ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF SCIENCE AND LETTERS

GRADUATION PROJECT

Hierarchical Approach in Large Scale Image Recognition with Deep Learning

Azmi Can Özgen

Mustafa Can Uslu

Onur Kaplan

Physics Engineering Department

2015-2016 Fall

Abstract

Today, machine learning is a research area in computer science with increasing number of applications in a variety of domains. Besides, our understanding of its theoretical basis is improving continuously. Also, deep learning is a branch of this area that has become very popular and powerful with impressive results in recent years.

In this research, we used convolutional neural networks in which one of the most successful technique in image recognition task, on three different datasets and analyzed them.   The dataset, ImageNet was the main objective. We used ImageNet with available algorithms and frameworks. Nevertheless, our own architectures and techniques helped us at understanding these available structures.

First chapter is an introduction that tells about the datasets and the objectives. The second chapter consists of theoretical expressions about the architectures and general specifications of deep neural networks and convolutional neural networks. Third chapter is about our approach to ImageNet classification problem. While there are comparisons about the accuracies and properties of different architectures in the fourth chapter, all these results are evaluated in the last chapter.

The work can be found at our GitHub repository.

Acknowledgment

This BSc thesis is based upon a mutual study of a collaborative team. Many people contribute supportive ideas on this work. Our supervisor Ahmet Levent Subaşı and our mentor Berkin Malkoç come in the first place of this list. With their precious supports we have placed as one of the innovative ideas in ImageNet Large Scale Visual Recognition Challenge 2015. Also, special thanks to Altan Çakır and Tolga Birkandan for their beneficial comments about this paper. We are appreciated to NVIDIA and IBM supported us with resources in the project. We are thankful to Miletos for their hospitality. Also, we would like to thank who illuminates us in the darkest time of the nights, Thales.

The last but not the least, we appreciate to our families for being alongside us always.

Contents

Abbreviations

CV	: Computer Vision
AI	: Artificial Intelligence
ANN	: Artificial Neural Network
CNN	: Convolutional Neural Network
RL	: Reinforcement Learning
SL	: Supervised Learning
USL	: Unsupervised Learning
DL	: Deep Learning
ILSVRC	: ImageNet Large Scale Visual Recognition Challenge
SGD	: Stochastic Gradient Descent
CONV	: Convolutional Layer
POOL	: Pooling Layer
FC	: Fully Connected Layer
LRN	: Local-Response-Normalization Layer
DROP	: Dropout Layer
ReLU	: Rectified Linear Unit
MNIST	: Mixed National Institute of Standards and Technology
CIFAR	: Canadian Institute for Advanced Research
SLIC	: Simple Linear Iterative Clustering
CPU	: Central Processing Unit
GPU	: Graphical Processing Unit
OCR	: Optical Character Recognition

1. Introduction

1.1 Computer Vision

Computer Vision (CV) is a computer science discipline that examines the process of extracting information from images via computers. Inputs might be in the form of not only pictures but also video sequences or multi-dimensional images such as medical scans. Some of the sub-domains of CV are scene classification, motion estimation, image restoration, object pose estimation, video tracking and object detection.

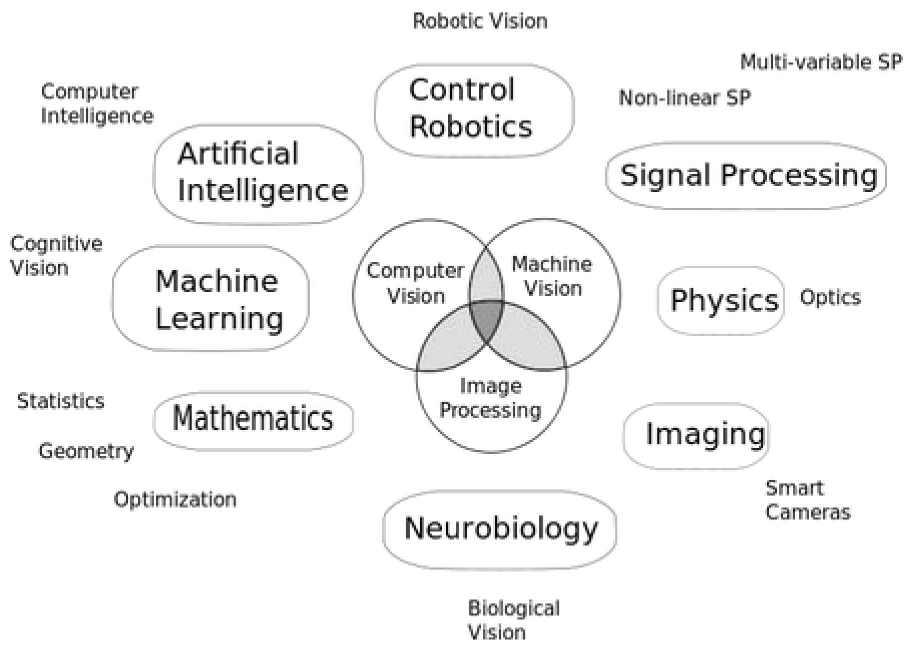


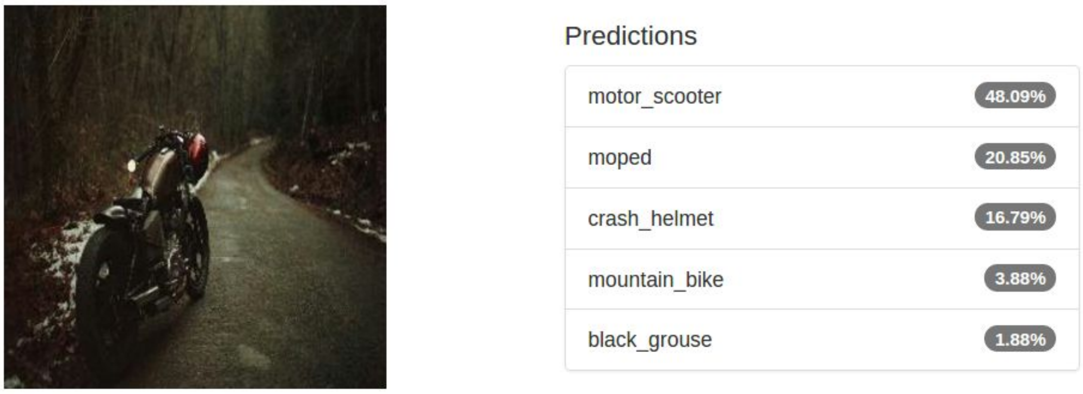
Figure 1.1 The relationship of CV and other fields[1]

Artificial Intelligence (AI), optics, neurobiology and various disciplines have a lot in common with CV. Pattern recognition and learning techniques from AI, interaction of light with surfaces from optics and neural networks from neurobiology carried CV up to now. Also those interactions create new research fields in CV such as Artificial Neural Networks (ANNs) which is the main concern of this thesis.

[1] Wikipedia, the free encyclopedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/File:CVoverview2.svg, Retrieval date 13 Jan 2016.

1.1.1 Object Detection

Object detection consists of two parts as object recognition and localization. Object recognition is determining the class of object, whereas localization is specifying the location of object. This concept is used in many applications like systems for automation, monitoring and security surveillance, Optical Character Recognition (OCR), Face Recognition and License Plate Matching and so on.



Predictions

motor_scooter	48.09%
moped	20.85%
crash_helmet	16.79%
mountain_bike	3.88%
black_grouse	1.88%

Figure 1.2 Object recognition result of Hierarchical GoogleNet on DIGITS

Since objects may look different in lighting, color, viewing direction, size and shape, there are several methods for object detection. One of the most common technique is edge detection.

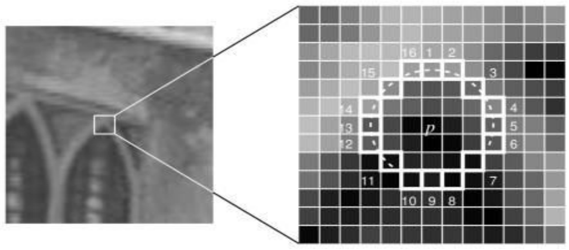


Figure 1.3 OpenCV FAST Algorithm for corner / edge detection[2]

As seen in Figure 1.3, corner or edge detection is one way to separate specific objects in an image. Fundamental purpose is to detect object as in the Figure 1.4 whether

[2] FAST Algorithm for Corner Detection — OpenCV 3.0.0-dev documentation. (n.d.). Retrieved from http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html, Retrieval date 13 Jan 2016.

edge or color specifications are used or not. In an image each object may exist more than once or there may be too many different objects. Also, similar objects may be too close each other. Because of these various reasons object localization is a difficult task.

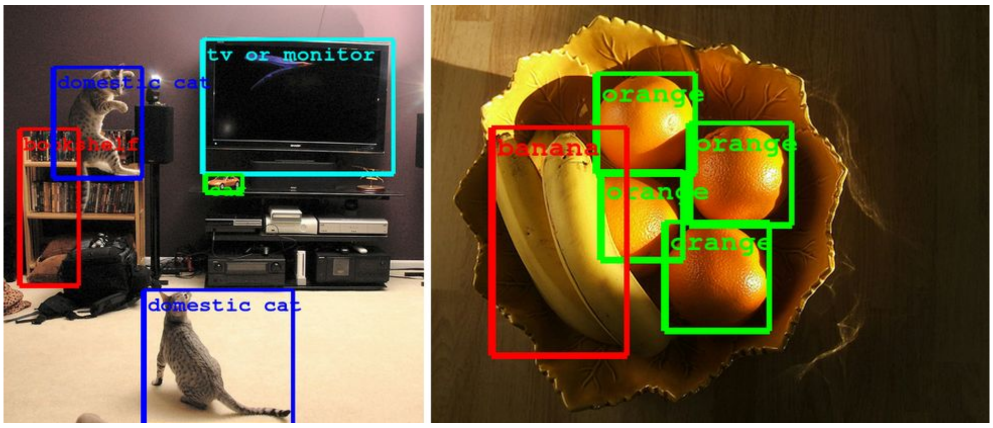


Figure 1.4 Object detection example with GoogLeNet[3]

1.2 Deep Learning and Artificial Neural Networks

1.2.1 Deep Learning

Deep learning (DL) is based on a set of algorithms that attempt to model high level abstraction in data by using multiple processing layers with complex structures[4]. Deep refers to high level of abstraction and non-linear transformations. By the help of high level abstraction, feature engineering of conventional machine learning can be left behind.

[3] Research Blog: Building a deeper understanding of images. (n.d.). Retrieved from http://googleresearch.blogspot.com.tr/2014/09/building-deeper-understanding-of-images.html, Retrieval date 13 Jan 2016.

[4] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. doi:10.1038/nature14539

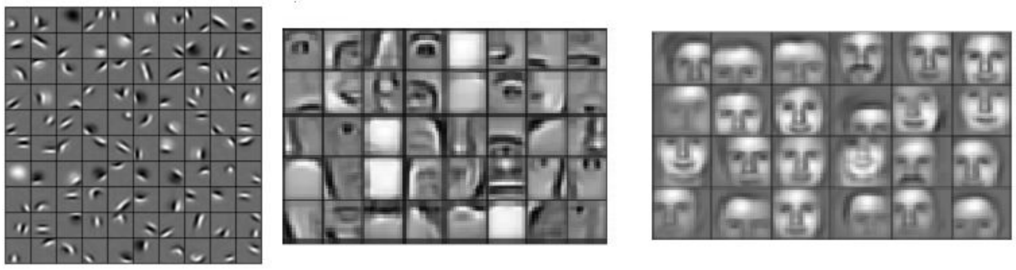


Figure 1.5 Learned hierarchical features from a DL algorithm[5]

DL can be divided into three following types as Supervised Learning (SL), Unsupervised Learning (USL) and Reinforcement Learning (RL). SL is learning from examples provided by an external supervisor. Data comes with its label. Algorithm just tries to find a mapping from input to output. However, USL is more similar to clustering. Algorithm should cluster the data according to features it detects. RL is similar to USL that have no labels with data. Yet, an objective, named score, about to be maximized is provided in RL. For example, in Super Mario Bros. game, the problem is that Mario ought to collect as much as coins. Objective is the score that can be seen in top left side of the image below since it is known that score increases as Mario collects coins.

