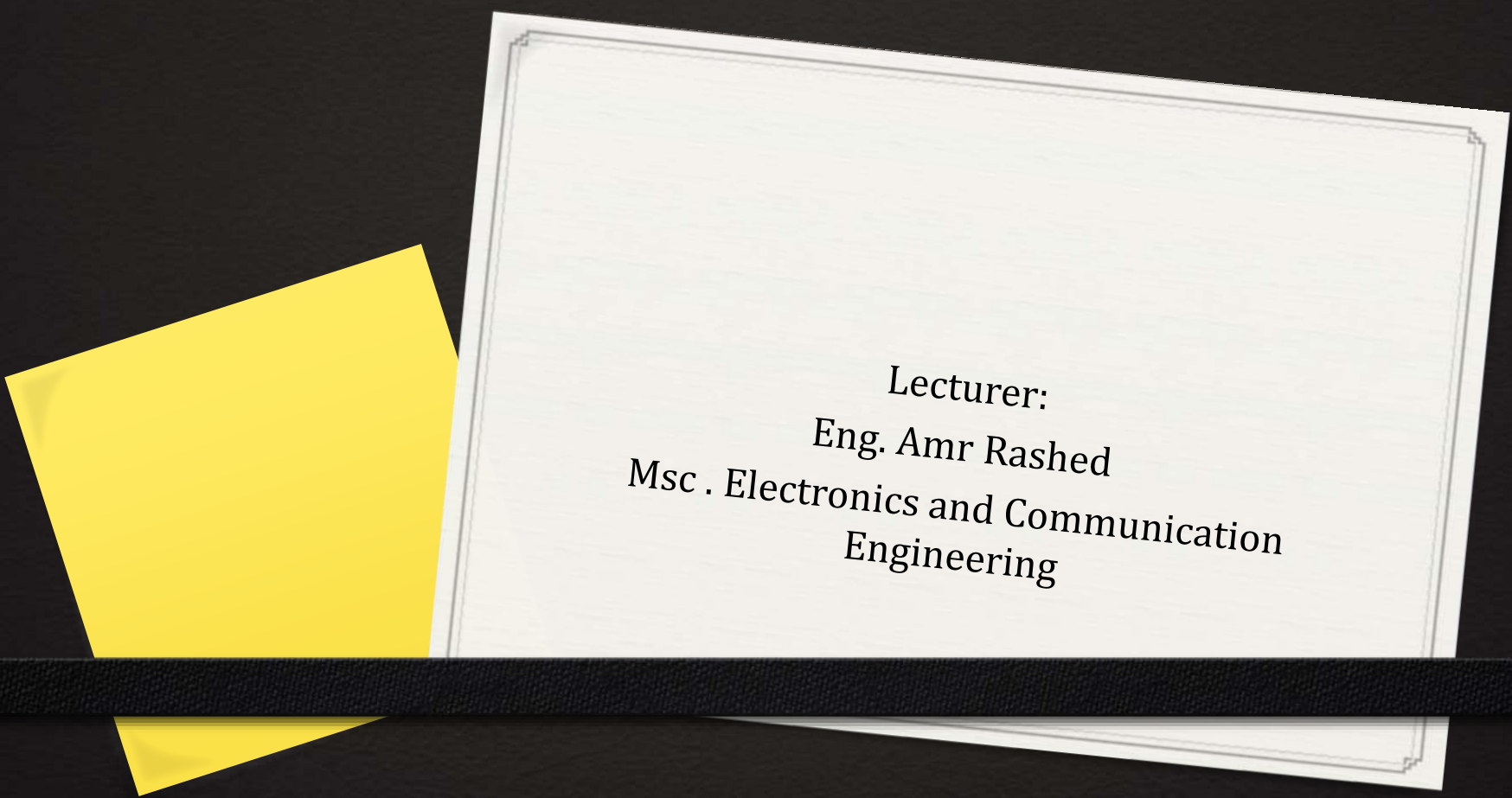


Deep Learning Tutorial



Lecturer:
Eng. Amr Rashed
Msc . Electronics and Communication
Engineering

Deep Learning

AGENDA

MOTIVATIONS, WHY DEEP NN



A Brief History, MACHINE LEARNING BASICS



BIG PLAYERS, APPLICATIONS



WHAT IS DEEP LEARNING, DEEP LEARNING BASICS



DEEP NN ARCHITECTURE, PROBLEM SPACE



Types of Networks , Convolution Neural Networks



RECURRENT NN, DEEP LEARNING TOOLS



DEEP LEARNING in Bioinformatics, Conclusion

Motivations



I have worked all my life in machine learning, and **I've never seen one algorithm knock over benchmarks like deep learning**. Andrew Ng(Stanford & Baidu)



Deep learning is an algorithm which has **no theoretical limitations of what it can learn**; the more data you give and the more computational time you provide; the better it is-Geoffrey Hinton(Google)

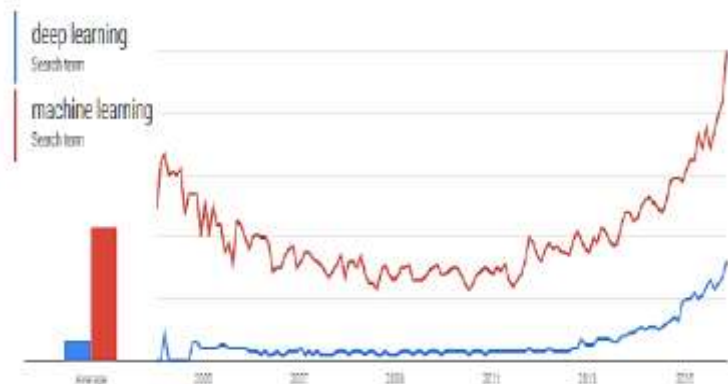
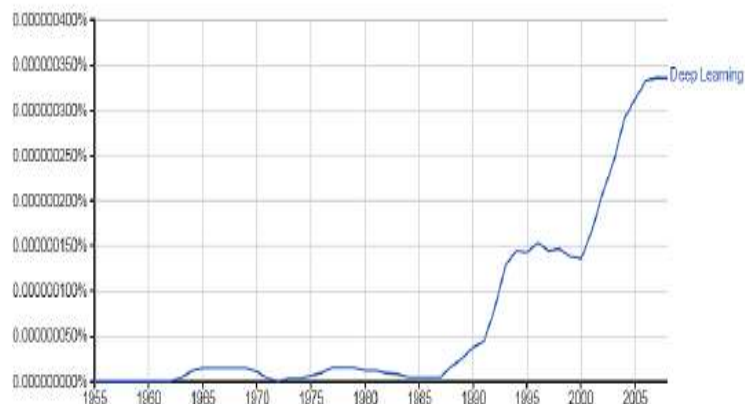


Human-level artificial intelligence has **the potential to help humanity** thrive more than any invention that has come before it –Dileep George(Co-Founder Vicarious)



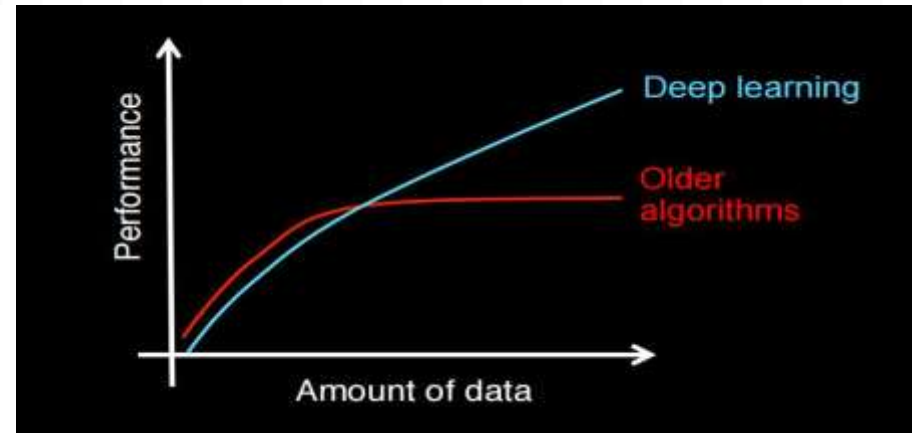
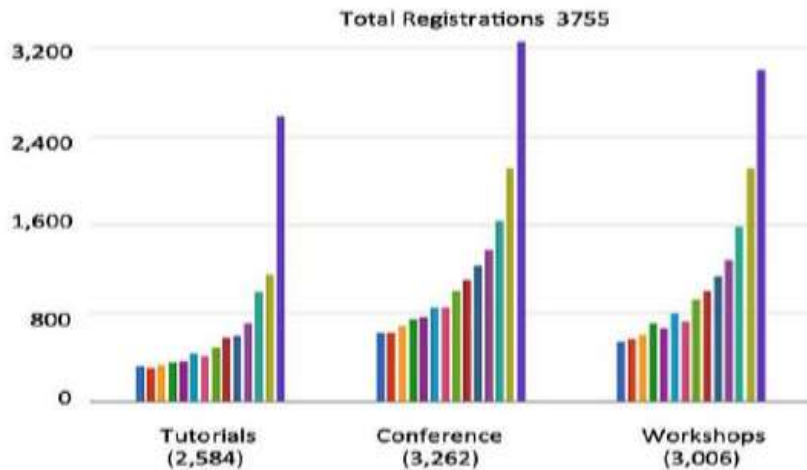
For a very long time it will be a **complementary tool** that human scientists and human experts can use to help them with the things that humans are not naturally good- Demis Hassabis (Co-Founder Deep Mind)

Motivations–Google NGram–Google Trend



ImageNet: The "computer vision World Cup"

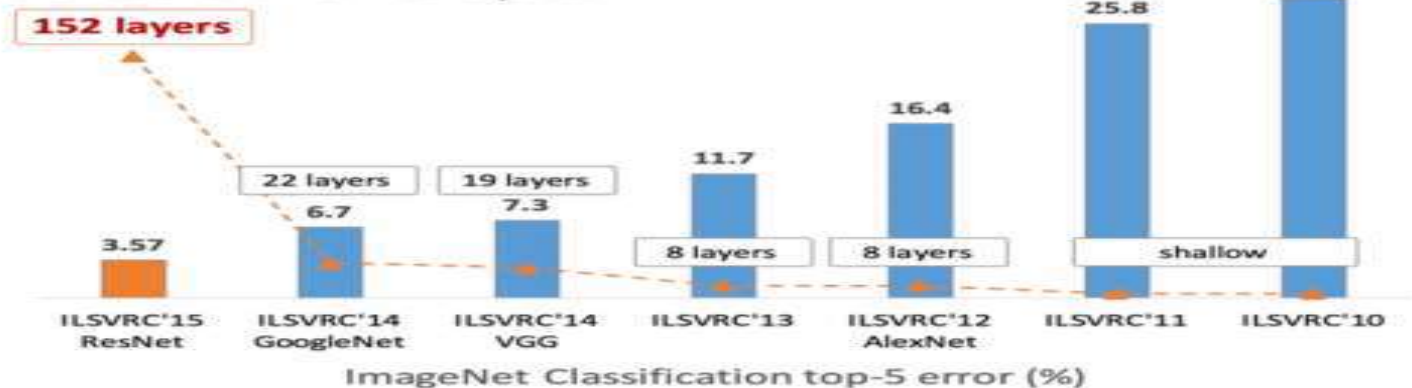
Motivations



NIPS(Computational Neuroscience Conference) Growth

Amount of Data vs Performance

Revolution of Depth



The Problem Space

- Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image.
- For humans, this task of recognition is one of the first skills we learn from the moment we are born .
- Without even thinking twice, we're able to quickly and seamlessly identify the environment and objects that surround us.
- When we see an image we are able to immediately characterize the scene and give each object a label, all without even consciously noticing.
- These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.

The Problem Space : inputs & outputs



What We See

```
08 02 22 97 38 18 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 40 87 17 40 98 43 49 48 04 56 42 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 92
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 96 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 44 70
67 26 20 68 02 62 12 20 95 43 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 94 89 14 88 34 89 43 72
21 36 23 09 75 00 76 44 20 45 35 14 00 41 33 97 34 31 33 95
78 17 83 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 25 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 88 66
88 36 48 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 34
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 14
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 97 05 54
01 70 54 71 83 51 54 69 16 82 33 48 61 43 52 01 89 19 47 48
```

What Computers See

- Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers
- These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer.
- The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for cat, .15 for dog, .05 for bird, etc).

The Problem Space :What We Want the Computer to Do

- 0 To be able to differentiate between all the images it's given and figure out the unique features that make a dog a dog or that make a cat a cat.
- 0 This is the process that goes on in our minds subconsciously as well.
- 0 When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs.
- 0 In a similar way, the computer is able perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers.

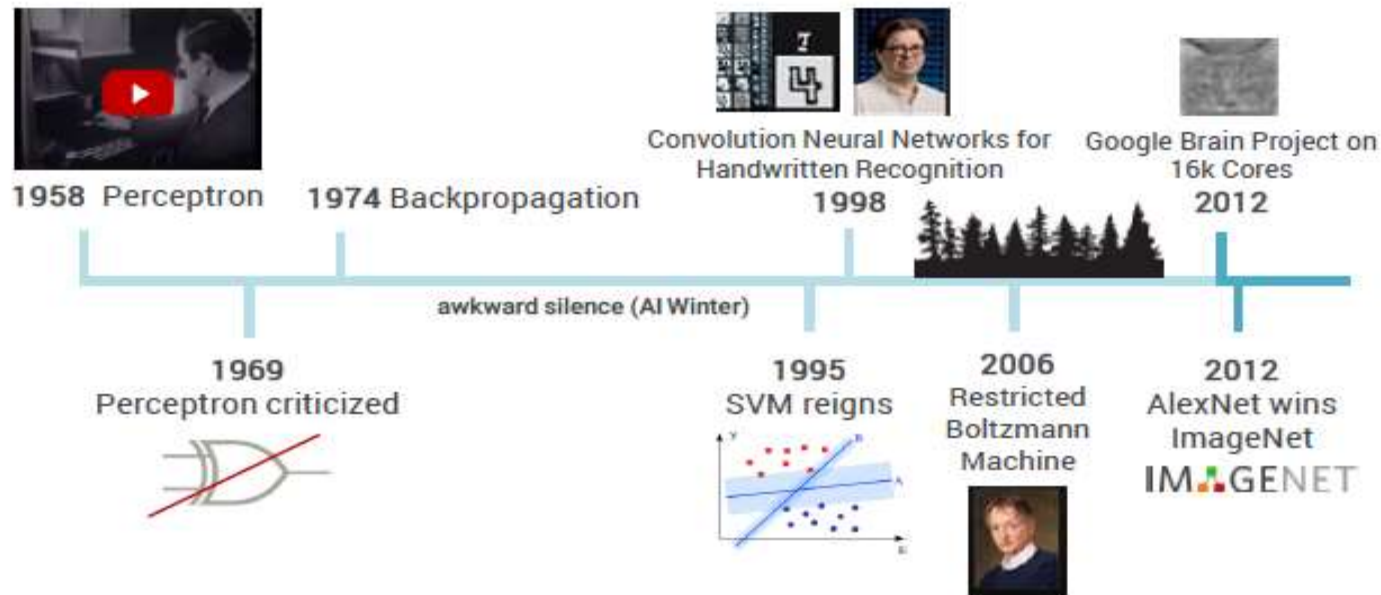
Why Deep Neural Network

- Imagine you have extracted features from sensors .
- The dimension of each sample is around 800
- You have 70,000 samples (trial)
- What method would you apply?
- **You may have several ways to do it**
- Reduce the dimension from 800 to 40 by using a feature selection or dim. reduction technique
- What you did here is “Finding a good representation”
- - Then, you may apply a classification methods to classify 10 classes
- But, what if**
- You have no idea for feature selection?
- The dimension is much higher than 800 and you have more classes.

