On the Representativeness of Convolutional Neural Networks Layers

Dario GARCIA-GASULLA [a,1], Jonatan MORENO [a] Raúl RAMOS-POLLÁN [c]
Romel CASADIEGOS BARRIOS [c] Javier BÉJAR [b] Ulises CORTÉS [a,b]
Eduard AYGUADÉ [a,b] Jesús LABARTA [a,b] Toyotaro SUZUMURA [a,d]

[a] *Barcelona Supercomputing Center, SPAIN*
[b] *Universitat Politècnica de Catalunya - Barcelona TECH, SPAIN*
[c] *Universidad Industrial de Santander, Bucaramanga, COLOMBIA*
[d] *IBM T.J. Watson Research Center, USA*

Abstract. Convolutional Neural Networks (CNN) are the most popular of deep network models due to their applicability and success in image processing. Although plenty of effort has been made in designing and training better discriminative CNNs, little is yet known about the internal features these models learn. Questions like, what specific knowledge is coded within CNN layers, and how can it be used for other purposes besides discrimination, remain to be answered. To advance in the resolution of these questions, in this work we extract features from CNN layers, building vector representations from CNN activations. The resultant vector embedding is used to represent first images and then known image classes. On those representations we perform an unsupervised clustering process, with the goal of studying the hidden semantics captured in the embedding space. Several abstract entities untaught to the network emerge in this process, effectively defining a taxonomy of knowledge as perceived by the CNN. We evaluate and interpret these sets using WordNet, while studying the different behaviours exhibited by the layers of a CNN model according to their depth. Our results indicate that, while top (*i.e.*, deeper) layers provide the most representative space, low layers also define descriptive dimensions.

Keywords. Convolutional Neural Networks, Visual Embeddings, Unsupervised Learning

1. Introduction

Deep learning (DL) networks are representation learning techniques [6], which build a very rich descriptive language by processing high-dimensional data. In the context of image data, Convolutional Neural networks (CNN) have shown remarkable performance, approaching human-level on object detection, segmentation and classification. Nevertheless, little is yet known about the nature of the models learnt by CNN, or how to exploit them. In this paper we seek to provide insight into this topic, by exploring the vector embedding space that originates from internal CNN activations. Building vector repre-

[1]Corresponding Author: Omega building, Office 207, C/Jordi Girona, 1-3, Campus Nord UPC, Barcelona, 08034, SPAIN. E-mail: dario.garcia@bsc.es

sentations of images and then classes (as defined in §3), we evaluate clusters that can be found unsupervisedly in that space (shown in §4), considering both their interpretability (through WordNet mappings and visual representation) and the relevance of CNN layers given their depth.

2. Related Work

Some previous works have explored the properties of CNN models through the extraction of activations. [2] were among the first to do so, using features extracted from various layers to identify classes from different data sets. Significantly for the work here presented, authors briefly explore the clustering capability of various layers, noticing that top layers can discriminate outdoor from indoor classes. Nevertheless, no thorough evaluation on this behaviour is performed. In [9], authors use the first fully connected layer of a model, for attribute detection. By applying a support vector machine on the 4096 composing features, certain abstract classes can be identified, such as *is male*, *has glasses*. Since only a top, fully connected layer is used, the number of descriptive features is relatively small. Middle convolutional layers of a CNN provide millions of features, which may also help in the process. These middle layers, as shown by [7] are relevant enough as to be reusable across data sets.

3. Methodology

Given an input image, each layer of a trained CNN model is filled with activation values as the image is processed forward within the network. Each of those activations corresponds to the occurrence (or lack of) of a pre-trained pattern (*i.e.*, a filter) in a specific location. The combined activations obtained within a given layer is therefore a representation of the input image at that specific CNN layer, capturing what the CNN perceives at that layer's depth, and can be explored separately.

Our methodology (first introduced in [1]) starts with the definition of a CNN model and the identification of a subset of layers of interest. Using those, we process a labelled data set of images, producing a vector representation for each image. Once all image vectors are computed, we obtain high-level abstractions by aggregating all vectors sharing the same label. The result of this process are class vectors, each corresponding to a well defined concept as labelled in the original dataset. To study the relations among those class vectors we compute their vector distances, and build the corresponding distance matrix. This whole process is detailed in the remaining of this section. The resultant distance matrix is then used to explore the representativeness of the approach, and the semantics hidden in the model, as described in §4.

