound on ImageNet dataset compared to the Celebrity10K dataset. We see that besides forming tighter clusters, the samples seem to have much less spread.

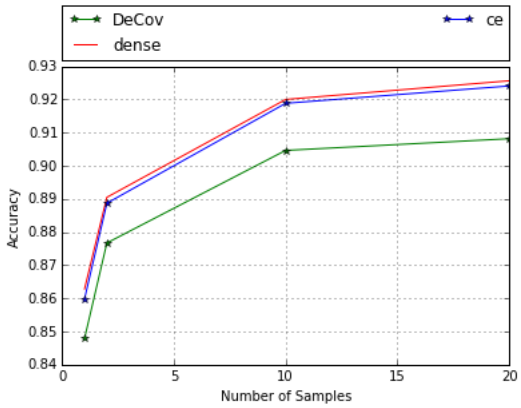

Figure 4.3: Plot of the overall Classification accuracy using various methods of Representation Learning on Celebrity 10K dataset

	Novel Classes		
n	Baseline	Dropout	Dense
1	4.6%	9.01%	10.53%
1	51.58%	54.10%	57.55%

Table 4.4: Accuracy on Novel Classes of ImageNet dataset with varying number of samples for Novel Classes

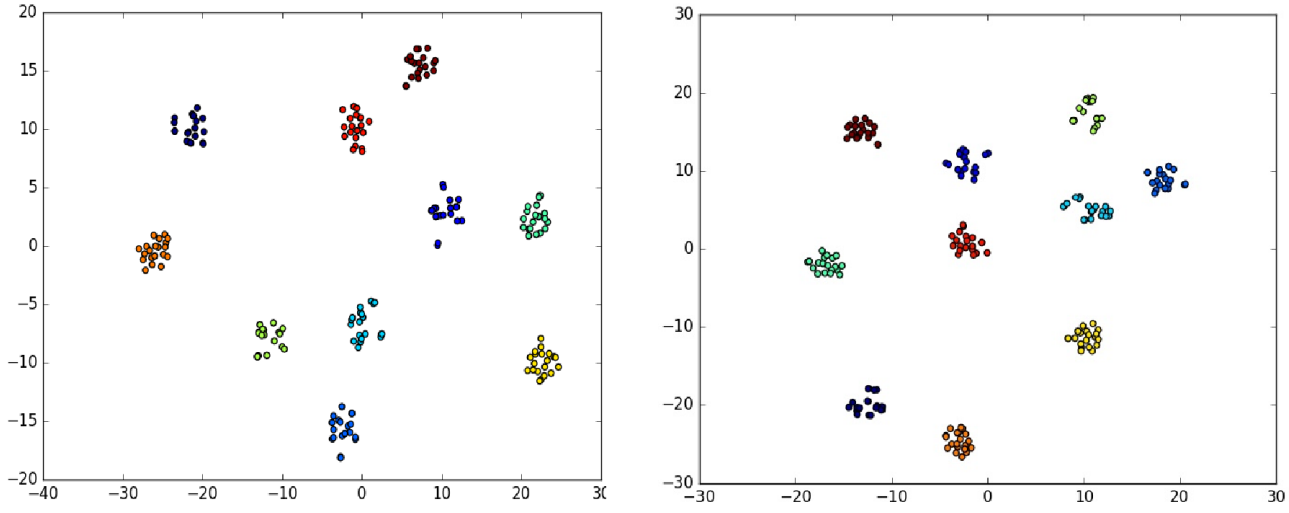

Figure 4.4: TSNE Embeddings of the novel class samples. The figure on left shows embeddings generated using the feature extractor trained with Cross Entropy Loss. The figure on left shows embeddings generated using the feature extractor trained with Dense loss.Visualization created using L. V. D. MAATEN, https://lvdmaaten.github.io/tsne/

n	Base Classes		
	Baseline	Dropout	Dense
1	80.9%	81.86%	84.28%
20	73.81%	73.78%	77.18%

Table 4.5: Accuracy on Base Classes of ImageNet with varying number of samples for Novel Classes

