Estabilishing Semantic Relationships among object classes using Deep Networks to Image Clasification

Sabyasachi Sahoo ,Vineetha Kondameedi

May 4, 2015

Abstract

Neural networks are a powerful technology for classification of visual inputs arising from documents. Convolutional neural networks provide an efficient method to constrain the complexity of feed-forward neural networks by weight sharing and restriction to local connections. In, this project we try to explore the power of the Convolutional neural Networks in feature extraction and eventually establish semantic relationship between object classes. We see various ways of visualizing this semantic relationship among object classes. We present a hybrid scheme, a combination of CNN and ANN, to bring out this semantic relationship. Using this semantic relationship and other factors, we go further showing how existing standard image classification algorithms can be improved.

1 Introduction

Most of the real world information is high dimensional working on which is not only time consuming but also computationally costlier. This so called 'curse of dimensionality' plays a vital role in domain of Artificial Intelligence, since here in we deal with very large amount of high dimensional data. While the data is high dimensional the most important information lies in the low dimensional subspace. Hence, we need to come up with efficient algorithms which process this high dimensional data and come up with lower dimensional representation of it.

In case of object recognition we try to extract the features from the objects which preserve the original class variability. Hence, it is of pivotal importance that we extract these features located in the lower dimensions with very little loss in the information. Image classifiers require these preprocessed features as input for classification. In most of the methods these preprocessing stage is handcrafted which is very laborious and time taking. Deep learning provides exciting advantages in this case where in we attempt to model high-level abstractions in data by using model architectures composed of multiple non-linear transformations. An algorithm is deep if the input is passed through several non-linearity before being output. Artificial Neural Networks and Convolutional Neural networks are one among the deep learning architectures. [2]

The following report is organized as follows. In Section 2 We give a brief description of type of deep architechtures we have used.In Section 3 We describe

the methodology employed for estabilishing semantic relationships. Section 4 talks about Hybrid CNN-ANN scheme Section 5 We present Results and Analysis Section 6 Deals Conclusion and Future Work.

2 Deep Architectures

2.1 Convolutional Neural Networks

In this Section we give a detailed description of how a Convolutional Neural Network works. Convolutional neural networks (CNNs) are a type of deep model that can act directly on the raw inputs and give out the features. CNNs are a biologically-inspired trainable architecture that can learn invariant features. Each stage in a Convolutional Neural Networks is composed of a filter bank, nonlinearities, and feature pooling layers. A CNN consists alternating pairs of convolution and max pooling layers. [3]

2.1.1 Convolution

Convolution in mathematical terms is an integral that expresses the amount of overlap of one function g as it is shifted over another function f. A convolution layer applies a set of filters that process small local parts of the input where these filters are replicated along the whole input space. These "replicated" units form a feature map, which share the same parameterization, i.e. the same weight vector and the same bias.

Mathematically, a feature map is obtained by convolving the input image with a linear filter, adding a bias term and then applying a non-linear function. If we denote the k-th feature map at a given layer as h^k , whose filters are determined by the weights w^k and bias b^k, then feature map is given by

$$h_i j^k = f((w^k * x)_i j + b_k)$$

Where $f(u)$ is a non-linear function like $tanh$ for transformation/mapping purposes.

2.1.2 Max-Pooling

Max-pooling, is a form of non-linear down-sampling. A max-pooling layer generates a lower resolution version of the convolution layer activations by taking the maximum filter activation from different positions within a specified window. This adds translation invariance and tolerance to minor differences of positions of objects parts. Advantages of Max-Pooling: 1. Eliminates non-maximal values, it reduces computation for upper layers. 2. It provides a form of translation invariance. Avoids problems due to light perturbation.

