# Note:

- **Pecha Kucha** project presentation **next week**
  - Submit your MS-PPT slides two days prior (**May 11, 5pm**) by **email**
  - Read my email as well as the assignment handout. **8 slides with 50** seconds each. Will be automatically played! You will have NO CONTROL!!! **So MAKE SURE YOU PRACTICE!**

- Presentation in alphabetical order (See email)

- Project Report due in next week (**5/15 Thr, 10pm**).
  - Two additional days
  - Read the assignment thoroughly
  - Late policy will apply.





# Deep Neural Network

CSC 872

Pattern Analysis and Machine Intelligence

# Outline

- What is deep learning (DL)?
- Limitation of prior arts in AI/ML
  - Scalability
  - Complexity
  - Frame Problem
  - Symbol Grounding Problem
  - Feature Engineering Problem
- Advent of Learning Theory & Fast Hardware moved us to Probabilistic modeling with Big Data (DS/DE).
- **End-to-end Learning with Representation Learning of distributed and hierarchical representation structure** is the key to solve the issues!



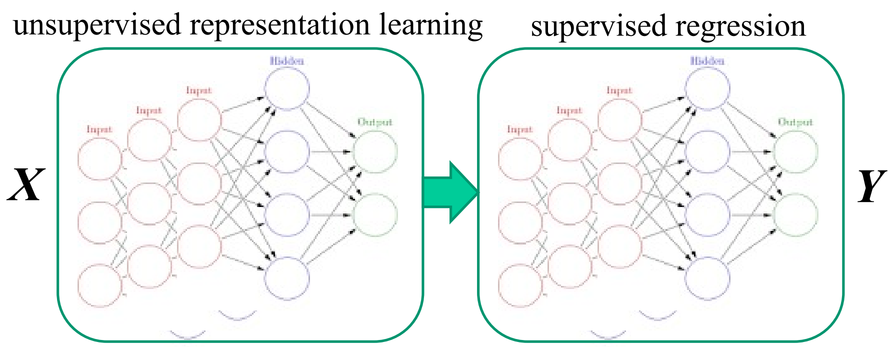


# What is Deep Learning?

- Machine Learning (ML) on Deep Neural Network (Deep NN) DNN
- Deep NN: NN with many stacked layers in various configurations
  IN H OUT
  - Add more layers to the multilayer perceptron
  - Complex Structure with many more unknown variables
  - More flexible to match with complex problem but
  - More difficult to train/learn
  - Require a lot of data to train/learn

# What is Deep Learning?

- Has seen a great recent success
  - Hierarchical Structure
  - Representation Learning
  - **Let's study first the foundation of WHY this works!**






# Recall Classic AI

- 1st AI Boom: 1960's
  - Tree-Search (Goal/Utility-based Agents)
  - Games: Puzzles and Chess
  - Can solve interesting problems if we can describe them in a tree-search formulation
  - **But did not work for large-scale problems!**
- Perceptron
  - First Artificial Neural Network in ML
  - Minsky's proof for the limit of ANN as a linear classifier
  - **Basically did not work for complex problems!**
  - Neuro Winer 1970~

# Classic AI: 2nd AI Boom (1980's)

Siri IBM Watson

- Let's incorporate explicit knowledge!
  - Rules with formal logic (Knowledge-based Agents)
  - Chat system (Eliza,1964) & Expert system (Mycin,1970)
    - Based on DB of standard templates & logical rules
    - Eventually evolved to Siri/ChatGPT and Watson, respectively
  - Work on small-scale problems but
- **Need to increase the size of knowledge to make it work for large-scale problems…**
  - But the amount of common sense/background required to describe a simple rule is **HUGE and COMPLEX**!
  - Basically **more data did not help to solve large-scale problems** ← time-knowledge trade-off



7

# And Neural Network?

- Let's introduce layered hierarchy !
  - **More complex structure to match complex problems?**
- Multilayer Feedforward Networks
  - Perceptron → MLP
  - Hidden layers introduced (Func() →Func(Func(Func)))
  - Backpropagation algorithm to solve a learning problem structurally more difficult than perceptron
  - Beat the Minsky's criticism (Linear → Non-linear)
  - More accurate system → More complex network structure
  - More complex structure → More difficult learning
  - **Adapting it to complex & large problem still failed…**
  - **Second Neuro Winter (~2007)**



8

## Summary: Situations before DL

- Things work with small toy problems but … controlled & focused
- **Failed when we apply them to large-scale and difficult/complex problems**
- More knowledge/data better?
  - Yes, but failed, could not handle too large a size of knowledge
- More complex structure better?
  - Yes, but failed, could not solve learning problem when making the system structure more complex
- AI Winter (1995~)





## How best to describe larger and more complex data then? KR - FE - PS

- Maybe we were formatting data wrong?
- Fundamental Question posed by the AI
- **Knowledge Representation**
  - = formal description of information
  - Semantic network (1960~)
  - Cyc (1984~) Knowledge Base of all common knowledge 39 year ↗ still going
  - Ontology (1990~)
- Needs
  - Scalability
  - Adaptability
  - Efficiency

## Any system design that works better with more data then?

- Yes, probabilistic system!
  - Led to the advent of **statistical ML** since 1990's
  - Such system can be learned by using statistics of data
  - More-Data = Better-System (Mathematical Proof)
  - Law of Large Numbers
    - With larger size data, sample mean converges to true (expected) probability
  - Central Limit Theorem
    - With larger size data, sample means follow the normal distribution regardless of data distribution
  - Some success with **Big Data** but still could not solve difficult problems. → **Why?**



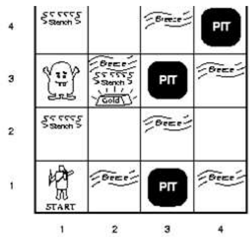


## Three Problems of Pre-DL ML/AI

- Fundamental Problems in Overall Design of Solution Architecture!
  - **Frame Problem**
  - **Symbol Grounding Problem**
  - **Feature Engineering Problem**
- Success of Deep NN can be attributed to offering an effective solution to these problems!

# Frame Problem: McCarthyHayes69’

- FOL ignores a lot of possibilities…




- ✓ New wampus?
- ✓ Moving pits?
- ✓ Escape routes?
- ✓ Explosion?
- ✓ Stock market crash?
- ✓ Flower blossom?