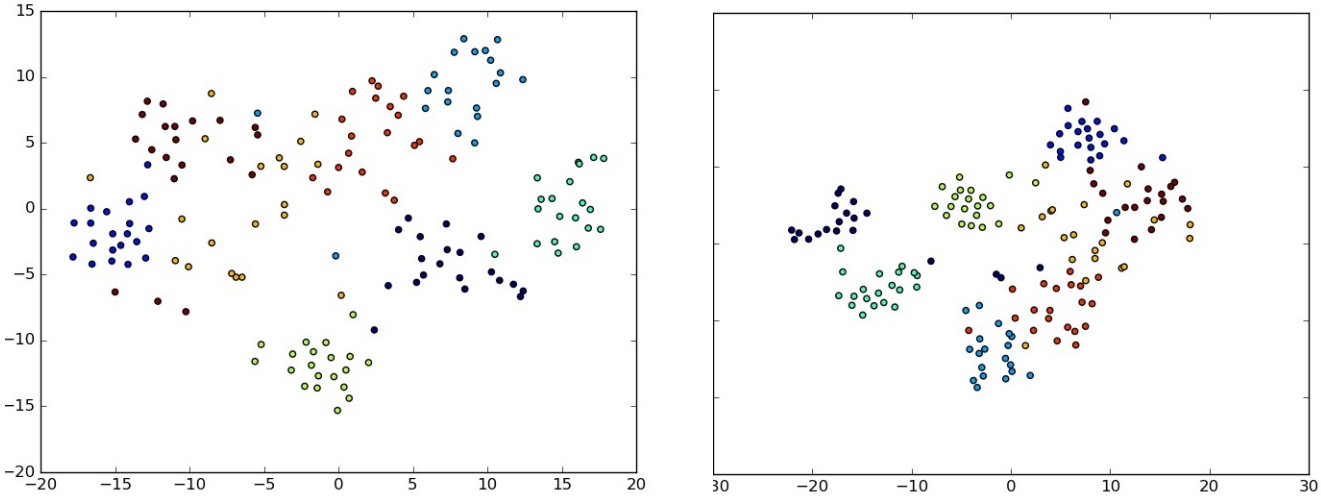

Figure 4.5: TSNE Embeddings of the novel class samples on Imagenet Dataset. The figure on left shows embeddings generated using the feature extractor trained with Cross Entropy Loss. The figure on right shows embeddings generated using the feature extractor trained with Dense loss.Visualization created using L. V. D. MAATEN, https://lvdmaaten.github.io/tsne/

Chapter 5

Future Work

The Low-shot recognition problem is a promising direction with many applications. Here we present a few possible directions for extending this work.

5.1 Variations of the Loss Function

We propose a few variations of the loss functions which can be more robust to outliers.

Bounding the loss with Upper Bound: In the current form, the loss function penalizes the distance between feature vector$f(x_i)$and it's mean vector μ_i. The Loss is computed as

$$L = \frac{1}{N}\sum_{i=0}^{N}(f(x_i) - \mu_{y_i})^2$$

This would cause outliers to effect the means. Especially given that the Celebrity data set is not clean and contains noise, we need to make sure that the noise does not deteriorate the model. We can mitigate this by bounding the loss by an upper bound (β). So the influence of outliers is bounded by β.

Loss function in this case would be

$$L = \frac{1}{N}\sum_{i=0}^{N} min((f(x_i) - \mu_{y_i})^2, \beta)$$

Bounding the loss with a Lower Bound: Points already close to the means do not influence the decision boundary as much. Penalizing them could cause an undesired shift in the features and the means. As long as a point is within some bound(let's call α), we should not penalize it. The loss function would then be

$$L = \frac{1}{N}\sum_{i=0}^{N} max((f(x_i) - \mu_{y_i})^2, \alpha)$$

Combining both the upper bound and lower bounds, we would get a loss function of the following form which is robust to outliers

$$L = \frac{1}{N}\sum_{i=0}^{N} min(max((f(x_i) - \mu_{y_i})^2, \alpha), \beta)$$

Mean Separation We are only enforcing that each point be closer to it's mean in the current loss function. It does not restrict anything about the location of different means with respect to each other. Ideally, for good classification, we would also want the means to be as far away from each other as possible. It is worth exploring this form by adding a term that pushes the means far away from each other. This could help avoid the issue of Dense loss degenerating to SGM loss when the mean parameters are learnt from the scratch.

$$L = \frac{1}{N}\sum_{i=0}^{N} (f(x_i) - \mu_{y_i})^2 + \sum_{k=0}^{C}\sum_{l=0}^{C} (\mu_k - \mu_l)^2$$

5.2 Variation of Distances

Our current Dense loss form computes the Euclidean distances of feature from the mean points. There is no reason to believe this is optimal. We can potentially use Kernel functions such as the Gaussian Kernel or Polynomial Kernel to compute distances.

Chapter 6

Conclusions

To summarize, we proposed a solution to the lowshot image recognition problem by introducing a novel loss function. We showed the performance of our loss function on two different datasets

- 1. Celebrity 10k Dataset from Microsoft Research
- 2. ImageNet 1K Dataset

We show that our Dense loss performs better than the traditional Cross Entropy loss on Lowshot Visual Recognition task. We show that our Dense loss helps not only in the Lowshot Recognition but also in general recognition. Most of the previous works [1] show a decline in the general recognition when addressing the Novel classes. We also observe that Dense loss does not result in any such degradation. We believe that this is probably because the there is an advantage to Dense loss because it is trying to learn better representations which are tight clusters for each class so it performs well on general classification and also lowshot.

The Lowshot recognition problem is very important many applications. This will provide promising deviation from systems needing a lot of examples and reduce the stress of data collection as huge effort. This will make Deep learning methods to be available for many tasks where data collection is costly or available data is too scarce.

Bibliography

[1] B. Hariharan and R. B. Girshick, "Low-shot visual object recognition", *CoRR*, vol. abs/1606.02819, 2016 (cit. on pp. ii, 1, 2, 4, 7, 15, 22, 26).

[2] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]", *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012. DOI: 10.1109/MSP.2012.2211477. [Online]. Available: http://dx.doi.org/10.1109/MSP.2012.2211477 (cit. on pp. 1, 2).

[3] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction", *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015, ISSN: 0036-8075. DOI: 10.1126/science.aab3050. eprint: http://science.sciencemag.org/content/350/6266/1332.full.pdf. [Online]. Available: http://science.sciencemag.org/content/350/6266/1332 (cit. on pp. 1, 2).

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *CVPR*, 2009 (cit. on pp. 1, 26).