3.1. Image Vectors

An image processed through a CNN produces activations on each of its layers. By capturing and extracting those activations we build vector representations of images (*i.e.*, a vector embedding). The number of activations per layer on typical CNN architectures may range between a few thousands (highest, deepest layers) to a few millions (lower

layers). As a result, the image vector representations obtained from the consideration of several layers can easily be composed by millions of values.

For a given data set of labelled images, we process each of those images independently through a pre-trained CNN model to generate their *image vectors*. Since image data sets typically contain thousands of images, in our experiments we will work with thousands of vectors (one for each image) composed by millions of values (one for each CNN feature captured). We explore these structures by considering each of those values as a distinct and independent data descriptor, only keeping track of which values correspond to which layers in the model. Formally,

$$\begin{aligned} &Given\ a\ set\ of\ layers\ L,\ an\ image\ vector\ IV = (act_1, ..., act_n), \\ &\qquad where\ \forall act_i \in IV,\ act_i \geq 0\ and\ layer(act_i) \in L \end{aligned} \tag{1}$$

3.2. Class Vectors

The capability of a single image to be representative of a class is limited by varying aspects such as perspective, illumination, scale, context, and many others. Indeed, a single two dimensional image only constitutes an incomplete description of an abstract entity (*i.e.*, a class). To extend such incomplete description, trying to consider as many visual aspects of an entity as possible, we consider the aggregation of the evidence provided by several different images. In the context of the vector embeddings we use here, we combine several image vectors belonging to the same class to generate what we call *class vectors*. Since we consider vector values as independent descriptors, we generate the class vector by independently aggregating each of the vector values through an arithmetic mean. Formally, given a set of n image vectors $IV = \{IV_1, ..., IV_n\}$ of a shared class C, all of equal length m ($\forall\ IV_i \in IV,\ |IV_i| = m$), the corresponding class vector CV is defined as follows:

$$CV = (act_1, ..., act_m), where\ \forall\ act_i \in CV,\ act_i = \frac{\sum_{x=1}^{n} IV_x[act_i]}{n} \tag{2}$$

Notice the resultant class vector have the same size than the aggregated image vectors. After aggregation, class vector values are normalized to unit by layer to balance the evidence provided by each layer on a single class vector. The activations of each layer are normalized using the corresponding euclidean norm of that layer. Qualitatively, a class vector represents a list of the visual features commonly found in an abstract class (*e.g.*, an elephant), as perceived by the deep learning network.

3.3. Distance Matrices

Depending on how many and how big are the layers extracted from the CNN, an aggregated and normalized class vector may contain millions of values. To compare any given pair of class vectors, we compute the similarity between their activations using the cosine measure. Formally:

Given two class vectors CV_1 and CV_2, composed by layers of equal number and size, the similarity between both is :

$$sim(CV_1, CV_2) = \frac{CV_1 \cdot CV_2}{||CV_1||\ ||CV_2||} \tag{3}$$

Using this similarity measure, for our experiments we will build a triangular distance matrix of size MxM given a set of M class vectors. This matrix will be used to compute clusters and correlations with WordNet distances.

4. Evaluation

Next we study the representativeness of the computed class vectors and distance matrices, obtained as described in §3. For that purpose we use the ILSVRC 2012 validation data set, containing 50,000 images labelled into 1,000 categories. Since the images of this data set are mapped to WordNet labels (*e.g.*, synsets) we can use WordNet as a validation tool. WordNet is a lexical database with no inherent information on the visual aspect of concepts (*i.e.*, synsets). However previous research showed a correlation between the space its taxonomy defines and the space defined by traditional bag of visual word models [12].

First, we perform an unsupervised clustering process on the class vectors, evaluating the quality of the clusters according to WordNet as described in §4.2. Then we analyze the correlation between class vectors distances and WordNet similarity metrics. This will give us insight into the semantics of the embedding space defined by the class vectors. Finally, to illustrate the contents of the embedding space, we will generate visualizations of selected class vectors through a deconvolution process.