- Alpha-Go works because the system only need to think of how to play GO game not worry about raising rents and job prospects…



# Frame Problem in General

- **Dennett (1994)**
  - **A battery is on a tray in a cave. But there is also a bomb on the same tray. To save the battery, a robot is sent to bring it out of the cave.**
  - **R1D1: Exploded.** It understood the task of bring out the battery and noticed of the bomb but did not understand that bringing out the tray will also bring out bomb.
  - **R1D2: Stucked.** Redesign it to consider consequences of action. When reaching the battery, it stopped and never moved, thinking all possible consequences of moving the battery out; would it make the bomb explode? ceiling falls? change the color of walls? and so on, forever.
  - Frame
  - **R2D1: NeverMoved.** Redesign it to not consider situations that are irrelevant. Before reaching the battery, it stopped and never moved, thinking all possible situations to sort out what is relevant and irrelevant.

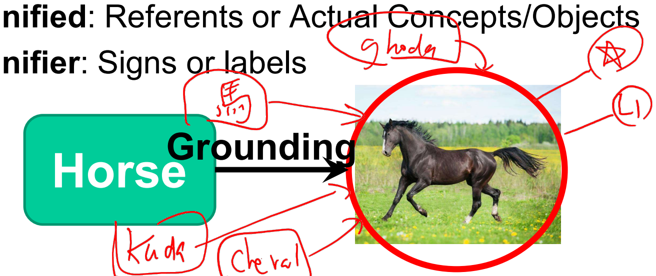
# Frame Problem in General

- Think Self-Driving: There are infinitely many things that must be considered to solve a task in general domains.
- Often ignore those that are not related to the given task and consider only related information (= frame).
- Selecting a correct frame among all possible things forces you to search among infinitely many choices so it can take forever…
- **It tells us that rule-based approach has a fundamental problem.**

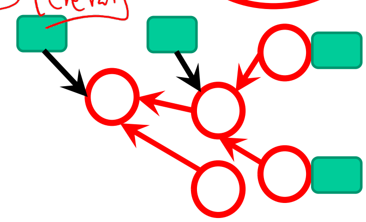


# Semiology-Semiotics: Saussure

- Linguistical relation of labels and concepts
  - **Signified**: Referents or Actual Concepts/Objects
  - **Signifier**: Signs or labels




Compositions of *Signifieds* create new *Signfied* that receives a ***Signifer***.



## Symbol Grounding Problem: Harnad '90

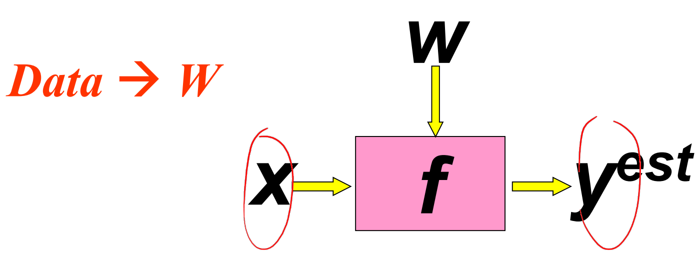
- Symbolic manipulation of ungrounded / meaningless signs cannot handle unknowns


- But we can imagine this is possible as long as we know horse and stripe…
- Can AI too?



## PR/ML: Learning Machine

- PR and ML focused on mathematical techniques to derive best ***W*** from given data

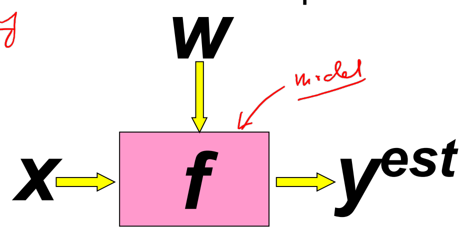



- But **you have to chose** what **factors** you extract from raw data to be used as inputs ***X*** & outputs ***Y*** (and the choice of ***f*** form as well).

## Feature Design

- To build an end-to-end ML/PR system, **engineers must find a set of factors, appropriate to a given problem**, to be extracted as inputs

*Data* $\xrightarrow{\text{learning}}$ *W*




- Features = Problem-specific factors to be used as ***X*** extracted from raw data. (I), (I, x, y), (r, g, b), (r, g, b, x, y)
- Feature design = act of finding appropriate features



## Feature Engineering Problem

- <u>Feature design turns out to be an art!!!</u>
- Found that **the overall performance changes a lot when using different features (sensitive)**
- This sensitivity are often larger with feature design than the choice of ML algorithm used
- Expert developers thus focused on hand-crafting best features to improve performances in ML/PR system research and development **(→ this is Feature Engineering!)**
- But this was difficult without any guiding theories and slowing the progress of Big Data ML
  Probabilistic

## Common Culprit

- **Human designer had to decide which aspects in the real world would be used as the factors in formulating problems to be solved.**
- **System failed when designer's choice was wrong!**

- Frame Problem
  - Designer manually defines problem-specific assumptions/frames
  - **Wrong inference → Bomb explode!**
- Symbol Grounding Problem
  - Manually designed labels used w/o signified model from data
  - **Cannot adapt to new concepts → Do not generalize/scale!**
- Feature Engineering Problem
  - Manually designed problem-specific features used
  - **Could not handle complex problem! ← Limit of BigData ML**

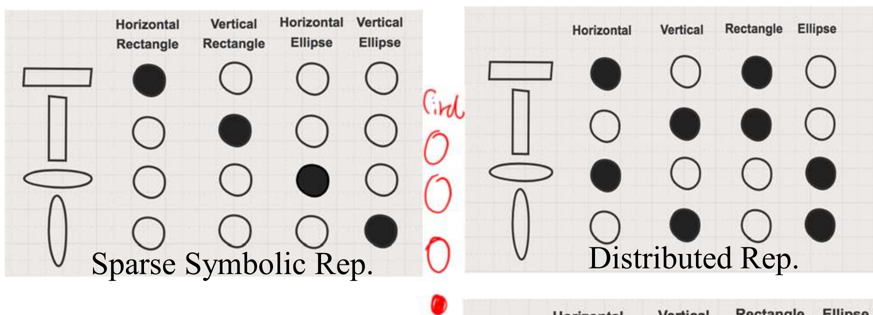


## Let data decide what factors to be used rather than human designer's choice!

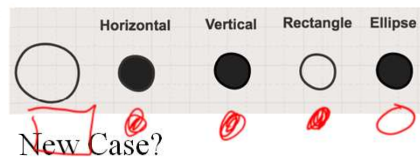
- **ML's Bottleneck = Feature Design**
- **Representation Learning**
- Data-Driven Feature Design (vs Human Intelligence)
- **Automation of Feature Design**
- Solves
  - Feature Engineering Problem (automate it)
  - Symbol Grounding Problem (add a process to extract signified from data)!