Why Deep Neural Network

- Drawbacks -Back-Propagation:
- Scalability -**does not scale well** over multiple layers.
 - very slow to converge.
 - “**Vanishing gradient problem** “ **errors shrink exponentially with the number of layers**
 - Thus makes poor use of many layers.
 - This is the reason most feed forward NN have only three layers.
- **It got stuck at local optima**
 - When the network outputs have not got their desired signals, the activation functions in the hidden layer are saturating.
 - These were often surprisingly good but there was no good theory.
- Traditional Machine Learning **doesn't work well when features can't be explicitly defined.**

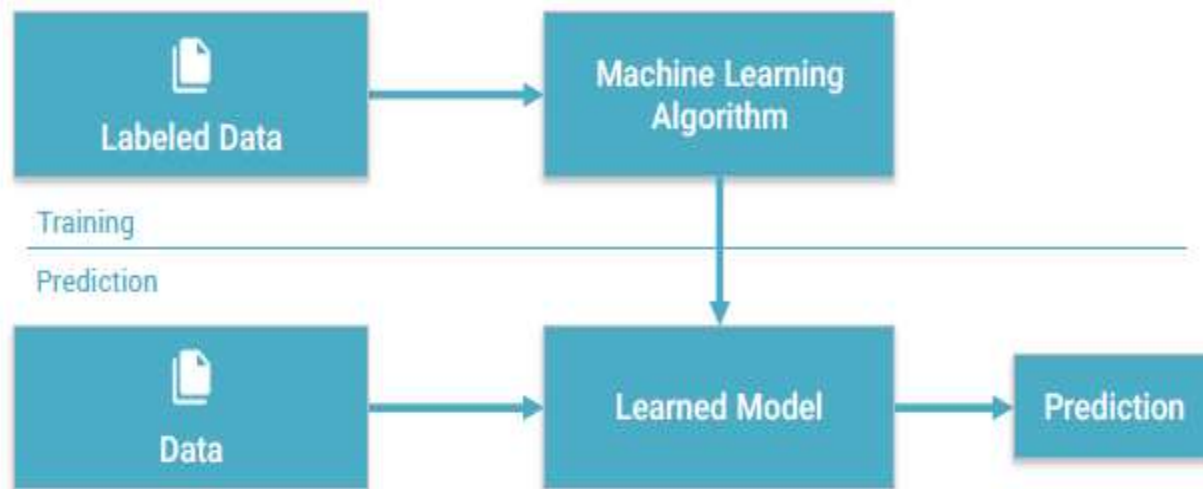
A Brief History



0 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time.

Machine Learning – Basics

Introduction



Machine Learning is a type of Artificial Intelligence that provides computers with the ability to learn without being explicitly programmed. Provides various techniques that can learn from and make predictions on data.

Machine Learning – Basics

Learning Approach



Supervised Learning : Learning with a **labeled training set**

Example : e-mail spam detector with training set of already labeled emails



Unsupervised Learning : **Discovering patterns** in unlabeled data

Example : cluster similar documents based on the text content



Reinforcement learning : learning based on **feedback** or reward.

Example : learn to play chess by wining or losing

Machine Learning – Basics

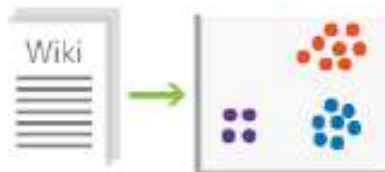
Problem Types



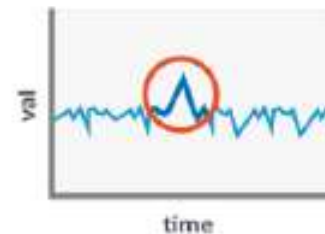
Classification
(supervised – predictive)



Regression
(supervised – predictive)



Clustering
(unsupervised – descriptive)



Anomaly Detection
(unsupervised – descriptive)

Big Players : Superstar Researcher



Geoffrey Hinton : University of Toronto
& Google



Yann Lecun : New York University
& Facebook



Andrew Ng : Stanford & Baidu



Yoshua Bengio : University of Montreal



Jürgen Schmidhuber : Swiss AI Lab &
NNAISENSE

Big Players : Companies



facebook



YAHOO!

Google



IBM



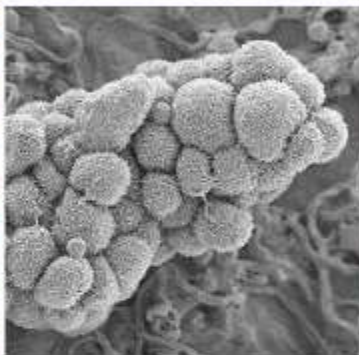
NVIDIA

Baidu 百度



Deep Learning : Applications

DEEP LEARNING EVERYWHERE



INTERNET & CLOUD

Image Classification
Speech Recognition
Language Translation
Language Processing
Sentiment Analysis
Recommendation

MEDICINE & BIOLOGY

Cancer Cell Detection
Diabetic Grading
Drug Discovery

MEDIA & ENTERTAINMENT

Video Captioning
Video Search
Real Time Translation

SECURITY & DEFENSE

Face Detection
Video Surveillance
Satellite Imagery

AUTONOMOUS MACHINES

Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

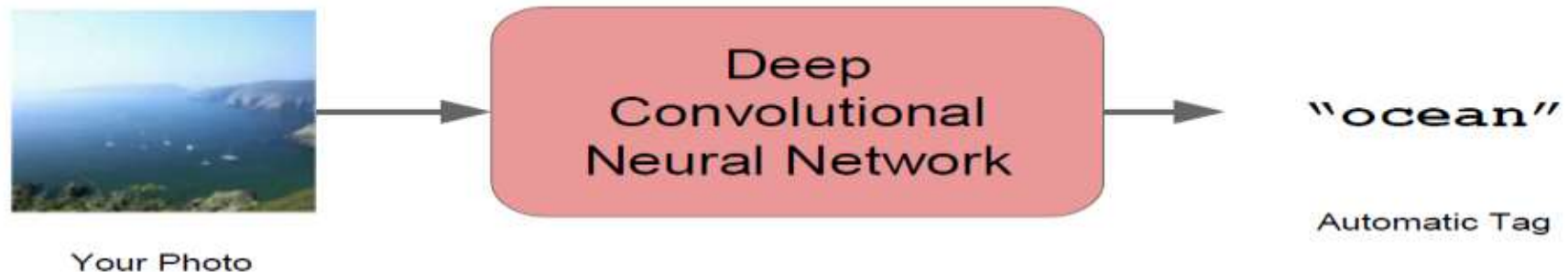
Google Application

0 What are some ways that deep learning is having a significant impact at Google?



Reduced word errors by more than 30%

Google Research Blog - August 2012, August 2015



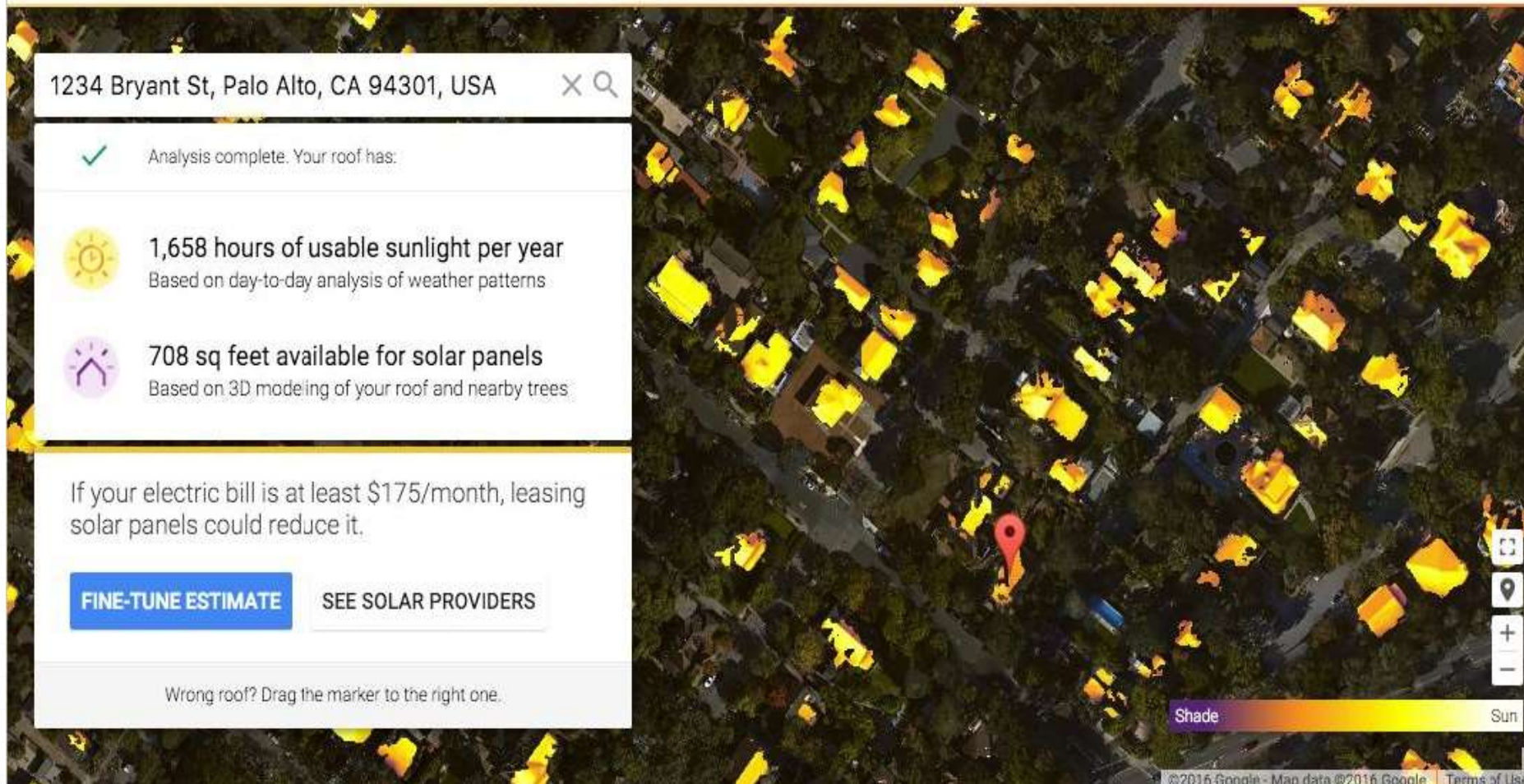
Search personal photos without tags.

Google Research Blog - June 2013

Google Application –Sunroof Project

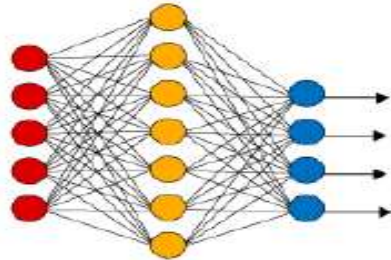
Google Project Sunroof

www.google.com/sunroof

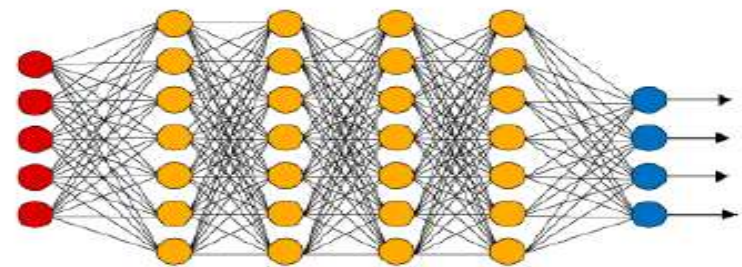


What is Deep Learning(DL)

Simple Neural Network



Deep Learning Neural Network



● Input Layer ● Hidden Layer ● Output Layer

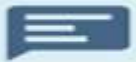
Stacked “Neural Network”. Is usually indicates “Deep Neural Network”.



Part or a powerful class of the machine learning field of learning representations of data. Exceptional effective at learning patterns.

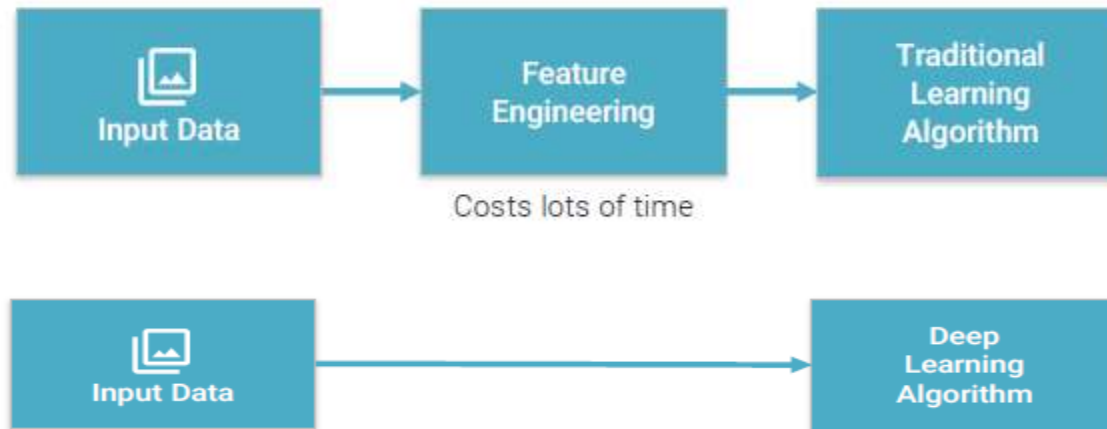


Utilize learning algorithms that derive meaning out of data by using **hierarchy** of multiple layers that **mimic the neural networks** of our brain.



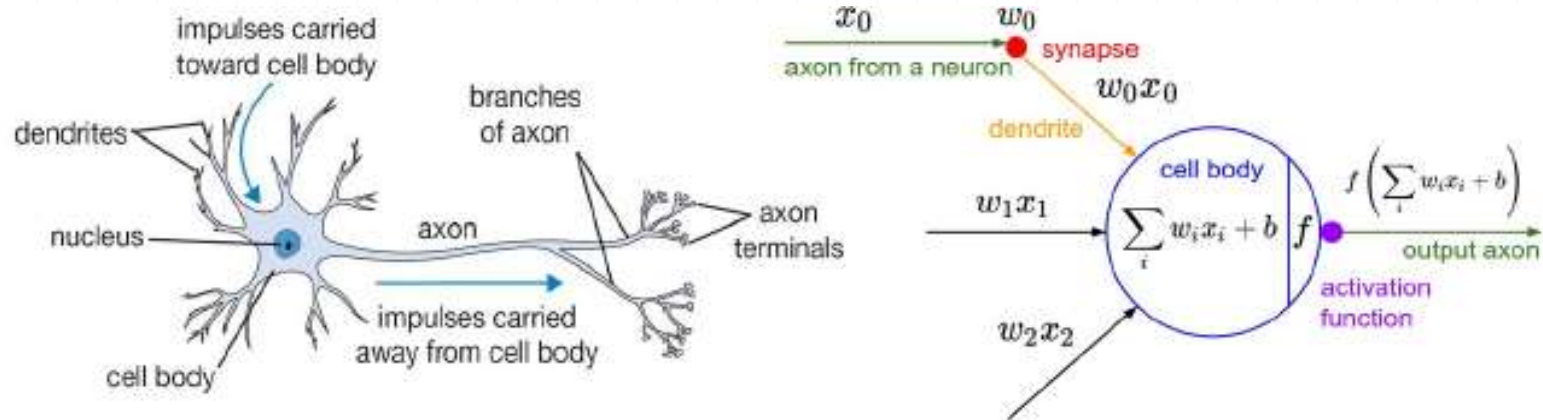
If you provide the system tons of information , it begins to understand it and respond in useful ways.