4.1. CNN Models and Data

Our goal is to build vector representations of images which are as rich in features as possible, with the goal in mind of enabling machine learning applications on top of those representations. Accordingly, we choose a CNN network capable of identifying many complex visual features from images. In most experiments shown in this section we use the GoogLeNet [11] architecture, a 22 layers CNN that won the ILSVRC 2014 visual

Table 1. For the GoogLeNet architecture, layers extracted and their size

Layer	inception_3a/output	inception_3b/output	inception_4a/output	inception_4b/output
Size	200,704	376,320	100,352	100,352
Layer	inception_4c/output	inception_4d/output	inception_4e/output	inception_5a/output
Size	100,352	103,488	163,072	40,768
Layer	inception_5a/output			
Size	50,176			

Table 2. For the VGG19 architecture, layers extracted and their size

Layer	conv2_2	conv3_4	conv4_4	conv5_4
Size	1,605,632	802,816	401,408	401,408

recognition challenge [10] for classifying images. We used the pre-trained model available in the Caffe deep learning framework [5], trained with 1.2M images of the ImageNet test set for the task of discriminating the 1,000 ImageNet hierarchy categories. Additionally, for the inter-architecture comparison of §4.2.1, we also use the VGG19 architecture. This architecture won the second place in the classification challenge at ILSVRC 2014.

For each of the 50,000 images obtained from the ImageNet 2012 validation set we extract a vector representation. Then we aggregate these image vectors to obtain the class vectors corresponding to the 1,000 labelled classes (*i.e.*, 1,000 class vectors) of the ImageNet hierarchy. For the GoogLeNet architecture, image vector are built from the output layers of the inception modules (see Table 1), having 1,235,584 features in total. For VGG we use a similar approach, extracting the output of all convolutional layers except the first one (see Table 2), resulting in 3,211,264 total features. In both cases we skip the first convolutional layer because the patterns found at that level are too simple (*e.g.*, straight lines [11]) to be of use for the representation of abstract classes.

4.2. Clustering of Class Vectors

To study the semantics captured in the embedding vector space defined in §3, we compute the clusters that can be found unsupervisedly in that space. For that purpose we use distance matrices built as previously defined, obtained from a set of 1,000 class vectors. Our goal is also to explore the different characterizations provided by the different layers found in a CNN, which is why we built different distance matrices corresponding to the extraction of different layers of the CNN.

Since we initially ignore the nature of the embedding space, we consider the existence of a variable number of clusters. Using spectral clustering [13], we identify k clusters, between 2 and 19, for each distance matrix. Each k-clustering provides an abstraction of image classes, but given the sub-symbolic nature of the features defining those classes (feature activations of a CNN) little can be said about their nature in sym-

Table 3. Distance matrix among 1,000 class vectors, considering nine different GoogLeNet layers (from inception/3a_output to inception/5b_output). Compute from 2 to 8 clusters through spectral clustering. For each clustering process and cluster identified, Table shows the WordNet synset with the best F1 measure. When relevant, second best synset by F1 measure is shown in parenthesis. Clustering experiments are independent and their relation has not been explored.

2			Living thing	Artifact				
3			Living thing	Artifact	Conveyance			
4		Mammal	Living thing	Artifact	Conveyance			
5		Mammal	Living thing	Artifact	Conveyance	Artifact (clothing)		
6	Bird	Mammal	Matter (reptile)	Artifact	Conveyance	Artifact (clothing)		
7	Bird	Mammal	Matter (reptile)	Instrumentality	Wheeled vehicle	Clothing	Craft	
8	Bird	Mammal	Matter (reptile)	Instrumentality	Wheeled vehicle	Clothing	Craft	Structure

bolic terms. To try to characterize those clusters symbolically we use the WordNet hypernym/hyponym lexical taxonomy to identify the WordNet synset that better describes each cluster. In detail, since every image class is mapped to a WordNet synset, we can compute a F1 score for every pair of WordNet synset and cluster, considering that a synset S applies to an image class IC if the synset associated with IC is S or a hyponym of S. For example, given a 2-clustering, if 90% of image classes which are hyponyms of the dog synset can be found within cluster A, and 85% of image classes of cluster A are hyponyms of dog, the F1 measure of dog for cluster A will be 0.87. We use the synset obtaining the best F1 measure for each cluster as that cluster label. Notice two clusters may share the same label.

To illustrate the type of cluster labelling that we obtain, Table 3 shows the results when using all available layers from the GoogLeNet architecture. Briefly, the first 2-clustering separates between living things and non-living things. Significantly, the same first-level categorization found by more traditional visual models in [12]. Means of transportation (*i.e.*, conveyance) are separated next, at 3-clustering. Later on, these are further separated between crafts (*i.e.*, ships and aircrafts) and wheeled vehicles. A cluster for birds and clothes is also reliably found. On the other hand, there is a cluster of mixed artifacts and instruments on all k-clusterings, likely due to the less coherent visual features of objects, and to the lack of specificity of WordNet synsets for characterizing items.