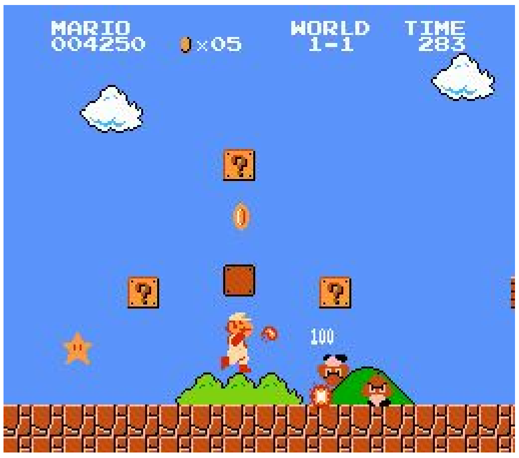


Figure 1.6 Mario collecting coins[6]

[5] Deep Learning in a Nutshell: Core Concepts | Parallel Forall. (n.d.). Retrieved from https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/, Retrieval date 13 Jan 2016.

[6] Retrieved from https://upload.wikimedia.org/wikipedia/en/5/50/NES_Super_Mario_Bros.png, Retrieval date 13 Jan 2016.

DL sounds like a recent research field, nonetheless Fukushima came up with Neocognitron in 1980[7]. Neocognitron is an ANN which is hierarchical and multi-layered. It has been used for OCR on handwritten characters and other pattern recognition tasks. It was also a great inspiration for Convolutional Neural Networks (CNNs). The first goal to achieve in DL history is the solution of the problem of training multi-layered networks. Not so long ago, LeCun and his team applied back propagation algorithm on recognizing handwritten ZIP codes on mail successfully[8]. After that, Frey showed that it is possible to train a network containing six Fully Connected Layers (FCs) and few hundred hidden units. Hinton and Dayan developed an algorithm named wake-sleep algorithm[9] that allowed Frey to train that network in two days.

Growing impact of DL on image recognition lets people to work on various problems such as speech recognition. In speech recognition task, input is audio instead of image and networks in image recognition are adapted to process audio instead of image. Late 1990s and first part of 2000s passed with the solution to overcome speech recognition problem. Meanwhile, different DL architectures developed such as CNNs, Recurrent Neural Networks, Deep Belief Networks etc. Nowadays, DL is taking a major place in industry.

[7] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, *36*(4), 193-202. doi:10.1007/bf00344251

[8] LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, 1, pp. 541–551, 1989.

[9] Hinton, Geoffrey E.; Dayan, Peter; Frey, Brendan J.; Neal, Radford (1995-05-26). "The wake-sleep algorithm for unsupervised neural networks". Science 268 (5214): 1158–1161. doi:10.1126/science.7761831.

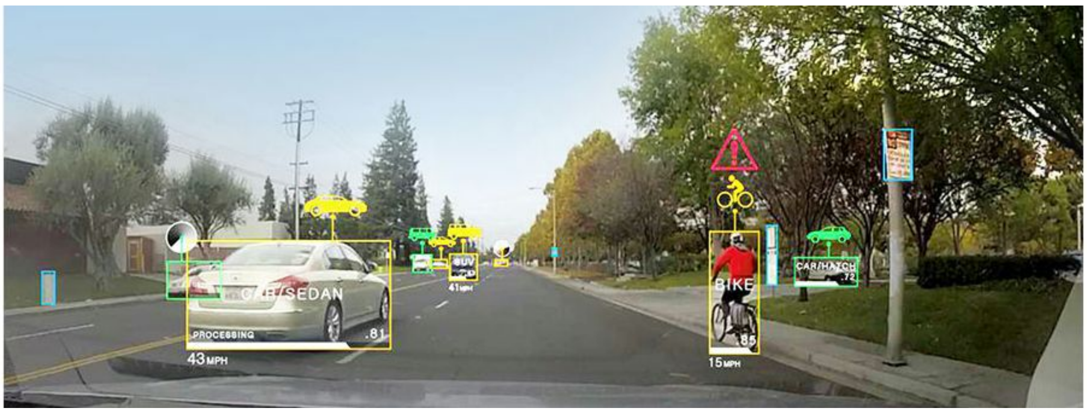


Figure 1.7 Self-driving cars using DL[10]

It is worthwhile to mention that the current progress in hardware technology is getting more important for DL. Increasing amount of calculations could not be handle by Central Processing Units (CPU). Fortunately, graphic cards with their own processors on them created a new era to DL. In the core of DL there is not much more than matrix operations which is quite suitable for parallel calculation. In short, Graphical Processing Unit (GPU) technology renewed interest of DL and made life easier.

1.2.2 Artificial Neural Networks

The goal of ANNs is to take a large dataset as training set, and then develop a system which can recognize new unseen examples. In other words, a uses the examples to automatically infer rules for recognizing other unknown examples.
As in all other machine learning techniques, ANNs work as in the diagram below.

[10] NVIDIA'S DRIVE PX Platform to Pave Way for Self-Driving Cars Using Deep Learning - Dataconomy. (n.d.). Retrieved from http://dataconomy.com/nvidias-drive-px-platform-to-pave-way-for-self-driving-cars-using-deep-learning/, Retrieval date 13 Jan 2016.

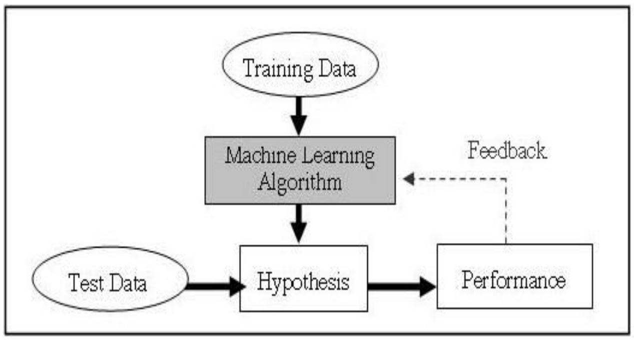


Figure 1.8 General machine learning diagram[11]

What this diagram tells us is that take as possible as more data, produce a model by training the algorithm on that data, test your model with some other data and finally adjust your model with the feedbacks that come from the of results of test data. The feedback is provided by an error term in the sense of a distance from the correct results. These steps continue iteratively until no longer improvement is obtained on our model. This generic machine learning process is also valid in DL identically. As we will see soon, questions like how you design the network architecture, how you measure the error at the output or how you minimize this error become very important. However, deep ANNs take on another very important part of the learning, feature extraction.

Normally, feature extraction requires different type of expertise for every specific dataset. This procedure is also called as 'feature engineering' since high level engineering is needed to preprocess the data before training. Yet, with the aid of deep ANNs, feature learning arise automatically without any specific domain expertise.

In feature learning process, multiple layers of nonlinear features are extracted and these features are passed into a classifier that combines all the features to learn correct labels. Non-linear features come out from nonlinear cumulative operations made by each neuron itself. Such features are shown for an image below.

[11] Retrieved from https://upload.wikimedia.org/wikipedia/commons/0/01/Machine_Learning_Technique..JPG, Retrieval date 13 Jan 2016.

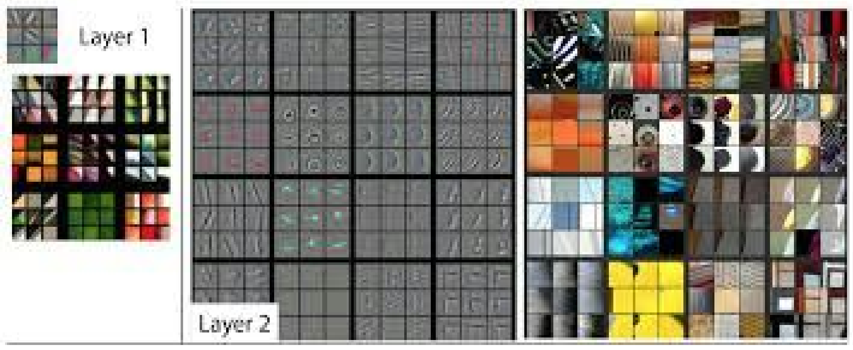


Figure 1.9 Some learned features in first two layers in a CNN for corresponding sample image patches[12]

As we said, shallow networks are inadequate at extracting complex features from detailed colored images whereas such very deep ANNs are more powerful at representing images in a very high dimensional feature spaces. In Figure 1.9, there are features like edges, blobs or lines extracted in the first two layers. However, these features are not enough to separate subtle patches of images even a human cannot distinguish. At some level of layers in CNN, features become more and more smaller and sharper as shown below.

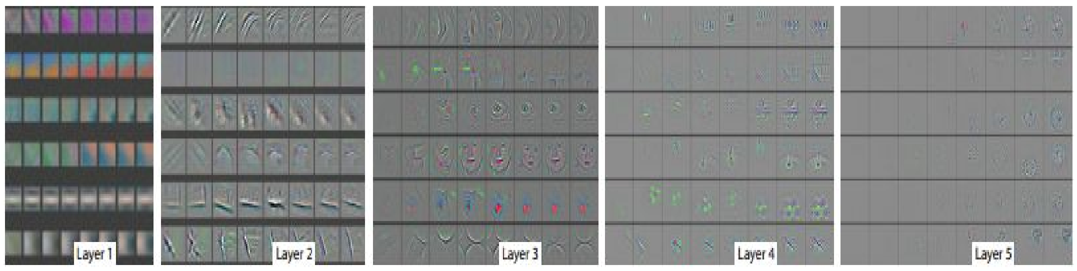


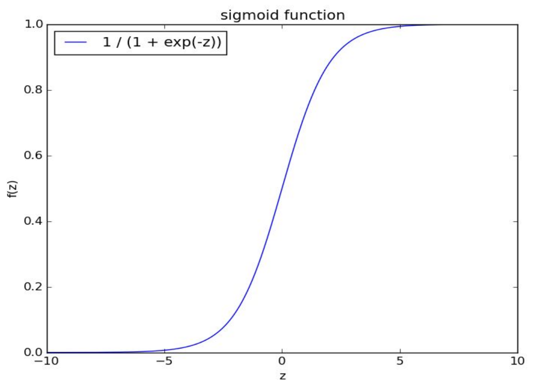
Figure 1.10 The evolution of features going deeper through layers[13]

Preliminary versions of deep ANN trials suffered from some problems such as the vanishing gradient problem where the gradients become too small to conduct a learning signal for very deep layers. But in recent years these problems were overcome with properly chosen activations functions combined with GPU based

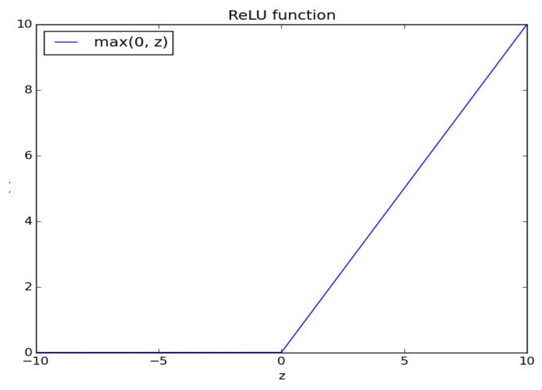
[12] Zeiler, Matthew D., and Rob Fergus. "Visualizing and Understanding Convolutional Networks." *Computer Vision – ECCV 2014 Lecture Notes in Computer Science* (2014): 818-33. Web.