2.1.3 Fully Connected Layers

While the lower-layers are composed to alternating convolution and max-pooling layers the upper-layers are fully-connected and correspond to a traditional Multi-Layer perceptron (hidden layer + logistic regression). The input to the first fully-connected layer is the set of all features maps at the layer below. Higher layers use more broad filters that work on lower resolution inputs to process

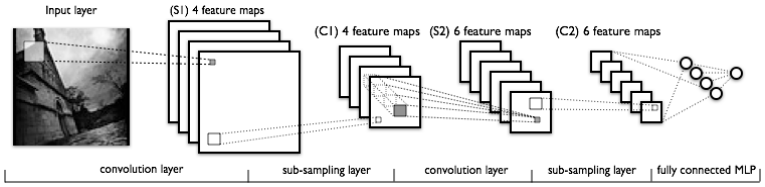

Figure 1: Depiction of CNN.

more complex parts of the input. Top fully connected layers finally combine inputs from all positions to do the classification of the overall inputs.Figure 1 shows a basic pipeline of how a CNN is implemented. [1]

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) are parallel computational models, comprising densely interconnected adaptive processing units. ANNs have multiple layers that are interconnected. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer. The input to each neuron is a linear combination of the outputs of each of the neurons in the lower layer. The output of the neuron is a nonlinear threshold applied to its input. Figure 2 is a schematic representation of ANN using one neuron/perceptron.

$$u_j = \sum(x_i.w_ij)eq1$$

$$y_j = Fth(u)j + t_j)eq2$$

For every neuron, j, in a layer, each of the i inputs, X_i, to that layer is multiplied by a previously established weight, w_ij. These are all summed together, resulting in the internal value of this operation, U_j. This value is then biased by a previously established threshold value, t_j, and sent through an activation function, F_th. This activation function is usually the sigmoid function, which has an input to output mapping. The resulting output, Y_j, is an input to the next layer or it is a response of the neural network if it is the last layer.

2.2.1 Learning weights by Backpropagation

Learning in a neural network is basically, computing the error using the training data and updating the weights by calculating the gradient of the error function. This updating of weights is called Backpropagation.
Backpropagation starts at the output layer with the following equations:

$$w_ij = w_i^ij + LR.e_j.X_ieq3$$

$$e_j = y_j, (1 - Y_j(.(d_j - Y_j)eq4$$

For the i^th input of the j^th neuron in the output layer, the weight w_ij is adjusted by adding to the previous weight value, w'^ij, a term determined by the product of a *learningrate*, LR, an error term, e_j, and the value of the i^th input, X_i. The error term, e_j, for the j^th neuron is determined by the product

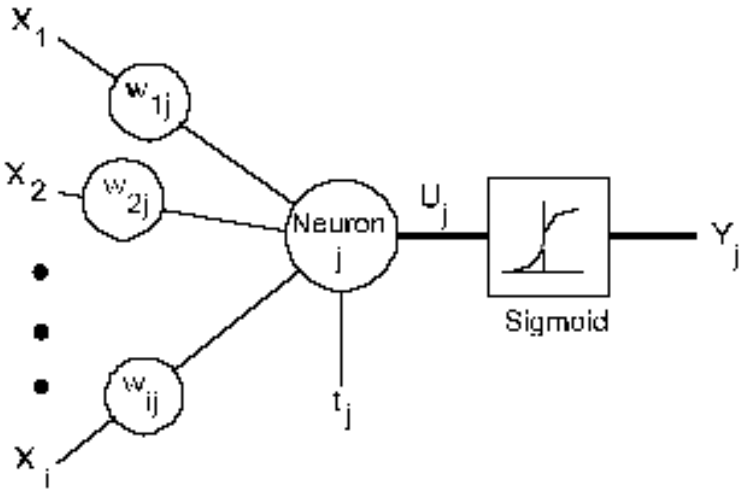

Figure 2: Model of ANN.

of the actual output, Y_j, its complement, $1 - Y_j$, and the difference between the desired output, dj, and the actual output.

Once the error terms are computed and weights are adjusted for the output layer, the values are recorded and the next layer back is adjusted. The same weight adjustment process, determined by Equation 3, is followed, but the error term is generated by a slightly modified version of Equation 4. This modification is:

$$e_j = Y_j.(1 - Y_j).\sum(e_k.w'_jk)eq5$$

Here, the difference between the desired output and the actual output is replaced by the sum of the error terms for each neuron, k, in the layer immediately succeeding the layer being processed times the respective pre-adjustment weights.