We evaluate the quality of the discovered sets numerically, using this labelling process. For each k-clustering, we compute its quality as the mean F1 score obtained on all its clusters. Results indicate that clustering quality decreases as the number of clusters increases, as shown in Figure 1. The best results are obtained for the 2-clustering, with an F1 score close to 0.9. To study the relevance of the various layers found in a CNN model, we separately consider the case of using all nine layers shown in Table 1, using only the

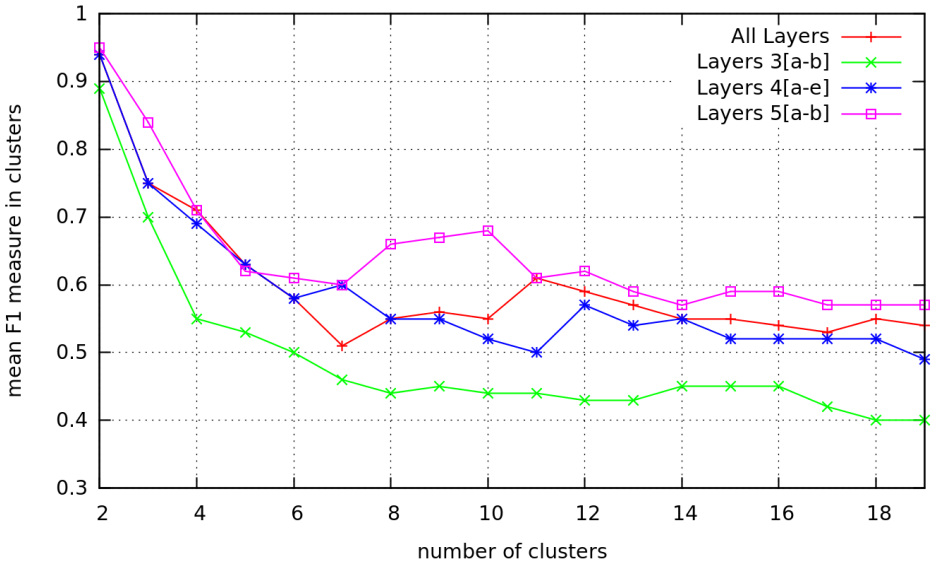

Figure 1. Given the distance matrix among class vectors built using all layers, layers 3a and 3b, layers 4a to 4e and layers 5a and 5b of the GoogLeNet architecture, chart shows the mean F1 score for WordNet on a variable number of clusters (from 2 to 19) PR curve of a LP score on two different graphs. Grey area shows the CAUC.

3a and 3b layers (layers found near the bottom of the CNN), using the 4a to 4e layers (layers found at the middle of the CNN), and a using the 5a and 5b layers (those with maximum discriminative power). Results indicate that top layers (5a and 5b) provide the best clusters, although similar results are obtained when using all layers. Since top layers capture the most complex visual patterns these also provide the most descriptive power. Nevertheless, as shown in Figure 1, middle and low layers also convey a relevant amount of visual semantics, since one can also find high quality clusters using them.

4.2.1. Comparison Between Architectures

A question that may arise is how dependant are these results on the specific CNN architecture being used. To investigate that point we apply the same methodology to the VGG19 network, using the layers defined in Table 2. After building the image vectors, class vectors and its corresponding distance matrix, we compute the same k-clustering and F1 scores. The comparison between the results of the VGG19 and the GoogLeNet architectures can be seen in Figure 2

