Chapter 6. Deep Learning for PHM

Prognostics and Health Management (PHM)

Byeng D. Youn

System Health & Risk Management Laboratory
Department of Mechanical & Aerospace Engineering
Seoul National University

CONTENTS

1 Introduction to deep learning

2 Unsupervised learning

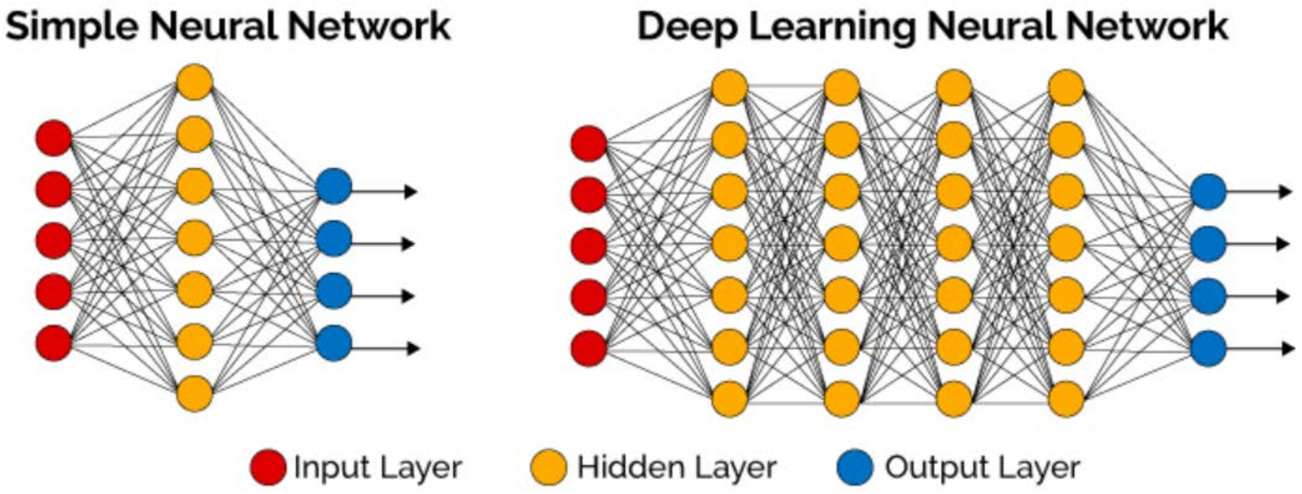
3 Supervised learning

4 Reinforcement learning

5 Case Study

What is Deep Learning?

- Learning higher level abstractions/representations from data.
- Motivation: how the brain represents and processes sensory information in a hierarchical manner.



Remind for Neural Networks

Deep Learning is based on neural networks

- Weighted sum followed by nonlinear activation function.
- Weights adjusted using gradient descent (η = learning rate) $w_{ij} \leftarrow w_{ij} + \eta \frac{\partial E}{\partial w_{ij}}$
- Minimize cost function : $E = \frac{1}{2}\sum_k (a_k - t_k)^2 \;\; or = \sum_k (t_k \log(a_k) + (1 - t_k)\log(1 - a_k))$

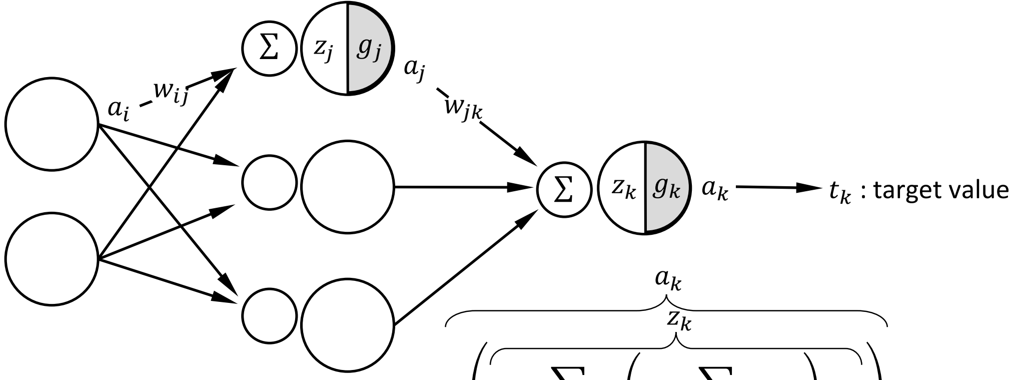
Error Back Propagation

- “propagate” backwards calculating all the error signal

1. Gradients for output layer weights

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\left(\frac{1}{2}\sum_k (a_k - t_k)^2\right)$$
$$= (a_k - t_k)\frac{\partial}{\partial w_{jk}}(a_k - t_k)$$
$$= (a_k - t_k)\frac{\partial}{\partial w_{jk}} a_k$$
$$= (a_k - t_k)\frac{\partial}{\partial w_{jk}} g_k(z_k)$$
$$= (a_k - t_k) g'_k(z_k)\frac{\partial}{\partial w_{jk}} z_k$$
$$= (a_k - t_k) g'_k(z_k) a_j$$
$$= \delta_k a_j$$

a_i : activity at input side of weight w_{ij}



$$a_k = \overbrace{g_k\left(\overbrace{b_k + \sum_j \underbrace{g_j\left(\underbrace{b_j + \sum_i a_i w_{ij}}_{z_j}\right)}_{a_j} w_{jk}}^{z_k}\right)}^{a_k}$$

$$\boxed{\begin{array}{l}\delta_k = (a_k - t_k) g'_k(z_k) \\ \text{error signal of neuron k in output layer}\end{array}}$$

g : activation fucntion
(sigmoid, tanh, Relu, etc)