# Distributed Representation

- Generalization/Scalability in symbol grounding problem can influence basic rep schemes




- Multilayer NN architecture thus helps to improve those above






https://www.oreilly.com/ideas/how-neural-networks-learn-distributed-representations



# Frame Problem?

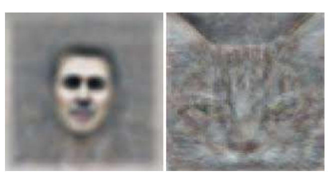
- Can be solved by using a flexible general-purpose learning machine that can be learned from data and from scratch!
- Do this without assumptions or learn the assumptions themselves from data
- **End-to-End learning** of Deep NN ← Became possible by the feature learning!

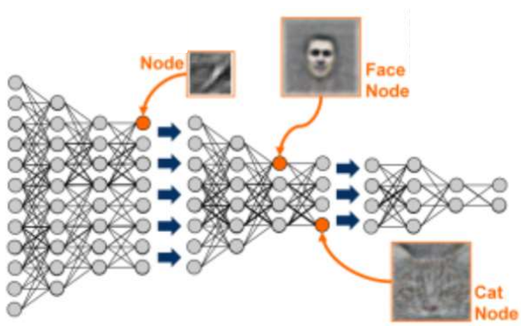
## How can we assure flexibility of the Learning Machine though?

- Hierarchical Structure by stacking many hidden layers between inputs and outputs!
- Func() → Func(Func(Func(Func(……))))))))
- Exponentially increase # learnable patterns
  = Scalable & Can tackle more complex prob.
- But more difficult learning! → Next Part
- Also inspired by how human brain processes visual information in our brain.
- Can DNN mimic how babies learn to see?

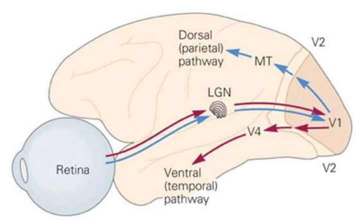


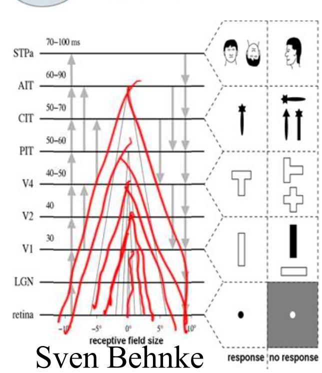
## Biologically-Inspired Hierarchical Representation of Visual Patterns






Google's Cat (Google X, 2012)




Sven Behnke

## Summary

- Deep Learning realizes
  - End-to-end learning
  - Representation learning
- Deep NN architecture realizes
  - Distributed Representation
  - Hierarchical Representation
- These design addresses
  - Scalability/Problem-Complexity
  - Frame Problem
  - Symbol Grounding Problem
  - Feature Engineering Problem

reason for success

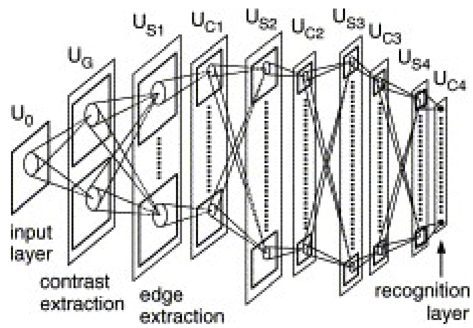


## Outline

- Deep Neural Network Architecture: Auto Encoder
- Deep Neural Network Architecture: ConvNets CNN
- What made it work: Various learning techniques to avoid overfitting
- ImageNet
- CNN variants
- Software Libraries
- Recurrent Neural Networks
- Generative Adversarial Network
- Ethics/Future

# Deep NN is not a new idea…

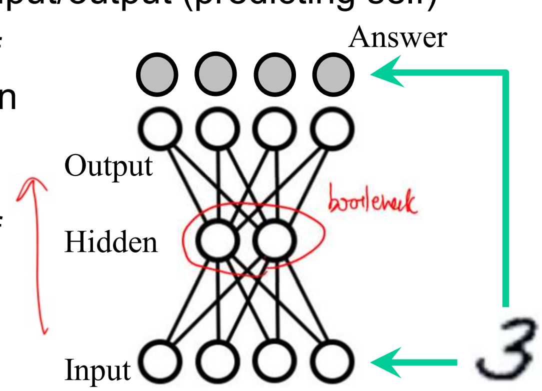
- Neocognitron (Fukushima, 1979)
  - End-to-end representation learning with distributed hierarchical structure
- But could not train it from data then…






# Auto-Encoder: First Step

- Hinton 2006
- NN with the same input/output (predicting self)
- With less number of hidden units, NN can **learn/extract compact and essential aspect** of the patterns
- Unsupervised Learning

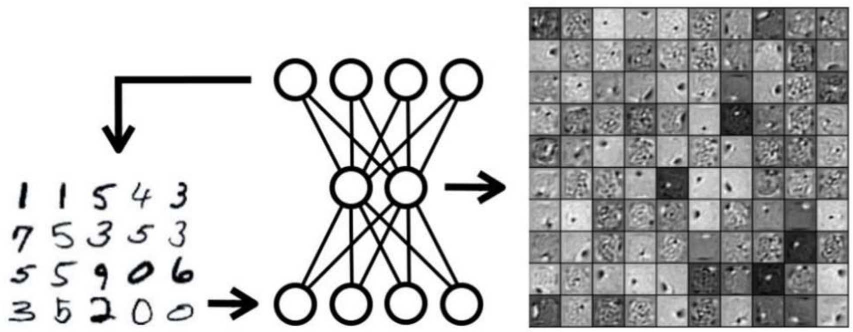



Matsuo, 2015 https://www.ipa.go.jp/files/000048577.pdf

## Representation Learning: Low Level

- Backprop w/ Gradient Descent…

V1




low-level visual features
learned from data

CSC872: PAMI – Kazunori Okada (C) 2025

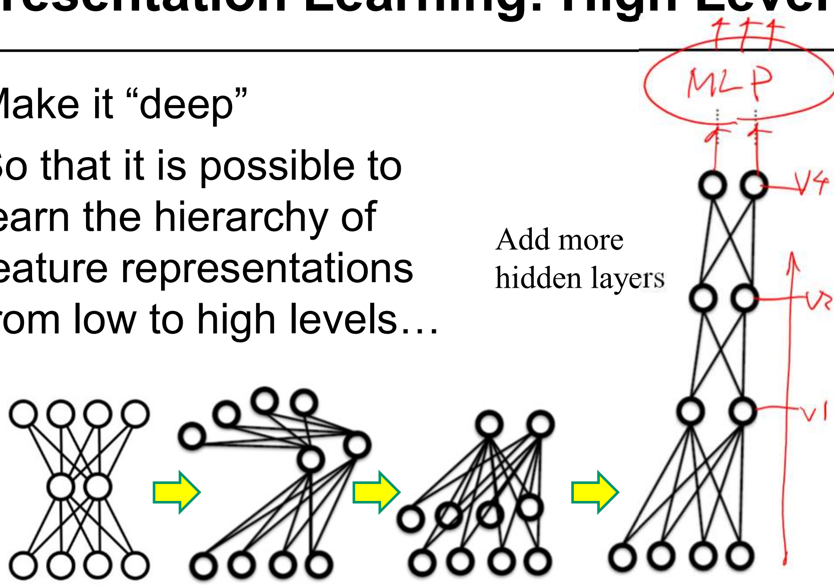
Matsuo, 2015 https://www.ipa.go.jp/files/000048577.pdf 31