[5] G. Koch, "Siamese neural networks for one-shot image recognition siamese neural networks for one-shot image recognition", 2015 (cit. on p. 1).

[6] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification", *CoRR*, vol. abs/1406.4773, 2014. [Online]. Available: http://arxiv.org/abs/1406.4773 (cit. on p. 2).

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682 (cit. on p. 2).

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *CoRR*, vol. abs/1207.0580, 2012 (cit. on p. 6).

[9] M. Cogswell, F. Ahmed, R. B. Girshick, C. L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations", *CoRR*, vol. abs/1511.06068, 2015 (cit. on p. 6).

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cit. on p. 8).

[11] ethereon, http://ethereon.github.io/netscope/#/editor/ (cit. on pp. 11, 12).

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *CoRR*, vol. abs/1512.03385, 2015 (cit. on pp. 8, 17).

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298594 (cit. on p. 8).

[14] ——, "Going deeper with convolutions", *CoRR*, vol. abs/1409.4842, 2015 (cit. on p. 8).

[15] L. V. D. MAATEN, https://lvdmaaten.github.io/tsne/ (cit. on pp. 18, 19).

[16] L. van der Maaten, "Accelerating t-sne using tree-based algorithms", *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2697068 (cit. on pp. 16, 17).

Appendices

Appendix A

Appendix

The ImageNet[4] dataset has a images from 1000 object categories. For the purpose of Low-shot learning, Hariharan *et al.*[1] created a low-shot classification scenario by splitting these into 389 base classes set for representation learning and tested on the 611 novel classes. We use the same split for our experiments. Below is the list of classes included in the base classes set and novel classes set.

A.1 Base Classes

1. n01807496 partridge
2. n02916936 bulletproof vest
3. n03794056 mousetrap
4. n03394916 French horn
5. n02342885 hamster
6. n02782093 balloon
7. n01847000 drake
8. n04044716 radio telescope
9. n04136333 sarong
10. n11879895 rapeseed
11. n03534580 hoopskirt
12. n02676566 acoustic guitar
13. n03000247 chain mail
14. n04482393 tricycle
15. n03062245 cocktail shaker
16. n09399592 promontory
17. n03127925 crate
18. n02264363 lacewing
19. n04542943 waffle iron
20. n01968897 chambered nautilus
21. n02169497 leaf beetle
22. n07615774 ice lolly
23. n01484850 great white shark

24. n02871525 bookshop
25. n03290653 entertainment center
26. n01819313 sulphur-crested cockatoo
27. n02102318 cocker spaniel
28. n03874293 paddlewheel
29. n02526121 eel
30. n02835271 bicycle-built-for-two
31. n03983396 pop bottle
32. n07749582 lemon
33. n07584110 consomme
34. n02091244 Ibizan hound
35. n04487394 trombone
36. n04517823 vacuum
37. n03950228 pitcher
38. n02321529 sea cucumber
39. n01773157 black and gold garden spider
40. n03271574 electric fan
41. n07697313 cheeseburger
42. n03602883 joystick
43. n02749479 assault rifle
44. n07583066 guacamole
45. n01943899 conch
46. n01582220 magpie
47. n04019541 puck
48. n02486261 patas
49. n03344393 fireboat
50. n03379051 football helmet
51. n03538406 horse cart
52. n01698640 American alligator
53. n02666196 abacus
54. n02009229 little blue heron
55. n02093754 Border terrier
56. n03627232 knot
57. n01491361 tiger shark
58. n02486410 baboon
59. n01756291 sidewinder
60. n02988304 CD player
61. n03843555 oil filter
62. n01693334 green lizard
63. n04548280 wall clock
64. n04336792 stretcher
65. n03425413 gas pump
66. n02783161 ballpoint
67. n02894605 breakwater
68. n02317335 starfish
69. n02134084 ice bear
70. n03201208 dining table
71. n04254680 soccer ball
72. n03814639 neck brace
73. n06874185 traffic light
74. n03495258 harp
75. n04086273 revolver

76. n03661043 library
77. n02119789 kit fox
78. n03376595 folding chair
79. n02206856 bee
80. n03527444 holster
81. n04579145 whiskey jug
82. n02096177 cairn
83. n02114548 white wolf
84. n04254777 sock
85. n04005630 prison
86. n03594734 jean
87. n03924679 photocopier
88. n01755581 diamondback
89. n03785016 moped
90. n03494278 harmonica
91. n04099969 rocking chair
92. n02268443 dragonfly
93. n02977058 cash machine
94. n01641577 bullfrog
95. n04350905 suit
96. n02012849 crane
97. n07734744 mushroom
98. n07565083 menu
99. n03916031 perfume
100. n02391049 zebra
101. n02111500 Great Pyrenees
102. n01806567 quail
103. n02106662 German shepherd
104. n01632458 spotted salamander
105. n03481172 hammer
106. n02704792 amphibian
107. n04525305 vending machine
108. n02091134 whippet
109. n04560804 water jug
110. n04192698 shield
111. n03388549 four-poster
112. n03854065 organ
113. n04116512 rubber eraser
114. n07930864 cup
115. n02690373 airliner
116. n03775071 mitten
117. n03485794 handkerchief
118. n04509417 unicycle
119. n03450230 gown
120. n01443537 goldfish
121. n01632777 axolotl
122. n04330267 stove
123. n01990800 isopod
124. n02086646 Blenheim spaniel
125. n02096051 Airedale
126. n04286575 spotlight
127. n02120505 grey fox