[13] Zeiler, Matthew D., and Rob Fergus. "Visualizing and Understanding Convolutional Networks." *Computer Vision – ECCV 2014 Lecture Notes in Computer Science* (2014): 818-33. Web.

training to dozens of layers of non-linear hierarchical features. A comparison of activation functions is in Figure 1.11



a)



b)

Figure 1.11 a) Sigmoid activation function. Saturation in terminal points cause vanishing gradients and learning become drastically slower. **b)** Rectified Linear Unit (ReLU) activation function does not bound gradients so that, learning does not get slower through consecutive layers.

We summarized some general principles about deep ANNs up to now. Now, we are going into CNNs which is a commonly used type of deep ANNs.

1.2.2.1 Convolutional Neural Networks

Deep ANNs are such networks which are more robust than typical shallow ANNs at inferring rules between inputs and outputs. In this research, we focused on CNNs that are commonly used on the image recognition problems.

In Figure 1.12, a CNN architecture can be seen. As a general deep ANN, such a CNN composed of many layers and also each layer looks like has layers within. These are called feature maps. Moreover, there are several types of layer like Convolutional Layer (CONV), Pooling Layer (POOL), FC etc. We will cover the properties of these layers in the second chapter and what they do and how such a network is so successful at recognizing images, even at a large scale dataset.

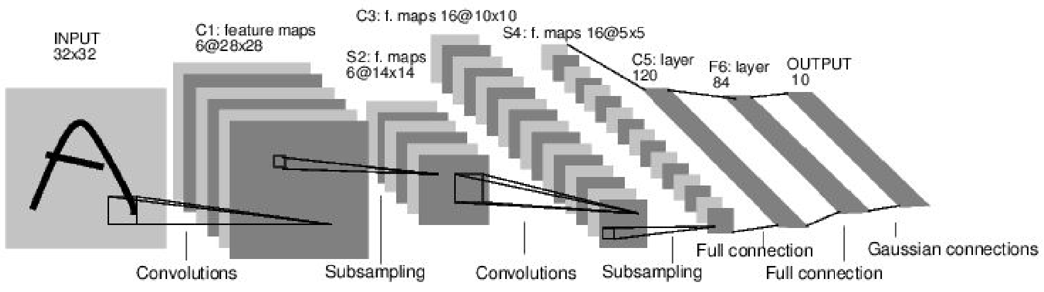


Figure 1.12 Architecture of LeNet-5, a CNN[14]

1.3 ImageNet and ImageNet Large Scale Visual Recognition Challenge

1.3.1 ImageNet

ImageNet[15] is an image dataset that prepared pursuant to WordNet[16] framework. Each node on ImageNet contains hundreds and thousands of images. There are 14,197,122 images and 21,841 indexed synsets[17]. Moreover, the dataset consists of nearly one million images with bounding box annotations to help localization problems. ImageNet project was started by Stanford Vision Laboratory from Stanford University[18] and sponsored by Princeton University, Google, A9[19].

[14] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi:10.1109/5.726791
[15] ImageNet. (n.d.). Retrieved from http://image-net.org, Retrieval date 13 Jan 2016.
[16] About WordNet - WordNet - About WordNet. (n.d.). Retrieved from https://wordnet.princeton.edu, Retrieval date 13 Jan 2016.
[17] ImageNet. (n.d.). Retrieved from http://image-net.org/about-stats, Retrieval date 13 Jan 2016.
[18] ImageNet. (n.d.). Retrieved from http://image-net.org/about-people, Retrieval date 13 Jan 2016.
[19] ImageNet. (n.d.). Retrieved from http://image-net.org/about-sponsors, Retrieval date 13 Jan 2016.

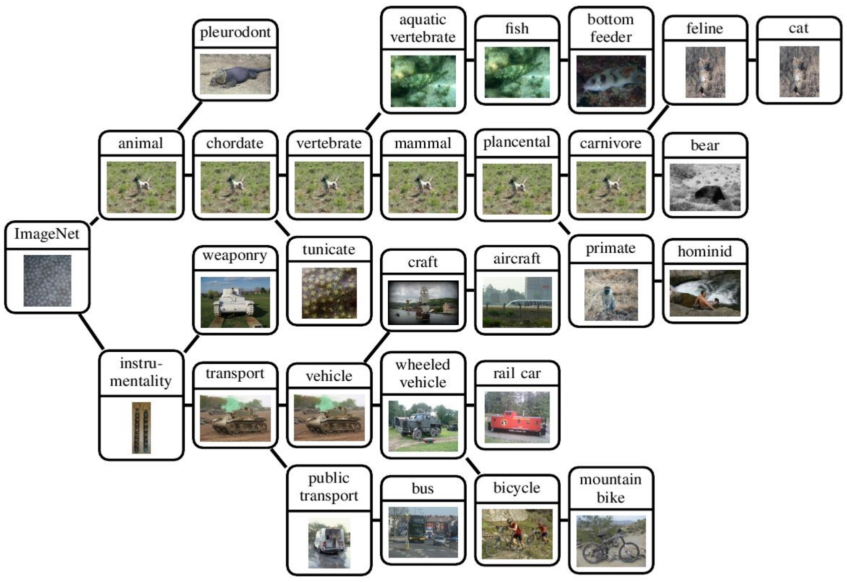


Figure 1.13 Some synsets from ImageNet with their prototype images[20]

1.3.2 ImageNet Large Scale Visual Recognition Challenge

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a yearly organized visual recognition competition which is supported by many of the leading universities and companies. In ILSVRC 2015, there were four different categories: object detection, object localization, object detection from video and scene classification. We involved this year in object localization category. Results are evaluated according to the top-5 accuracy and bounding box coordinates for predicted class labels since it is a localization task. This means that the results are expected to be five class labels in decreasing order of confidence and five bounding box coordinates one for each class labels. The quality of a localization labeling will be evaluated based on the label that best matches the ground truth label for the image and also the bounding box that overlaps with the ground truth. More than 50% area overlap is expected to be accounted as successful.

[20] Prototypes for ImageNet. (n.d.). Retrieved from http://groups.inf.ed.ac.uk/calvin/imagenet/prototypes.html, Retrieval date 13 Jan 2016.

1.3.3 Other Datasets

Data is one of the crucial part of machine learning applications. In this work our main focus was large scale image recognition and ImageNet dataset because of the competition ILSVRC 2015. But before dealing with ImageNet, we thought that we would try some smaller datasets to practice and we began to use Mixed National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research (CIFAR) datasets.

1.3.3.1 Mixed National Institute of Standards and Technology Dataset

MNIST is a classical but not trivial dataset to test image recognition algorithms. Dataset consist of 50,000 training, 10,000 validation and 10,000 test handwritten images. All images are 28x28 pixels in size and grayscale. Some samples are in Figure 1.14 below.

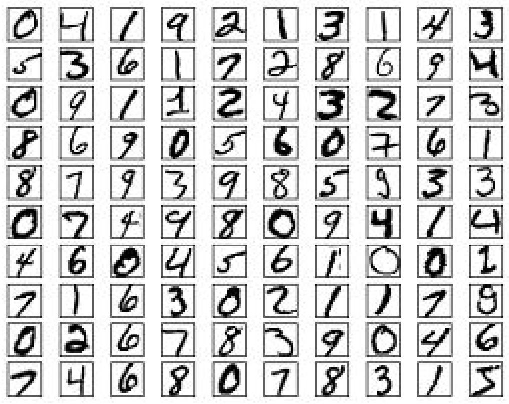


Figure 1.14 MNIST samples[21]

99.77% classification accuracy was one the best results achieved by Cireşan et al.[22] .Thus, classification MNIST dataset is an almost completed task. However, it is still good data to practice.

[21] Neural networks and deep learning. (n.d.). Retrieved from http://neuralnetworksanddeeplearning.com/chap6.html, Retrieval date 13 Jan 2016.

[22] Cireşan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, *32*, 333-338. doi:10.1016/j.neunet.2012.02.023

1.3.3.2 Canadian Institute for Advanced Research Dataset

CIFAR-10 and CIFAR-100 datasets were collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. Unlike MNIST, Images in CIFAR are colored and 32x32 pixels in size.

CIFAR-10 set consists of 10 classes as MNIST and they can be listed as airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. Each class contains 6000 images.

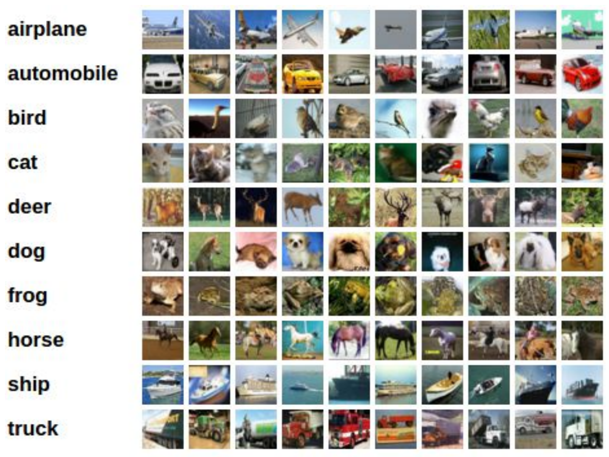
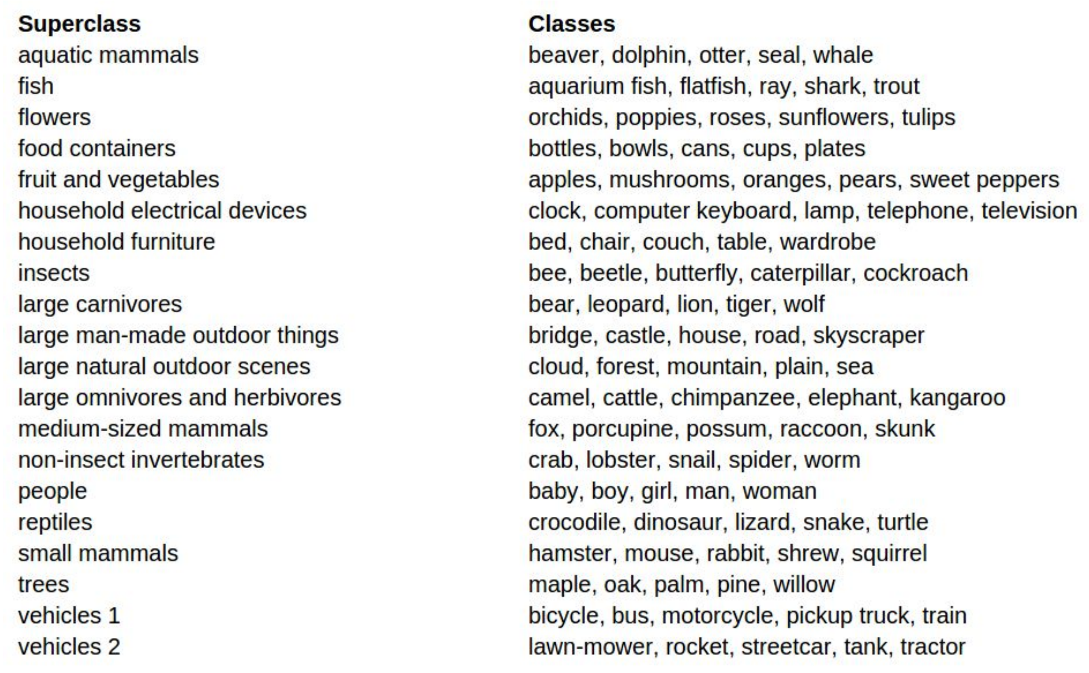


Figure 1.15 CIFAR-10 samples from each of 10 classes[23]

CIFAR-100 set consists of 100 classes and contains 600 images for each class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

[23] CIFAR-10 and CIFAR-100 datasets. (n.d.). Retrieved from https://www.cs.toronto.edu/~kriz/cifar.html, Retrieval date 13 Jan 2016.



Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

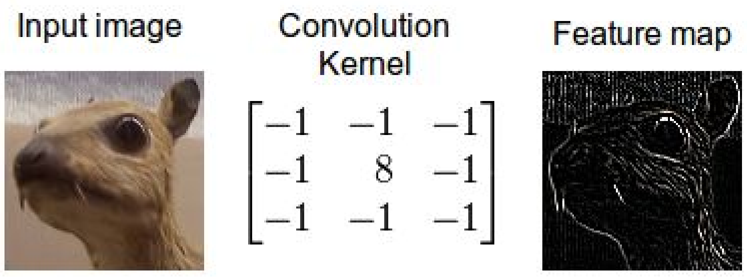
Figure 1.16 List of CIFAR-100 classes and superclasses

2. Convolutional Neural Networks

Convolution is a mathematical operation that combines two functions. It is also a filtering process, where the kernel filters the feature map for information of a certain kind. Therefore, convolution is a good choice for filtering desired features from images. Definition is

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \qquad (1)$$

Convolution is actually a description of correlation between two functions (*f* and *g* in Equation 1.). Or another point of view is that it is how similar *f* and *g*. In our case, convolutional filters are feature detectors and the image is filtered by kernel (specific feature) and if the image contains that feature, output is that much large.



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 2.1 Extracting edges with specific edge detector kernel[24]

In our method, as in Figure 1.12, there are specific layers that only apply this convolution operation. Furthermore, there may be more than one CONV in a network as we shall see in our implementations. As you can see in Figure 1.12 there are squares in each layer and there are also POOLs which are another specific layer type designed for CNN. For now let us explain these squares and another important term called parameter sharing.

2.1. Feature Maps and Parameter Sharing

Each square in a layer is a feature map in Figure 1.12. Feature maps are actually formed by neurons just like in ordinary FCs. Each feature map in a CONV focuses on some specific feature and each neuron in a map is connected to a local region in the image.

Parameter sharing is based upon a simple assumption. That is, if a feature appears at some point in the image, it is very likely to appear again at another place. And neuron must learn this feature independent from the position of the feature. Thus, each neuron in a feature map share the same parameters. This also reduces the number of parameters enormously.

[24] Understanding Convolution in Deep Learning - Deep Learning. (n.d.). Retrieved from http://timdettmers.com/2015/03/26/convolution-deep-learning/, Retrieval date 13 Jan 2016.

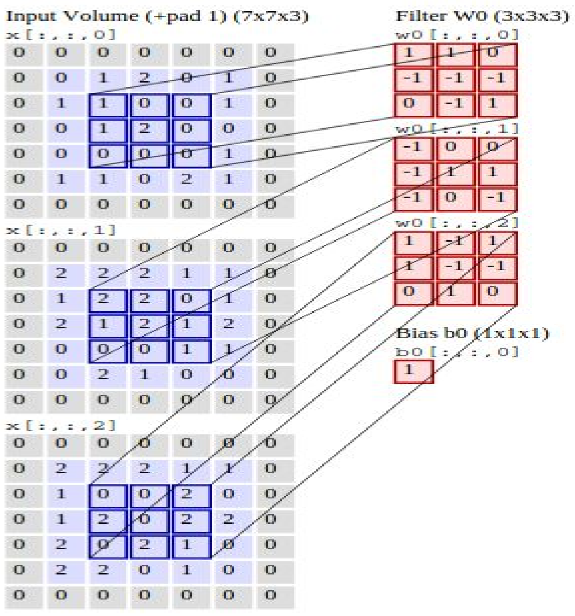


Figure 2.2 Original input image size is 5x5. Zero-padding makes the image 7x7. Local region size is 3x3. Kernel stride is 2x2 (kernel slides 2 pixels right and down). There are 3 feature maps and 9 neurons in each within (as a result of image and kernel size and stride amount). Each neuron looks into a different local receptive field in the image. And each pixel is connected with different parameters to its own neuron. So, each neuron learns only its own local field. Also, each neuron share the same parameter in a feature map.[25]

2.2. Number of Neurons

As is explained in section above, each neuron focuses on its own field. Then how many neurons are exactly in a feature map and in a CONV? In order to understand the fact, let us visualize the connection between local receptive fields and neurons.

[25] CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved from http://cs231n.github.io/convolutional-networks/, Retrieval date 13 Jan 2016.

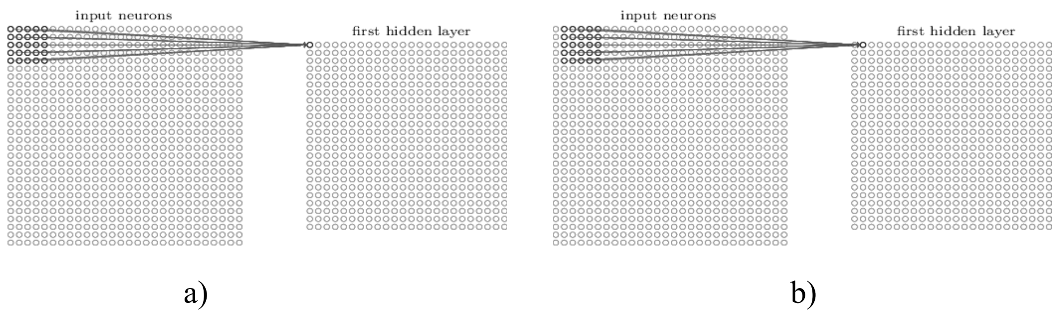


Figure 2.3 Input image is 28x28. 5x5 local receptive fields. Feature map size in the hidden layer is 24x24. **a)** The first one is connected the first neuron. **b)** The second one is connected the next neuron.[26]

In Figure 2.3 number of neurons depends on kernel size and stride. But in this example there is no padding. General case with padding is calculated like, $(W - F + 2P)/S + 1$ where W is the length of one side of image, F is the length of the side of receptive field (kernel), P is padding size, and S is the stride amount. So, simple example in Figure 2.3. with no padding, kernel size is 5 and stride is 1 gives us 24 correctly. It means we have $24x24$ neurons in one feature map. This amount must be multiplied with the amount of feature map in order to find total number of neurons in one CONV. Since all neurons in one map shares their parameters, total number of parameters in the example above just $5x5$. One important thing is that, if depth of image is more than one (i.e. image is colored) unlike the example, then the number of parameters becomes $5x5x3$. And again this number should be multiplied by the amount of feature map size to find the total number of parameters in one CONV.

2.3 Pooling Layer

We have seen that one of the layers in our architecture was POOL. Its fundamental purpose is to provide translational invariance[27]. Another benefit is to prevent one of the most common problem in ANNs, namely overfitting[28]. Besides, it reduces the amount of parameters as in ‘parameter sharing’.

[26] Neural networks and deep learning. (n.d.). Retrieved from http://neuralnetworksanddeeplearning.com/chap6.html, Retrieval date 13 Jan 2016.
[27] Retrieved from http://deeplearning.net/tutorial/lenet.html, Retrieval date 13 Jan 2016.
[28] Srivastava, N., Hinton, J., Krizhevsky A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Web.

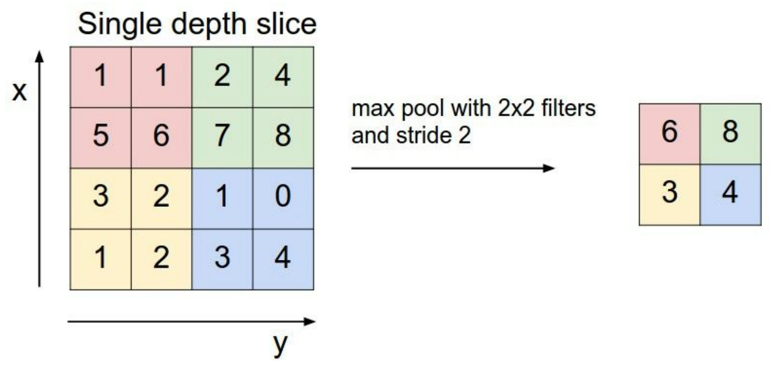


Figure 2.4 Max-POOL[29]

Generally, max and average operations are used as kernel functions in POOLs and these POOLs are named as Max-POOL and Average-POOL, respectively.

In Max-POOL, as seen in Figure 2.4, maximum pixel value is chosen to be next value for the next convolution operation in the coming layer and in Average-POOL, average of the pixel values is chosen to be next value for the next convolution operation in the coming layer. These operations are applied to every map independently. There are some hyper-parameters to be selected like kernel size and stride as in CONV. Yet, typical value for kernel size is 2x2 and stride is 2.
POOL prevents the layer from being dominated by some specific feature and allows the generalization. Nevertheless without POOL -putting CONVs consecutively can be used to not lose information at different layers. This idea is also used in GoogLeNet.

[29] CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved from http://cs231n.github.io/convolutional-networks/, Retrieval date 13 Jan 2016.

2.4 Fully Connected Layers

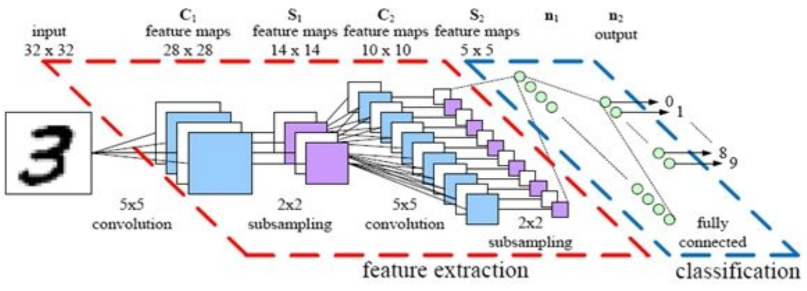


Figure 2.5 Classification after feature extraction[30]

Extracted features with consecutive CONV and POOLs need to be classified with matching labels. Actually, when we get the features, any machine learning classifier can work at this point. Conventional and practical reasons lead us to join the FCs to our CONV and POOLs to make classification. Usually, two or three FCs is enough to make the correct classification. Layer sizes (number of neurons) are the hyper-parameters to be chosen directly (not determined by any other hyper-parameter unlike CONV-POOLs).

2.5 Dropout Layers

Dropout Layers (DROPs) complement other regularization techniques to get rid of overfitting and to enhance generalization capacity of the networks. DROP prevents units from co-adapting too much[31]. Actually, DROP is not a proper layer and it acts on other layers that introduce parameters. DROPs drop the units of a layer with their connections with a predefined probability and allows remaining units to

[30] PARsE | Education | GPU Cluster | Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster. (n.d.). Retrieved from http://parse.ele.tue.nl/education/cluster2, Retrieval date 13 Jan 2016.

[31] Srivastava, N., Hinton, J., Krizhevsky A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Web.

communicate. Dropping a unit with its connections means that, weights that connects a unit to upper layers are set to zero.