What is Deep Learning (DL)



- Collection of simple, trainable mathematical functions.
- Learning methods which have deep architecture.
- It often allows end-to-end learning.
- It automatically finds intermediate representation. Thus, it can be regarded as a representation learning.

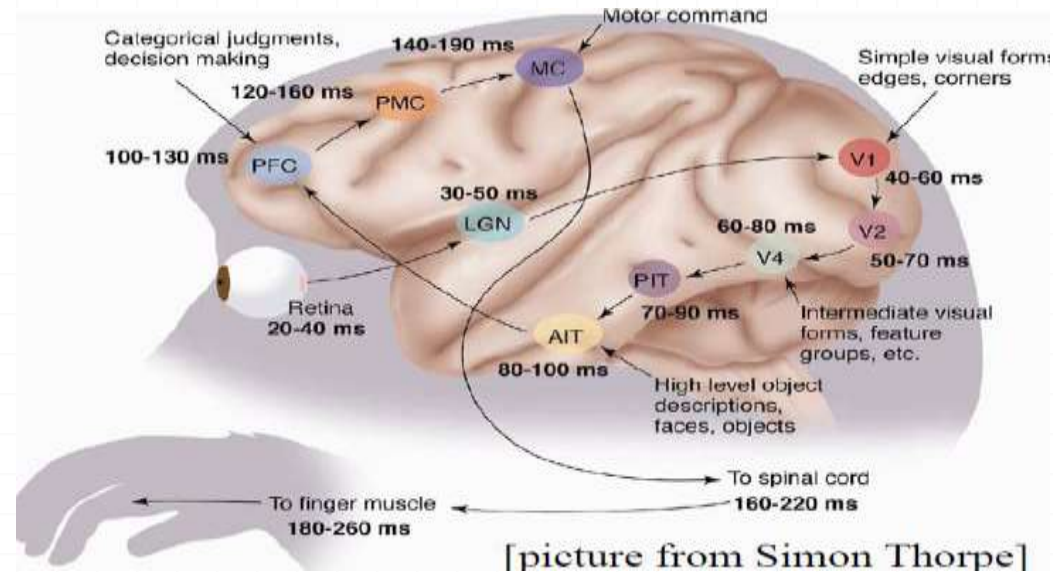
Deep Learning Basics :Neuron



An artificial neuron contains a nonlinear activation function and has several incoming and outgoing weighted connections.

Neurons are trained filters and detect specific features or patterns (e.g. edge, nose) by receiving weighted input, transforming it with the activation function and passing it to the outgoing connections

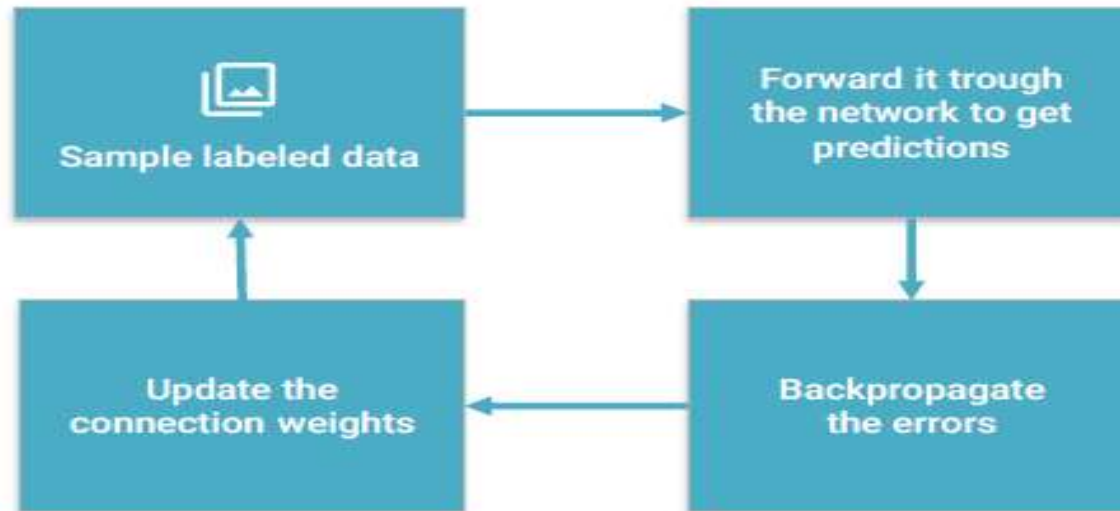
Deep Learning Basics :Neuron



Commonalities with real brains:

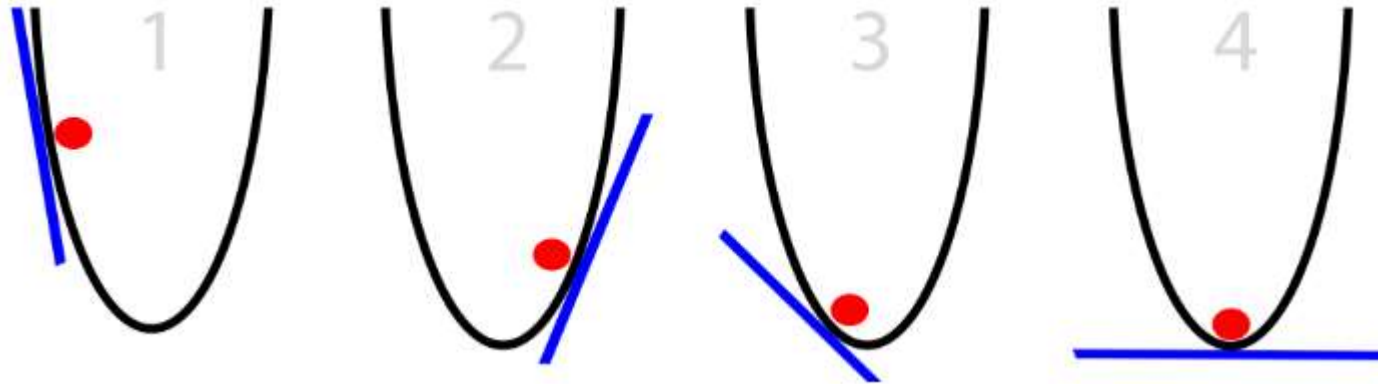
- Each neuron is connected to a small subset of other neurons.
- Based on what it sees, it decides what it wants to say.
- Neurons learn to cooperate to accomplish the task.
- The ventral pathway in the visual cortex has multiple stages
- There exist a lot of intermediate representations

Deep Learning Basics : Training process



Learns by generating an error signal that measures the difference between the predictions of the network and the desired values and then using this **error signal to change** the weights (or parameters) so the predictions get more accurate.

Deep Learning Basics :Gradient Descent



Gradient Descent/optimization **finds the (local)the minimum** of the cost function(used to calculate the output error) and is used to adjust the weights.

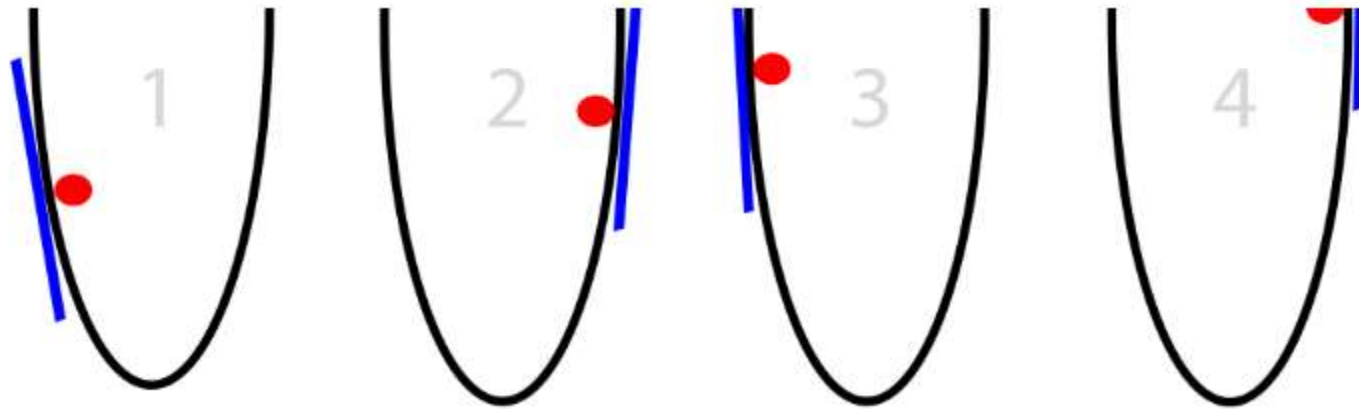
Curve represents is the network's error relative to the position of a single weight.
X axis \rightarrow weight Y axis \rightarrow error

Oversimplified Gradient Descent:

Calculate slope at current position

- If slope is negative, move right
- If slope is positive, move left
- (Repeat until slope == 0)

Deep Learning Basics :Gradient Descent Problems

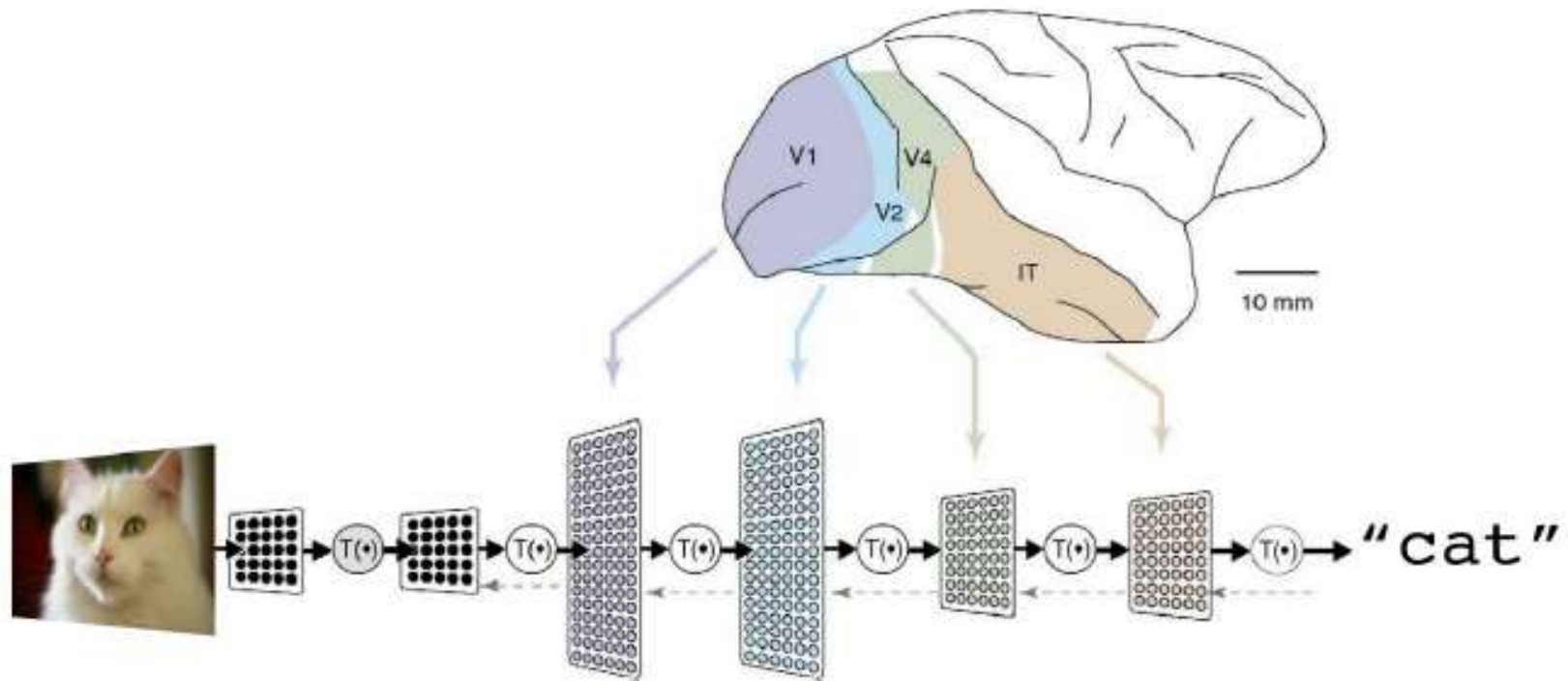


Problem 1: When slopes are too big → Solution 1: Make Slopes Smaller



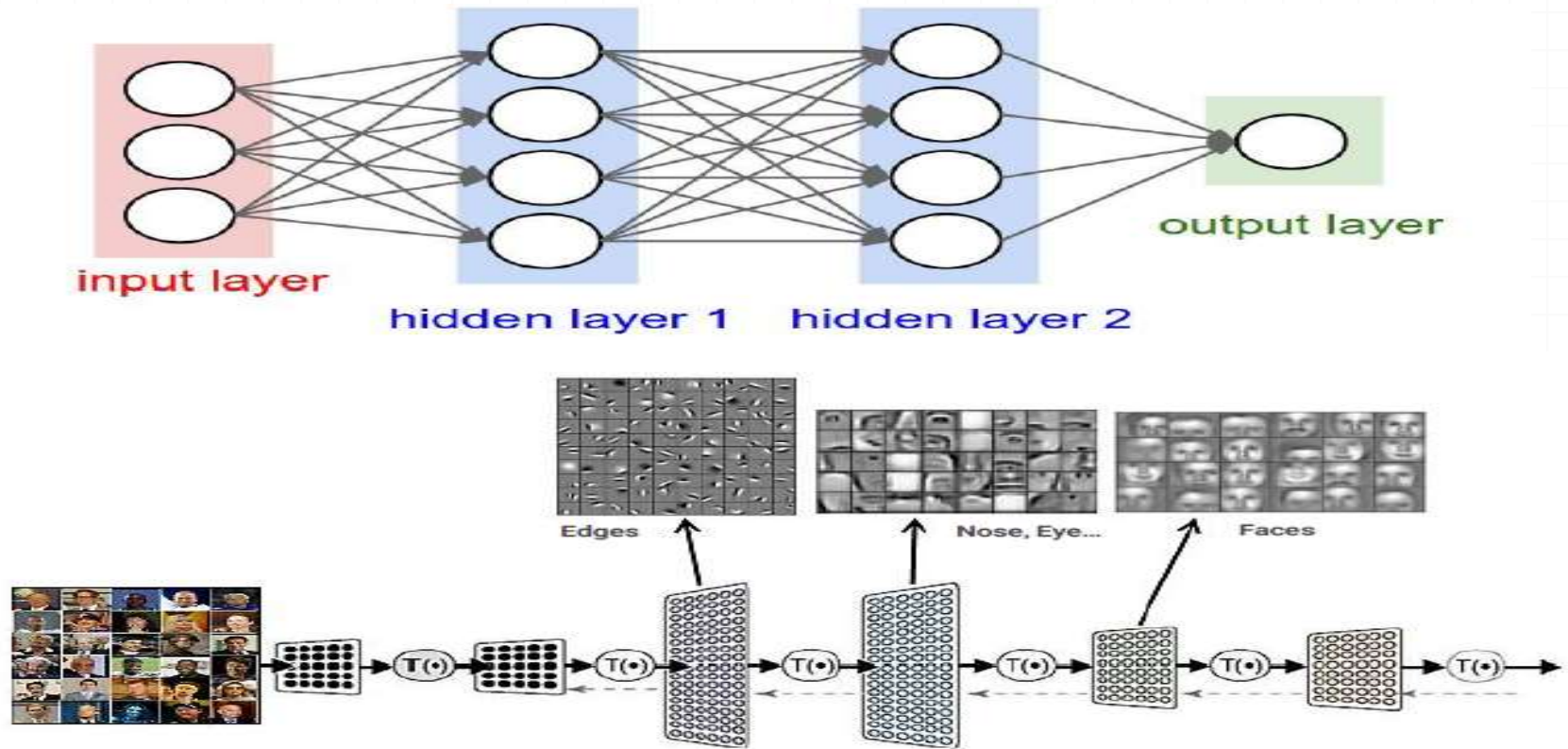
Problem 2: Local Minima-→Solution 2: Multiple Random Starting States

Deep Neural Network :Architecture



A Deep Neural Network consists of a **hierarchy of layers**, whereby each layer **transforms the input data** into more abstract representations (e.g edge -> nose -> face). The output layer combines those features to make predictions.