Remind for Neural Networks

Deep Learning is based on neural networks

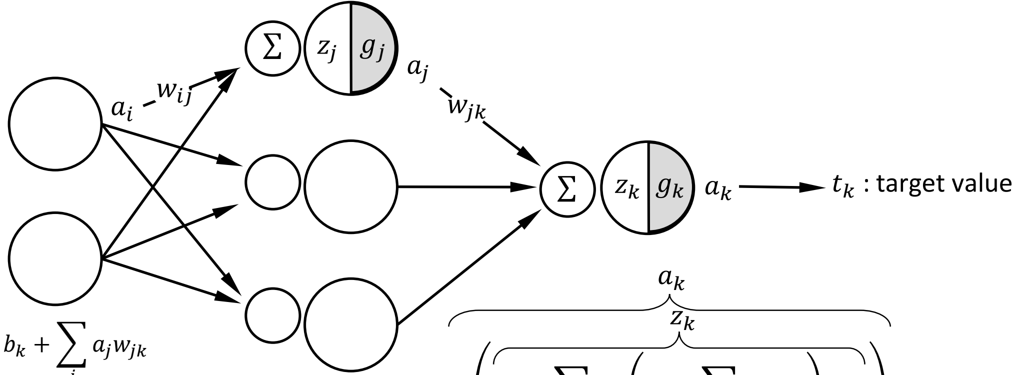
- Weighted sum followed by nonlinear activation function.
- Weights adjusted using gradient descent (η = learning rate) $w_{ij} \leftarrow w_{ij} + \eta \frac{\partial E}{\partial w_{ij}}$
- Minimize cost function : $E = \frac{1}{2}\sum_k (a_k - t_k)^2 \;\; or = \sum_k (t_k \log(a_k) + (1 - t_k)\log(1 - a_k))$

Error Back Propagation

- “propagate” backwards calculating all the error signal

2. Gradients for hidden layer weights

$$\begin{aligned}
\frac{\partial E}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}\left(\frac{1}{2}\sum_k (a_k - t_k)^2\right) \\
&= \frac{1}{2}\sum_k (a_k - t_k)\frac{\partial}{\partial w_{ij}} a_k \\
&= \sum_k (a_k - t_k)\frac{\partial}{\partial w_{ij}} g_k(z_k) \\
&= \sum_k (a_k - t_k) g_k'(z_k)\frac{\partial}{\partial w_{ij}} z_k \\
&= \sum_k (a_k - t_k) g_k'(z_k) w_{jk} g_j'(z_j) a_i \\
&= a_i g_j'(z_j) \sum_k \delta_k w_{jk} = a_i \delta_j
\end{aligned}$$



$$z_k = b_k + \sum_j a_j w_{jk}$$

$$\frac{\partial z_k}{\partial w_{ij}} = \frac{\partial z_k}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} = w_{jk} g_j'(z_j) a_i$$

$$a_k = \overbrace{g_k\left(\overbrace{b_k + \sum_j \underbrace{g_j\left(\underbrace{b_j + \sum_i a_i w_{ij}}_{z_j}\right)}_{a_j} w_{jk}}^{z_k}\right)}^{a_k}$$

$$\boxed{\delta_j = g_j'(z_j)\sum_k \delta_k w_{jk}}$$

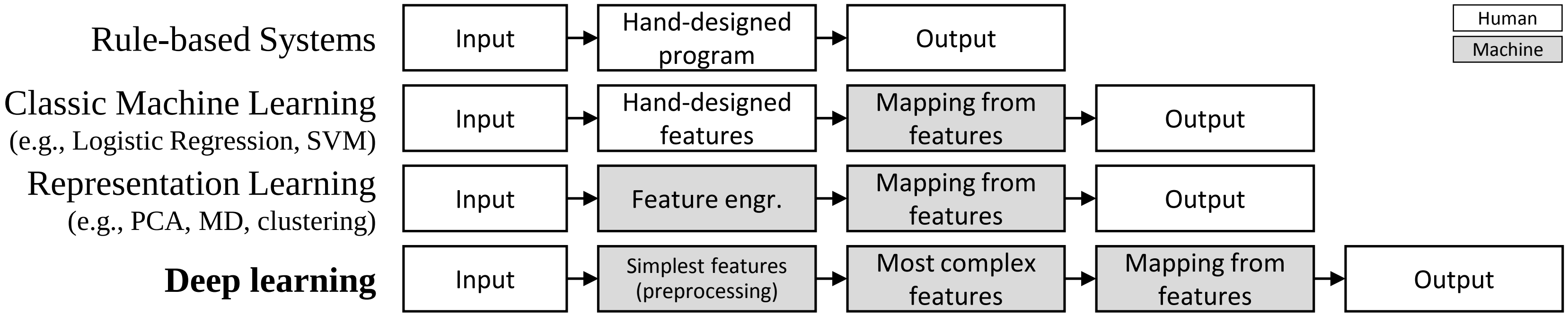
error signal of neuron j in hidden layer

g : activation fucntion
(sigmoid, tanh, Relu, etc)