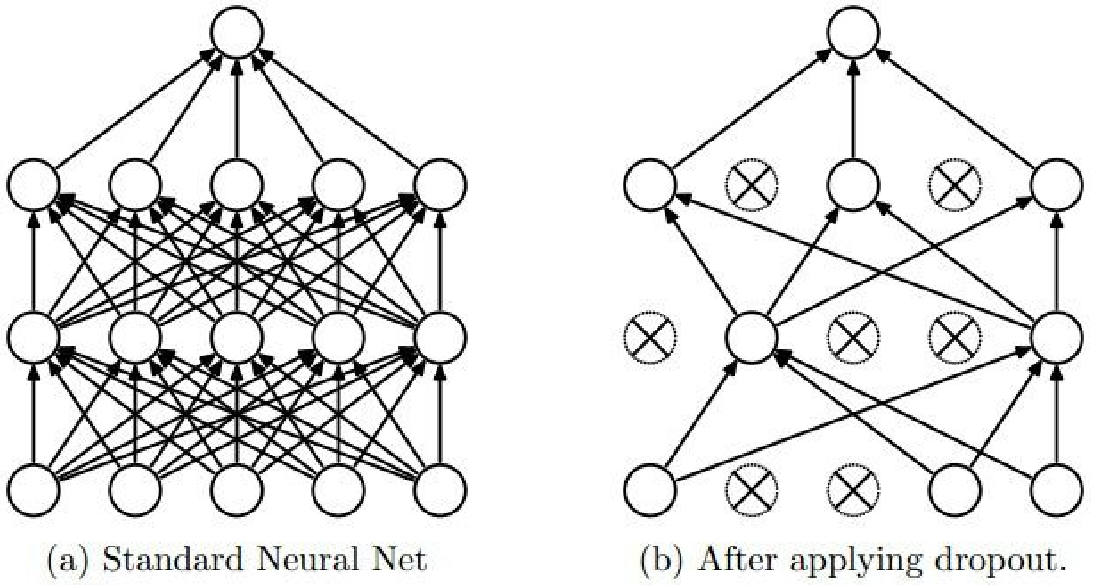


Figure 2.6 DROP[32]

By imposing DROP, different samples from the original network are selected in training phase. In testing phase, instead of imposing DROP to the network, all units are used by imposing normalization according to the DROP probability to the output of that layer.

2.6 Local-Response-Normalization Layers

ReLUs do not require input normalization to prevent them from saturating. In modern networks like GoogLeNet, there are Local-Response-Normalization Layers (LRNs) that aid generalization. After applying the ReLU nonlinearity, the response-normalized activity is applied in this layer. This type of response normalization implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs

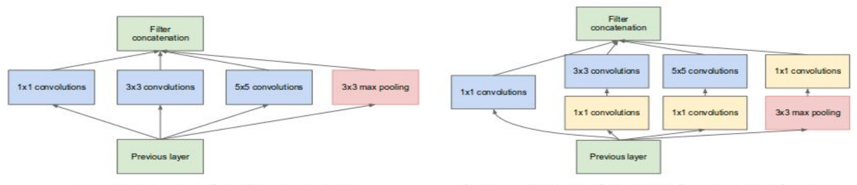
[32] Srivastava, N., Hinton, J., Krizhevsky A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Web.

computed using different kernels[33]. LRN reduces their top-1 and top-5 error rates by 1.4% and 1.2%, respectively in ImageNet according to Krizhevsky et al[34].

3. Hierarchical Approach

3.1 GoogLeNet Architecture

GoogLeNet is a CNN which consists of CONVs, POOLs, LRNs, FCs and DROPs with filter-concatenation operations and activation functions.
Architecture of GoogLeNet is shown in Figure 3.2.
In Figure 3.1, blue boxes represent CONVs and FCs. Red boxes represent POOLs with max or average operations. green boxes represent LRNs and Concatenation Operation over filters. yellow boxes represent Softmax activation functions and we take outputs of these yellow boxes as our prediction. Finally, white boxes represent inputs and outputs of the GoogLeNet.
GoogLeNet is a specific type of "Inception Network". Architecture of a general Inception Network is based on "Inception modules". Two types of Inception module are shown in Figure 3.1 below:



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3.1 Inception Module[35]

[33] Krizhevsky, A., Sutskever, I. & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Web.

[34] Krizhevsky, A., Sutskever, I. & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Web.

[35] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

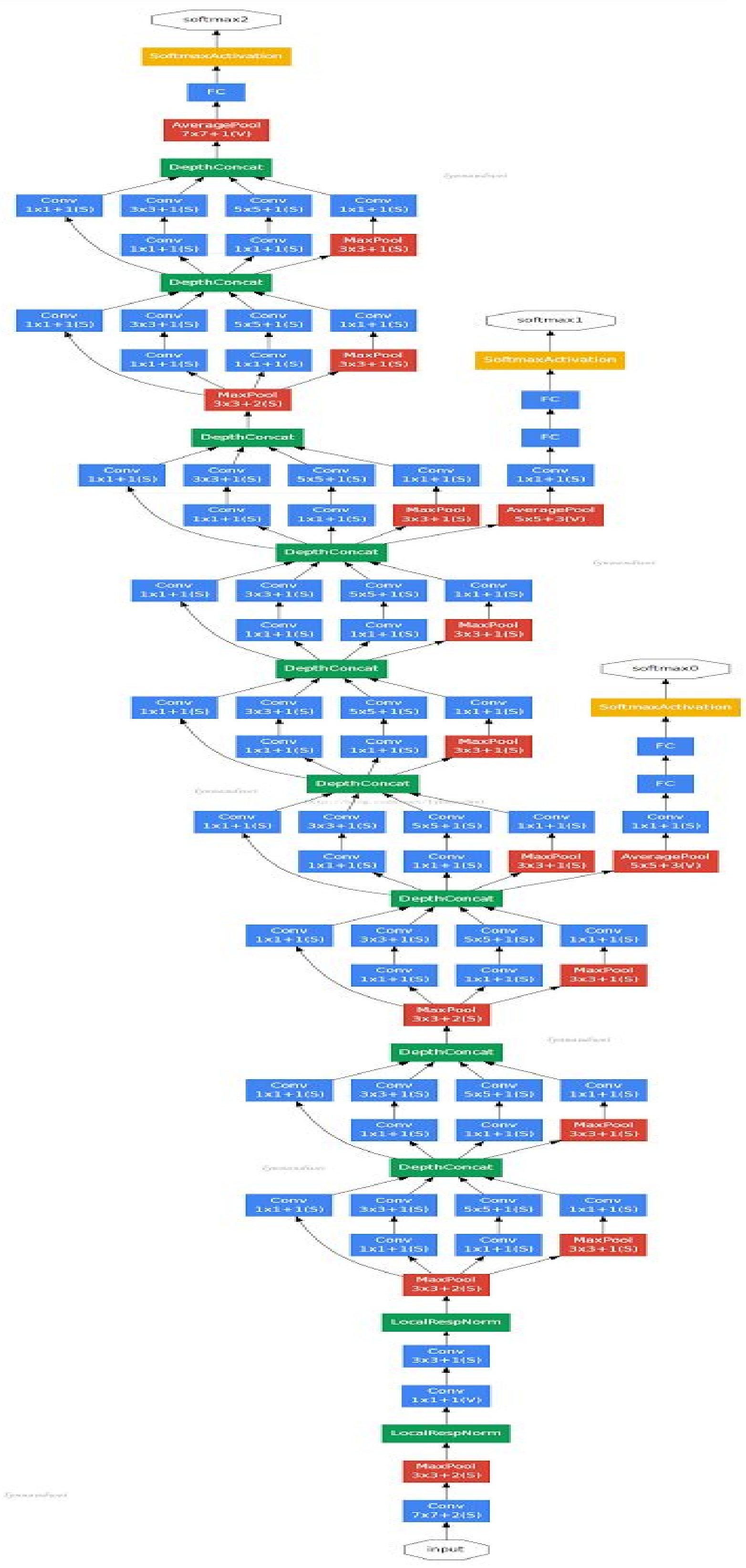


Figure 3.2 Architecture of the GoogLeNet[36]

36 Retrieved from http://img.blog.csdn.net/20150319150219047?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv

The main idea of the Inception architecture is based on finding out how an optimal local sparse structure in a convolutional vision network can be approximated and covered by readily available dense components[37].
Naive version of the Inception module (can be shown in Figure 3.1 (a)) consists of three CONVs and a Max-POOL with 3x3 kernel size. These CONVs differ by their kernel sizes which are 1x1, 3x3 and 5x5. Those layers in Inception module operate on same input which may be an output of previous layers and output of those layers are concatenated along filters (Filter-Concatenation) to construct the output of the Inception module which may be an input of the next layers.

One big problem with naive version of the Inception module is that even a modest number of 5x5 convolutions can be prohibitively expensive on top of a CONV with a large number of filters[38]. It is obvious that, as the network is getting deeper, number of filters increases consistently due to Filter-Concatenation operation at the end of the Inception module. To surpass this big problem, Inception module with dimension reductions can be used. In Inception module with dimension reductions, we add two CONVs with 1x1 kernel size, before CONVs with 3x3 and 5x5 kernel sizes, with decreasing number of filters with respect to number of filters in 3x3 and 5x5 expensive CONVs. Besides being used as reductions, they also include the use of ReLU activation which makes them dual-purpose[39]. Therefore, we increase non-linearity of our network by adding these layers with ReLU activations.

---

bHlubmFuZHdlaQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center, Retrieval date 13 Jan 2016.
[37] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.
[38] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.
[39] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

Inception module with dimension reductions (can be shown in Figure 3.2 (b)) consists of six CONVs and a Max-POOL with 3x3 kernel size. It includes three extra CONVs with 1x1 kernel sizes with respect to naive version. These extra layers are placed before 3x3 and 5x5 CONVs and after 3x3 Max-POOL. Similar to naive version, outputs of two 1x1, 3x3 and 5x5 CONVs are concatenated along filters to construct the output of the Inception module.

Building blocks of a general Inception Network are Inception Modules with separate CONVs, POOLs and LRNs. Generally, Inception modules are separated by POOLs to reduce resolution of output of the modules. On the other hand, for technical reasons (memory efficiency during training), it seemed beneficial to start using Inception modules only at higher layers while keeping the lower layers in traditional convolutional fashion[40].

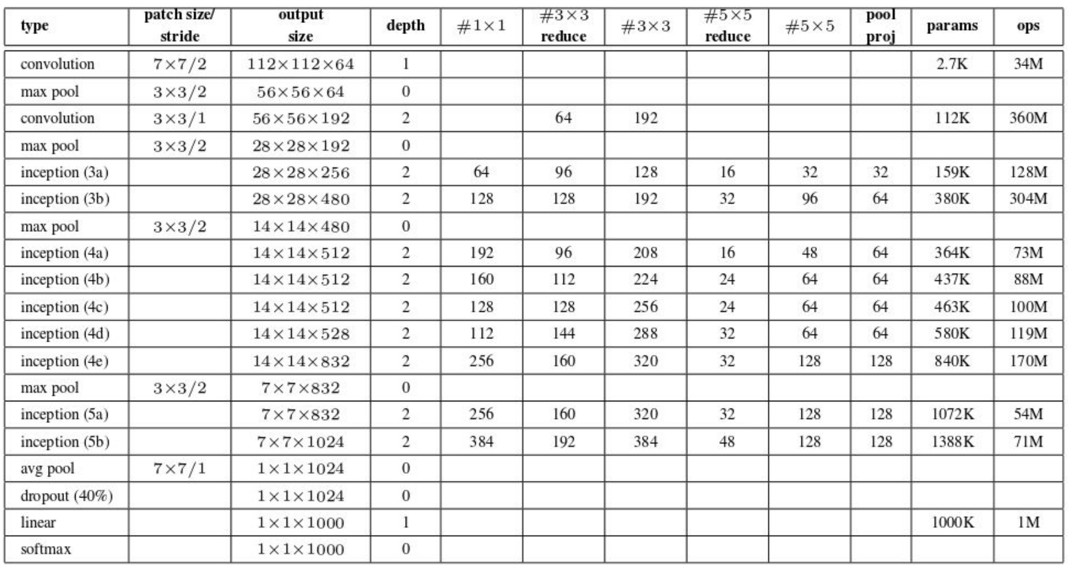
Now, after description of Inception Modules, we can analyse architecture of GoogLeNet in Figure 3.1. As we can see, GoogLeNet includes nine Inception modules with dimension reductions. Besides, as mentioned above, there is two Max-POOLs separating Inception modules and its lower layers are in the form of traditional CNN. Also, there are three FCs with softmax activations that are used as outputs, output layers, of the GoogLeNet.