128. n04485082 tripod
129. n01978455 rock crab
130. n03457902 greenhouse
131. n04493381 tub
132. n02859443 boathouse
133. n12768682 buckeye
134. n01687978 agama
135. n12144580 corn
136. n01560419 bulbul
137. n02992529 cellular telephone
138. n02488702 colobus
139. n02509815 lesser panda
140. n03127747 crash helmet
141. n04153751 screw
142. n04332243 strainer
143. n01806143 peacock
144. n02910353 buckle
145. n01829413 hornbill
146. n04200800 shoe shop
147. n03483316 hand blower
148. n03089624 confectionery
149. n04277352 spindle
150. n09428293 seashore
151. n04591713 wine bottle
152. n04606251 wreck
153. n02097658 silky terrier
154. n04252225 snowplow
155. n03769881 minibus
156. n03796401 moving van
157. n03444034 go-kart
158. n02167151 ground beetle
159. n04258138 solar dish
160. n03763968 military uniform
161. n03016953 chiffonier
162. n03393912 freight car
163. n01695060 Komodo dragon
164. n02091831 Saluki
165. n02410509 bison
166. n02009912 American egret
167. n04209239 shower curtain
168. n04039381 racket
169. n02422699 impala
170. n03929855 pickelhaube
171. n04456115 torch
172. n03218198 dogsled
173. n01728572 thunder snake
174. n04325704 stole
175. n02071294 killer whale
176. n02966193 carousel
177. n04296562 stage
178. n02117135 hyena
179. n02965783 car mirror

180. n01820546 lorikeet
181. n02123597 Siamese cat
182. n02930766 cab
183. n02786058 Band Aid
184. n03992509 potter's wheel
185. n03908618 pencil box
186. n02090379 redbone
187. n04552348 warplane
188. n01608432 kite
189. n02110185 Siberian husky
190. n02096437 Dandie Dinmont
191. n07742313 Granny Smith
192. n01945685 slug
193. n03496892 harvester
194. n03026506 Christmas stocking
195. n02018207 American coot
196. n01614925 bald eagle
197. n07747607 orange
198. n04131690 saltshaker
199. n07730033 cardoon
200. n04372370 switch
201. n02091635 otterhound
202. n04081281 restaurant
203. n03467068 guillotine
204. n07932039 eggnog
205. n02364673 guinea pig
206. n03384352 forklift
207. n04557648 water bottle
208. n01735189 garter snake
209. n03920288 Petri dish
210. n07860988 dough
211. n02108551 Tibetan mastiff
212. n04311004 steel arch bridge
213. n07579787 plate
214. n04264628 space bar
215. n02100877 Irish setter
216. n01616318 vulture
217. n02510455 giant panda
218. n04553703 washbasin
219. n02268853 damselfly
220. n03788195 mosque
221. n02454379 armadillo
222. n03532672 hook
223. n03291819 envelope
224. n02114855 coyote
225. n02110806 basenji
226. n13044778 earthstar
227. n02606052 rock beauty
228. n02408429 water buffalo
229. n03814906 necklace
230. n02089078 black-and-tan coonhound
231. n04310018 steam locomotive

232. n03884397 panpipe
233. n09246464 cliff
234. n02643566 lionfish
235. n02825657 bell cote
236. n03197337 digital watch
237. n04118538 rugby ball
238. n03355925 flagpole
239. n04266014 space shuttle
240. n04591157 Windsor tie
241. n02109525 Saint Bernard
242. n03131574 crib
243. n02006656 spoonbill
244. n04162706 seat belt
245. n02769748 backpack
246. n02480495 orangutan
247. n04435653 tile roof
248. n01729977 green snake
249. n01592084 chickadee
250. n02808440 bathtub
251. n02437616 llama
252. n03476991 hair spray
253. n02951585 can opener
254. n02011460 bittern
255. n04613696 yurt
256. n03297495 espresso maker
257. n03014705 chest
258. n03240683 drilling platform
259. n02823750 beer glass
260. n02917067 bullet train
261. n07718747 artichoke
262. n02091467 Norwegian elkhound
263. n01742172 boa constrictor
264. n04525038 velvet
265. n12985857 coral fungus
266. n01514859 hen
267. n04589890 window screen
268. n02791124 barber chair
269. n02423022 gazelle
270. n02412080 ram
271. n03947888 pirate
272. n01981276 king crab
273. n03345487 fire engine
274. n02870880 bookcase
275. n04346328 stupa
276. n02447366 badger
277. n04423845 thimble
278. n01494475 hammerhead
279. n02236044 mantis
280. n11939491 daisy
281. n03208938 disk brake
282. n02445715 skunk
283. n02112706 Brabancon griffon