Deep Neural Network :Architecture



Consists of one input ,one output and multiple fully-connected hidden layers in between. Each layer is represented as a series of neurons and **progressively extracts higher and higher-level features** of the input until the final layer essentially makes a decision about what the input shows. The more layer the network has, the higher-level features it will learn.

Deep Neural Network : Architecture types

Feed-Forward

- Convolutional neural networks
- De-convolutional networks

Bi-Directional

- Deep Boltzmann Machines
- Stacked auto-encoders

Sequence -Based

- RNNs
- LSTMs

Types of Networks used for Deep Learning

Convolutional Neural Networks(Convnet,CNN)

Recurrent Neural Networks(RNN)

Long Short Term Memory (LSTM) networks

Deep/Restricted Boltzmann Machines (RBM)

Deep Q-networks

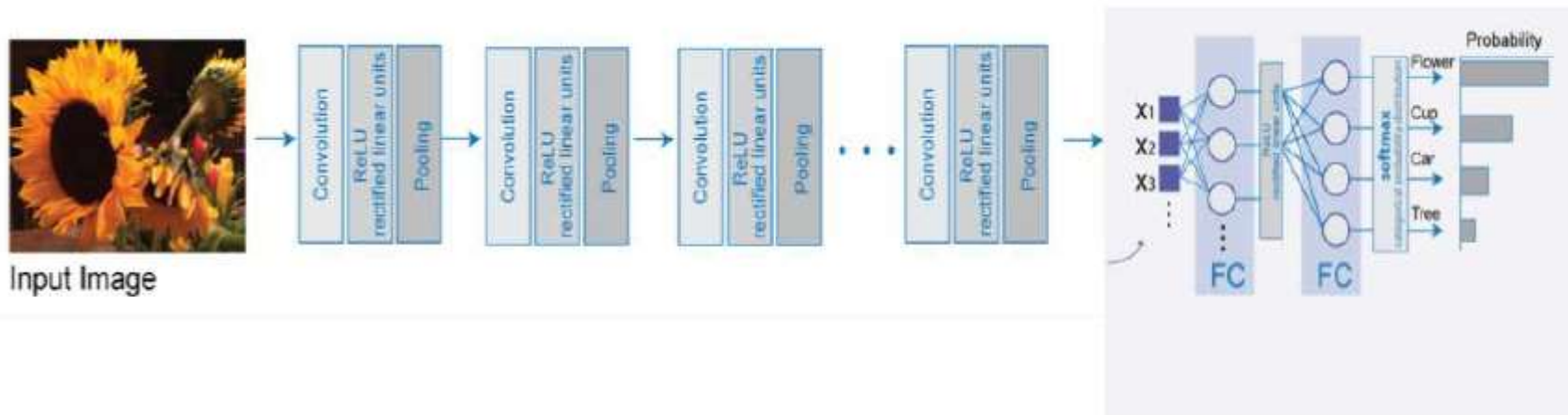
Deep Belief Networks(DBN)

Deep Stacking Networks

Convolution Neural Networks(CNN):Basic Idea

- This idea was expanded upon by a fascinating experiment by Hubel and Wiesel in 1962 where they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation.
- For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges. Hubel and Wiesel found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception.
- This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

Convolution Neural Networks(CNN)



DEFENITION

- Convolutional neural networks learn a complex representation of visual data using vast amounts of data .they are **inspired by human visual system** and learn **multiple layers of transformations** , which are applied on top of each other to extract progressively more **sophisticated representation of the input** .

Notes

- Inspired by the visual cortex and Pioneered by Yann Lecun (NYU).
- CNN have multiple types of layers ,the first of which is the Convolutional layer.

CNN Layers

1

- Convolution Layer

2

- Batch Normalization Layer

3

- RELU Layer

4

- Local Response Normalization Layer

5

- Max and AVG Pooling

6

- Dropout Layer

7

- Fully Connected Layer (FC)

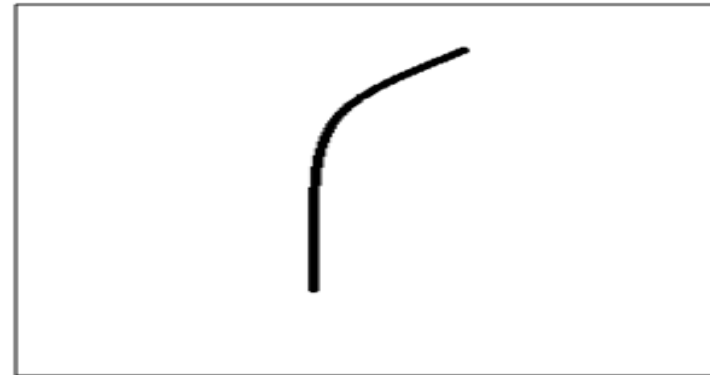
8

- Output Layer (SOFT MAX ,Regression)

CNN Structure :First Layer – Convolution

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Convolution layer is a feature detector that auto magically learns to filter out not needed information from an input by using convolution kernel.

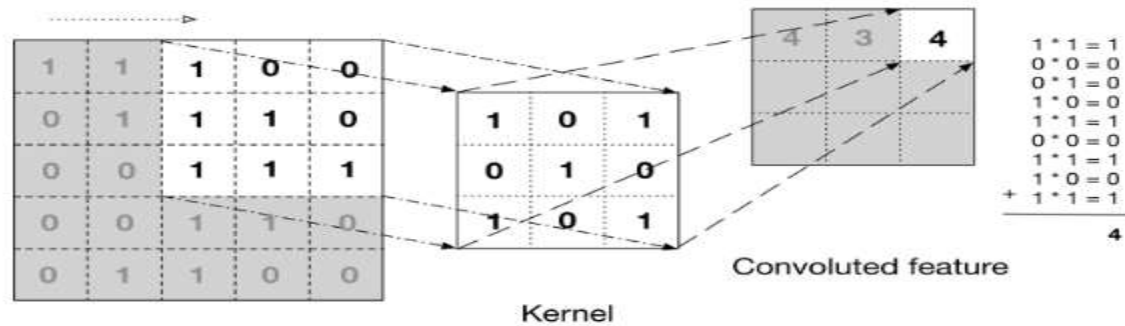


Original image



Visualization of the filter on the image

CNN Structure :First Layer – Convolution



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = $(50 \times 30) + (50 \times 30) + (50 \times 30) + (20 \times 30) + (50 \times 30) = 6600$ (A large number!)



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

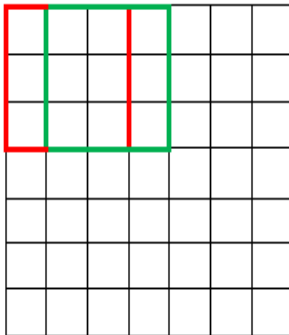
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

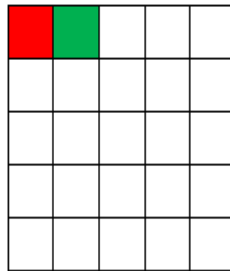
Multiplication and Summation = 0

Stride and Padding

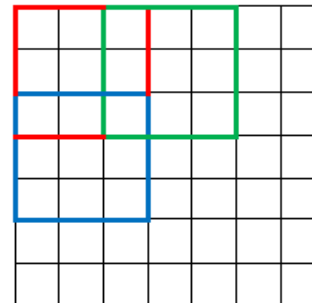
7 x 7 Input Volume



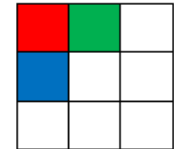
5 x 5 Output Volume



7 x 7 Input Volume



3 x 3 Output Volume

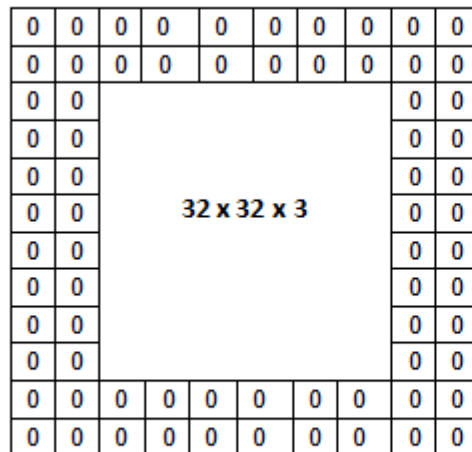


Stride set to 1

Stride set to 2



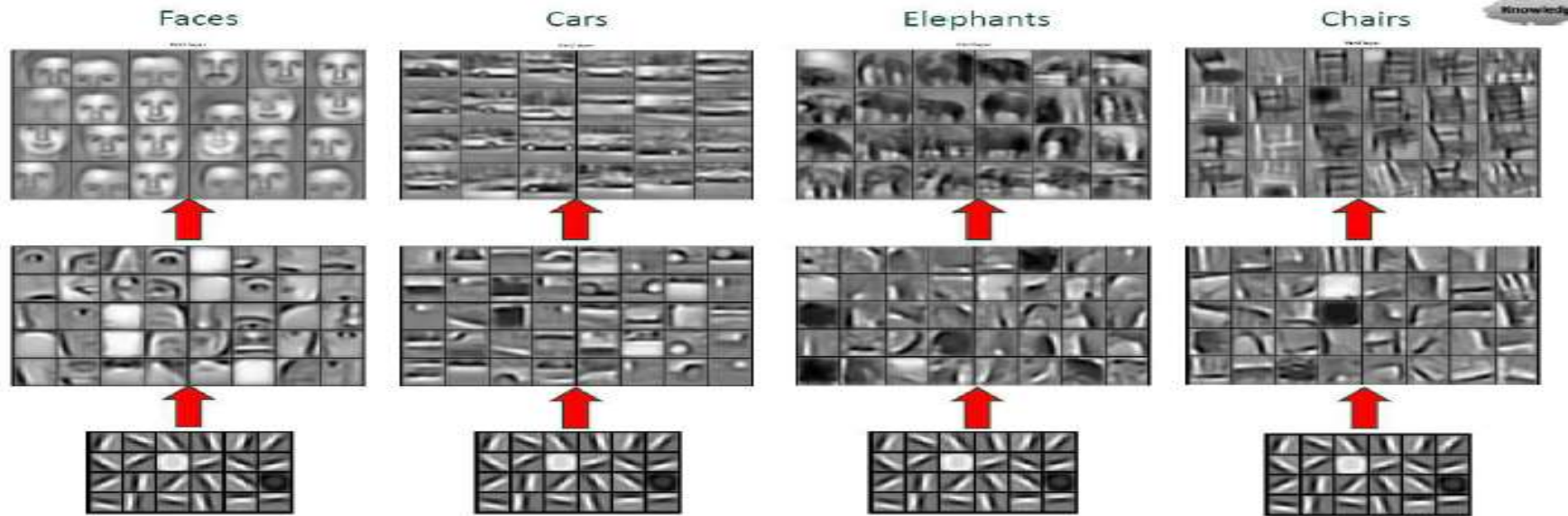
36



36

The input volume is $32 \times 32 \times 3$. If we imagine two borders of zeros around the volume, this gives us a $36 \times 36 \times 3$ volume. Then, when we apply our conv layer with our three $5 \times 5 \times 3$ filters and a stride of 1, then we will also get a $32 \times 32 \times 3$ output volume.

Examples of Actual Visualizations



"Convolutional deep belief networks for scalable unsupervised learning of hierarchical representation", Lee et al., 2012

End-to-End Learning

A pipeline of successive Layers

- Layers produce features of higher and higher abstractions
 - Initial Layer capture low-level features (e.g. edges or corners)
 - Middle Layer capture mid-level features (object parts e.g. circles, squares, textures)
 - Last Layer capture high level, class specific features (object model e.g. face detector)
- Preferably, input as raw as possible
 - Pixels for computer vision, words for NLP.

Batch Normalization Layer

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

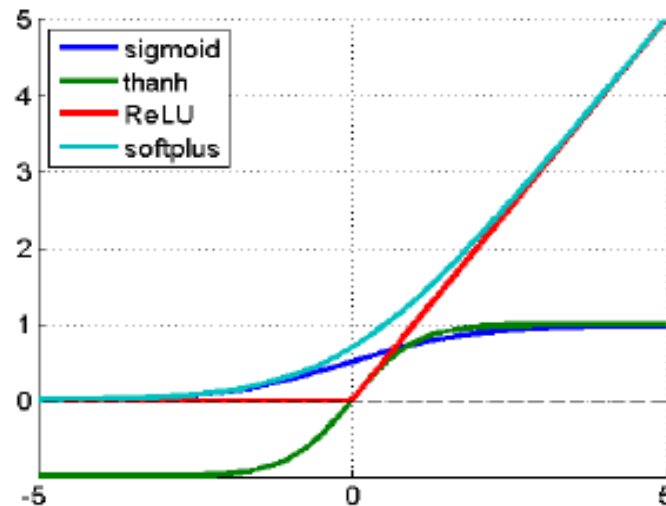
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

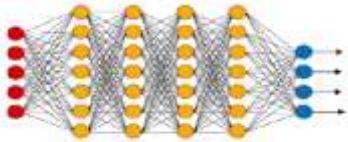
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

- 0 Batch Normalization is a technique to provide any layer in a Neural Network with inputs that are zero mean/unit variance.
- 0 Use batch normalization layers between convolutional layers and nonlinearities such as ReLU layers to speed up network training and reduce the sensitivity to network initialization.
- 0 The layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by an offset β and scales it by a scale factor γ . β and γ are themselves learnable parameters that are updated during network training.

Nonlinear Activation Function(Relu)



- Rectified Linear Unit (ReLU) module .
- Activation function $a=h(x)=\max(0,x)$.



Most deep networks use ReLU $-\max(0,x)$ -nowadays for hidden layers ,since it trains **much faster** ,is more expressive than logistic function and **prevents the gradient vanishing problem**.



Non-linearity is needed to learn complex (non linear) representations of data ,otherwise the NN would be just a linear function .

Vanishing Gradient Problem

- 0 It is a difficulty founds in training ANN with gradient-based learning methods and backpropagation.
- 0 In such methods, each of the neural network's weights receives an update proportional to the gradient of the error function with respect to the current weight in each iteration of training.
- 0 Traditional activation functions such as the hyperbolic tangent function have gradients in the range $(-1, 1)$, and backpropagation computes gradients by the chain rule.
- 0 This has the effect of multiplying n of these small numbers to compute gradients of the "front" layers in an n -layer network.
- 0 This means that the gradient (error signal) decreases exponentially with n while the front layers train very slowly.