The learning rate, LR, applies a greater or lesser portion of the respective adjustment to the old weight. If the factor is set to a large value, then the neural network may learn more quickly, but if there is a large variability in the input set then the network may not learn very well or at all. The revised weight adjustment process is given below:

$$w_ij = w'_ij + (1 - M).LR.e_j.X_j + M.(w'_ij - w''_ij)eq6)$$

This is similar to Equation 3, with a momentum factor, M, the previous weight, w'_ij, and the next to previous weight, w''_ij, included in the last term. This extra term allows for momentum in weight adjustment.

3 Experiments

We used PASCAL VOC 2012 dataset [4]

CONTENTS: 20 classes: Aero plane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV,sofa,bird. Total – 40,000 images. Minimum 500 images for each class.

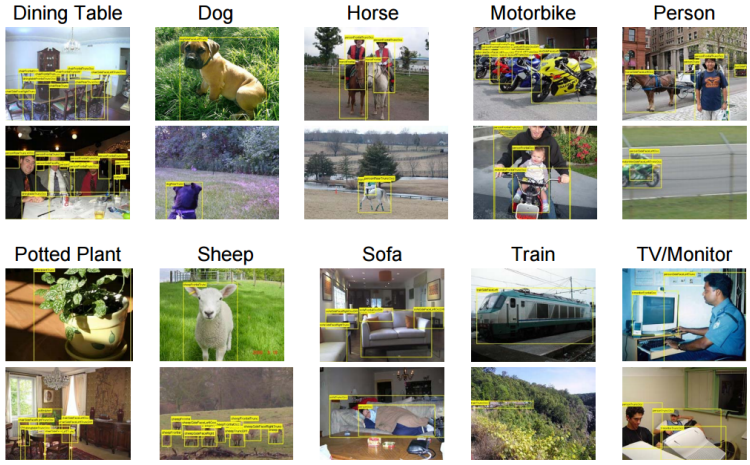

Figure 3: Example Images from PASCAL DATASET.

Annotations: • Complete annotation of all objects (XML format). • Annotated in one session with written guidelines.

3.1 Establishing Semantic Relationships Using Deep

3.1.1 Preprocessing

The aim of this project is to exploit the power of Convolutional Neural Networks for establishing semantic relationships. For this purpose, we have used Matconvnet for implementing Convolutional Neural Networks. 'vl_simplenn' is a wrapper around MatConvNet core computational blocks that implements a CNN with a simple linear structure. We have used pre-trained filters of Imagenet in our model. The output of all the FC6 and FC7 layers are collected and then dimensionality reduction was done on the data. The data of all the layers of all the classes is stored in one single matrix.

3.1.2 Dimensionality Reduction

STEP 2: Dimensionality reduction is performed using principal component analysis. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate.

3.1.3 Clustering

The basic idea is that, when we perform clustering, we expect similar objects to be mapped to same cluster. We have employed K-means for this purpose. The reason for employing K-means is due to its simplicity of application and less computationally expensive, when compared to other clustering methods. When