284. n03124170 cowboy hat
285. n02814860 beacon
286. n02099429 curly-coated retriever
287. n12057211 yellow lady's slipper
288. n04429376 throne
289. n02087046 toy terrier
290. n03045698 cloak
291. n03930630 pickup
292. n01688243 frilled lizard
293. n07697537 hotdog
294. n04540053 volleyball
295. n02395406 hog
296. n03673027 liner
297. n04584207 wig
298. n03792972 mountain tent
299. n02105641 Old English sheepdog
300. n02092002 Scottish deerhound
301. n04065272 recreational vehicle
302. n02865351 bolo tie
303. n03976657 pole
304. n01773797 garden spider
305. n02112350 keeshond
306. n02120079 Arctic fox
307. n02492035 capuchin
308. n04146614 school bus
309. n01798484 prairie chicken
310. n04252077 snowmobile
311. n02966687 carpenter's kit
312. n04507155 umbrella
313. n02403003 ox
314. n04389033 tank
315. n03134739 croquet ball
316. n01630670 common newt
317. n07753113 fig
318. n04070727 refrigerator
319. n02361337 marmot
320. n04548362 wallet
321. n01910747 jellyfish
322. n02114367 timber wolf
323. n03594945 jeep
324. n03976467 Polaroid camera
325. n03478589 half track
326. n04239074 sliding door
327. n03792782 mountain bike
328. n04120489 running shoe
329. n03787032 mortarboard
330. n04285008 sports car
331. n03216828 dock
332. n02481823 chimpanzee
333. n02281406 sulphur butterfly
334. n01914609 sea anemone
335. n02442845 mink

336. n02108000 EntleBucher
337. n02992211 cello
338. n02077923 sea lion
339. n03400231 frying pan
340. n03658185 letter opener
341. n01795545 black grouse
342. n02107908 Appenzeller
343. n04179913 sewing machine
344. n03544143 hourglass
345. n02099267 flat-coated retriever
346. n03417042 garbage truck
347. n02951358 canoe
348. n02066245 grey whale
349. n07880968 burrito
350. n02776631 bakery
351. n01950731 sea slug
352. n02123045 tabby
353. n02488291 langur
354. n02356798 fox squirrel
355. n02999410 chain
356. n03085013 computer keyboard
357. n02417914 ibex
358. n02326432 hare
359. n03777754 modem
360. n02640242 sturgeon
361. n01537544 indigo bunting
362. n02974003 car wheel
363. n03630383 lab coat
364. n01873310 platypus
365. n12267677 acorn
366. n02226429 grasshopper
367. n02108915 French bulldog
368. n03670208 limousine
369. n02088238 basset
370. n02099601 golden retriever
371. n02747177 ashcan
372. n01665541 leatherback turtle
373. n02105056 groenendael
374. n01883070 wombat
375. n04597913 wooden spoon
376. n04590129 window shade
377. n03187595 dial telephone
378. n03706229 magnetic compass
379. n03770439 miniskirt
380. n04428191 thresher
381. n02276258 admiral
382. n01622779 great grey owl
383. n02100735 English setter
384. n02939185 caldron
385. n03841143 odometer
386. n07802026 hay
387. n02794156 barometer
388. n02841315 binoculars
389. n02834397 bib

A.2 Novel Classes

1. n10565667 scuba diver
2. n02978881 cassette
3. n03126707 crane
4. n07693725 bagel
5. n03710193 mailbox
6. n02105412 kelpie
7. n07753275 pineapple
8. n01818515 macaw
9. n02802426 basketball
10. n03908714 pencil sharpener
11. n03535780 horizontal bar
12. n09468604 valley
13. n03877845 palace
14. n02094114 Norfolk terrier
15. n03781244 monastery
16. n02113023 Pembroke
17. n03443371 goblet
18. n02256656 cicada
19. n01677366 common iguana
20. n02087394 Rhodesian ridgeback
21. n02111129 Leonberg
22. n02074367 dugong
23. n02892767 brassiere
24. n03724870 mask
25. n02536864 coho
26. n01728920 ringneck snake
27. n04204347 shopping cart
28. n03888257 parachute
29. n02483362 gibbon
30. n02128757 snow leopard
31. n04154565 screwdriver
32. n04344873 studio couch
33. n01770393 scorpion
34. n09256479 coral reef
35. n07720875 bell pepper
36. n02107574 Greater Swiss Mountain dog
37. n03196217 digital clock
38. n03498962 hatchet
39. n07875152 potpie
40. n07714571 head cabbage
41. n06785654 crossword puzzle
42. n01871265 tusker
43. n02091032 Italian greyhound
44. n02095889 Sealyham terrier
45. n01796340 ptarmigan
46. n04265275 space heater
47. n03961711 plate rack
48. n01496331 electric ray
49. n03272010 electric guitar

50. n01770081 harvestman
51. n03388043 fountain
52. n03188531 diaper
53. n02102480 Sussex spaniel
54. n02837789 bikini
55. n02089973 English foxhound
56. n01924916 flatworm
57. n04335435 streetcar
58. n01534433 junco
59. n04090263 rifle
60. n02090622 borzoi
61. n03721384 marimba
62. n03710637 maillot
63. n02097209 standard schnauzer
64. n02109047 Great Dane
65. n02095314 wire-haired fox terrier
66. n03717622 manhole cover
67. n01601694 water ouzel
68. n13040303 stinkhorn
69. n02056570 king penguin
70. n09835506 ballplayer
71. n01739381 vine snake
72. n02480855 gorilla
73. n03207941 dishwasher
74. n02667093 abaya
75. n04149813 scoreboard
76. n01740131 night snake
77. n02233338 cockroach
78. n02687172 aircraft carrier
79. n02101556 clumber
80. n03720891 maraca
81. n01530575 brambling
82. n03388183 fountain pen
83. n04074963 remote control
84. n01797886 ruffed grouse
85. n02799071 baseball
86. n09229709 bubble
87. n03873416 paddle
88. n02058221 albatross
89. n02097298 Scotch terrier
90. n03871628 packet
91. n03742115 medicine chest
92. n04118776 rule
93. n03777568 Model T
94. n02879718 bow
95. n02883205 bow tie
96. n02102177 Welsh springer spaniel
97. n03637318 lampshade
98. n01694178 African chameleon
99. n03179701 desk
100. n02328150 Angora
101. n03529860 home theater