31

## Representation Learning: High Level

- Make it “deep”
- So that it is possible to learn the hierarchy of feature representations from low to high levels…

Add more
hidden layers



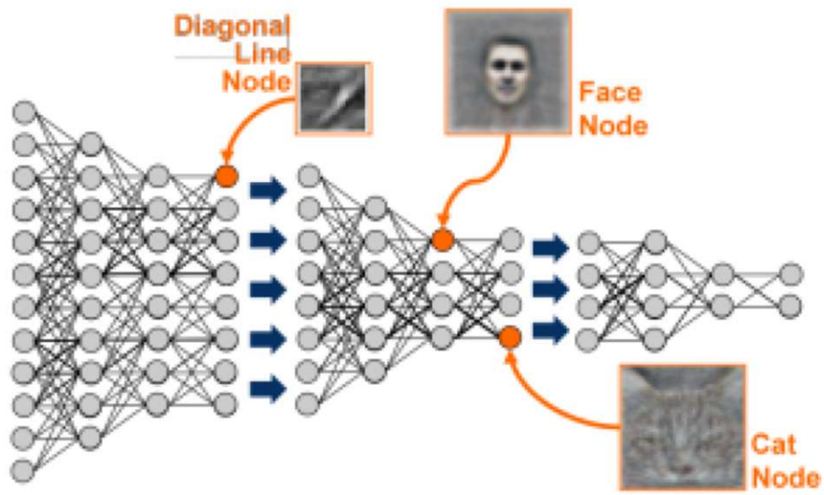

CSC872: PAMI – Kazunori Okada (C) 2025

Matsuo, 2015 https://www.ipa.go.jp/files/000048577.pdf 32

32

# Google's Cat (2012)

AlexNet






# Pre-train & Fine-tuning

- Final classifier: stacking the deep auto encoder and a fully-connected multilayer feedforward network MLP
- **Pre-training**: unsupervised learning of auto encoder
- **Fine-tuning**: supervised learning of fully-connected multilayer network by backpropagation
- Alternative to stacked auto encoder
  - Stacked restricted Boltzmann machine
  - Stacked denoise autoencoders
  - Stacked kernel PCA/semi-supervised embedding/ISA etc

Transfer Learning

# Convolutional Neural Net (CNN)

- Le Cun
- LeNet (1989) consists of
  - Convolution Layer
  - Transfer function (ReLU) Layer
  - Pooling Layer
  - Fully connected Layer

Hierarchy

Receptive field

Feature Maps

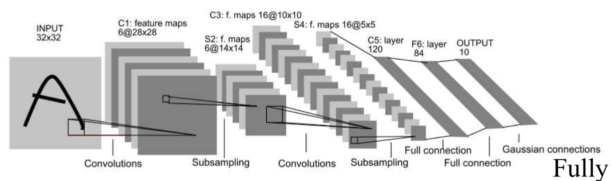
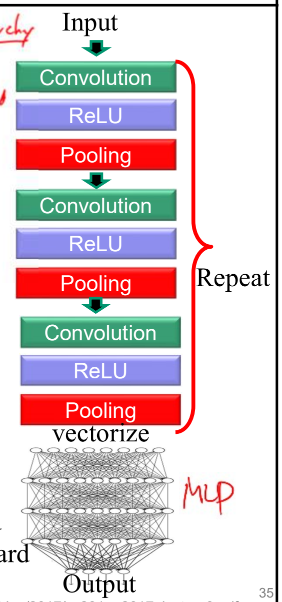



Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.




Fully connected Feedforward Net

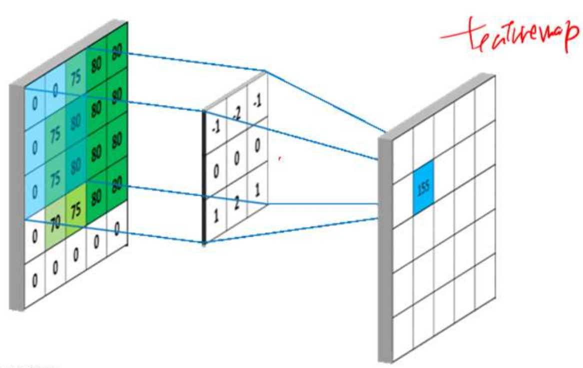


Source of the slide: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

35

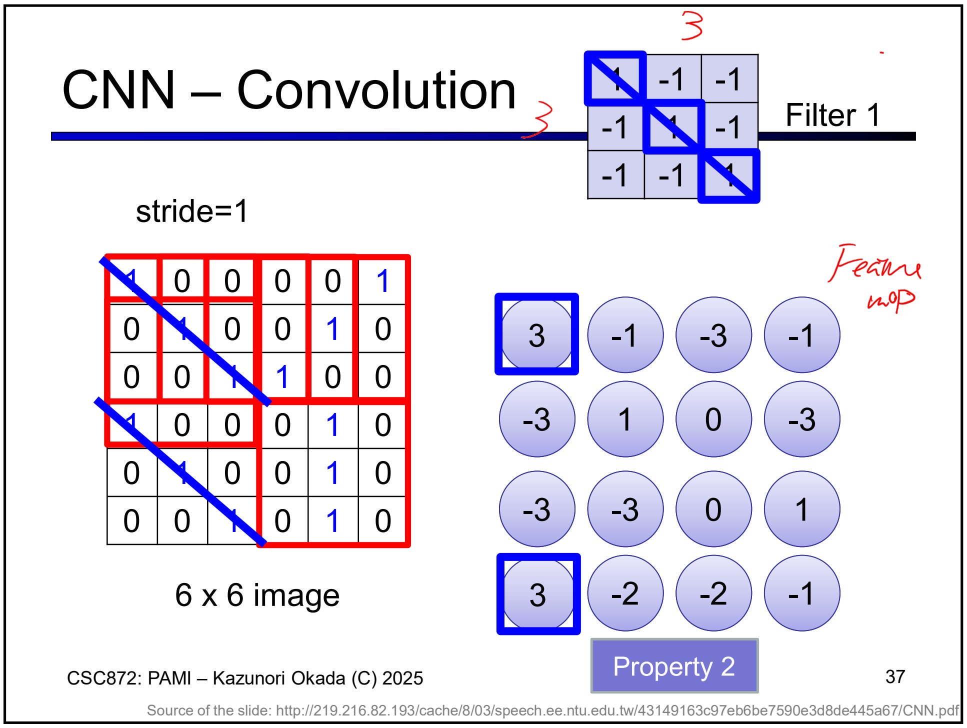
# Convolution

- Local filter is scanned over images to compute a response by computing the sum of pairwise products
- Stride: skipping the interval of scans
- Results goes through transfer function (Sigmoid/ReLU)