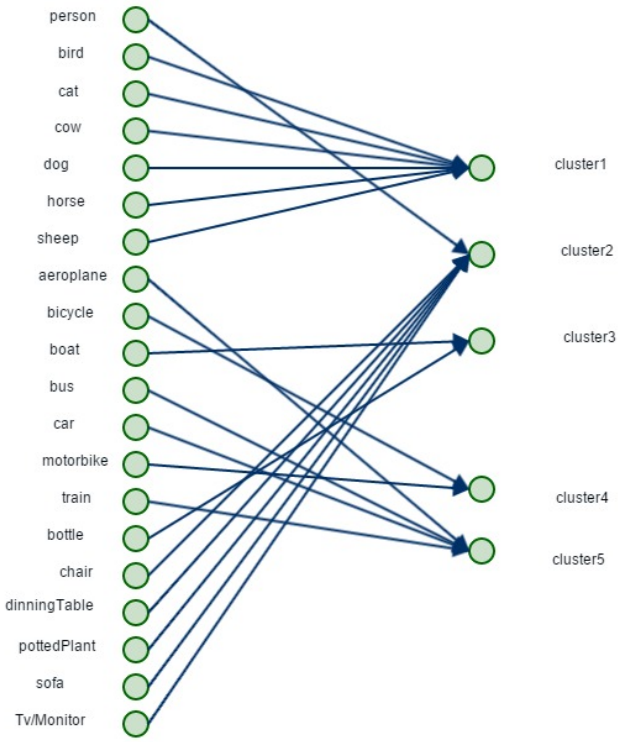

Figure 4: BiPartite Graph of Clustering

we perform K-means on the data with varying the number of clusters, we get different objects mapped into same cluster. Value 'k' is chosen by trial and error and the one, which groups similar classes to same cluster is taken. Below is the bi-partite graph which shows how object classes are mapped to the clusters.

Intuitively, we can say that objects that are mapped to same cluster are similar. The results out of clustering also prove this. Thus we can establish a semantic relationship between the classes. But, the main drawback of this method is choosing the proper value of k. Also, since k-means is a linear clustering method it cannot capture all the information in the higher dimensional space. Other Non-linear clustering methods exist but they are computationally expensive, and also they will take exponential amount time. Therefore they are usually not a feasible alternative. We also have used hierarchical clustering on FC7 vectors, and the dendrogram so obtained is as follows:

4 Hybrid CNN-ANN

The main drawback with most of the clustering algorithms, linear or non - linear is that they mostly don't account for the ground truth labels of the points. Linear clustering algorithm try to group close-by data points together, and non-linear algorithms try to cluster based on some kernel(like in kernel k-Means)

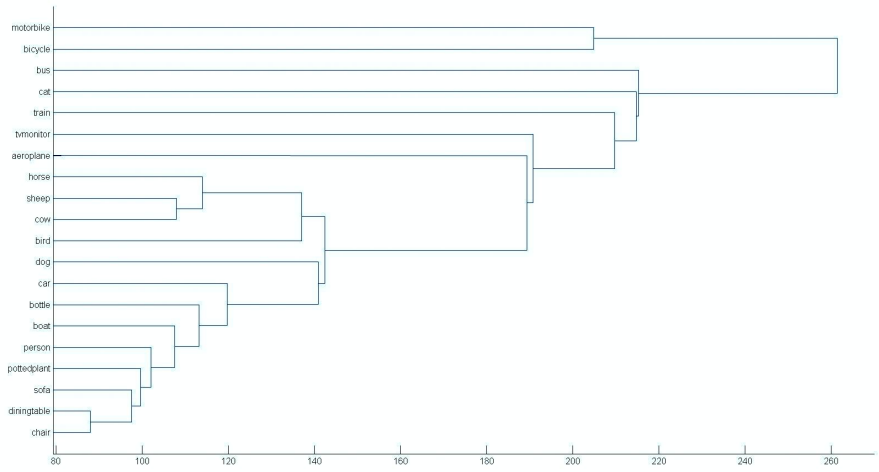

Figure 5: AGGLOMERATIVE CLUSTERING DENDOGRAM PLOT

or as per some manifold present in the dataset (like in spectral clustering). The output from CNN i.e. last Fully Connected (FC) layer is also called as FC7 layer. We know that these FC7 layers capture most or nearly all of the semantic information of the input image to CNN and one might expect FC7 vector similar images to be close to each other in N-dimensional space but this is not true as seen above. What we need is to make use of ground truth labels of the dataset and then somehow try to find semantic relationship among different object classes.

This can be nicely done by using a simple (yet powerful) Artificial Neural Network layer that classifies the FC7 vectors into object classes. The classifier can be trained for the dataset using ground truth labels of every input image. Once the classifier is trained properly, we can use it generate a 'score' vector that gives probability of that FC7 vector being mapped to any of the predefined object classes. The object class for which the classifier generates the maximum score for an input image should be the object class of that image. This is usually done in simple image Classification experiments.