102. n04612504 yawl
103. n02099712 Labrador retriever
104. n03995372 power drill
105. n04033995 quilt
106. n02795169 barrel
107. n02504458 African elephant
108. n03729826 matchstick
109. n03042490 cliff dwelling
110. n02980441 castle
111. n04532670 viaduct
112. n04209133 shower cap
113. n01882714 koala
114. n07768694 pomegranate
115. n04152593 screen
116. n04251144 snorkel
117. n04356056 sunglasses
118. n03249569 drum
119. n04037443 racer
120. n02112018 Pomeranian
121. n02607072 anemone fish
122. n02493793 spider monkey
123. n02669723 academic gown
124. n01644373 tree frog
125. n02492660 howler monkey
126. n04404412 television
127. n04476259 tray
128. n03314780 face powder
129. n04033901 quill
130. n02487347 macaque
131. n04505470 typewriter keyboard
132. n02699494 altar
133. n03837869 obelisk
134. n01749939 green mamba
135. n03125729 cradle
136. n02950826 cannon
137. n03733131 maypole
138. n03887697 paper towel
139. n03942813 ping-pong ball
140. n02948072 candle
141. n03095699 container ship
142. n02807133 bathing cap
143. n04483307 trimaran
144. n04147183 schooner
145. n03697007 lumbermill
146. n04228054 ski
147. n02443484 black-footed ferret
148. n03372029 flute
149. n02804610 bassoon
150. n04579432 whistle
151. n02088632 bluetick
152. n02280649 cabbage butterfly
153. n03761084 microwave

154. n02086910 papillon
155. n02114712 red wolf
156. n03000134 chainlink fence
157. n01776313 tick
158. n02085936 Maltese dog
159. n02110341 dalmatian
160. n02398521 hippopotamus
161. n03743016 megalith
162. n01697457 African crocodile
163. n01824575 coucal
164. n04040759 radiator
165. n07717556 butternut squash
166. n04523525 vault
167. n03868863 oxygen mask
168. n01774750 tarantula
169. n03874599 padlock
170. n03599486 jinrikisha
171. n02107142 Doberman
172. n01682714 American chameleon
173. n04041544 radio
174. n02115641 dingo
175. n07760859 custard apple
176. n02279972 monarch
177. n02002724 black stork
178. n01955084 chiton
179. n02007558 flamingo
180. n04204238 shopping basket
181. n04550184 wardrobe
182. n03929660 pick
183. n01748264 Indian cobra
184. n01843065 jacamar
185. n03590841 jack-o'-lantern
186. n03935335 piggy bank
187. n02096585 Boston bull
188. n04366367 suspension bridge
189. n03793489 mouse
190. n03838899 oboe
191. n02514041 barracouta
192. n02281787 lycaenid
193. n02692877 airship
194. n02086079 Pekinese
195. n04554684 washer
196. n04229816 ski mask
197. n03649909 lawn mower
198. n03110669 cornet
199. n04447861 toilet seat
200. n02123159 tiger cat
201. n03804744 nail
202. n02107683 Bernese mountain dog
203. n04201297 shoji
204. n02025239 ruddy turnstone
205. n02094433 Yorkshire terrier

206. n02396427 wild boar
207. n03764736 milk can
208. n03676483 lipstick
209. n01855032 red-breasted merganser
210. n02037110 oystercatcher
211. n09421951 sandbar
212. n02093991 Irish terrier
213. n04328186 stopwatch
214. n02051845 pelican
215. n01704323 triceratops
216. n02641379 gar
217. n03899768 patio
218. n02493509 titi
219. n01983481 American lobster
220. n03666591 lighter
221. n01768244 trilobite
222. n03832673 notebook
223. n03633091 ladle
224. n02116738 African hunting dog
225. n03657121 lens cap
226. n03786901 mortar
227. n03895866 passenger car
228. n01978287 Dungeness crab
229. n09193705 alp
230. n02088364 beagle
231. n03773504 missile
232. n02097474 Tibetan terrier
233. n07695742 pretzel
234. n03146219 cuirass
235. n03445777 golf ball
236. n02128925 jaguar
237. n07753592 banana
238. n02128385 leopard
239. n03930313 picket fence
240. n03902125 pay-phone
241. n02823428 beer bottle
242. n02089867 Walker hound
243. n03982430 pool table
244. n02111889 Samoyed
245. n04442312 toaster
246. n04515003 upright
247. n03325584 feather boa
248. n03272562 electric locomotive
249. n03623198 knee pad
250. n02797295 barrow
251. n04376876 syringe
252. n02088094 Afghan hound
253. n02092339 Weimaraner
254. n01729322 hognose snake
255. n01532829 house finch
256. n04487081 trolleybus
257. n03598930 jigsaw puzzle