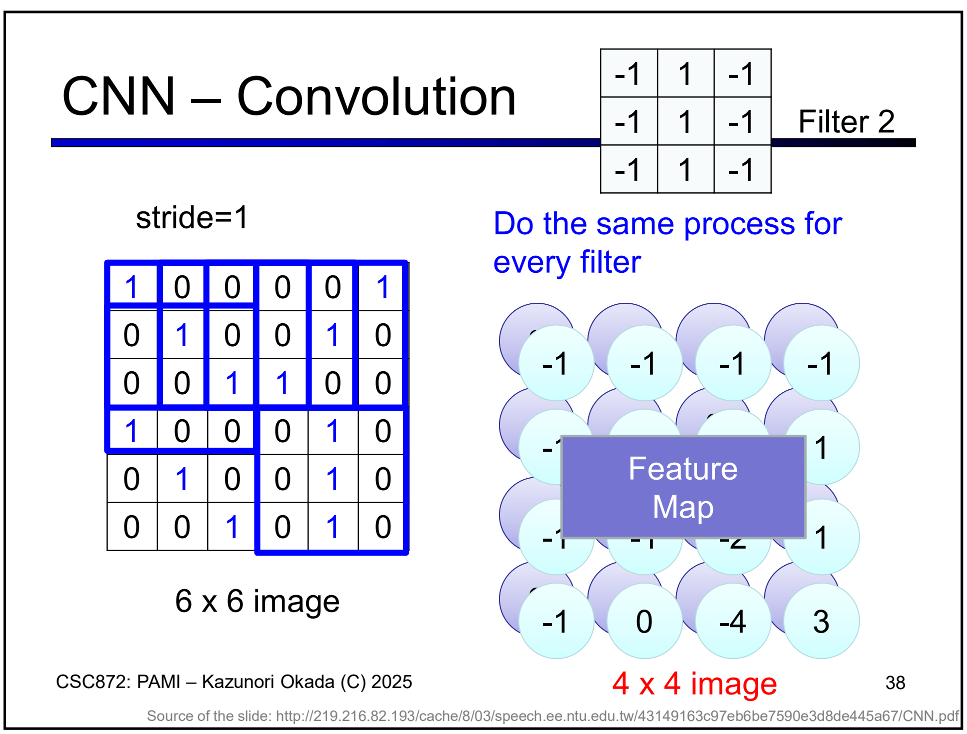





36


CNN – Convolution
3
3
1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
stride=1
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Feature map
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
Property 2
CSC872: PAMI – Kazunori Okada (C) 2025
37
Source of the slide: http://219.216.82.193/cache/8/03/speech.ee.ntu.edu.tw/43149163c97eb6be7590e3d8de445a67/CNN.pdf

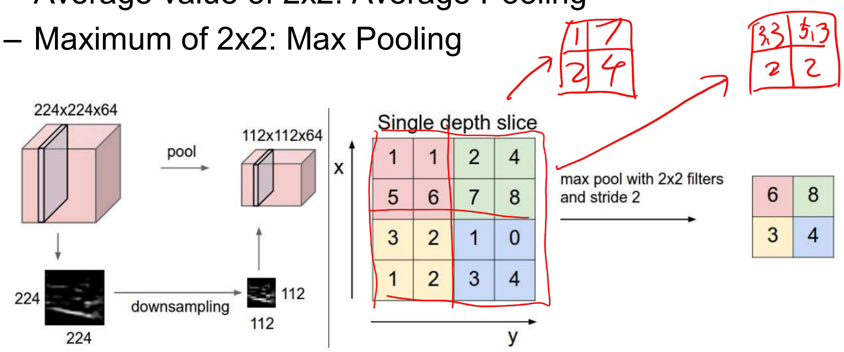
37


CNN – Convolution
-1 1 -1
-1 1 -1
-1 1 -1
Filter 2
stride=1
Do the same process for every filter
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
-1 -1 -1 -1
Feature Map
-1 0 -4 3
4 x 4 image
CSC872: PAMI – Kazunori Okada (C) 2025
38
Source of the slide: http://219.216.82.193/cache/8/03/speech.ee.ntu.edu.tw/43149163c97eb6be7590e3d8de445a67/CNN.pdf

38

## Max Pooling and Sub-Sampling

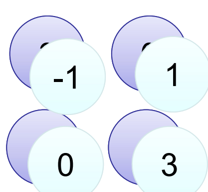
- Reducing resolution by replacing a patch with
  - Random sample among 2x2: Sub Sampling
  - Average value of 2x2: Average Pooling
  - Maximum of 2x2: Max Pooling