The main point that we are missing here is, that the score generated for every image belonging to a particular class, is not only exactly trying to predict true object class of that image, but it also stores probabilities of it getting mapped to other object classes. So according to this, we can say that the next best guess object class for an image will be more close to its true class. And same will be true for all the images of a dataset. This can be simply explained in a following example.

Let's say we have three object classes, namely cat, dog and sheep. And we input a cat image to CNN that generates a FC7 layer, which is further fed to trained classifier. Let us assume the classifier is trained to less than 100 percent accuracy, which is mostly the case in practical scenario. The trained classifier will (or in most cases should) correctly classify that image as cat. If the classifier makes the second best guess for image as dog, then it clearly implies that object

class ‘dog’ and ‘cat’ must be closely related, more than either of cat or dog being closely related to sheep. This is quite intuitive since, the input vector to classifier is a FC7 layer with all the characteristics or features of the input image and classifier will get mostly confused among similar object classes, in our case, ‘dog’ and ‘cat’. Same can be said for all the images with more umber of object classes. Using this we can effectively establish semantic relationship among various object classes of a dataset using their ground truth label.

The same is done for PASCAL dataset of 20 classes. First we train a classifier to about 91 percent accuracy. Then FC7 vectors of all the images is fed to the classifier, for which classifier generates a score vector of size 20x1. Now we take score vectors of all the images of the same object class, and we generate 1 score vector which would represent that object class from all other remaining score vectors. The single representation score vector for an object class must be so taken that there is minimum loss of information while converting from many score vectors of an object class to a single score vector. One may use any of the measures of central tendency, like mean, mode, geometric mean, harmonic mean, etc. We took mean of the score vectors of an object class to find its representative score vector.

The same was repeated for all the 20 classes. Their respective representative vectors are concatenated one after other to form a matrix. We call, the matrix so formed, as similarity matrix. This is because this matrix now contains maximum values in the cells for which the corresponding column and row object class will be semantically most similar. However, we noticed that the matrix so formed is asymmetric in nature and this is something which we don’t want, but we expected this. This is because, a ‘sheep’ class image may be more similar to a ‘cat’ image but vice-versa wouldn’t be true, since a ‘cat’ class image will be more similar to ‘dog’ class image. So we would like this matrix to be symmetric since similarity is two-sided relationship, i.e. cat is as similar to a dog, as much as dog is similar to dog. To achieve this, we take another matrix, which is transpose of similarity matrix and average out all the elements of both the matrix to get a final symmetric matrix.

This symmetric similarity matrix exactly should represent which class is similar to which other class. The similarity can be shown in a hierarchical fashion in a dendrogram plot. The classes can be linked by finding maximum value in whole of the matrix, and linking the row and column class into one common class (or in a sense a cluster) and then finding next max in the matrix. This is repeated till all the object classes have been accounted for. The dendrogram so formed is plotted below.

We can see the dendrogram plot so formed is much better than dendrogram formed from simple agglomerative clustering of representative FC7 vectors of the object classes. We can see, sofa and chair are more similar to each other than anything else, and they both are together similar to dining table. Similarly, cat and dog are most similar to each other and horse, cow, sheep are again similar among each other. Also, since all of these are animals, they all are getting linked to each other, before they can get linked to some other class.

Hence we can claim now that the dendrogram so formed from our similarity matrix captures semantic relationship among object classes, in the best possible manner. And like the title of this project suggests, it was done using deep (ANN) network. It can be explained as follows: Grouping classes only based on linear clustering algorithm fails miserably because images of same object class need