258. n01667778 terrapin
259. n02787622 banjo
260. n02971356 carton
261. n13054560 bolete
262. n07718472 cucumber
263. n07613480 trifle
264. n02101006 Gordon setter
265. n07754684 jackfruit
266. n03938244 pillow
267. n02219486 ant
268. n03791053 motor scooter
269. n03770679 minivan
270. n06794110 street sign
271. n01917289 brain coral
272. n02095570 Lakeland terrier
273. n01855672 goose
274. n03877472 pajama
275. n01986214 hermit crab
276. n02028035 redshank
277. n03958227 plastic bag
278. n03018349 china cabinet
279. n07892512 red wine
280. n02096294 Australian terrier
281. n02422106 hartebeest
282. n02437312 Arabian camel
283. n07745940 strawberry
284. n02113712 miniature poodle
285. n02701002 ambulance
286. n03980874 poncho
287. n07248320 book jacket
288. n02843684 birdhouse
289. n03530642 honeycomb
290. n02483708 siamang
291. n02165456 ladybug
292. n02105855 Shetland sheepdog
293. n01980166 fiddler crab
294. n01644900 tailed frog
295. n02497673 Madagascar cat
296. n03461385 grocery store
297. n03041632 cleaver
298. n03956157 planetarium
299. n04461696 tow truck
300. n03476684 hair slide
301. n03733805 measuring cup
302. n02727426 apiary
303. n03710721 maillot
304. n04458633 totem pole
305. n04259630 sombrero
306. n15075141 toilet tissue
307. n13133613 ear
308. n01984695 spiny lobster
309. n04357314 sunscreen

310. n01689811 alligator lizard
311. n04254120 soap dispenser
312. n03662601 lifeboat
313. n03485407 hand-held computer
314. n03803284 muzzle
315. n07715103 cauliflower
316. n03944341 pinwheel
317. n12620546 hip
318. n02119022 red fox
319. n01985128 crayfish
320. n03075370 combination lock
321. n02457408 three-toed sloth
322. n03680355 Loafer
323. n04326547 stone wall
324. n02093647 Bedlington terrier
325. n04562935 water tower
326. n04026417 purse
327. n02672831 accordion
328. n02110627 affenpinscher
329. n04125021 safe
330. n07716358 zucchini
331. n04238763 slide rule
332. n03891332 parking meter
333. n02174001 rhinoceros beetle
334. n02895154 breastplate
335. n04270147 spatula
336. n02132136 brown bear
337. n02105251 briard
338. n03133878 Crock Pot
339. n06359193 web site
340. n07873807 pizza
341. n03825788 nipple
342. n04501370 turnstile
343. n13052670 hen-of-the-woods
344. n02130308 cheetah
345. n01828970 bee eater
346. n02277742 ringlet
347. n01877812 wallaby
348. n04355933 sunglass
349. n03891251 park bench
350. n04380533 table lamp
351. n02113186 Cardigan
352. n02123394 Persian cat
353. n03617480 kimono
354. n01817953 African grey
355. n12998815 agaric
356. n02017213 European gallinule
357. n02013706 limpkin
358. n02108089 boxer
359. n02097130 giant schnauzer
360. n02963159 cardigan
361. n03857828 oscilloscope

362. n04371430 swimming trunks
363. n02168699 long-horned beetle
364. n02190166 fly
365. n03459775 grille
366. n01774384 black widow
367. n02817516 bearskin
368. n03482405 hamper
369. n07836838 chocolate sauce
370. n02098413 Lhasa
371. n04347754 submarine
372. n07717410 acorn squash
373. n04367480 swab
374. n03690938 lotion
375. n02109961 Eskimo dog
376. n01744401 rock python
377. n04275548 spider web
378. n07871810 meat loaf
379. n02877765 bottlecap
380. n04008634 projectile
381. n01664065 loggerhead
382. n01631663 eft
383. n02104029 kuvasz
384. n07590611 hot pot
385. n02108422 bull mastiff
386. n04409515 tennis ball
387. n02094258 Norwich terrier
388. n01629819 European fire salamander
389. n07714990 broccoli
390. n07831146 carbonara
391. n07920052 espresso
392. n02389026 sorrel
393. n03180011 desktop computer
394. n03250847 drumstick
395. n03255030 dumbbell
396. n01531178 goldfinch
397. n03775546 mixing bowl
398. n01843383 toucan
399. n01734418 king snake
400. n02033041 dowitcher
401. n02415577 bighorn
402. n02808304 bath towel
403. n04599235 wool
404. n03866082 overskirt
405. n04141327 scabbard
406. n02259212 leafhopper
407. n01692333 Gila monster
408. n02927161 butcher shop
409. n03207743 dishrag
410. n04522168 vase
411. n02397096 warthog
412. n04398044 teapot
413. n03063689 coffeepot