In a traditional CNN, there is only one output layer. However, gradients, which is calculated using Gradient-Descent Algorithm, flowing from output layer to other layers in Back-Propagation are diminished with increasing number of layers. Hence, it is hard for lower layers to get their share of information from gradients. To get rid of this vanishing gradient problem, three output layers in different depths are used in GoogLeNet. Therefore, By adding auxiliary classifiers connected to these intermediate layers, we would expect to encourage discrimination in the lower stages in the classifier, increase the gradient signal that gets propagated back, and provide additional regularization[41].

More detailed architecture of the GoogLeNet is shown in Figure 3.3 below:

[40] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

[41] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.



type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 3.3 Inception Module[42]

Activation function for all CONVs in GoogLeNet is ReLU. Besides, it can be seen that the number of layers with parameters is 22 and the number of total layers is 27 if POOLs are included.

3.2 Hierarchical GoogleNet

Hierarchical GoogLeNet is an adaptation of the GoogLeNet to use hierarchical structure in the data for ILSVRC – Localization competition. Architecture of the Hierarchical GoogLeNet is the same as architecture of plain GoogLeNet.

3.2.1 Inputs

Images for training and testing the network are delivered by ImageNet. Some samples from ILSVRC – Localization dataset are given in Figure 3.4 below

[42] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

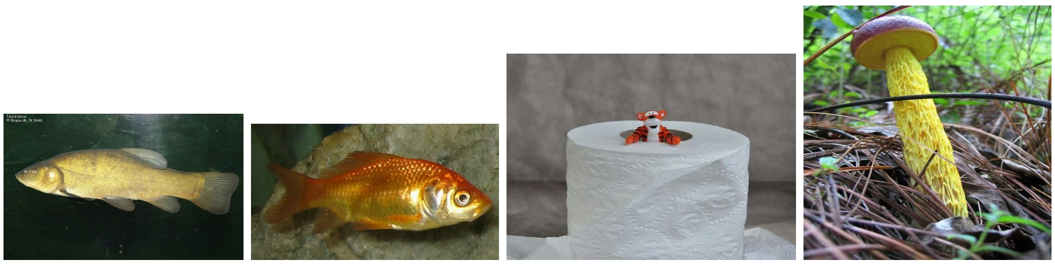


Figure 3.4 Sample Images

As mentioned earlier, each of these these images that are used for training or testing the network contains one target object and they are hand-labeled (by ImageNet) according to the class of the objects. 1000 object categories are obtained with hand-labeling.

There is a hierarchical structure in these object categories and this hierarchical structure of the ImageNet is compatible with WordNet Tree[43,44]. In WordNet Tree, objects are hierarchically categorised into a tree structure. The 1000 object categories contain both internal nodes and leaf nodes of ImageNet, but do not overlap with each other[45].

We can explain this hierarchical structure with an example. There are two object categories in our dataset named “tench” and “goldfish”. These two object categories are labeled as two different classes. However, we know that both of them are fish and they share some features of the fish. Therefore, it is obvious that, if we can take advantage of these similar features that describes the fish, we can get superior classification. In other words, if we can classify all fish species as fish and all dog species as dog, for example, in the lower layers of the network, we may expect superior classification for 1000 object categories at the higher layers.

[43] LabelMe tree. (n.d.). Retrieved from http://people.csail.mit.edu/torralba/research/LabelMe/wordnet/test.html, Retrieval date 13 Jan 2016.

[44] About WordNet - WordNet - About WordNet. (n.d.). Retrieved from https://wordnet.princeton.edu, Retrieval date 13 Jan 2016.

[45] ILSVRC2015. (n.d.). Retrieved from http://image-net.org/challenges/LSVRC/2015/, Retrieval date 13 Jan 2016.

For example, if we search sample objects above in ImageNet search site[46], then we get a screen with examples of these objects and an information parsed from WordNet tree as seen below.

Table 3.1 WordNet features of some samples

Object	Synset	Definition	Depth in WordNet
tench	tench, Tinca tinca	freshwater dace-like game fish of Europe and western Asia noted for ability to survive outside water.	15
goldfish	goldfish, Carassius auratus	small golden or orange-red freshwater fishes of Eurasia used as pond or aquarium fishes.	15
toilet tissue	toilet tissue, toilet paper, bathroom tissue	a soft thin absorbent paper for use in toilets.	6
bolete	bolete	any fungus of the family Boletaceae.	6

Synset column represents synonyms for that object noun, Definition column represents the explanation of that object in WordNet and Depth in WordNet column represents depth of the node that represents that object.

If we select correct definition for an object and then search the node that represents the object in WordNet tree, we get a collection of super nodes, that encapsulates and defines the object in a more abstract level.

For example, Super nodes for the above objects in WordNet tree are given below

[46] ImageNet. (n.d.). Retrieved from http://image-net.org/explore, Retrieval date 13 Jan 2016.

Table 3.2 WordNet trees

Object Name	Depth 1	Depth 2	Depth 3	Depth 4	Depth 5
tench	'entity'	'physical_entity'	'object'	'whole'	'living_thing'
goldfish	'entity'	'physical_entity'	'object'	'whole'	'living_thing'
toilet tissue	'entity'	'abstraction'	'relation'	'part'	'substance'
bolete	'entity'	'physical_entity'	'object'	'whole'	'living_thing'

Object Name	Depth 6	Depth 7	Depth 8	Depth 9	Depth 10
tench	'organism'	'animal'	'chordate'	'vertebrate'	'aquatic_vertebrate'
goldfish	'organism'	'animal'	'chordate'	'vertebrate'	'aquatic_vertebrate'
toilet tissue	'material'	'paper'	'tissue'	‘toilet_tissue'	
bolete	'organism'	'fungus'	'bolete'		

Object Name	Depth 11	Depth 12	Depth 13	Depth 14	Depth 15
tench	'fish'	'bony_fish'	'teleost_fish'	'soft-finned_fish'	'cypriniform_fish'
goldfish	'fish'	'bony_fish'	'teleost_fish'	'soft-finned_fish'	'cypriniform_fish'

Object Name	Depth 16	Depth 17
tench	'cyprinid'	'tench'
goldfish	'cyprinid'	'goldfish'

We can see that at a smaller depth in WordNet tree two different objects collapse into the same node, hence same class and that super class describes both of these objects, both visually and biologically.

As mentioned earlier, there are three output layers in GoogLeNet architecture in different depths. Hence, if we choose two super classes for the objects, we can train GoogLeNet with these two super classes and with original 1000 classes. Plain GoogLeNet ignores this hierarchical structure and trains the network using 1000 classes in all output layers.

To impose this hierarchical structure into the network, we repeat the steps explained above for all of the 1000 object categories and we come up with 69 and 333 super classes. Now, we can train the network with 69, 333, 1000 classes in output layers instead of only 1000 classes.

In plain GoogLeNet, inputs are of size 224x224, mean subtracted, colored images both in training and testing phases. In both training and testing phase, these images are rescaled to multiple scales and then 224x224 crops are extracted from them before input the images to the network. Multiple scales in training provide more scale invariance to the object sizes. Also, various interpolation methods are used when resizing the inputs, such as bilinear, area, nearest neighbour and cubic. To enhance dataset further and to get more regularization, photometric distortions[47] are applied to the images.

As in plain GoogLeNet, inputs of the hierarchical GoogLeNet are of size 224x224, mean subtracted, colored images both in training and testing phases. Also, in

[47] Howard, A. (2013). Some Improvements on Deep Convolutional Neural Network Based Image Classification. Web.

Hierarchical GoogLeNet same images are used as in plain GoogLeNet. However, in Hierarchical GoogLeNet, there is only one scale (256x256) for rescaling. It is known that CNNs have scaling invariance up to limited factor and the aim of multiple scale images in plain GoogLeNet is to beat this limitation. It is obvious that, using images with multiple scales in training will be beneficial for scale invariance. However, due to time limitations, we have restricted ourselves to single scale.

3.2.2 Training and Testing Phase

3.2.2.1 Plain GoogLeNet

In training phase, seven versions of GoogLeNet architecture is developed and trained, independently, to get an ensemble prediction in testing phase. These 7 models are trained with the same initialization of weights and biases and same learning rate policies[48].

Actually, initialization of weights and biases in these independent models should better be different. However, especially in deeper ANNs, these initial parameters must be chosen carefully, to be able to train network. With uncarefully chosen initial parameters, norm and variance of the gradients (during the backpropagation phase), are decreased drastically[49]. Hence, to transfer back propagated gradients smoothly, initialization of weights and biases is important.

To get rid of the dependence on the initial parameters, there is an approach called "Batch Normalization"[50]. Batch Normalization suggests that with normalizing the distribution inputs of each layer to fixed mean and fixed variance, more smooth gradients can be obtained. Therefore, using Batch Normalization approach, initial parameters of those independent models can be chosen differently without much

[48] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015): n. pag. Web.

[49] Glorot, Xavier, and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks (2010). Web.

[50] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Web.

effort. On the other hand, order and scale of the input images to train these seven models are different.

In training phase of the seven networks, batched inputs are feed-forwarded to the network to calculate Negative-Log-Likelihood loss. The goal is to minimize loss function via Stochastic-Gradient-Descent (SGD) in backpropagation[51].

In each network, as already mentioned, there are three output layers with 1000 classes, hence there are three losses. Total loss of each network is calculated by weighted sum of the losses of output layers. Weights for losses from first two output layers are 0.3 and the weight for loss at the last output layer is 1.0. After calculation of total loss for each network, gradients are back-propagated using SGD.

In training with SGD, momentum parameter is chosen 0.9 and learning rate is dropped by 4% each 8 epochs.

In testing phase, inputs are rescaled to four different scales where the shorter dimension of these images to be 256, 288, 320 and 352[52].

When making prediction for an image in a single scale, left, center and right squares are cropped from image. Also, if the input is a portrait image, top, center and bottom squares are cropped from image. After getting squares, 224x224 crops are taken from four corners and center of the square. On the other hand, these squares are rescaled to 224x224. Also, using mirrored versions of these 224x224 cropped and rescaled images, inputs of the network is enhanced. At the end, there are 12 versions of a square, hence, there are 36 different versions of an image at a single scale.

Four different scales are used for an image, so we have 144 different versions for an image. These different 144 versions of the image are input to each of the seven trained networks and the softmax probabilities are averaged over multiple crops and

[51] Retrieved from http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/, Retrieval date 13 Jan 2016.

[52] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

over all the individual classifiers to obtain the final prediction[53]. Also, final predictions of these seven networks are averaged using “Polyak Averaging”[54].

3.2.2.2 Hierarchical GoogLeNet

Unlike plain GoogLeNet, we trained only one network due to time limitations and hence, we could not use any model averaging or any ensemble prediction. As in plain GoogLeNet, batched inputs are feed-forwarded to the network to calculate Negative-Log-Likelihood loss with the goal of minimizing loss function via SGD in back-propagation.