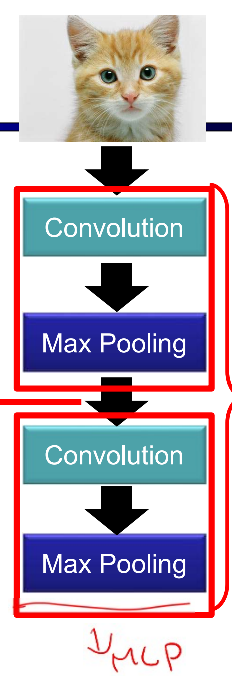

Figures from: Jianping Fan, UNC Charlotte



## Stacking Layers




A new image




Can repeat many times

Smaller than the original image

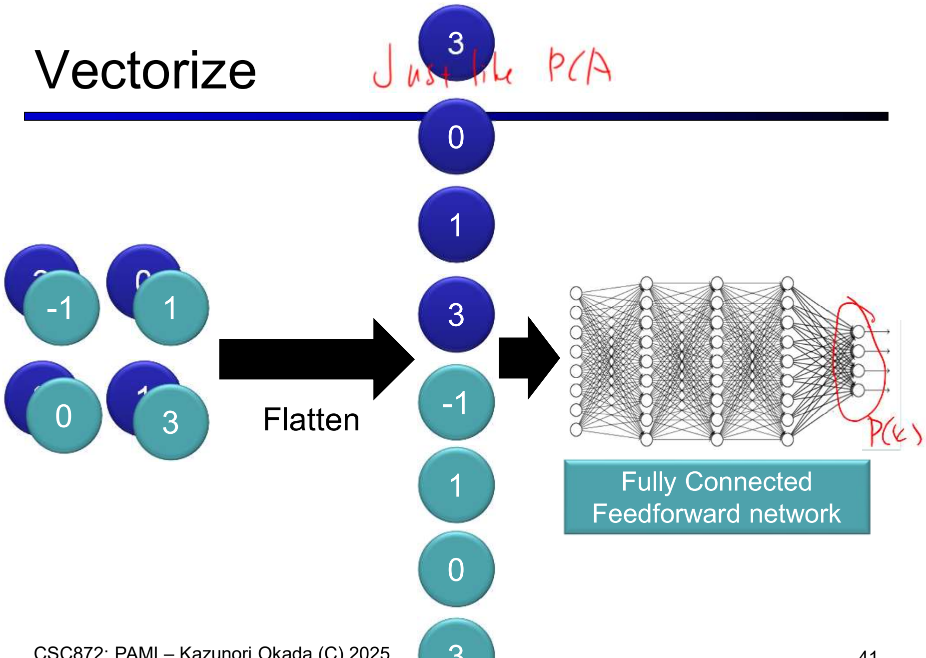
The number of the channels is the number of filters



Source of the slide: http://219.216.82.193/cache/8/03/speech.ee.ntu.edu.tw/43149163c97eb6be7590e3d8de445a67/CNN.pdf

# Vectorize

Just like PCA






Source of the slide: http://219.216.82.193/cache/o/cs/speech.ee.ntu.edu.tw/43149163c97eb6be7590e3d8de445a67/CNN.pdf

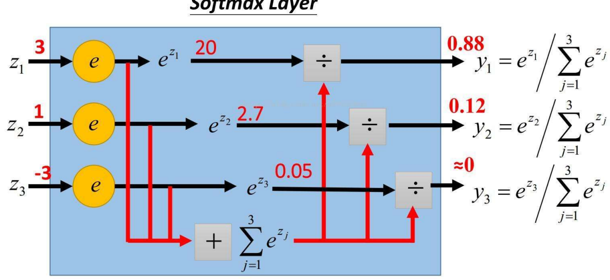


# Softmax Unit

- Used for output unit to convert values into a probability distribution

  - Softmax layer as the output layer

***Probability***:
- $1 > y_i > 0$
- $\sum_i y_i = 1$




$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad j = 1, 2, \ldots, K.$$

Figures from: Ruiyun Yu, Northeastern University

## Making the Learning of CNN work?

- Learning is essentially done by the **backpropagation** algorithm from last lecture, but it got more difficult when making NN architecture deeper and more complicated
- **More variables = Tend to overfit**
- Break Through: find strategies to sabotage/regularize the learning process, adding noises and breaking connections lead to **robustness to avoid overfit**
- Longer time still required to learn with more variables
- **Advent of GPU**: improving throughput and making it possible to do massive learning required to solve large-scale/complex prob.

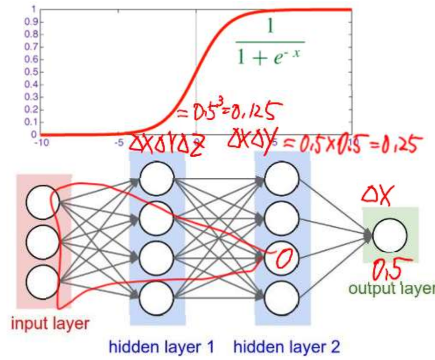


## Stochastic Gradient Descent

- Recall batch vs online perceptron **delta rule** derived by the gradient descent algorithm
- Stochastic Gradient Descent
  - **Online Gradient Descent**
  - Approximation of the batch version
  - Converges better because of high frequency of weight updates
  - Minibatch GD: randomly sample a subset of small batch from data then sequentially perform gradient descent

# Vanishing Gradient Problem

- Backpropagation propagates gradient of errors from output layers to input layers using the chain rule (product of probs)
- Sigmoid as transfer function: gradient [0,1]
- So as we get closer to the input layer, propagated error become a product of many less-then-1 values → exponentially get smaller!
- Stop changing weights = Stop learning (Delta = 0)
- Shadowing
- **Deeper the net, more chance of having gradient vanished!**
- **Sigmoid won't work deep!**

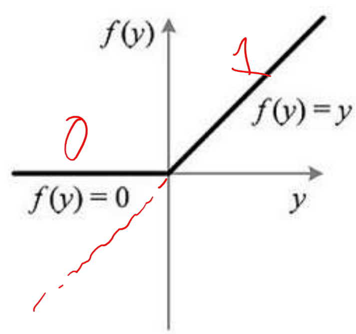



Figures from: Jianping Fan, UNC Charlotte



# Rectified Linear Units (ReLU)

- max(0,y)
- Gradient is the constant (1) for y > 0 so it would not vanish by going through many layers back
- More efficient (no exp comp)
- Sparsity (y<0) → Regularization = Avoid overfitting



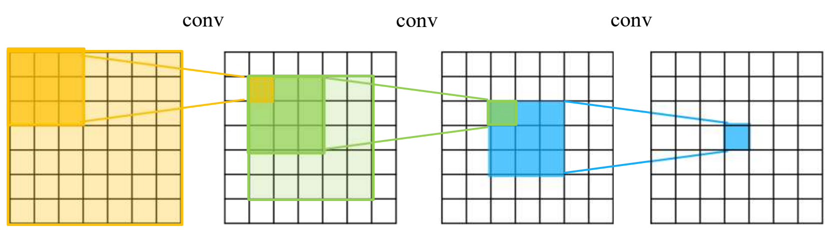

$$ReLU(x) = \begin{cases} x & if\, x > 0 \\ 0 & if\, x \leq 0 \end{cases}$$



Figures from: Ruiyun Yu, Northeastern University

## Why Max Pooling?

- Pooling creates overlapping receptive fields
- Max Pooling is most commonly used
  - Shift Invariants: slight misplacement accommodated
  - Additional non-linearity: more expressive representation
  - Efficiency: computes faster than average pooling






Figures from: Jianping Fan, UNC Charlotte





## Dropouts

- Randomly ignore neurons in hidden layers for updating during learning
- This acts as practically-efficient regularizer for deep learning
  - Adding noise to learning process for robustness
  - Sparse activation of units during the learning
  - Thus avoid overfitting
  - Also efficient to do