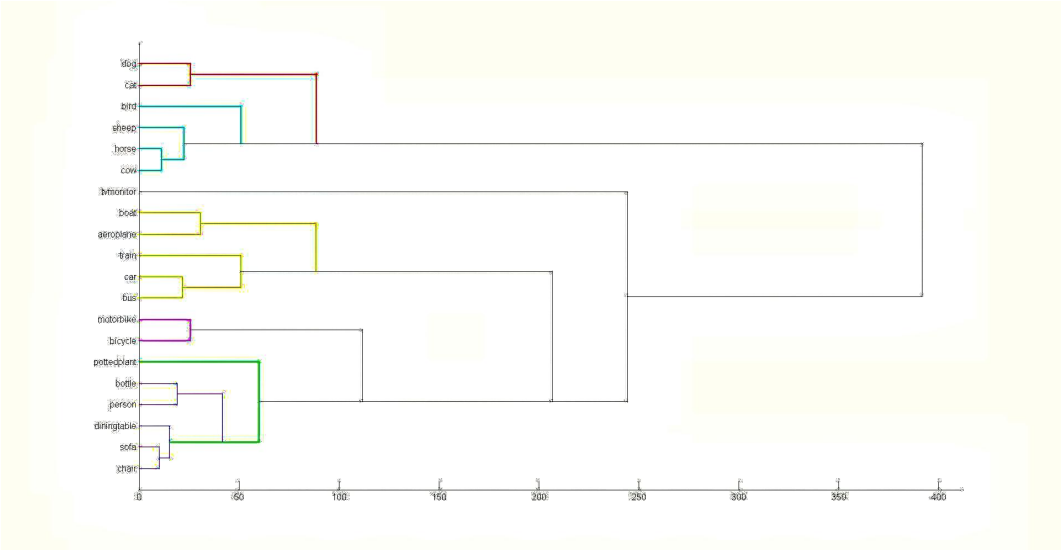

Figure 6: SIMILARITY MATRIX DENDOGRAM PLOT

not be close-by in the high dimensional space. But what one needs to do is to, not only use some non-linear algorithm to group the object classes (which can also learn some manifolds present in dataset and hence form decision boundary) but also effectively make use of ground truth labels for the image data points available with us. The simple classifier in our case does the same in best possible manner.

5 Exploiting Semantic relationship among object classes in Image classification

In the previous section, we saw that semantic relationship can be effectively established among object classes of a dataset of images using a hybrid combination of CNN and ANN. However, we would want to further extend this idea and use it to make improvements in the area of image classification. Our goal from now on, will be to improve object class classification accuracy for any image (test or training image), by applying a variety of strategies discussed in the section. Using these strategies, we can not only improve our classification accuracy, but in case of any misclassification, we try to increase our accuracy for next best guess object class for the input image. In the end, we will try to explain, why some methods will and should perform than other methods and under what constraints. And all of this is possible by making use of sematic relationships among object classes, which is present in FC7 vectors and score vectors of every image.

We will consider a simple image classification experiment and suggest methodologies to improve for that experiment. These methodologies can be further used along with other complex and more accurate image classification algorithms. In a simple image classification, an image is fed to pre-trained CNN to generate FC7 vectors. In our case, we chose the convolutional neural network trained

on Imagenet dataset, which is much larger than PASCAL dataset in terms of number of images and number of classes. The Imagenet dataset has about 1000 classes and all the 20 classes of PASCAL dataset is already present in them. All the convolutions on any image is henceforth done using this CNN to generate FC7 vectors. We chose the FC7 vectors of all the images because, classification using this layer is giving maximum classification accuracy when trained for only an hour on our personal systems.

FC7 vectors obtained are then fed into a pre-trained classifier that generates a score vector for the input image, whose maximum value corresponds to the object class the classifier predicts the image to belong to. The next maximum value corresponds to the class the classifier predicts the second best guess object class for the input image. The same is implemented in our 'query.m' MATLAB function. The function accepts an image, number of guesses to be displayed, and returns its best guess object class, subsequent similarity of that image to the object classes of PASCAL dataset as input.