Deep Learning

- Complex models with large number of parameters
  - Hierarchical representations
  - More parameters = more accurate on training data
  - Simple learning rule for training (gradient-based)
- Massive data
  - Needed to get better generalization performance
  - As sensory input dimension grows, required data exponentially increase (curse of dimensionality).
- A great deal of computing power needed: GPU, etc.
  - GPUs are suitable to handle such time consuming problems.

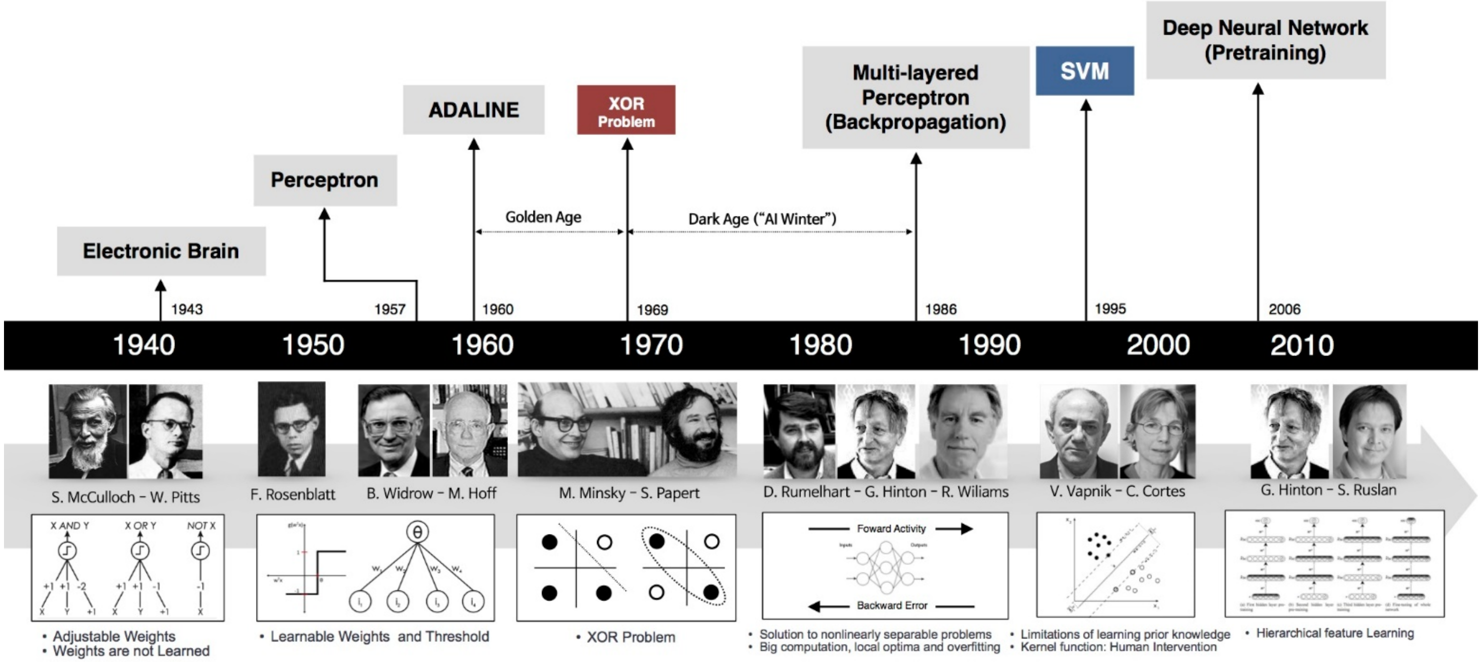
In the Context of AI/ML



Fully Automated Feature Discovery!