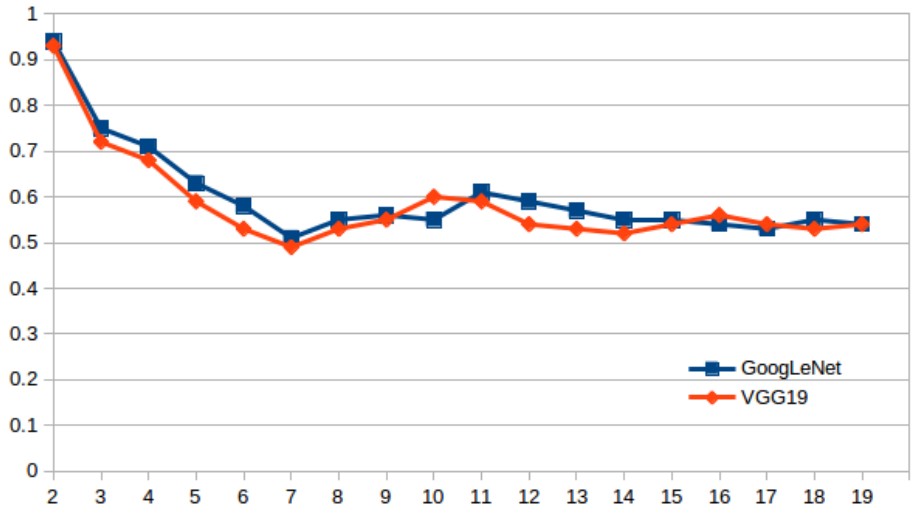

Figure 2. Given the distance matrix among class vectors built using all layers, layers 3a and 3b, layers 4a to 4e and layers 5a and 5b of the GoogLeNet architecture, chart shows the mean F1 score for WordNet on a variable number of clusters (from 2 to 19).

4.3. Visualization of Class Vectors by Layer

The unsupervised clustering process shown here can be understood as a methodology to automatically learn high-level concepts through CNN. In our experiments we are able to interpret clusters thanks to the WordNet mappings of classes. However on a fully automatic process that information may not be available. In other words, the CNN may be able to identify new entities, but a human user may be unable to understand the nature of those entities, since their representations are just vectors.

To enable a fully independent learning process, we explore the generation of images from class vectors. Inspired by the process described in [3], we adapt their implemen-

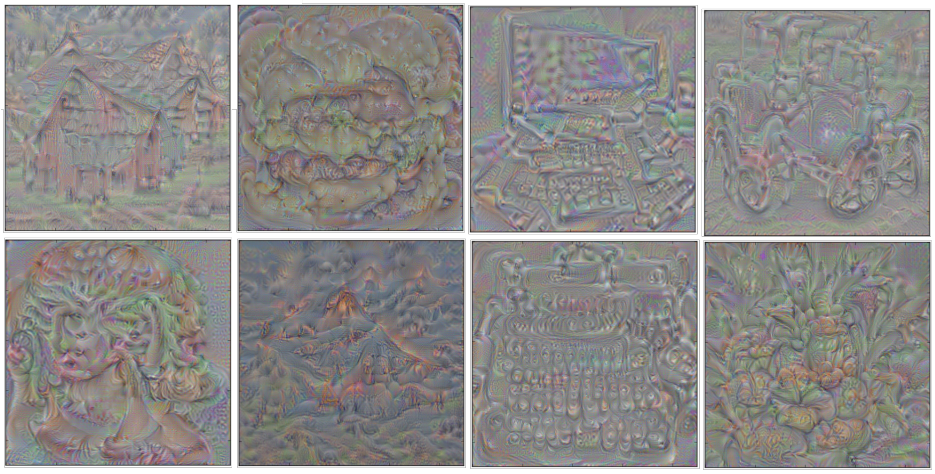

Figure 3. Visual reconstruction of various class vectors. From top to bottom, left to right, classes shown correspond to the WordNet synsets: n02793495_barn, n07697313_cheeseburger, n03642806_laptop,_laptop_computer, n03777568_Model_T, n04584207_wig, n09472597_volcano, n04505470_typewriter_keyboard and n07753275_pineapple,_ananas. Best seen in color.

tation to reconstruct content instead of texture. To do so we compute the loss function on the activations instead of on the Gram matrix. The resultant process generates visual representations (*i.e.*, full images) corresponding to the contents of the class vectors. A sample of those visualizations are shown in Figure 3. For these visualizations only top layers were used (layers 5a and 5b) since these maximize visual interpretability.

5. Discussion

Table 3 shows how a hierarchy of concepts emerges from our methodology making intuitive sense (*i.e.*, finer grained concepts are separated as the number of clusters are increased) with close correspondence to WordNet's concept ontology as implied by its hypernym/hyponym relationship. Although our validation methodology is flawed (*e.g.*, WordNet measures linguistic similarities, while we compute visual similarities) results indicate that certain abstractions can be made almost perfectly (living things vs non-living thing).