Weights and biases of the Hierarchical GoogLeNet are initialized as in Plain GoogLeNet.

In Hierarchical GoogLeNet, there are three output layers with 69, 333 and 1000 classes, hence there are three losses. As in plain GoogLeNet, total loss of each network is calculated by weighted sum of the losses of output layers. Weights for losses from first two output layers (these output layers have 69 and 333 classes, respectively.) are 0.3 and the weight for loss at the last output layer, which has 1000 classes, is 1.0. After calculation of total loss for each network, gradients are back-propagated using SGD.

In training with SGD, momentum parameter is chosen to be 0.9 and initial learning rate is chosen 0.01. Learning rate is dropped by 10% each 33 epochs. We trained the network for 100 epochs. Also, we added L2-Norm Regularization into our loss function with a weight of 0.0005.

In testing phase, instead of rescaling to multiple scales, we rescaled images to a single scale (256x256) as in training phase and we only cropped center of the rescaled image to make predictions. Whereas plain network made predictions with 144 crops per image, with seven different models, we made prediction with only one

[53] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Web.

[54] Polyak, B. T., & Juditsky, A. B. (1992). Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, *30*(4), 838-855. doi:10.1137/0330046

crop per image and only one model. These simplifications are done due to time limitation.

4. Results and Analysis

4.1 Architecture and Hyper-parameter Tests on Mixed National Institute of Standards and Technology Dataset and Canadian Institute for Advanced Research Datasets

This part mostly covers the results of different network architectures with different parameters on MNIST and CIFAR-10 datasets.

Implementations are based upon Theano[55], a Python numerical computation library. Some advantages of this library are efficient symbolic calculation (automatically calculates gradients), transparent use of GPU, speed and stability optimizations.

Models were trained mostly with adaptive learning rate, which provides great improvement in training times comparing to constant learning rate. A variation of 'bold driver'[56] technique used for adaptation of learning rate. It starts from the value of 10^{-2}. At the end of each epoch, if accuracy increases and learning rate is above 10^{-3} learning rate raises 5% up, if is below 10^{-3} it raises 90% up. If accuracy decreases, learning rate halves. When learning rate lowers too much, learning becomes too slow and training should be interrupted. Thus, in our examples 30 consecutive epochs with no improvement in validation accuracy leads to stopping of the training. This causes unequal epoch numbers in comparisons below.

L1 regularization technique used in all examples with constant $\lambda = 0.1$. SGD technique was used for optimization with no momentum. Also DROP was used in all FCs with 0.5 probability.

[55] Welcome — Theano 0.7 documentation. (n.d.). Retrieved from http://deeplearning.net/software/theano/, Retrieval date 13 Jan 2016.

[56] Duffner, Stefan, and Christophe Garcia. "An Online Backpropagation Algorithm with Validation Error-Based Adaptive Learning Rate." *Lecture Notes in Computer Science Artificial Neural Networks – ICANN 2007* (2007): 249-58. Web.

First results from MNIST with adaptive learning rate and just two FCs (no CONV-POOL). Number of neurons and activation functions are compared.

Table 4.1 MNIST accuracies with 2 FCs

name	# of epochs	# of neurons	activatio n function	validation accuracy	test accuracy
net1	58	100	sigmoid	92.44%	91.50%
net2	106	100	ReLU	97.87%	97.69%
net3	93	1000	ReLU	98.27%	98.10%
net4	104	1000	sigmoid	92.51%	91.72%

Table 4.1 tells us two critical results. ReLU activation function is superior over sigmoid activation function (even in a shallow network) and neuron number is important for learning capacity. We will be showing results with just ReLU activation function from now on and ReLU activation function is a good choice in most cases. Also increasing the number of neurons may not be necessary all the time since the improvement of accuracy is not guaranteed. Besides this makes the number of parameters exponentially large and hence the training times much longer.

The next comparison is the kernel sizes in CONVs. In both MNIST and CIFAR-10 there are 2 CONV-POOL (1 CONV + 1 POOL + 1 CONV + 1 POOL) with concatenated 3 FCs. Stride is 1x1, feature maps are 20 + 40 and neuron numbers in FCs are 1000 + 100 and softmax layer in the end.

Table 4.2 MNIST accuracies with 2 CONV-POOLs + 3 FCs

name	# of epochs	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net10	94	1000, 100	20, 40	5x5, 5x5	99.39%	99.54%
net12	183	1000, 100	20, 40	9x9, 7x7	99.20%	99.31%

Table 4.3 CIFAR-10 accuracies with 2 CONV-POOLs + 3 FCs

name	# of epochs	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net10	142	1000, 100	20, 40	5x5, 5x5	75.87%	75.25%
net12	152	1000, 100	20, 40	9x9, 7x7	69.23%	68.79%

Both dataset results show that increasing kernel sizes does not mean improvement in classification even with larger number of epochs. Actually smaller kernel size is related to catching subtle features in the image. In both image set, 5x5 kernels are better than 9x9 + 7x7 kernel at fitting to features.

The next comparison is about the order the hidden layers with different number of feature maps. Same architecture as ‘net10’ but with feature maps 80 + 40 and 40 + 80. Other parameters such as learning rate neuron numbers in FCs were same as ‘net10’.

Table 4.4 MNIST accuracies with 2 CONV-POOLs + 3 FCs

name	# of epochs	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net13	89	1000, 100	40, 80	5x5, 5x5	99.48%	99.46%
net14	105	1000, 100	80, 40	5x5, 5x5	99.44%	99.53%

Table 4.5 CIFAR-10 accuracies with 2 CONV-POOLs + 3 FCs

name	# of epochs	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net13	120	1000, 100	40, 80	5x5, 5x5	76.98%	75.47.%
net14	197	1000, 100	80, 40	5x5, 5x5	77.47%	77.37%

Table 4.4 and Table 4.5 do not actually tell much critical things. With different epoch numbers 'net14' seems to be more accurate than 'net13' in CIFAR-10. Yet, in MNIST validation accuracy drops down a bit, whereas test accuracy rises up a bit. Probably, these trials are not enough to decide which feature map number strategy (expanding or narrowing) order is the best.

Now, the results for slightly larger networks will be compared. We almost come to a limit with respect to computation power. Even with powerful GPUs, 3 CONV-POOL plus 3 FCs increase the training times a couple of hours. Yet you will see that these results are the best one in accuracies. Let us focus now just on MNIST and analyze it.

Table 4.6 MNIST accuracies with 2 CONV-POOLs + 1 CONV + 3 FCs

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net15	97	300, 100	80, 40, 20	5x5, 3x3, 3x3	99.35%	99.35%
net16	125	300, 100	160, 80, 40	5x5, 3x3, 3x3	99.36%	99.37%
net17	97	1024, 128	256, 256, 128	5x5, 3x3, 3x3	99.38%	99.49%
net18	378	1024, 512	512, 256, 256	5x5, 3x3, 3x3	99.50%	99.47%

It looks like the best result network is 'net18' since its average accuracy of validation and test set is the highest one. We can say 'net18', which has largest number of parameters, has the highest learning capacity. and without allowing overfitting. DROP and other regularization techniques prevent overfitting. Larger number of epochs also seem to bring about a higher learning capacity, since it was able to continue to learn without being interrupted by the algorithm.

All the networks shown in Table 4.6 have same number of layers and no POOL after the final CONV because of the image size in this layer is not suitable for applying pooling operation. Padding could be utilized at this point, but we just wanted to be consistent with other previous networks.

Actually, all these networks were trained just once. Namely, saved models were not be continued from the previous parameters. Yet this could be meaningful if dataset would be expanded and this expanded dataset would be trained with saved parameters. But of course with much longer training times.
Similar architectures for CIFAR-10 dataset are as below.

Table 4.7 CIFAR-10 accuracies with 2 CONV-POOLs + 1 CONV + 3 FCs

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net15	164	200, 100	80, 40, 20	5x5, 3x3, 3x3	74.65%	73.57%
net16	115	300, 100	160, 80, 40	5x5, 3x3, 3x3	76.88%	75.50%
net17	145	1024, 128	512, 256, 128	5x5, 3x3, 3x3	78.20%	77.10%
net18	489	1024, 512	512, 256, 256	5x5, 3x3, 3x3	79.36%	78.30%

Parameters are almost the same as in Table 4.6, apart from small adjustments applied. Again the best results came with ‘net18’. Since CIFAR-10 dataset is “harder” than MNIST, this result is not a surprise. But there may be much improvement than MNIST with a little bit of effort, of course.
There are some experimental network architectures also tested on CIFAR-10.

Table 4.8 CIFAR-10 accuracies with 1 CONV-POOL + 3 CONVs + 1 CONV-POOL + 3 FCs

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net21	78	1024, 1024	32, 64, 128, 128, 128	5x5, 3x3, 3x3, 3x3, 3x3	72.96%	72.38%

‘net21’ was eventually interrupted because of being too expensive in terms of computation. As we said it was just experimental and it is just about testing computation power with deeper ANNs. Another example is more interesting. It has only one FC with softmax activation in the end.

Table 4.9 CIFAR-10 accuracies with 1 CONV-POOL + 3 CONV + 1 CONV-POOL + 1 FC

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net22	111	-	32, 64, 128, 128, 128	5x5, 3x3, 3x3, 3x3, 3x3	58.21%	56.36%

As can be seen, such deeper CNNs need a classifier in the end. Without FCs it does not give much hope.

Table 4.10 CIFAR-10 accuracies with 1 CONV-POOL + 1 FC

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net24	48	-	64	3x3	62.01%	61.10%
net26	50	-	64	5x5	60.81%	59.80%
net27	40	-	64	7x7	58.62%	58.84%

Table 4.10 shows us the kernel size comparison as in Table 4.2 and Table 4.3, but in a more specific and with more simple architecture. As it can be seen with increasing kernel sizes, accuracies decrease systematically with smaller epoch numbers.
This situation could be interpreted as explained above, smaller kernel sizes are good at catching details in an image. Therefore, deeper ANNs with smaller kernels are generally better choice than shallow networks with larger kernels.
Then the last network tested on CIFAR-10 below. This is the only network that can exceed 80% accuracy for CIFAR-10.

Table 4.11 CIFAR-10 accuracies with 3 CONV-POOLs + 3 FCs
(with continued model of 'net31')

name	# of epoch	# of neurons	# of feature maps	kernel sizes	validation accuracy	test accuracy
net31	372	1024, 1024	32, 64, 128	5x5, 3x3, 3x3	80.11%	79.75%
net31*	227	1024, 1024	32, 62, 128	5x5, 3x3, 3x3	80.60%	80.00%

The network in the second row is the continued model with last saved parameters of 'net31'. Network specifications were given just by convention (changing architecture is not possible for a continued model). As we said, 'net31' were interrupted automatically where the learning cannot go any further and model parameters saved. After that with these parameters models were continued with normal starting learning rate 10^{-2}.