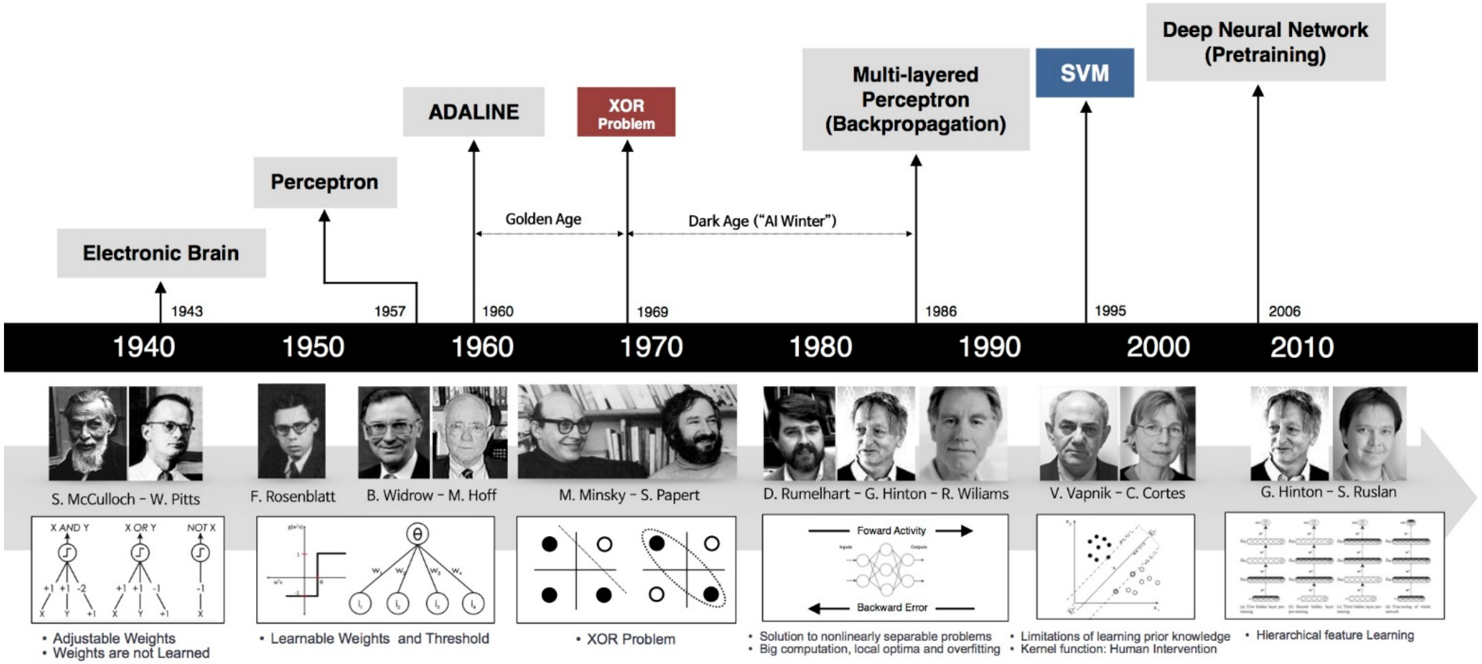
Neural Networks - History

- Simplified modeling of a neuron (McCulloch & Pitts, 1943).
- Introducing <u>the first Perceptron </u>(single layer network) (Frank, 1957).
- Skepticism analytics on the limitations of Perceptron: XOR problem (Minsky, 1969)
  - The First NNs Winter
- Error backpropagation: Powerful learning scheme for MLP (Honton, 1986)
  : Limitation of scale up to larger problem, and SVM rise (Cortes & Vapnik, 1995).
  - The second NNs Winter



Neural Networks - History

- Rebranding as ‘Deep Learning’
  : Introducing unsupervised pretraining and deep belief nets (Hinton, 2006)
- Breakthrough
  : Large datasets, GPU, new architectures, regularizations, optimizers (2012~)
  - Appearance of large, high-quality labeled datasets
  - Massively parallel computing with GPUs
  - Backprop-friendly activation: Relu, etc
  - Improved architectures: Resnets, inception.
  - Software platforms
  - New regularization techniques: dropout, batch-norm
  - Robust optimizers: SGD, ADAM



Current Trends

- Deep belief networks (based on Boltzmann machine)
- Convolutional neural networks
- Deep recurrent neural networks using (LSTM)
- Deep Q-learning Network (extensions to reinforcement learning)
- Applications to diverse domains.
  - Vision, speech, video, NLP, PHM, etc.
- Lots of open source tools available.

Rank	Framework (Library)	Language
1	Tensorflow	C++, Python
2	Keras	Python
3	Caffe	C++, Python
4	Theano	Python
5	PyTorch	Python
6	MXNet	C++, etc.
7	Torch	C, Lua
8	CNTK	C++
9	dlib	C++
10	Deeplearning4j	C++, Java

Three Training Manners of Machine Learning

1. Unsupervised learning
2. Supervised learning
3. Reinforcement learning

Yann LeCun in his many talks this year has repeatedly hammered away at this analogy:

If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don′t know how to make the cake.