The partitioning of image vectors into subsets obtained from different CNN layers shows coherent behaviour as seen in Figure 1, providing evidence that layers holding more abstract information might be more useful for representing concepts. Nevertheless, all layers show descriptive power. Fine tuning this aspect might help reduce the computational resources for storing image vector and producing and clustering class vectors.

Rather surprisingly, results are very similar for the two CNN architectures tested, as shown in Figure 2. Even though both architectures have significantly different topologies, their embedding spaces seem to be consistent. Since both networks were trained with the same data set, our hypothesis is that the features needed to optimally discriminate the training data acts as a centroid during training, causing both network vector embeddings to be correlated. This is further supported by the fact that for the VGG19 network one obtains a hierarchy of clusters very similar to the one shown in Table 3.

5.1. Conclusions and Future Work

This work set forth to extract semantically meaningful representations from images using the activations of pre-trained CNNs. We hereby show that these image vectors hold correlated semantics to those of WordNet's hypernym/hyponym relations, and consider this as a first cornerstone piece of evidence for their potential utility for further tasks ranging from machine learning to image understanding. The methodology described here may accept many variations including the usage of different CNNs to extract image vectors, distance metrics, weighting schemes for activations extracted from different layers, clustering options, *etc.*Whereas it is yet to be understood how these variations behave under different contexts, we believe this fact fuels the potential applicability of the work here presented. It also defines our priority lines of future work.

Even though the data used for evaluation (validation data of ImageNet 2012) was not used to trained the network models, we are aware that the classes of objects are common. To explore the limits of this approach, we intend to evaluate the clustering of data completely disjoint from the training data. In this context we believe that explicit ontologies might provide more robust validation (WordNet is not available for all data sets, and regardless it is not a perfect baseline) and would like to see our method tested against other semantic structures or ontologies, even specialized ones, such as SNOMED [4] for medical image datasets. Finally, we expect image vectors to be used in specific machine learning or image processing tasks providing end to end validation of our work.

Acknowledgements

This work was partially supported by the IBM/BSC Technology Center for Supercomputing (Joint Study Agreement, No. W156463), by the Spanish Government through Programa Severo Ochoa (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project and by the Generalitat de Catalunya (contracts 2014-SGR-1051).

References

[1] Garcia-Gasulla, D., Béjar, J., Cortés, U., Ayguadé, E. and Labarta, J. *A visual embedding for the unsupervised extraction of abstract semantics.* arXiv preprint arXiv:1507.08818 (2015).

[2] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T. *Decaf: A deep convolutional activation feature for generic visual recognition*. arXiv preprint arXiv:1310.1531 (2013).

[3] Gatys, L., Ecker, A. and Bethge, M.. *Texture synthesis using convolutional neural networks.* In Advances in Neural Information Processing Systems **29** (2015), 262-270.

[4] Benson, T. *Principles of health interoperability HL7 and SNOMED*. Springer Science & Business Media (2012).

[5] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T. *Caffe: Convolutional Architecture for Fast Feature Embedding*. arXiv preprint arXiv:1408.5093 (2014).

[6] LeCun, Y., Bengio, Y. and Hinton, G. *Deep learning*. Nature **521(7553)** (2015), 436-444.

[7] Oquab, M., Bottou, L., Laptev, I. and Sivic, J.. *Learning and transferring mid-level image representations using convolutional neural networks*. In Computer Vision and Pattern Recognition (2014), 1717-1724.

[8] Pedersen, T., Patwardhan, S. and Michelizzi, J. *WordNet:: Similarity: measuring the relatedness of concepts*. In Demonstration papers at hlt-naacl (2004), 38-41.

[9] Razavian, A., Azizpour, H., Sullivan, J. and Carlsson, S. *CNN features off-the-shelf: an astounding baseline for recognition*. In Computer Vision and Pattern Recognition Workshops (2014), 512-519.

[10] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. and Fei-Fei, L. *ImageNet Large Scale Visual Recognition Challenge*. International Journal of Computer Vision **115(3)** (2015), 211–252.

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. *Going deeper with convolutions*. arXiv preprint arXiv:1409.4842 (2014).

[12] Deng, J., Berg, A., Li, K., and Fei-Fei, L. *What does classifying more than 10,000 image categories tell us?* In Computer Vision–ECCV (2010),71-84.

[13] Ng, A., Jordan, M. and Weiss, Y. *On spectral clustering: Analysis and an algorithm* In Advances in neural information processing systems **2** (1998), 849-856.