One thing to note about the simple image classification experiment explained above is that, the test image which classifier is going to classify is not previously seen by classifier. Hence it may or may not give correct class label for the input image. However, we would still expect next best guess to be accurate for a good classifier. Even for those cases, simple image classification algorithm fails miserably sometimes. To overcome that, we propose the following methods:

Once we get FC7 vector for an input image, then instead of directly feeding it into classifier, we can find its nearest (FC7 vector) training data point, and then assign the score vector generated for the training data point to the image for its object class classification. The nearest training data point can be found in a number of linear and non-linear ways. Here, we have implemented 1-Nearest Neighbor (linear) algorithm and found it to be working better than original image classification algorithm. Even if best guess object class for input image turns out to be incorrect for both these algorithms, this variant is better at predicting its next best guesses in most of the case. The same is implemented in our 'query_fc7_1nn.m' MATLAB function. The function accepts an image, number of object classes of classifier, and the distance method to be used for finding 1-Nearest Neighbor. It returns the best guess object class, and subsequent best guess object classes for the image and similarity of each object class with that image.

As pointed out above, the above algorithm uses linear 1-NN algorithm to find best guess object classes for the image. One effective and efficient way of doing it non-linearly is by using Deep networks. So in next algorithm, we propose that instead of finding 1-NN for FC7 vector of the input image, first feed the image into the trained classifier and then with the score vector generated for input image, find its 1-Nearest Neighbor among score vectors of all training data points, and assign that nearest data point's score vector to the image for its best guess object class classification. The same is implemented in our 'query_score_1nn.m' MATLAB function. The function accepts an image, number of object classes of classifier, and the distance method to be used for finding 1-Nearest Neighbor. It returns the best guess object class, and subsequent best guess object classes for the image and similarity of each object class with that image.

The above mentioned algorithm works fairly good and sometimes better than simple image classification experiment at making first best guess object class.

However it can be (in some cases only) further improved, if instead of finding 1-Nearest Neighbor data point among score vectors, we can find k-Nearest neighbor object classes for that image and assign object classes successively in the same order. Here for finding best guess object classes for any image, instead of looking at the score vectors of the (training data points) Nearest Neighbors of image, we look at the actual ground truth class labels of the score vectors to assign best guess object class to the input image. The same is implemented in our 'query_score_knn.m' MATLAB function. The function accepts an image, number of object classes of classifier, and the distance method to be used for finding k-Nearest Neighbor. It returns the best guess object class, and subsequent best guess object classes for the image and similarity of each object class with that image.

We also propose two other linear methods of image class classification (without using classifier) which are based on clustering. One way is to take the FC7 vector of the input image and compare it with object class representatives (which happens to be centroid of all FC7 vectors of an object class) to find possible best guess object class. It can be done by finding k-Nearest Neighboring object class centroids for input image FC7 vector. It gives decent results for those object classes close to each other linearly in higher dimensions. But only in rare case, it can give better result than our standard image classification experiment.

Other way is to take FC7 vector of input image and find its 1-Nearest Neighboring centroid obtained from k-means clustering and map all the object classes of that cluster to the given image. It simple makes use of semantic relationship present among object classes in linear sense and isn't explicitly used for image classification for finding best guess object class. Rather it can be used for giving a fixed set of object classes which are closer to the input image than any other image. The same is implemented in MATLAB function 'query_kmeans.m' which accepts an image and distance method to be used for finding 1-nearest neighbor and returns a list of object classes for that cluster centroid to which it was mapped.

To explicitly see which object class is similar to which other object classes of PASCAL dataset, we can efficiently use the similarity matrix. In the function 'query_similarity.m', which accepts an object class and number of similar object classes to be displayed as input and returns the similar objects and similarity of each object class with the input object class.

5.1 RESULTS AND ANALYSIS

We take a Scoobydoo (a popular cartoon's image of a dog) as input image and give it to various functions implementing various algorithms discussed above. The function 'query.m' implementing simple image classification algorithm fails to classify it as a dog, rather classifying it as a dining table. The function 'query_fc7_1nn.m' happens to fail even worse in this case by classifying it as a potted plant. However, the non-linear function 'query_score _1nn.m' outperforms the standard image classification algorithm by correctly classifying it as dog. The other non-linear function, 'query _score_knn.m', in this cases, matches the accuracy of standard image classification algorithm by misclassifying it as a dining table.