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

https://medium.com/intuitionmachine/predictive-learning-is-the-key-to-deep-learning-acceleration-93e063195fd0

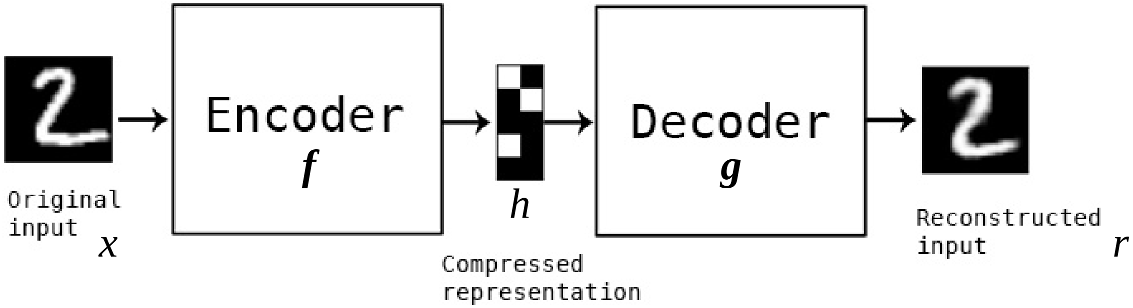
Autoencoder

What is an Autoencoder?

- Artificial neural network that is trained to attempt to copy its input to output
- Has a hidden layer *h* that describes the code used to represent the input

General Structure of an Autoencoder

- Maps an input *x* to an output *r* through an internal representation code *h* (called reconstruction)
  - It has a hidden layer *h* that describes a code used to represent the input
- The network has two parts
  - The "encoder" function $h=\boldsymbol{f}(x)$
  - A "decoder" that produces a reconstruction $r=\boldsymbol{g}(h)$



Autoencoder

Rational of an Autoencoder

- An autoencoder that simply learns to set $g\left(f(x)\right) = x$
- Autoencoders are designed to be unable to copy perfectly
  - They are restricted in ways to copy only approximately
  - Copy only input that resembles training data
- Because a model is forced to prioritize which aspects of input should be copied, it often learns useful properties of the data
- Modern autoencoders have generalized the idea of encoder and decoder beyond deterministic functions to stochastic mappings $p_{encoder}(h|x)$ and $p_{decoder}(x|h)$

Use of Autoencoder

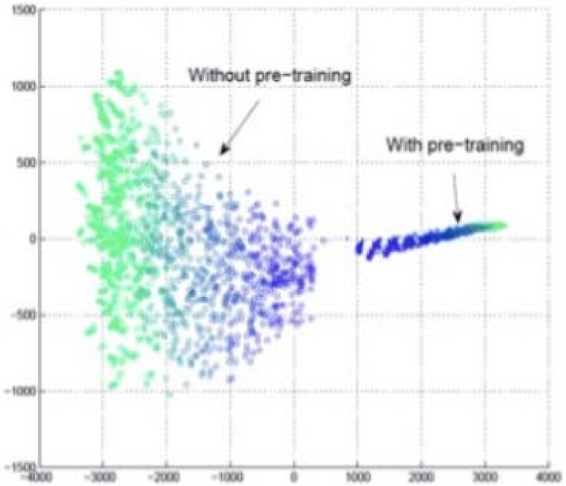
- The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction
- It roles essential parts in a greedy layer-wise pretraining of MLP in the next section

Greedy Layer-wise Pretraining

- Pretraining skills on Unsupervised Manner
  - Autoencoder → Stacked-Autoencoder(SAE, LeCun,2007)

Why is Greedy Layer-wise Pretraining working?

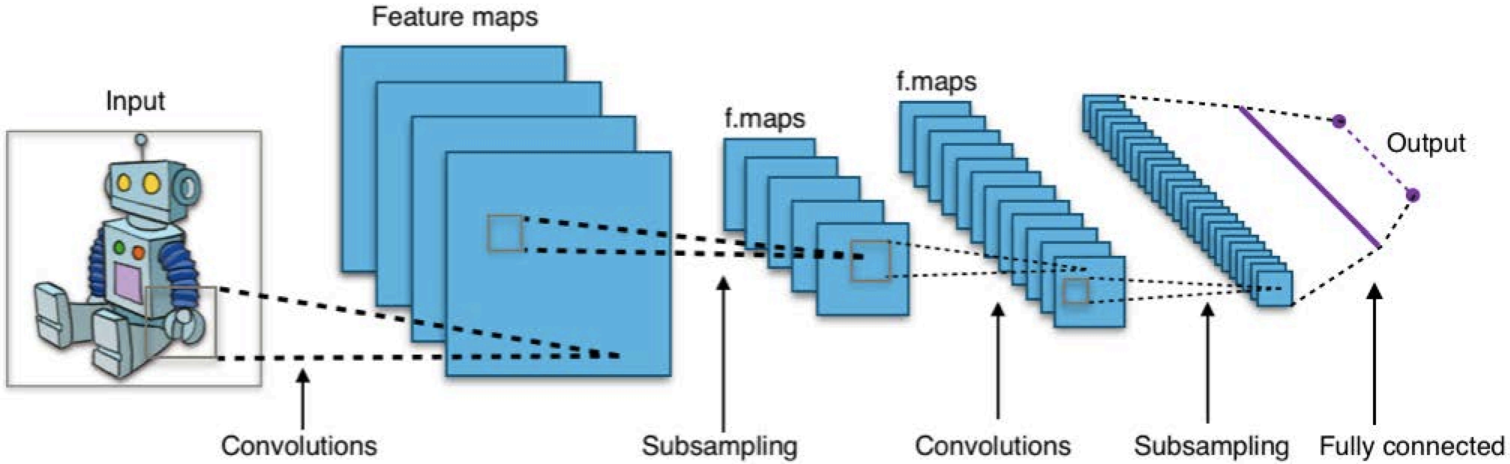
- Regularization hypothesis:
  Representations good for P(x) are good for P(y|x)
- Optimization hypothesis:
  Unsupervised initializations start near better local minimum of supervised training error
- Minima otherwise not achievable by random initialization
- It is no longer necessary, its purpose was to find a good initialization for the network weights in order to facilitate convergence when a high number of layers were employed. Nowadays we have ReLU, dropout and batch normalization.