## Backpropagation for CNN's End-to-End Learning

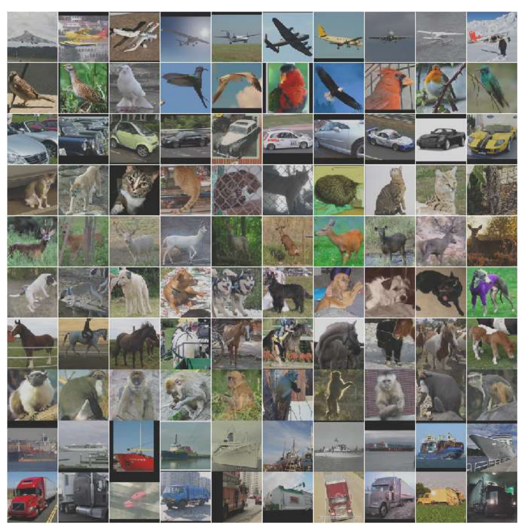
- Use the least sum of squares or other cost func.
- Solve it by stochastic gradient descent
  - MaxPooling layers do not involve learnable weights
  - Conv layers' weights updated by a convolution like procedure based on backpropagation through conv process
- **Transfer Learning:** reuse convolutional feature maps trained with a large dataset (pretrain/freezing: may take a long time!) and fine-tune the fully-connected network part by backpropagation (quick)





## ImageNet Challenge 2012

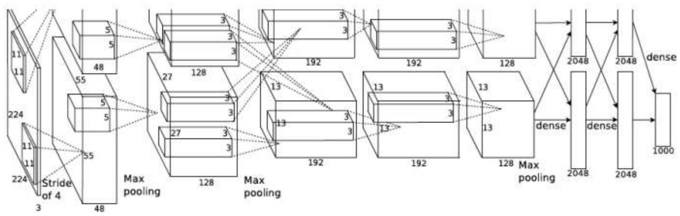
- Scene analysis: most difficult computer vision task
- ~14 million labeled images, 20k classes
- Image gathered from internet
- Human labels via Amazon Turk
- Challenge: 1.2 million training images with 100 classes



# Alex Net

## ImageNet Challenge 2012

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data ($10^6$ vs. $10^3$ images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Better regularization for training (DropOut)




A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

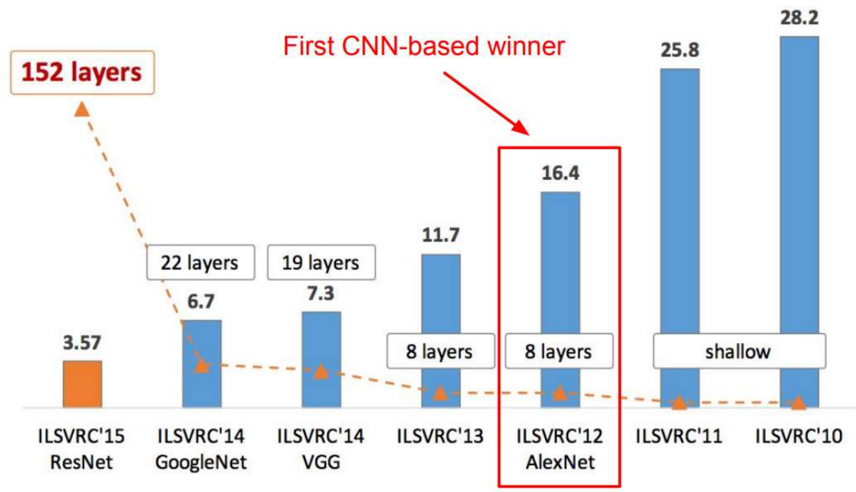


# CNN Variants: Getting deeper…

- VGGNet
  - Symonyan&Zisserman 2015, ILSVRC-2014 Runnerup
  - Smaller 3x3 Conv filters
  - Deeper network: 16~19 layers (AlexNet was 8 layers)
- GoogleNet
  - Szegedy 2015, ILSVRC-2014 Winner
  - Deeper: 22 layers
  - Focused on computational efficiency
- ResNet
  - He 2015, ILSVRC-2015 Winner
  - Extremely deep: 152 layers
  - Skip connections → Residual mapping

Feature Engineering
↓
Architecture Engineering!

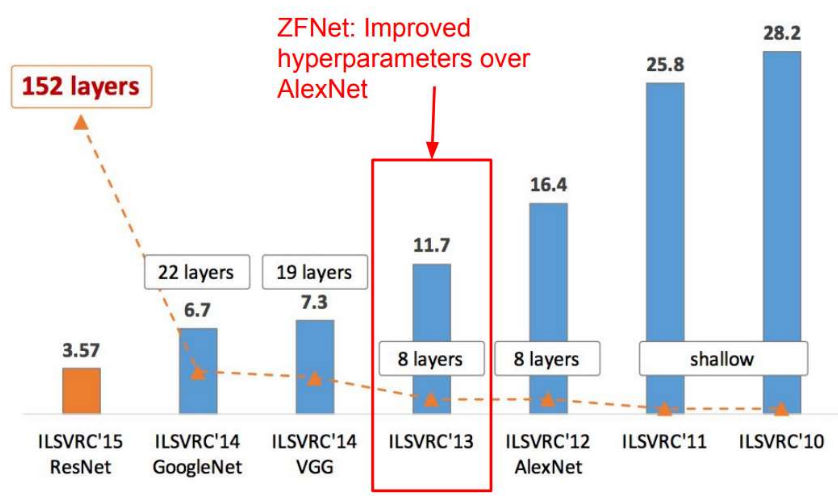
# ImageNet Large Scale Visual Recognition Challenge(ILSVRC) winners




CSC872: PAMI – Kazunori Okada (C) 2025 Source of the slide: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

53

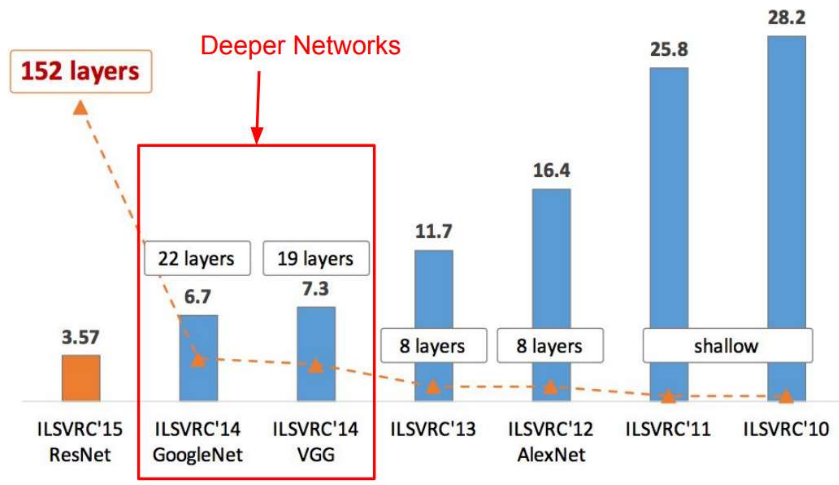
# ImageNet Large Scale Visual Recognition Challenge(ILSVRC) winners