Next we give it an image of a UFO and a small image of person standing below it. Except 'query_fc7_1nn.m' function, which correctly classifies it as

Figure 7: Test Image Scooby Doo

Figure 8: Test Image UFO

an aeroplane, all other algorithms, including the standard image classification algorithm classifies it as a person (for very small instance of person present in the image).

Next we give an image of iPhone to all functions. The standard image classification algorithm and 'query_score_1nn.m' classifies it incorrectly as bottle while others classify it correctly as a tvmonitor.

In all the experiments we can see that the algorithms suggested are either at par or better than standard image classification algorithm used to find best guess object class for an input image. And all of this was done because of semantic relationship among object classes present in FC7 and/or score vectors. They can be explained as follows:

In all the algorithms suggested, we are trying to map an input image to nearest data point from the training set and then try to classify it. The main motivation for doing this is because as seen from the clustering algorithms above,

Figure 9: Test Image iphone

the FC7 layers have features showing semantic relationship among object classes. This is ideology implemented in 'query_fc7 _1nn.m' in linear sense.

Now in non-linear sense, when we use the classifier and then find its neighboring training data point, along with taking the advantage of presence of semantic relationship among object classes, we also benefit from the fact that the classifier is hard trained as per its training data points. Hence instead of directly classifying the image as per its score vector generated, it would be better if we classified the image as per some of its nearby training data point (in linear or nonlinear sense) for which the classifier has already been trained to do. In worst case, if standard image classification fails, this will also fail, but mostly, this classifies better than the standard classification algorithm.

This ideology is implemented in function 'query_score_1nn.m'. However, one point that we are missing here is, instead of classifying an image only according to its 1-Nearest neighbor present, it would be better to find best guess object class for image as per k-Nearest neighboring different object classes (and not k-Nearest neighboring different data points). The object classes of those k-nearest neighboring object classes are determined directly from the ground truth label. So this eliminates the error due to classifier that might have come in between while trying to generate scores for those training k-Nearest Neighboring object class data points.

The last two algorithm work equally well in most cases relying on a certain tradeoff. The query_score_knn.m function works on the ideology that any error due to misclassification of trained classifier should be removed by directly taking ground truth labels of the training data-point into account. However the previous algorithm makes use of the fact that if a hard trained classifier is misclassifying a training data point, then it is only because that data point has more features or characteristics of some other object class as compared to its original object class. This can be helpful in classifying test images, which might have varied type of characteristics, not previously seen by the classifier. Hence in terms of performance both these algorithms are working quite well, competent with each other.

6 CONCLUSION

Deep Networks are state-of-the-art methods for learning problems in perception. They are known to work well in capturing the semantics in audio/visual inputs. Specifically, in vision, a class of deep networks - Convolutional Neural Networks (CNNs), have recently surpassed the recognition accuracies of many machine learning methods. The individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. Generally, in image classification, the output from the last fully connected layer is fed to a classifier to identify the object classes in the image. This project studies the semantic relationship between different object classes as brought out by the outputs of the FC layers. This involved analyzing the outputs of FC layers and studying their relation with the final object labels. We have performed independent unsupervised and supervised clustering on the outputs of FC layers and visualized the relationship between clusters or groups of classes formed by FC7 layer outputs and their correspondences with the final object classes.

We have also proposed few algorithms which are variant of standard image

classification and have shown that these algorithms will perform either equivalent or better than the standard image classification algorithm in most cases. Along with visualizations like bipartite graph obtained from kMeans clustering and dendrogram plot using agglomerative clustering, we use the concept of similarity matrix to actually bring out semantic relationship among object classes in most logical sense. And the same was depicted by its corresponding dendrogram plots.

In future, one could adopt a variety of non-linear methods, which we haven't explored to bring out semantic relationship among object classes, which also may further improve image classification accuracy.

References

[1] Convolutional neural networks (lenet). http://deeplearning.net/tutorial/lenet.html. Accessed: 2015-05-01.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[3] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237, 2011.

[4] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 891–898. IEEE, 2014.