https://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder

Convolutional Neural Networks

- Starts as another approach to solve “vanishing gradients” in backpropagation.
- The “neocognition” was introduced in 1980 which is inspired by animal’s visual cortexes and it reduces the number of weights and enables deep networks.
- Lecun developed LeNet-5 CNN network in 1998, that classifies digits.
- The research accelerated by the development of GPU computing.
- It is often used in image recognition, video analysis, natural language process, and Go.

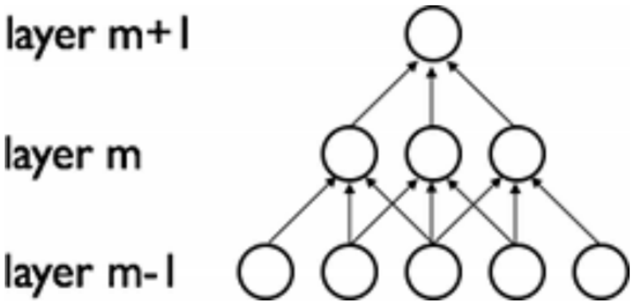


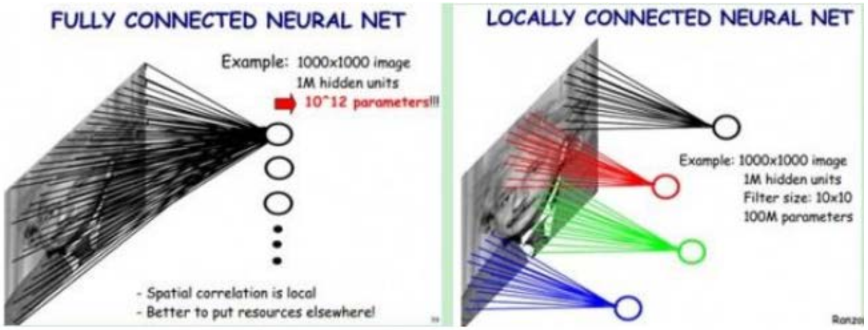
The Core Idea of CNNs

- The visual cortex contains a complex arrangement of cells. These cells are sensitivity to small sub-regions of the visual field, called a receptive field.
- Sparse connectivity
- Shared weights

Sparse connectivity

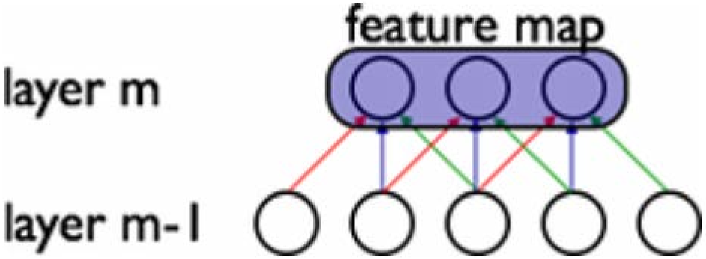
- CNNs exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers.
- In the below figure, units in layer m have receptive fields of width 3 in the input retina and are thus only connected to 3 adjacent neurons in the retina layer.





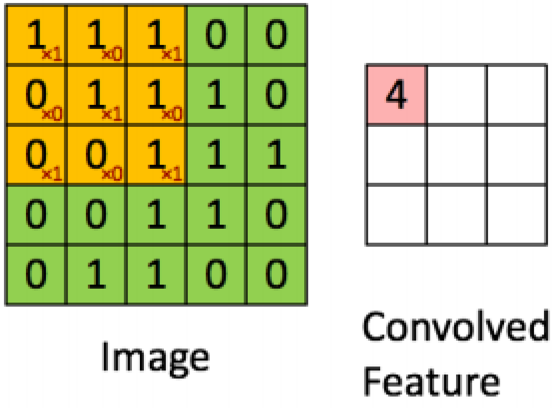
Shared Weights

- In addition, in CNNs, each filter h_i is replicated across the entire visual field.
- These replicated units share the same parameterization (weight vector and bias) and form a feature map.



The Convolutional Layer

- A feature map is obtained by repeated application of a function across subregions of the entire image.
- In other words, by convolution of the input image with a linear filter, adding a bias term and then applying a non-linear function.
- If we denote the k-th feature map at a given layer as h^k, whose filters are determined by the weights W^k and bias b_k, the feature map is obtained as $h_{ij}^k = \tanh\left(\left(W^k * x\right)_{ij} + b_k\right)$
- To form a richer representation of the data, each hidden layer is composed of multiple feature maps.
- The weights W of a hidden layer can be represented in a 4D tensor (*x*, *y*, R-G-B, kernel(feature)).