414. n03788365 mosquito net
415. n02101388 Brittany spaniel
416. n03709823 mailbag
417. n07614500 ice cream
418. n02110063 malamute
419. n04127249 safety pin
420. n01833805 hummingbird
421. n02115913 dhole
422. n03888605 parallel bars
423. n01784675 centipede
424. n04479046 trench coat
425. n04235860 sleeping bag
426. n13037406 gyromitra
427. n02909870 bucket
428. n02085620 Chihuahua
429. n04263257 soup bowl
430. n02129165 lion
431. n02129604 tiger
432. n01685808 whiptail
433. n03452741 grand piano
434. n09288635 geyser
435. n02102973 Irish water spaniel
436. n02788148 bannister
437. n03445924 golfcart
438. n01872401 echidna
439. n04208210 shovel
440. n02814533 beach wagon
441. n03424325 gasmask
442. n02113624 toy poodle
443. n01675722 banded gecko
444. n02490219 marmoset
445. n02231487 walking stick
446. n03933933 pier
447. n03998194 prayer rug
448. n03447447 gondola
449. n02113799 standard poodle
450. n04604644 worm fence
451. n02106382 Bouvier des Flandres
452. n02229544 cricket
453. n02107312 miniature pinscher
454. n01580077 jay
455. n02708093 analog clock
456. n03937543 pill bottle
457. n01930112 nematode
458. n02112137 chow
459. n02124075 Egyptian cat
460. n03642806 laptop
461. n04067472 reel
462. n02106166 Border collie
463. n02489166 proboscis monkey
464. n02125311 cougar
465. n04536866 violin

466. n02098286 West Highland white terrier
467. n04141975 scale
468. n02027492 red-backed sandpiper
469. n03970156 plunger
470. n02104365 schipperke
471. n02100236 German short-haired pointer
472. n01518878 ostrich
473. n01440764 tench
474. n02444819 otter
475. n04486054 triumphal arch
476. n04243546 slot
477. n02484975 guenon
478. n02137549 mongoose
479. n02443114 polecat
480. n02093859 Kerry blue terrier
481. n03868242 oxcart
482. n03223299 doormat
483. n02869837 bonnet
484. n02002556 white stork
485. n02346627 porcupine
486. n04141076 sax
487. n02100583 vizsla
488. n03017168 chime
489. n02906734 broom
490. n02105162 malinois
491. n03776460 mobile home
492. n02860847 bobsled
493. n02804414 bassinet
494. n01860187 black swan
495. n01498041 stingray
496. n01773549 barn spider
497. n03692522 loupe
498. n03347037 fire screen
499. n03259280 Dutch oven
500. n02085782 Japanese spaniel
501. n02655020 puffer
502. n02793495 barn
503. n02730930 apron
504. n02815834 beaker
505. n02088466 bloodhound
506. n03337140 file
507. n04417672 thatch
508. n02093428 American Staffordshire terrier
509. n03691459 loudspeaker
510. n01944390 snail
511. n04311174 steel drum
512. n03782006 monitor
513. n04399382 teddy
514. n02177972 weevil
515. n02172182 dung beetle
516. n02791270 barbershop

517. n02097047 miniature schnauzer
518. n03447721 gong
519. n09472597 volcano
520. n09332890 lakeside
521. n07716906 spaghetti squash
522. n03876231 paintbrush
523. n03991062 pot
524. n04069434 reflex camera
525. n04418357 theater curtain
526. n04467665 trailer truck
527. n04111531 rotisserie
528. n03028079 church
529. n02777292 balance beam
530. n02979186 cassette player
531. n04465501 tractor
532. n02110958 pug
533. n01751748 sea snake
534. n03584829 iron
535. n02319095 sea urchin
536. n03404251 fur coat
537. n01514668 cock
538. n03124043 cowboy boot
539. n03047690 clog
540. n02892201 brass
541. n04392985 tape player
542. n01775062 wolf spider
543. n04370456 sweatshirt
544. n03100240 convertible
545. n03141823 crutch
546. n03967562 plow
547. n03903868 pedestal
548. n01558993 robin
549. n03063599 coffee mug
550. n02090721 Irish wolfhound
551. n01753488 horned viper
552. n01667114 mud turtle
553. n04133789 sandal
554. n02133161 American black bear
555. n02018795 bustard
556. n02106030 collie
557. n07711569 mashed potato
558. n07684084 French loaf
559. n03595614 jersey
560. n02099849 Chesapeake Bay retriever
561. n04049303 rain barrel
562. n02098105 soft-coated wheaten terrier
563. n06596364 comic book
564. n04004767 printer
565. n02840245 binder
566. n04317175 stethoscope
567. n04355338 sundial
568. n03109150 corkscrew

569. n02363005 beaver
570. n02165105 tiger beetle
571. n02325366 wood rabbit
572. n02138441 meerkat
573. n03840681 ocarina
574. n04443257 tobacco shop
575. n02093256 Staffordshire bullterrier
576. n03759954 microphone
577. n02113978 Mexican hairless
578. n02504013 Indian elephant
579. n03220513 dome
580. n03065424 coil
581. n03032252 cinema
582. n03954731 plane
583. n02127052 lynx
584. n04592741 wing
585. n03000684 chain saw
586. n02105505 komondor
587. n04273569 speedboat
588. n02494079 squirrel monkey
589. n02106550 Rottweiler
590. n04532106 vestment
591. n02790996 barbell
592. n03733281 maze
593. n02134418 sloth bear
594. n02500267 indri
595. n10148035 groom
596. n03160309 dam
597. n02981792 catamaran
598. n04009552 projector
599. n03977966 police van
600. n02086240 Shih-Tzu
601. n03584254 iPod
602. n04371774 swing
603. n04023962 punching bag
604. n01737021 water snake
605. n02102040 English springer
606. n02441942 weasel
607. n04596742 wok
608. n02111277 Newfoundland
609. n01669191 box turtle
610. n03492542 hard disc
611. n04462240 toyshop