CSC872: PAMI – Kazunori Okada (C) 2025 Source of the slide: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

54

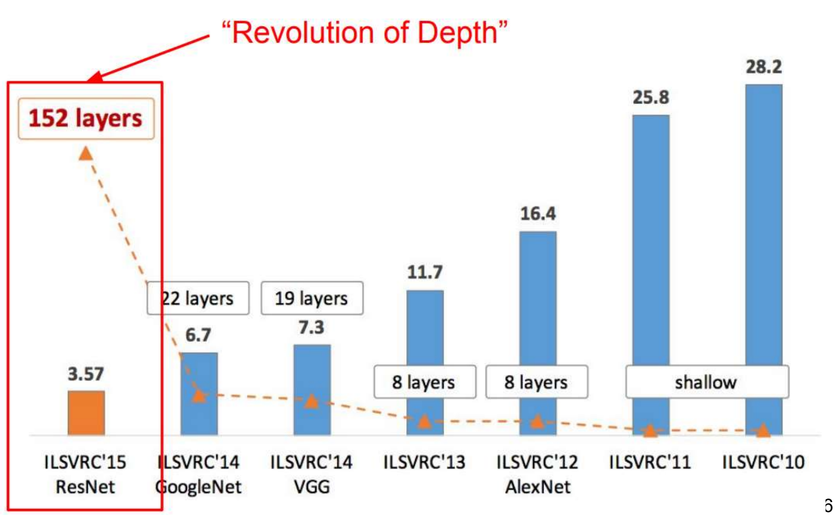
# ImageNet Large Scale Visual Recognition Challenge(ILSVRC) winners




CSC872: PAMI – Kazunori Okada (C) 2025 Source of the slide: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

55

# ImageNet Large Scale Visual Recognition Challenge(ILSVRC) winners




CSC872: PAMI – Kazunori Okada (C) 2025 Source of the slide: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

56

## Better than human?

- Captcha?

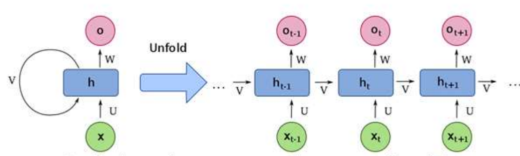
| | Top 5 error |
|---|---|
| Imagenet 2011 winner (not CNN) | 25.7% |
| Imagenet 2012 winner | 16.4% (Krizhesvky et al.) |
| Imagenet 2013 winner | 11.7% (Zeiler/Clarifai) |
| Imagenet 2014 winner | 6.7% (GoogLeNet) |
| Baidu Arxiv paper:2015/1/3 | 6.0% |
| Human: Andrej Karpathy | 5.1% |
| MS Research Arxiv paper: 2015/2/6 | 4.9% |
| Google Arxiv paper: 2015/3/2 | 4.8% |


Matsuo, 2015 https://www.ipa.go.jp/files/000048577.pdf




## Recurrent Neural Network: RNN

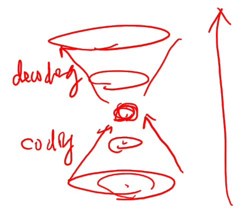
- Modeling and prediction of sequential data
  - Speech
  - Text
  - Video




- Recurrence unfolded over time → Deep Structure
- Suffers from Vanishing&Exploding Gradient
  - Remedy1: Long Short-Term Memory (LSTM)
  - Remedy2: Gated Recurrent Unit (GRU)
  - Replace standard units by them

# Encoder-Decoder Architecture

- Encoder: Texts → Thoughts
- Decoder: Thoughts → Texts
- Machine Translation:
  - Train an encoder with English
  - Train a decoder with Spanish
  - Concatenate them → English to Spanish Translation
- Video Captioning:
  - Train an encoder with Video (Video → Concepts)
  - Train a decoder with English (Concepts → Texts)
  - Concatenate them → Video to Text Captions!








# Generative Adversarial Network: GAN

- Goodfellow 2014
- Generative model for unsupervised learning
- Two neural networks one generative and the other discriminative compete each other game theoretically.
  - Generative network tries to fool the discriminative network.
  - Discriminative network try to distinguish the real ones from the fake ones

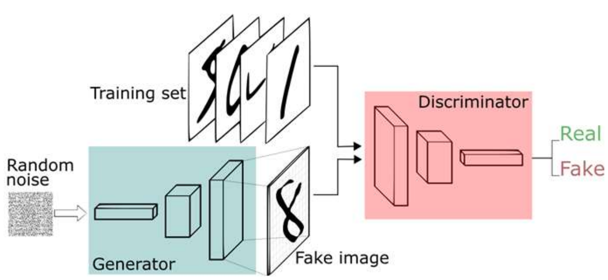



Image by *Thalles Silva*

## Image generated by GANs

Sample shoes images
from Zappos.com
[Yu and Grauman 2014]

Random image samples
from Generator G(z)
DCGAN [Radford et al. 2015]



## Software Library

- Tensor Flow (from Google Brain)
- Keras (high level Python API)
- Caffe (from Berkeley-AI)
- Microsoft Cognitive Toolkit (graphical)
- PyTorch (python, GPU)
- Theano, DeepLearning4j, Appache Mxnet, Caffe2, Torch, Chainer, Dlib, Paddlepaddle,
- Matalb: e.g. https://www.mathworks.com/matlabcentral/fileexchange/59223-convolution-neural-network-simple-code-simple-to-use

# Ethics/Future

- Deep Fake/Google's Lip Reading 96% success
- We have to continue figuring out how best to integrate it in our society
- We can solve problems but don't know why
  - → Going back to pre-enlightenment before 16thC
  - → People knew they don't die if they grill meat but did not know why then Pasteur discovered the reason
  - → Changing the basic mode of R&D to "solve it first then figure out why" from "develop a theory to solve it"
  - e.g., Alpha GO, ChatGPT
- IoT and 5G would increase available data for DL exponentially → Chicken race continues
- Induction vs Deduction (Distributed vs Symbolic)





# Summary

- Deep Neural Network
  - Frame Problem
  - Symbol Grounding Problem
  - Feature Engineering Problem
  - End-To-End Representation Learning
  - Distributed Representation
  - Auto Encoder
  - Convolutional Neural Nets
  - CNN Variants
  - Recurrent Neural Nets
  - Generative Adversarial Nets
- Next: Final Project Presentations
  - 5/11 5pm Due for **8 slides ONLY**
  - Each has **6-minutes-plus-40-seconds presentation**, Take a note and ask questions on *Canvas* after!