Subsampling

- Subsampling, or down-sampling, refers to reducing the overall size of a signal. In many cases, such as audio compression for music files, subsampling is done simply for the size reduction.
- But in the domain of 2D filter outputs, subsampling can also be thought of as increasing the position invariance of the filters. The specific subsampling method used in LeNets is known as 'max pooling'.

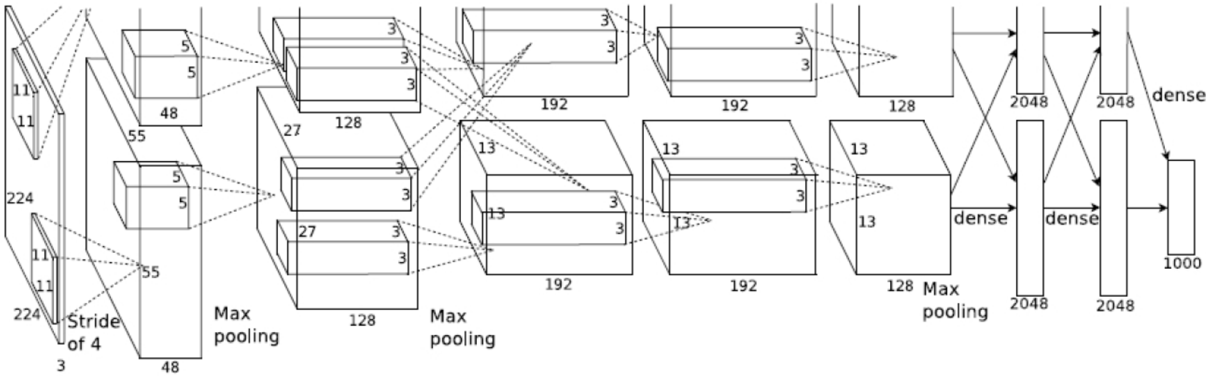
Max-Pooling

- Reducing computation for upper layers
- Preventing over-fitting

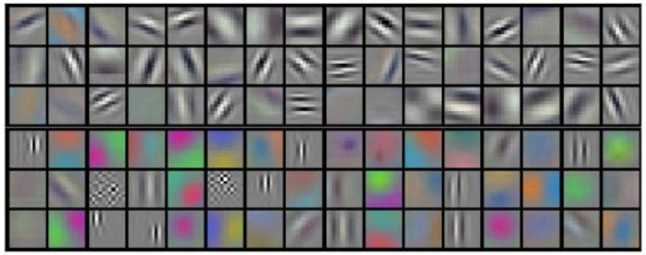
Convolutional Neural Networks

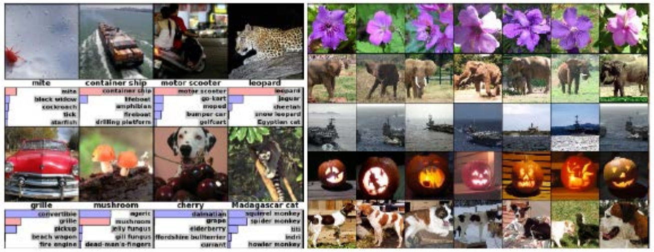
- Convolve several small filters on the input image
- Subsample this space of filter activations
- Repeat steps 1 and 2 until your left with sufficiently high level features.
- Use a standard a standard FFNN to solve a particular task, using the results features as input.

Deep Convolutional Neural Networks



- Krizhevsky et al. (2012)
- Applied to ImageNet competition (1.2 million images, 1,000 classes).
- Network: 60 million parameters and 650,000 neurons.
- Top-1 and top-5 error rates of 37.5% and 17.0%.
- Trained with back-propagation.

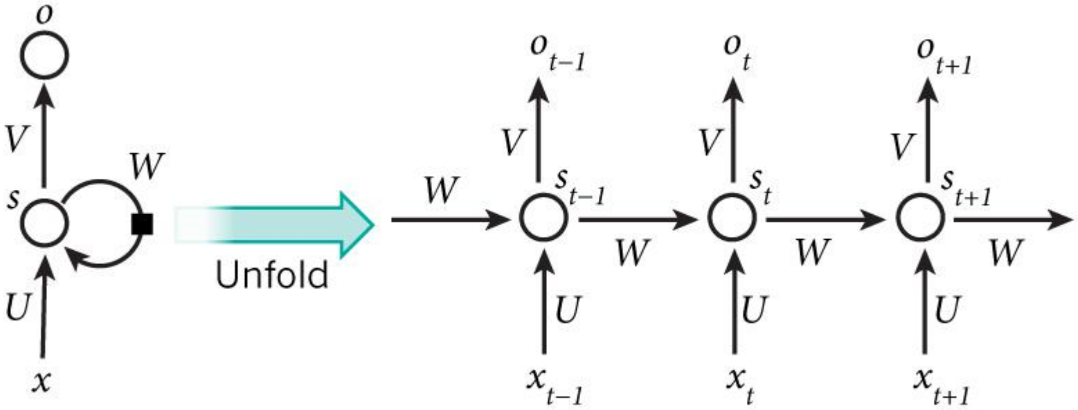




Recurrent Neural Network

Introduction

- A class of artificial neural network
  - where connections between units form a directed cycle