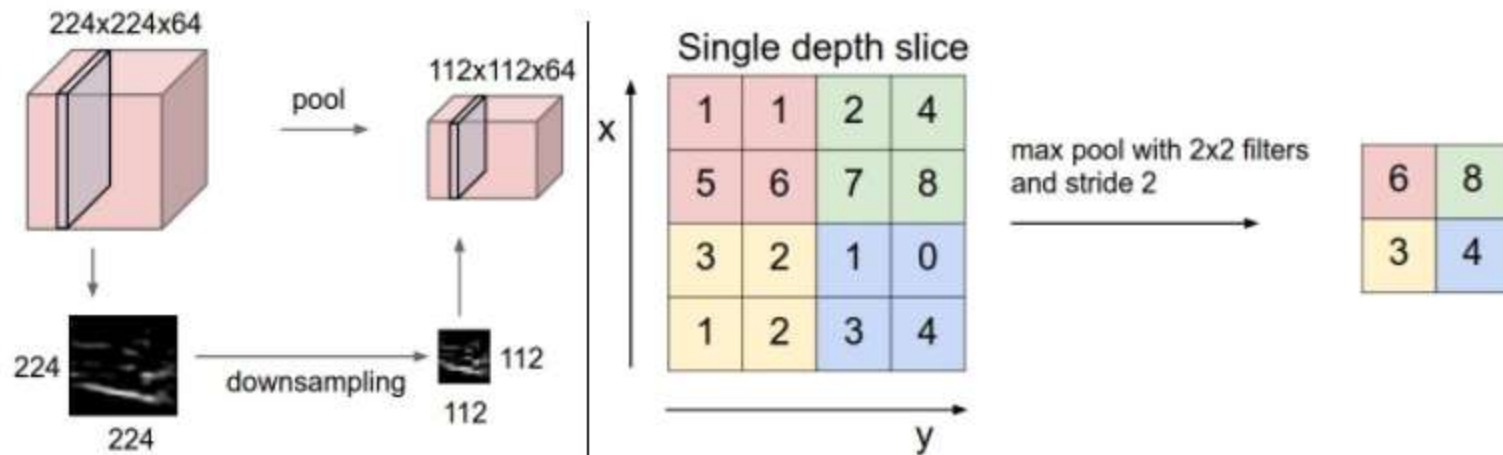
Local Response Normalization Layer

$$x' = \frac{x}{\left(K + \frac{\alpha * ss}{\text{windowChannelSize}}\right)^\beta}$$

where K , α , and β are the hyperparameters in the normalization, and ss is the sum of squares of the elements in the normalization window.

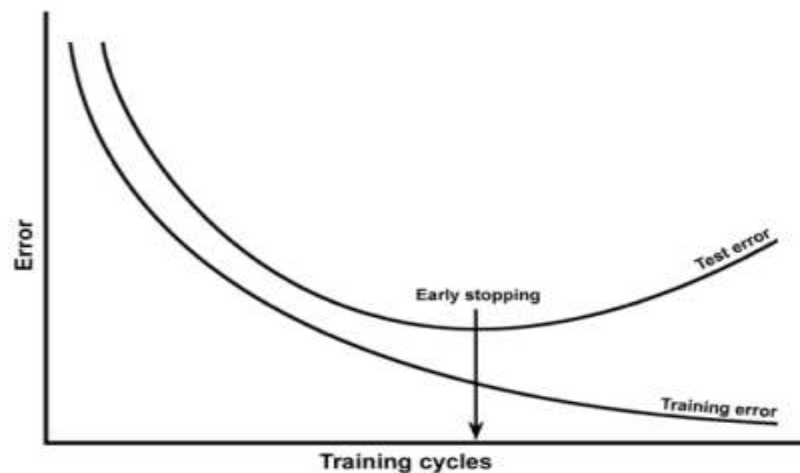
- we want to have some kind of inhibition scheme.
- Neurobiology concept “lateral inhibition”.
- Useful when we are dealing with ReLU neurons(unbounded activations)
- Detect high frequency features with a large response.
- If we normalize around the local neighborhood of the excited neuron, it becomes even more sensitive as compared to its neighbors.
- Inhibit the responses that are uniformly large in any given local neighborhood.
- If all the values are large, then normalizing those values will decrease all of them.
- Inhibition and boost the neurons with relatively larger activations.
- Two types of normalizations available in Caffe (same and cross channel)

Max and AVG Pooling Layer



- Pooling layers compute the max or average value of a particular feature over a region of the input data (downsizing of input images).
- Also helps to detect objects in some unusual places and reduces memory size.
- Aggregate multiple values into a single value
- – Reduces the size of the layer output/input to next layer ->Faster computations
- – Keeps most important information for the next layer
- • Max pooling/Average pooling

Over Fitting



- 0 **Over Fitting.** This term refers to when a model is so tuned to the training examples that it is not able to generalize well for the validation and test sets.
- 0 **A symptom of over Fitting** is having a model that gets 100% or 99% on the training set, but only 50% on the test data.
- 0 Implement dropout layers in order to combat the problem of over fitting to the training data.

Dropout Layers

- after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples.
- The idea of dropout is simplistic in nature. This layer "drops out" a random set of activations in that layer by setting them to zero. Simple as that.
- **Benefits**
- it forces the network to be redundant. By that the network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out.
- It makes **sure that the network isn't getting too "fitted" to the training data** and thus helps alleviate the over fitting problem.
- An important note is that this layer is only used during training, and not during test time.

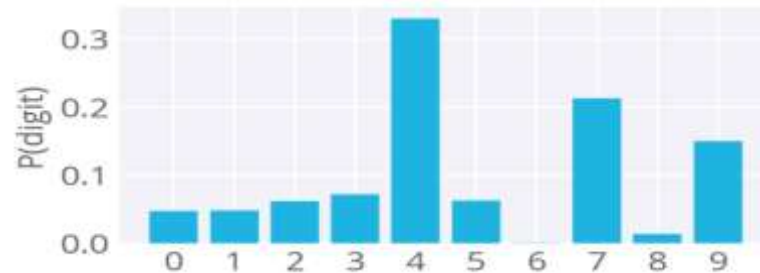
Output Layer –Soft-max Activation Function



$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$



For the example image above, the output of the softmax function might look like:



- For classification problems the sigmoid function can only handle two classes, which is not what we expect.
- The softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function.
- But it also divides each output such that the total sum of the outputs is equal to 1 .
- The output of the softmax function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true.

Output Layer –Regression

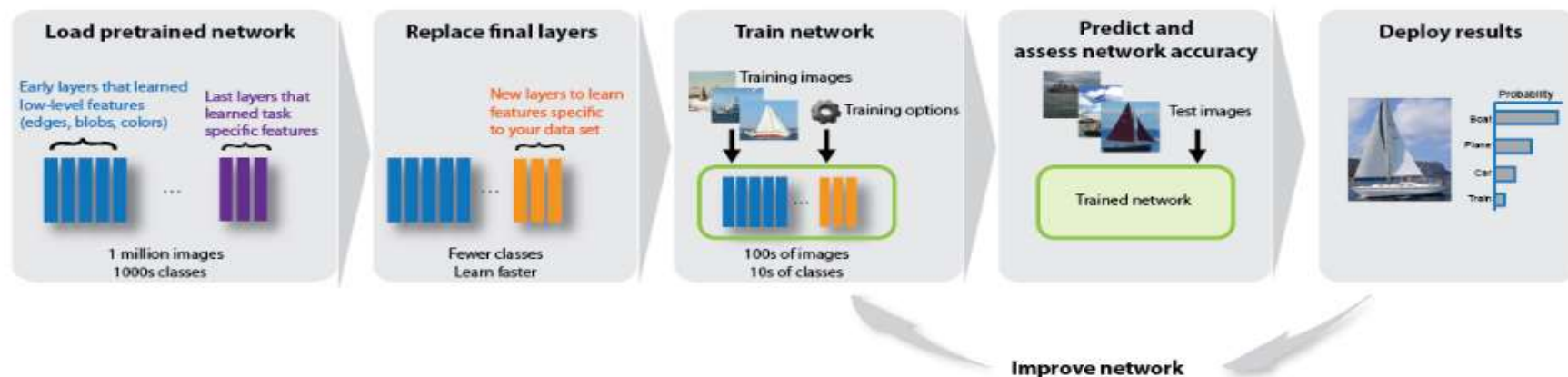
- 0 You can also use ConvNets for regression problems, where the target (output) variable is continuous.
- 0 In such cases, a regression output layer must follow the final fully connected layer. You can create a regression layer using the regressionLayer function.
- 0 The default loss function for a regression layer is the mean squared error.

$$MSE = E(\theta) = \sum_{i=1}^n \frac{(t_i - y_i)^2}{n},$$

- 0 where t_i is the target output, and y_i is the network's prediction for the response variable corresponding to observation i .

Transfer Learning

Reuse Pretrained Network

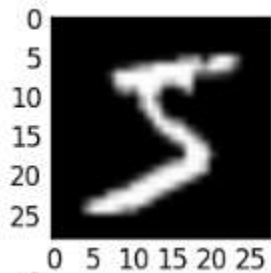


- 0 **Transfer learning** is the process of taking a pre-trained model (the weights and parameters of a network that has been trained on a large dataset by somebody else) and “fine-tuning” the model with your own dataset.
- 0 Pre-trained model will act as a feature extractor.
- 0 Freeze the weights of all the other layers .
- 0 Remove the last layer of the network and replace it with your own classifier.
- 0 Train the network normally.

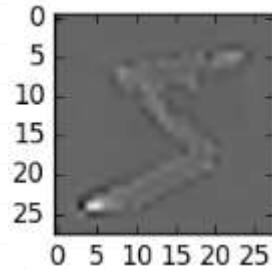
Transfer Learning:Cont.

- Transfer from A (image recognition) to B (radiology images)
- When transfer learning makes sense
 1. Task A and B have same input X.
 2. You have a lot more data for Task A than Task B.
 3. Low level features from A could be helpful for learning B.
 4. Weights initialization.
- When transfer learning doesn't makes sense
 1. You have a lot more data for Task B than Task A.

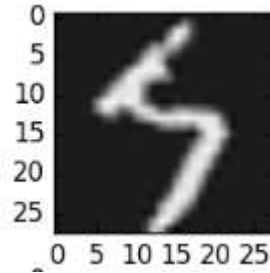
Data Augmentation Techniques



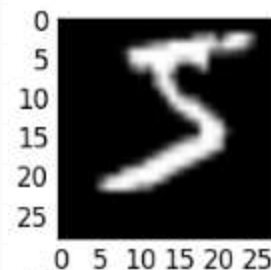
Example MNIST
images



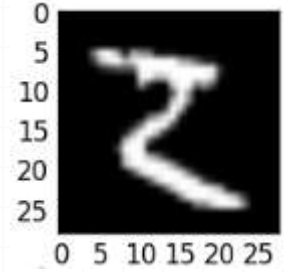
ZCA Whitening



Random Rotations



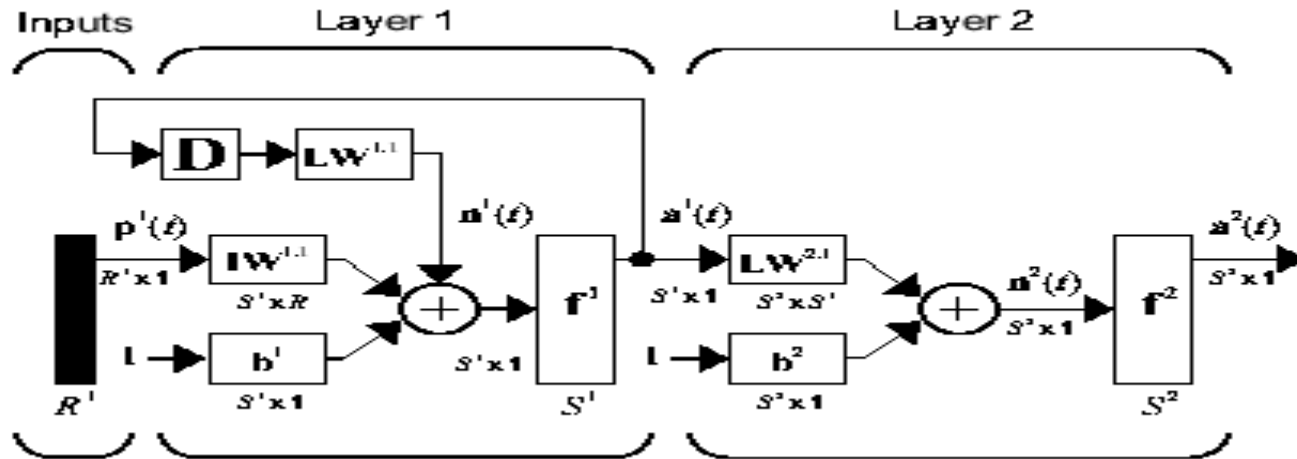
Random shift



Random flip

- Approaches that alter the training data in ways that change the array representation while keeping the label the same.
- They are a way to artificially expand your dataset. Some popular augmentations people use are gray scales , horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more .

Recurrent Neural Network (RNN)



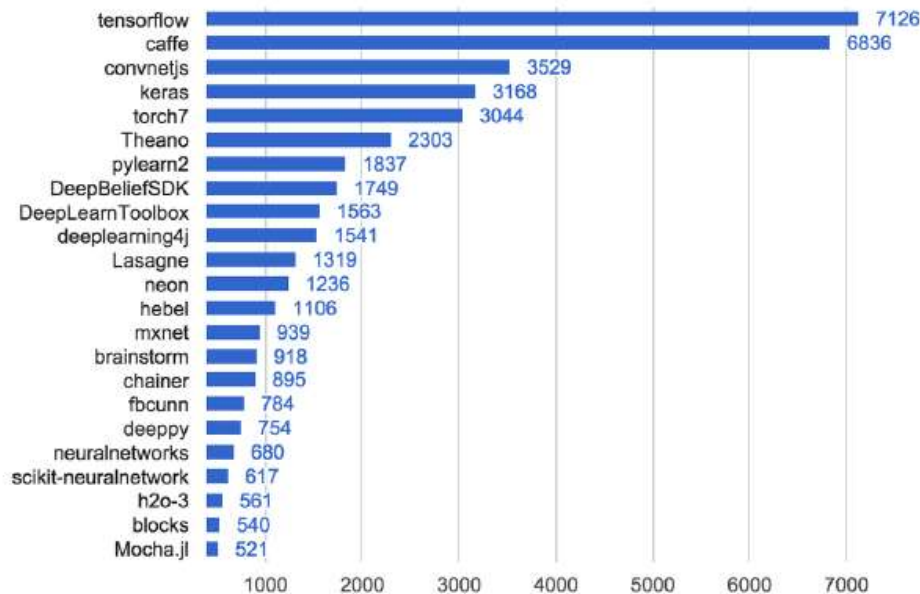
RNNs are **general computers which can learn algorithms to map input sequences to output sequences** (flexible -sized vectors). The output vector's contents influenced by the entire history of input.