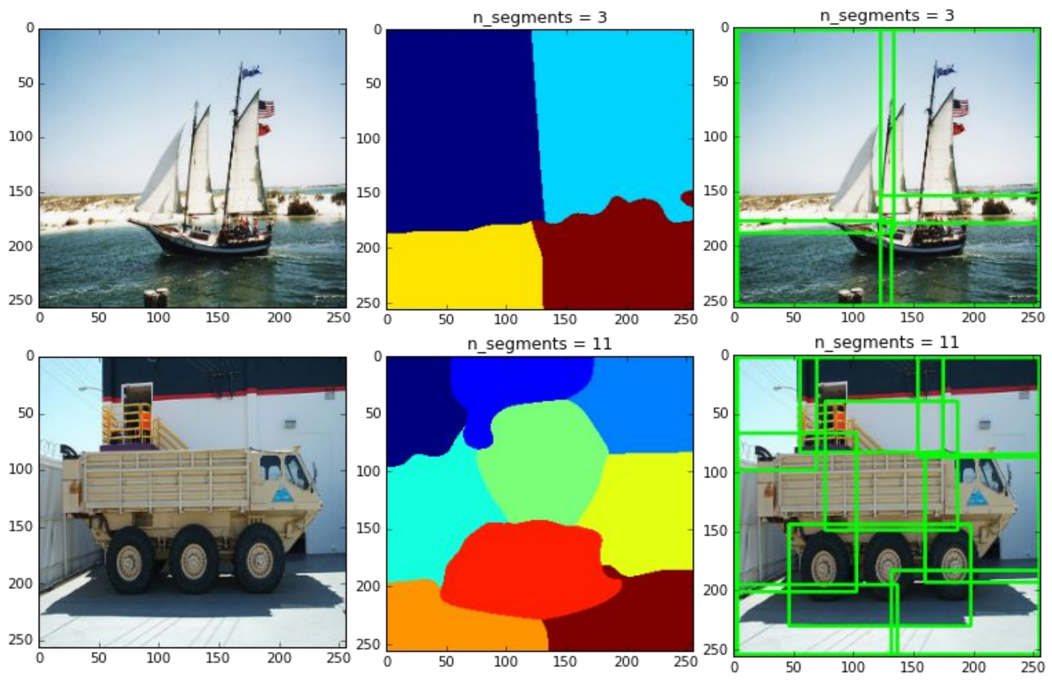
4.2 Localization Study on ImageNet Large Scale Visual Recognition Challenge - Localization Competition

Up to now, we only discussed and shared the results about classification task. The second leg of the competition is localization actually. We separated the localization part from the training process. We actually just applied some traditional image processing algorithms on the images. The technique is an image segmentation method called Simple Linear Iterative Clustering (SLIC)[57]. This method was implemented using SLIC method from the Python image processing library Scikit-image[58]. The algorithm simply performs k-means based upon local color information. The list of chosen number of centers for every image were

[57] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(11), 2274-2282. doi:10.1109/tpami.2012.120

[58] Comparison of segmentation and superpixel algorithms — skimage v0.12dev docs. (n.d.). Retrieved from http://scikit-image.org/docs/dev/auto_examples/segmentation/plot_segmentations.html?highlight=slic, Retrieval date 13 Jan 2016.

{2, 3, 5, 7, 11 ,15} for k-means. The number of generated image segments for every image was 45 on the average. Two examples on sample images from dataset are given below.



a) b) c)

Figure 4.1 a) Original images **b)** Segmented images with 3 and 11 center points for k-means **c)** Segmented parts on the original images

As in Figure 4.1, crops are found by drawing bounding boxes on generated segments. Then all these crops for an image were tested by our trained model. For each of top-5 class labels, the average of coordinates of matched crops with the labels were taken to be bounding box coordinates. If there were no matched crops with the labels, the bounding box coordinates of the label with the highest confidence percent were chosen. Finally, these matched bounding box coordinates were associated with predicted top-5 class labels.

4.3 Result of Hierarchical GoogLeNet in ImageNet Large Scale Visual Recognition Challenge - Localization Competition

Results of Hierarchical GoogLeNet in ILSVRC - Localization competition are described below. In localization competition, performance of the Hierarchical GoogLeNet is evaluated according to the classification performance and localization performance in test set of localization competition.

Classification performance is evaluated according to the Top-5 Error, which means that if one of the Top-5 most probable predictions of the network matches with the ground truth label, then it is counted as a true prediction.

Localization performance is evaluated according to the metric below:

$$e = \frac{1}{n} \cdot \sum_{k} min_i min_m max\{d(c_i, C_k), f(b_i, B_{km})\}$$

Figure 4.2 Localization performance metric[59]

In localization task, 5 bounding boxes, for each Top-5 most probable classes, are predicted by network. The quality of a localization labelling is evaluated based on the label that best matches the ground truth label for the image and also bounding box that overlaps with the ground truth[60].

As written in Figure 4.2, in above equation, ground truth labels are C_k where $k = 1, 2, ..., n$ with n class labels. For each ground truth class label C_k, the ground truth bounding boxes are B_{km} where $m = 1, 2, ..., M_k$. M_k is the number of instances of the k^{th} object in the current image and $d(c_i, C_k) = 0$ if $c_i = C_k$ and *1*

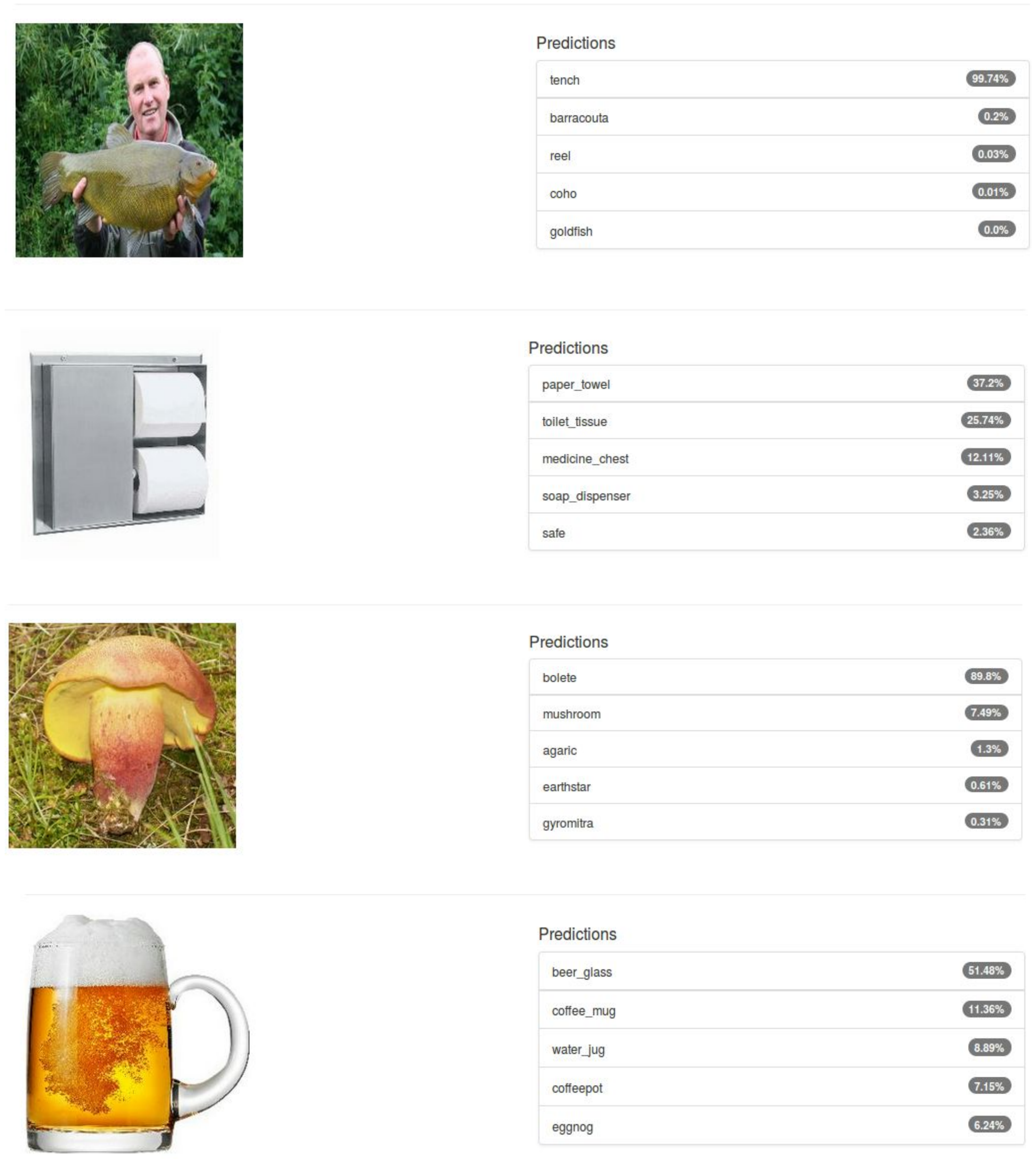
[59] ILSVRC2015. (n.d.). Retrieved from http://image-net.org/challenges/LSVRC/2015/, Retrieval date 13 Jan 2016.

[60] ILSVRC2015. (n.d.). Retrieved from http://image-net.org/challenges/LSVRC/2015/, Retrieval date 13 Jan 2016.

otherwise. Also, $f(b_i, B_k) = 0$ if b_i and B_k have more than *50%* overlap, and *1* otherwise.

According to these metrics, Top-5 Classification Error and Localization Error of Hierarchical GoogLeNet is *0.11433* and *0.589338*, respectively.

Some of the predictions of Hierarchical GoogLeNet is given in Figure 4.3:



Predictions

tench	99.74%
barracouta	0.2%
reel	0.03%
coho	0.01%
goldfish	0.0%

Predictions

paper_towel	37.2%
toilet_tissue	25.74%
medicine_chest	12.11%
soap_dispenser	3.25%
safe	2.36%

Predictions

bolete	89.8%
mushroom	7.49%
agaric	1.3%
earthstar	0.61%
gyromitra	0.31%

Predictions

beer_glass	51.48%
coffee_mug	11.36%
water_jug	8.89%
coffeepot	7.15%
eggnog	6.24%

Figure 4.3 Some of the predictions of Hierarchical GoogLeNet

5. Conclusion and Discussion

CNNs have become the most preferred architecture by the ANNs community in recent years. Despite that, its theoretical ground goes to 80's, it became popular with the developments of DL. And, the understanding of deep ANNs has changed by using different techniques such as activation functions and by using developed GPUs for the purpose of computation.

All works in this paper were done during the competition ILSVRC 2015. And all the time we have dealt with just CNNs for training process. We trained CNNs that are in different structure and with different hyper-parameters. Before passing into ILSVRC - Localization dataset we worked on MNIST and CIFAR-10 to test these structures. Then, for ILSVRC - Localization competition Caffe framework used with GoogleNet for classification task, combining with our hierarchical approach to dataset and a localization method called SLIC.

Our hierarchical approach to GoogLeNet provides us to divide 1000 classes into smaller hierarchical groups. These groups increased our control on this gigantic dataset and allowed the network to learn the 1000 classes with more eligible way. Actually this approach based upon a shared semantic origins between 1000 classes. Many different classes have common visual patterns. We believe that these patterns must be encoded some specific parts on the network architecture. And each part should decide whether the image has these patterns or not. This approach looks alike the feature maps in the CONVs that focus on some specific patterns in the image. In the network scale, hierarchy comes into play and each part of the network with its own output layer decide the correct classes that is determined before.

One drawback is that these hierarchical classes must be determined before training process. In this work these classes were selected manually. Automatically generated hierarchical classes would be a great leap for this approach.

The other drawback is that localization. It is hard to incorporate the localization process into training. We used an image segmentation method independent from the training process. However the most image segmentation method fails when used on its own. All such techniques should be embedded into the network to make the network a whole system. Therefore CNNs are very successful at classifying visual inputs. However visual recognition is not just formed by classifying, but also localizing. And in order to create a complete visual recognition ‘machine’, more adaptive or flexible network architectures are needed.

APPENDIX A

Hierarchical GoogLeNet was implemented in Caffe framework[61].

Works on hierarchical label selection, selected labels, codes to train Hierarchical GoogLeNet and learned parameters can be found in our GitHub repository[62].

[61] http://caffe.berkeleyvision.org/
[62] https://github.com/pathintegral/hierarchical-googlenet