Applications

- In time series prediction , adaptive robotics, handwriting recognition ,image classification, speech recognition, stock market prediction, and other sequence learning problems .every thing can be processed sequentially

Deep Learning Tools



	Caffe	Torch	Theano	TensorFlow
Language	C++, Python	Lua	Python	Python
Pretrained	Yes ++	Yes ++	Yes (Lasagne)	Inception
Multi-GPU: Data parallel	Yes	Yes cunn. DataParallelTable	Yes platoon	Yes
Multi-GPU: Model parallel	No	Yes fbcunn.ModelParallel	Experimental	Yes (best)
Readable source code	Yes (C++)	Yes (Lua)	No	No
Good at RNN	No	Mediocre	Yes	Yes (best)

Platform

- Ersatz Labs - cloud-based deep learning platform [<http://www.ersatz1.com/>]
- H2O – deep learning framework that comes with R and Python interfaces [<http://www.h2o.ai/verticals/algos/deep-learning/>]

Framework

- Caffe - deep learning framework made with expression, speed, and modularity in mind. Developed by the Berkeley Vision and Learning Center (BVLC) [<http://caffe.berkeleyvision.org/>]
- Torch - scientific computing framework with wide support for machine learning algorithms that puts GPUs first. Based on Lua programming language [<http://torch.ch/>]

Library

- Tensorflow - open source software library for numerical computation using data flow graphs from Google [<https://www.tensorflow.org/>]
- Theano - a python library developed by Yoshua Bengio's team [<http://deeplearning.net/software/theano/>]

What changed



Big Data
(Digitalization)



Computation
(Moore's Law, GPUs)



Algorithmic
Progress

Imagenet
,youtube-8M
,AWS public
dataset, Baidu's
Chinese speech
recognition

Usage Requirements



Large data set with good quality (input –output mapping)



Measurable and describable goals (define the cost)



Enough computing power(AWS GPU Instance)



Excels in tasks where the basic unit (pixel, word) has very little meaning in itself, but **the combination of such units has a useful meaning.**

Deep Learning : Benefits

Robust

- No need to design the features ahead of time –features are automatically learned to be optimal for the task at hand
- Robustness to natural variations in the data is automatically learned

Generalizable

- The same neural network approach can be used for many different applications and data types

Scalable

- Performance improves with more data ,method is massively parallelizable

Deep Learning: Weakness

1

- Deep learning **requires a large dataset**, hence long training period.

2

- In term of cost, Machine learning methods like SVM and other tree ensembles are very easily deployed even by relative machine learning novices and can usually get you reasonably good results

3

- Deep learning methods **tends to learn everything**. It's better to encode prior knowledge about structure of images (or audio or text).

4

- The learned features are often **difficult to understand**. Many vision features are also not really human-understandable (e.g, concatenations/combinations of different features).

5

- Requires **a good understanding of how to model** multiple modalities with traditional tools.

Pre-trained Deep Learning models

AlexNet (2012)

- Authors :Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton
- Winner 2012.
- Trained the network on ImageNet data, which contained over 15 million
- used for classification with 1000 possible categories
- Use 11x11 sized filters in the first layer.
- Used ReLU for the nonlinearity functions .
- Used data augmentation techniques that consisted of image translations, horizontal reflections, and patch extractions.
- Implemented dropout layers.
- Trained the model using batch stochastic gradient descent, with specific values for momentum and weight decay.
- Trained on two GTX 580 GPUs for **five to six days**
- This model achieved an 15.4% error rate.

ZF Net (2013)

built by Matthew Zeiler and Rob Fergus from NYU.

Winner 2013.

Very similar architecture to AlexNet, except for a few minor modifications.

ZF Net trained on only 1.3 million images.

ZF Net used filters of size 7x7

Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent.

Trained on a GTX 580 GPU for **twelve days**. Developed a visualization technique named Deconvolutional Network, which helps to examine different feature activations and their relation to the input space. Called “deconvnet” because it maps features to pixels (the opposite of what a convolutional layer does).

This model achieved an 11.2% error rate

Pre-trained Deep Learning models

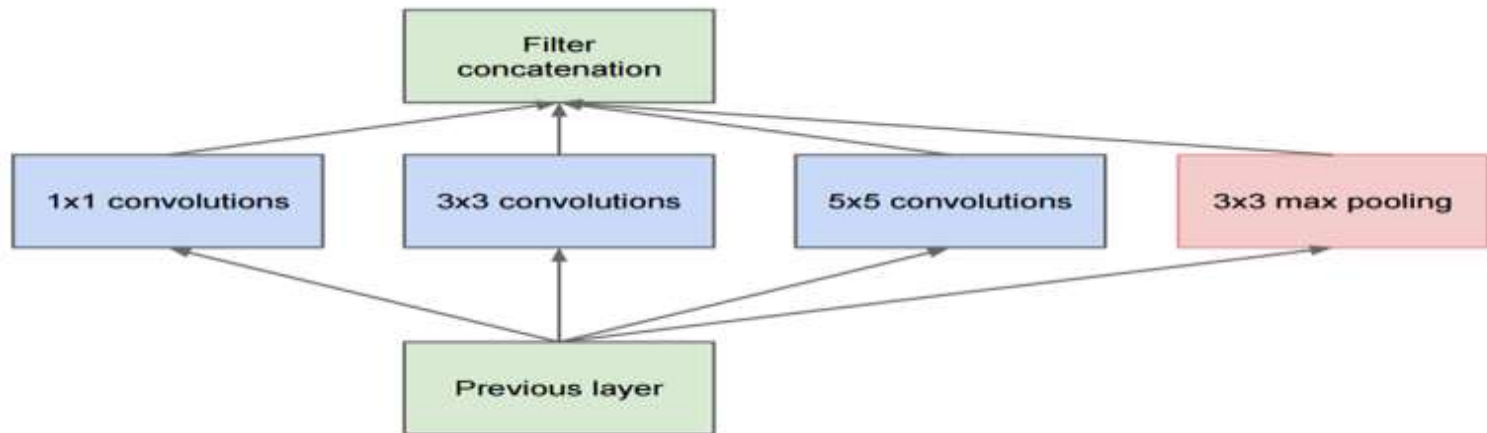
VGG Net (2014)

- Built by Karen Simonyan and Andrew Zisserman of the University of Oxford
- Not a winner .
- 19 layer CNN , used 3x3 sized filters with stride and pad of 1 , 2x2 max pooling layers with stride 2.
- The combination of two 3x3 conv layers has an effective receptive field of 5x5.
- Decrease in the number of parameters.
- Also, with two conv layers, we're able to use two ReLU layers instead of one. 3 conv layers back to back have an effective receptive field of 7x7.
- Interesting to notice that the number of filters doubles after each max pool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth.
- Worked well on both image classification and localization tasks.
- used a form of localization as regression Built model with the Caffe toolbox.
- Used scale jittering as one data augmentation technique during training.
- Used ReLU layers after each conv layer and trained with batch gradient descent.
- Trained on 4 Nvidia Titan Black GPUs for **two to three weeks**.
- This achieved an 7.3% error rate

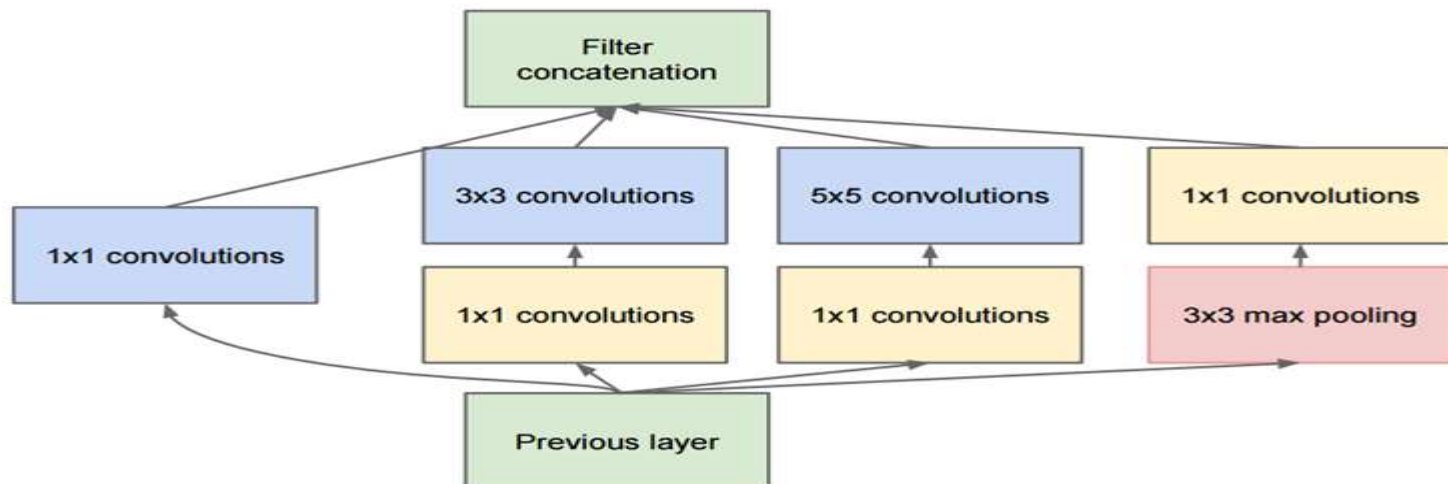
Google Net (2015)

- GoogLeNet is a 22 layer CNN.
- Winner 2014
- Used 9 Inception modules in the whole architecture, with over 100 layers in total.
- No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume.
- Uses 12x fewer parameters than AlexNet.
- During testing, multiple crops of the same image were created, fed into the network, and the softmax probabilities were averaged to give us the final solution.
- Utilized concepts from R-CNN for their detection model.
- This model places notable consideration on memory and power usage
- Trained on “a few high-end GPUs **within a week**”.
- This achieved an 6.7% error rate

Google Net : Inception Module

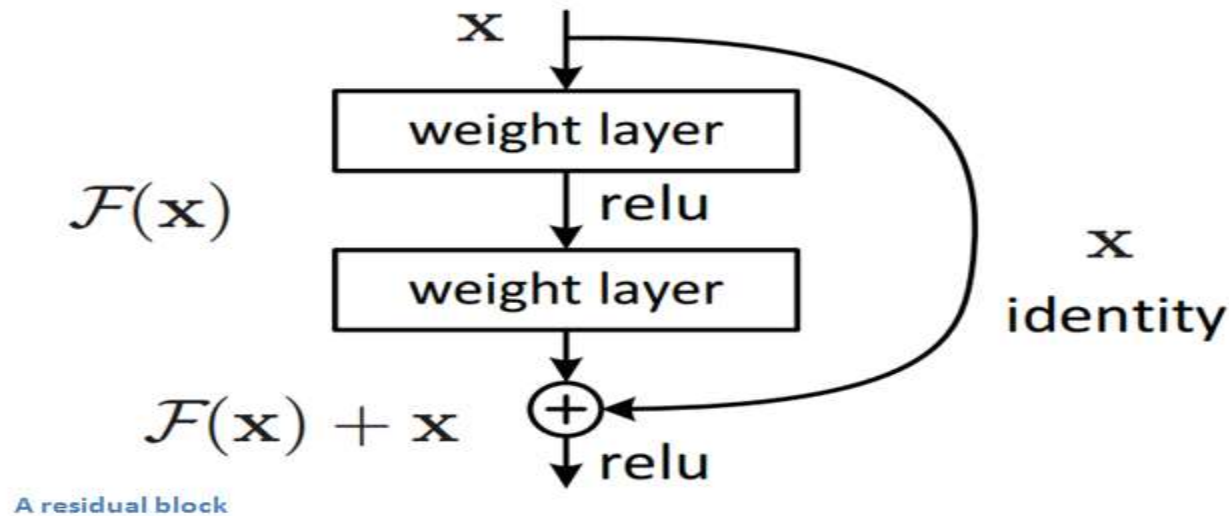


Naïve idea of an Inception module



Full Inception module

Pre-trained Deep Learning models



Microsoft ResNet (2015)

input x go through conv-relu-conv series.

152 layer, "Ultra-deep" – Yann LeCun.

Winner 2015.

After only the *first 2* layers, the spatial size gets compressed from an input volume of 224×224 to a 56×56 volume.

increase of layers in plain nets result in higher training and test error (author claims).

The authors believe that "it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping

The group tried a 1202-layer network, but got a lower test accuracy, presumably due to over-fitting.

Trained on an 8 GPU machine for **two to three weeks**.

Achieved an 3.6 % error rate.

2016 LSVRC winner(CUImage team)

- Compared with CUImage submission in ILSVRC 2015, the new components are as follows.
 - (1) The models are pretrained for 1000-class object detection task using the approach in [a] but adapted to the **fast-RCNN** for faster detection speed.
 - (2) The region proposal is obtained using the improved version of CRAFT in [b].
 - (3) A GBD network [c] with 269 layers is fine-tuned on 200 detection classes with the gated bidirectional network (GBD-Net), which passes messages between features from different support regions during both feature learning and feature extraction. The GBD-Net is found to bring ~3% mAP improvement on the baseline 269 model and ~5% mAP improvement on the Batch normalized GoogleNet.
 - (4) For handling their long-tail distribution problem, the 200 classes are clustered. Different from the original implementation in [d] that learns several models, a single model is learned, where different clusters have both shared and distinguished feature representations.
 - (5) Ensemble of the models using the approaches mentioned above lead to the final result in the provided data track.
 - (6) For the external data track, we propose object detection with landmarks. Comparing to the standard bounding box centric approach, our landmark centric approach provides more structural information and can be used to improve both the localization and classification step in object detection. Based on the landmark annotations provided in [e], we annotate 862 landmarks from 200 categories on the training set. Then we use them to train a CNN regressor to predict landmark position and visibility of each proposal in testing images. In the classification step, we use the landmark pooling on top of the fully convolutional network, where features around each landmark are mapped to be a confidence score of the corresponding category. The landmark level classification can be naturally combined with standard bounding box level classification to get the final detection result.
 - (7) Ensemble of the models using the approaches mentioned above lead to the final result in the external data track. The fastest publicly available multi-GPU caffe code is our strong support [f].

2017 LSVRC winner(BDAT team)

- 0 Adaptive attention[1] and deep combined convolutional models[2,3] are used for LOC task.
- 0 Deep residual learning for image recognition.
- 0 Scale[4,5,6], context[7], sampling and deep combined convolutional networks[2,3] are considered for DET task.
- 0 Object density estimation is used for score re-rank

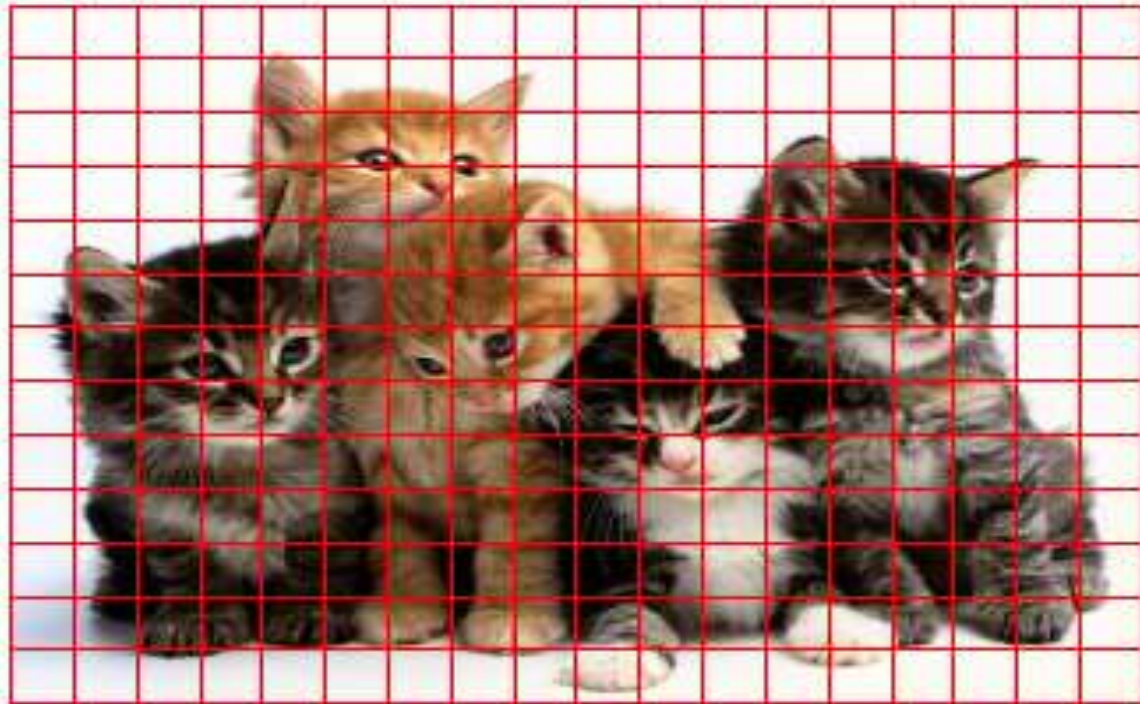
Object Recognition

Goal:



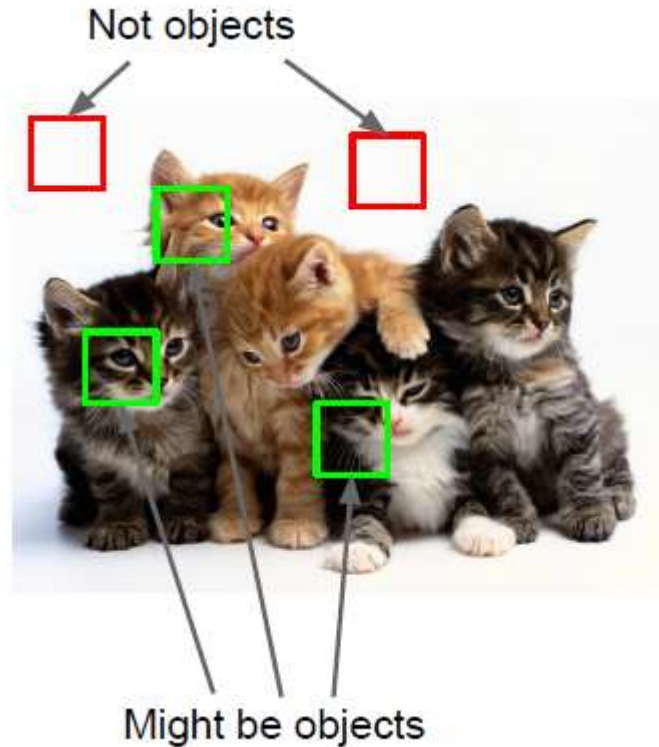
○ Problem: Where do we look in the image for the object?

Solution 1: Exhaustively search for objects.



- **Problem:** Extremely slow, must process tens of thousands of candidate objects.

Solution 2:



Running a scanning **detector** is cheaper than running a recognizer, so do that first.

1. Exhaustively search for candidate objects with a generic detector.
2. Run recognition algorithm only on candidate objects.

Problem: What about oddly-shaped
objects? Will we need to scan with windows of many different shapes?

Segmentation



- **Idea:** If we correctly segment the image before running object recognition, we can use our segmentations as candidate objects.
- **Advantages:** Can be efficient, makes no assumptions about object sizes or shapes.

Segmentation is Hard



Kittens are distinguishable by color (sort of), but not texture.



Chameleon is distinguishable by texture, but not color.

- 0 As we saw in Project 1, it's not always clear what separates an object.

Segmentation is Hard



Wheels are part of the car, but not similar in color or texture.



How do we recognize that the head and body/sweater are the same “person”?

0 As we saw in Project 1, it's not always clear what separates an object.

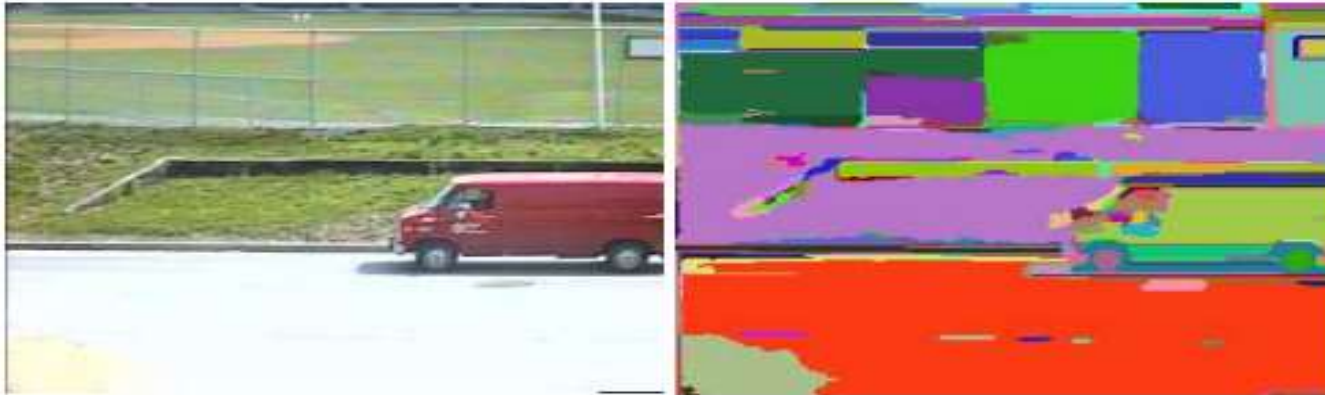
Selective Search

Goals:

1. Detect objects at any scale.
 - a. Hierarchical algorithms are good at this.
2. Consider multiple grouping criteria.
 - a. Detect differences in color, texture, brightness, etc.
3. Be fast.

Idea: Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.

Selective Search



Step 1: Generate initial sub-segmentation

Goal: Generate many regions, each of which belongs to at most one object.

Using the method described by Felzenszwalb et al. from week 1 works well.

Selective Search

Step 2: Recursively combine similar regions into larger ones.

Greedy algorithm:

1. From set of regions, choose two that are most similar.
2. Combine them into a single, larger region.
3. Repeat until only one region remains.

This yields a hierarchy of successively larger regions, just like we want.

Similarity

What do we mean by “**similarity**”?

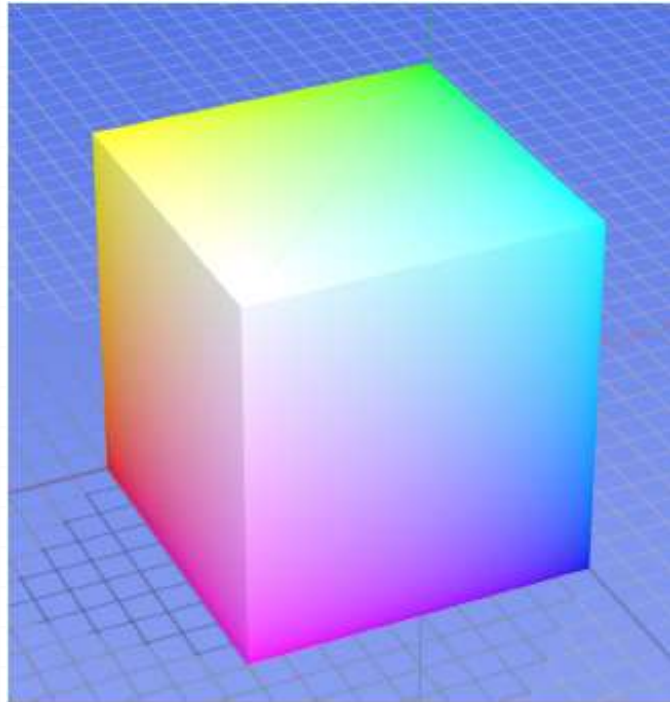
Goals:

1. Use multiple grouping criteria.
2. Lead to a balanced hierarchy of small to large objects.
3. Be efficient to compute: should be able to quickly combine measurements in two regions.

Two-pronged approach:

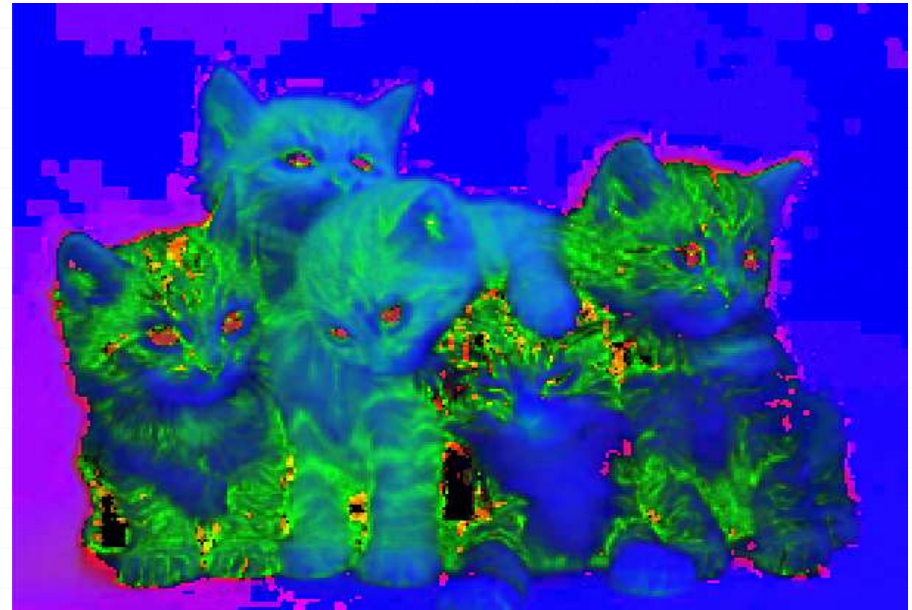
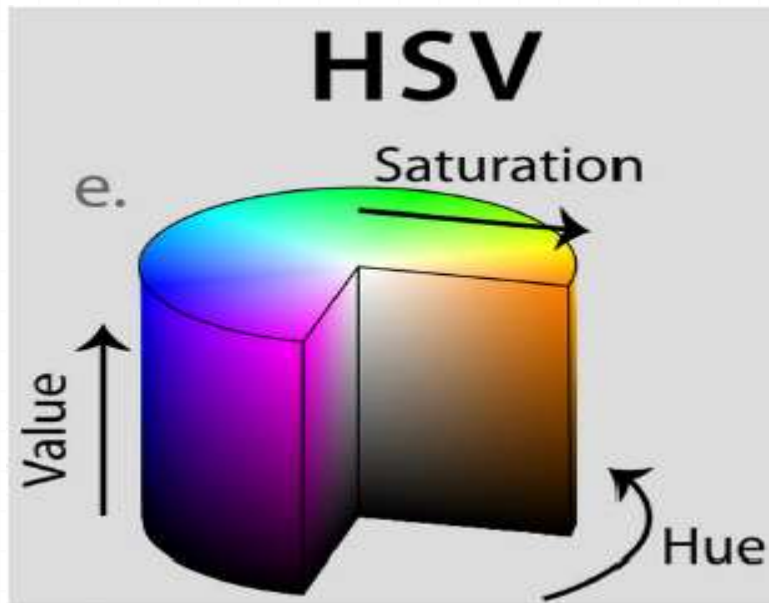
1. Choose a **color space that captures interesting things**.
 - a. Different color spaces have different invariants, and different responses to changes in color.
2. Choose a **similarity metric for that space that captures everything we're** interested: color, texture, size, and shape.

Similarity



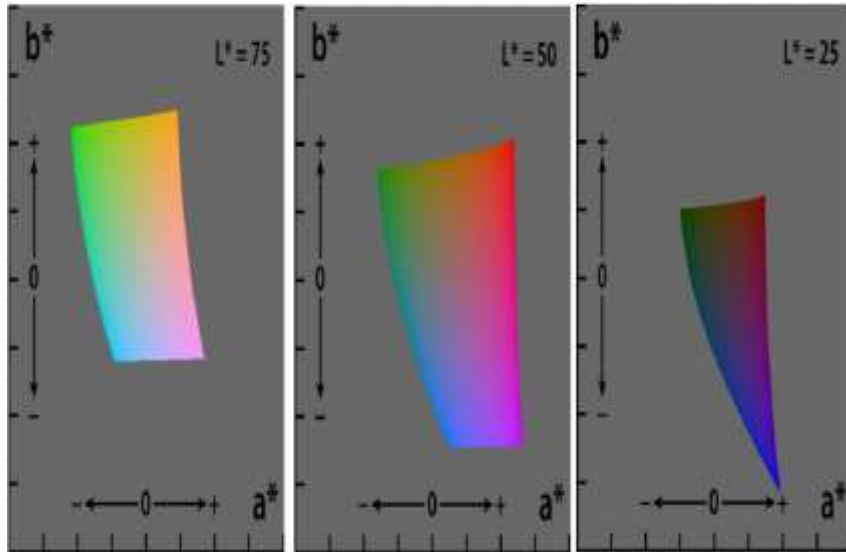
- **RGB (red, green, blue) is a good baseline, but changes in illumination (shadows, light intensity) affect all three channels.**

Similarity



- HSV (hue, saturation, value) encodes color information in the **hue channel**, which is invariant to changes in lighting. Additionally, saturation is insensitive to shadows, and value is insensitive to brightness changes.

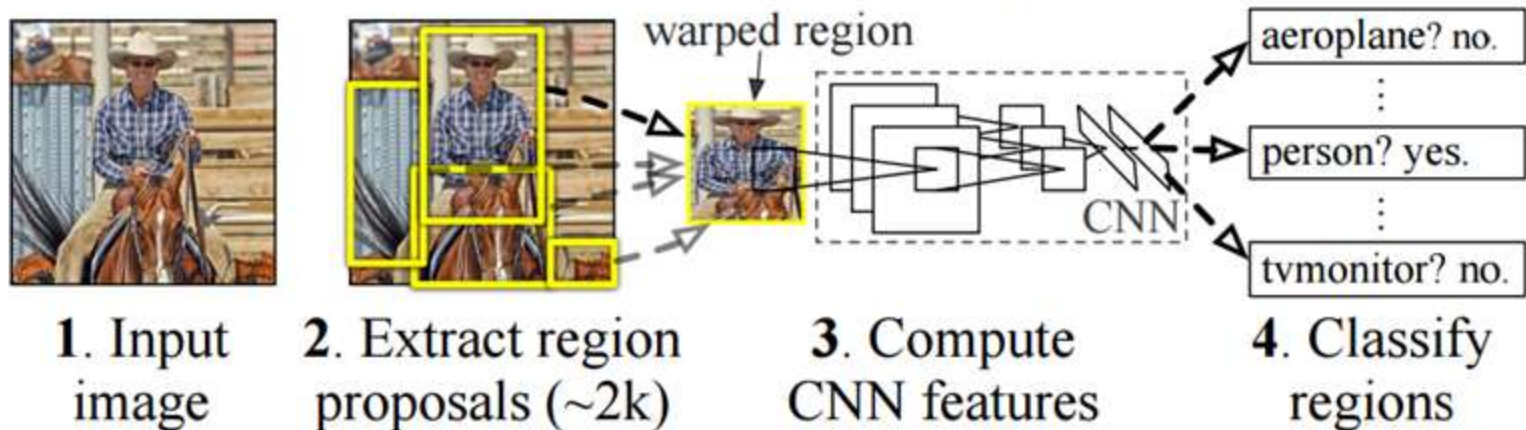
Similarity



- Lab uses a lightness channel and two color channels (a and b). It's calibrated to be *perceptually uniform*. Like HSV, it's also *somewhat invariant to changes in brightness and shadow*.

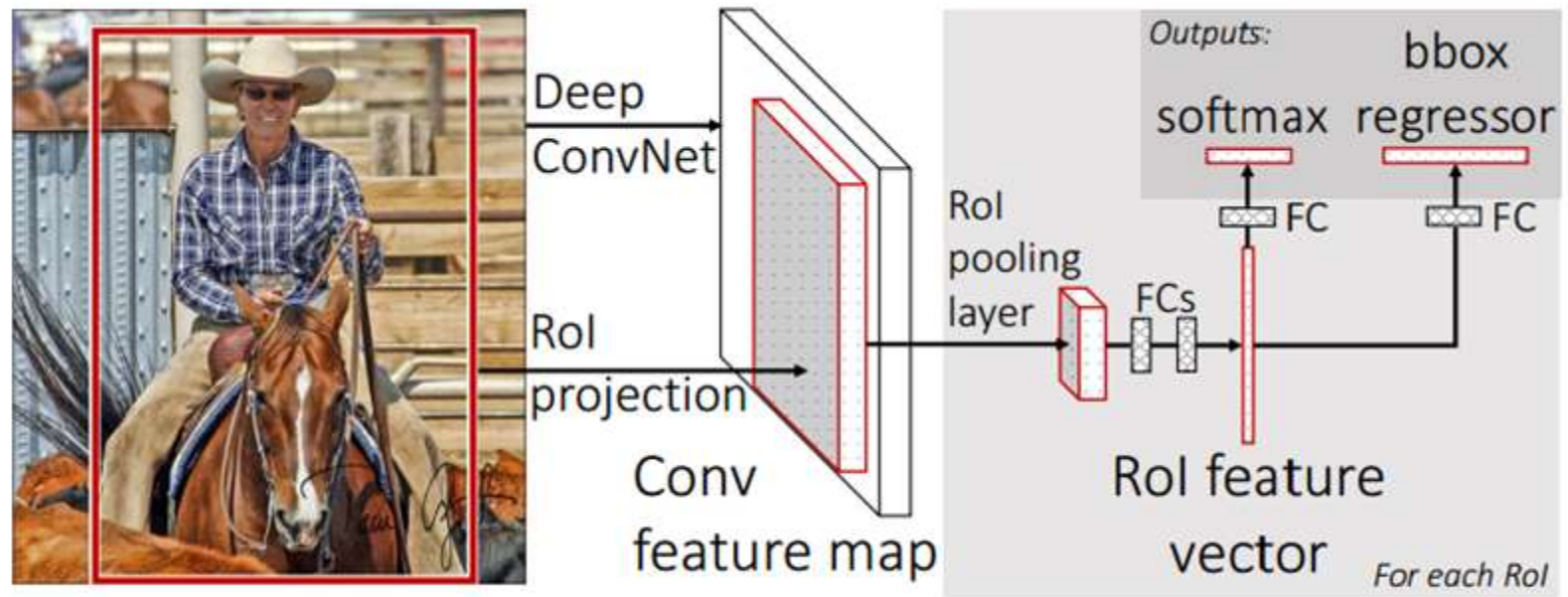
Region Based CNNs(R-CNN)

R-CNN: *Regions with CNN features*



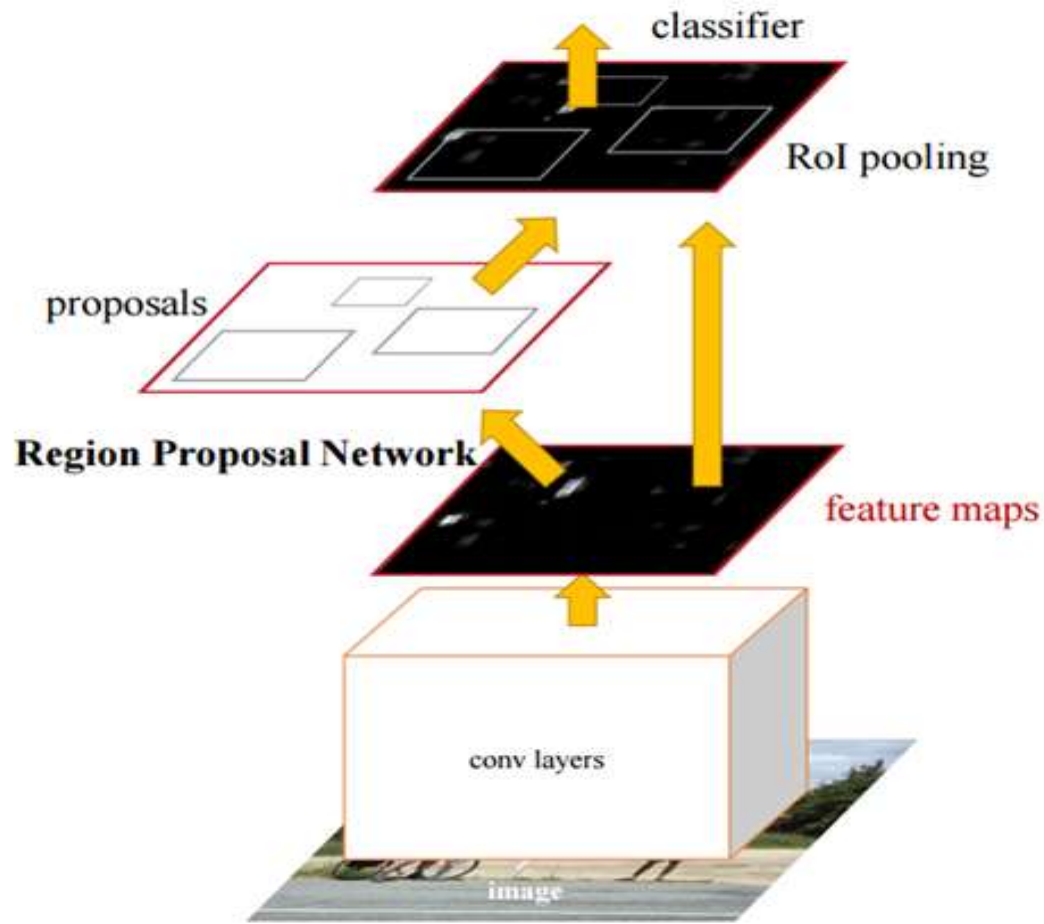
R-CNN workflow

Fast R-CNN



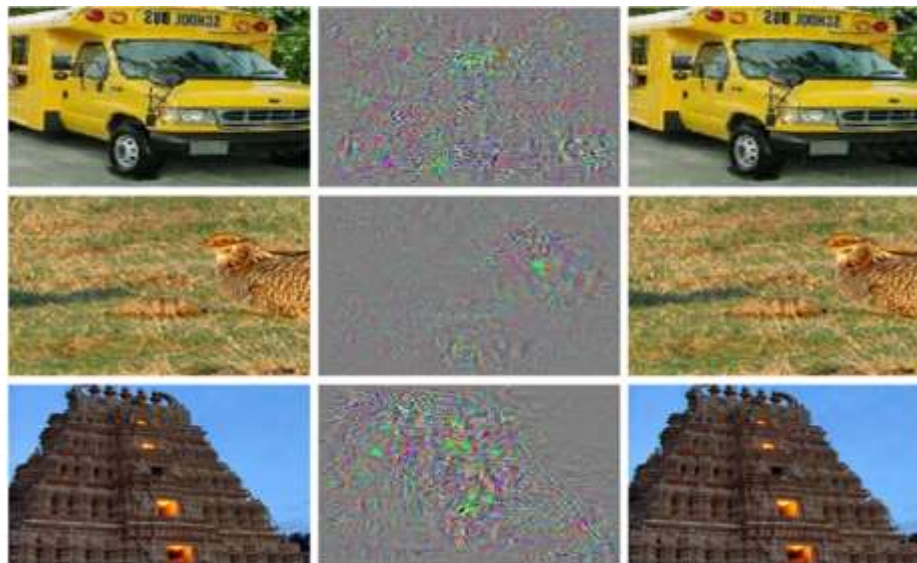
Fast R-CNN workflow

Faster R-CNN



faster R-CNN workflow

Generative Adversarial Networks

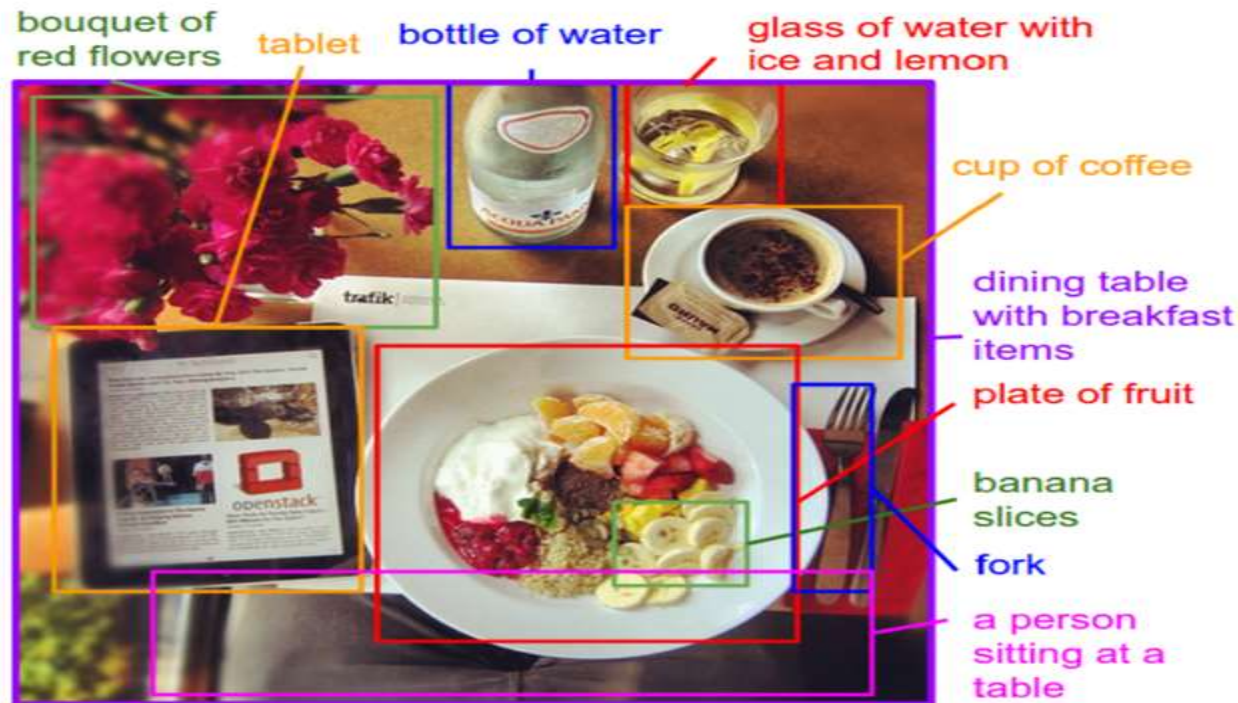


- According to Yann LeCun, these networks could be the next big development.
- The idea is to simultaneously train two neural nets.
- The first one, called the Discriminator $D(Y)$ takes an input (e.g. an image) and outputs a scalar that indicates whether the image Y looks “natural” or not.
- The second network is called the generator, denoted $G(Z)$, where Z is generally a vector randomly sampled in a simple distribution (e.g. Gaussian). The role of the generator is to produce images so as to train the $D(Y)$ function to take the right shape (low values for real images, higher values for everything else).

Generative Adversarial Networks : Importance

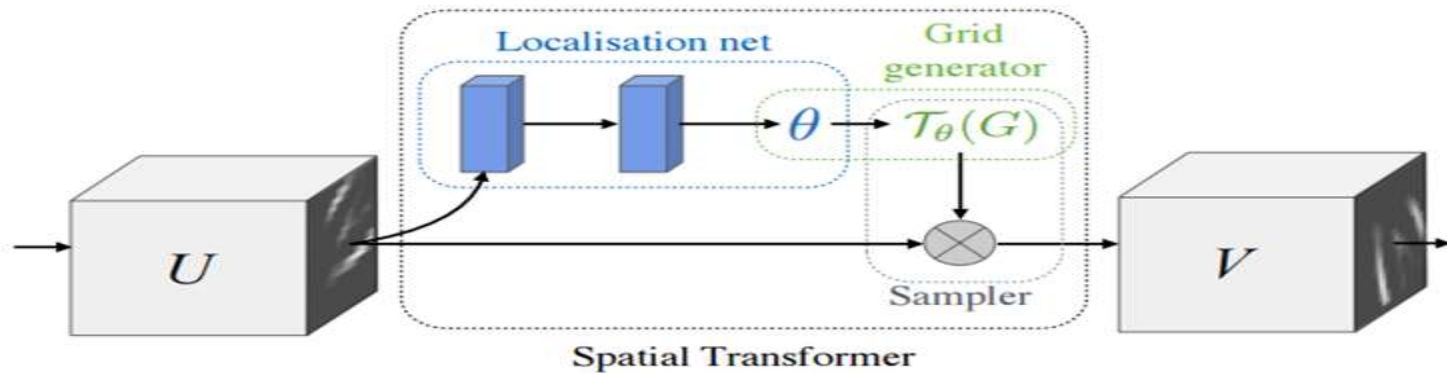
- The discriminator now is aware of the “internal representation of the data” because it has been trained to understand the differences between real images from the dataset and artificially created ones.
- It can be used as a feature extractor for CNN.
- You can just create really cool artificial images that look pretty natural to me.

Generating Image Descriptions



- 0 What happens when you combine CNNs with RNNs. you do get one really amazing application.
- 0 Combination of CNNs and bidirectional RNNs to generate natural language descriptions of different image regions

Spatial Transformer Network



A Spatial Tranformer module

- 0 The basic idea is that this module transforms the input image in a way so that the subsequent layers have an easier time making a classification.
- 0 The module consists of:
- 0 A localization network which takes in the input volume and outputs parameters of the spatial transformation that should be applied.
- 0 The creation of a sampling grid that is the result of warping the regular grid with the affine transformation (θ) created in the localization network.
- 0 A sampler whose purpose is to perform a warping of the input feature map.

Spatial Transformer Network

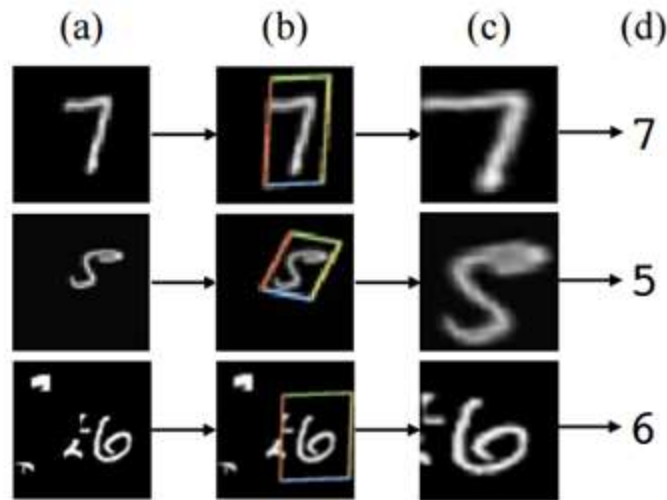


Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

- 0 This Network implements the simple idea of making affine transformations to the input image in order to help models become more invariant to translation, scale, and rotation.

Conclusion



Significant advances in **deep reinforcement and unsupervised learning**



Bigger and **more complex architectures** based on various interchangeable modules/techniques



Deeper models that can learn from much fewer training cases



Harder problems such as **video understanding and natural language processing** will be successfully tackled by deep learning algorithms

Conclusion



Machines that **learn to represent the world** from experience



Deep learning is **no magic** ! Just statistics in a black box , but exceptional effective at learning patterns.



We haven't figured out **creativity** and human-**empathy**.



Transitioning from research to consumer products .will make the tools you use every day **work better, faster and smarter**

Online Courses

- 0 <https://www.coursera.org/specializations/deep-learning>
- 0 https://www.youtube.com/watch?v=fTUwdXUFfI8&index=2&t=895s&list=PLAI6JViu7Xmd_UKcm-Mnxe1lvQI_qmW
- 0 <https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>