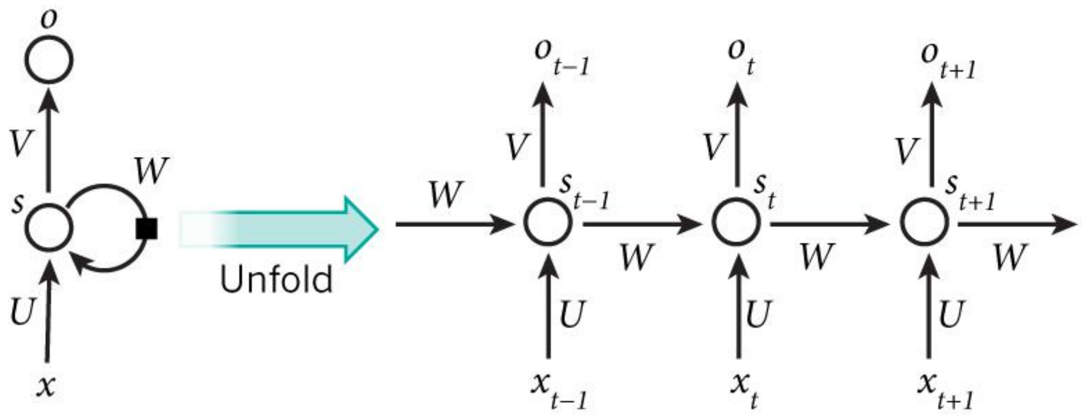
- RNN processes an input sequence one element at a time
  - maintain a ‘state vector’ in their hidden units
- The vector implicitly contains information about
  - The history of all the past elements of the sequence
- RNN is better for tasks that involve sequential inputs
  - Such as speech, language, and genome

U : weights of input
V : weights of output
W : previous weights of hidden states

Recurrent Neural Network

RNN Training: Back-propagation through time (BPTT)

- Standard backpropagation does not work on RNN because of recurrent weights
- Unfolding makes it clear how to apply backpropagation to train RNN
  - Consider the outputs of the hidden units at different time steps as if they were the outputs of different neurons in a deep network



- Copy the input and hidden unit activations for several previous time steps
- The more layers we maintain, the more history is included in our gradient computation
- This approach has become known as Back Propagation Through Time (BPTT)
- Introduce the new constraint that the weights at each level be identical

Fig from LeCun, Bengio, & Hinton (2015)

Recurrent Neural Network

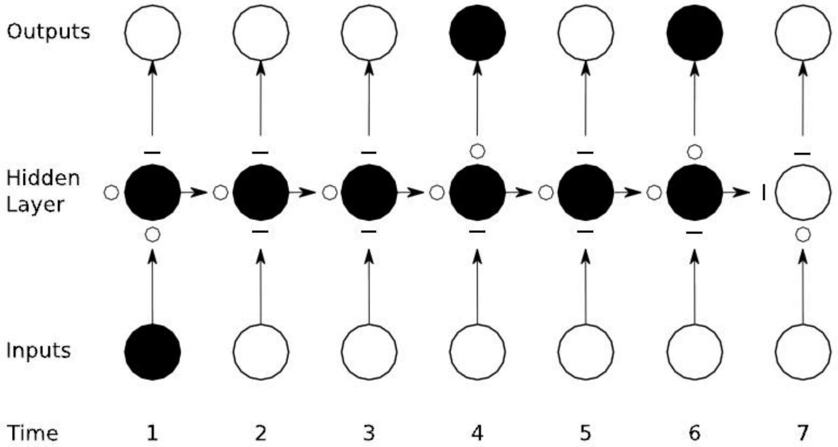
Challenges in RNN training

- Backpropagated gradients either grow or shrink at each time step
  - So over many time steps they typically explode or vanish
- The exploding gradient problem
  - When the weights in the matrix are large (i.e., the leading eigenvalue of the weight matrix is larger than 1.0)
  - Large gradient signals can cause learning to diverge
  - Easy to detect: overflow in gradient computation ➔ training cannot continue
- The vanishing gradient problem
  - When the weights in a weight matrix are small (i.e., the leading eigenvalue of the weight matrix is smaller than 1.0)
  - Learning either becomes very slow or stops working altogether
  - Learning long-term dependencies in the data becomes hard
  - Can go undetected while drastically hurting training
- In practice (with vanishing gradient)
  - the error information quickly disappears during backpropagation and cannot inform most previous steps of the error
  - Learning long-term dependency becomes problematic

Recurrent Neural Network

LSTM-RNN

- LSTM-RNN can preserve gradient information
  - Hidden layer units formed with Long Short-Term Memory (LSTM) cells can store and access information over long periods of time



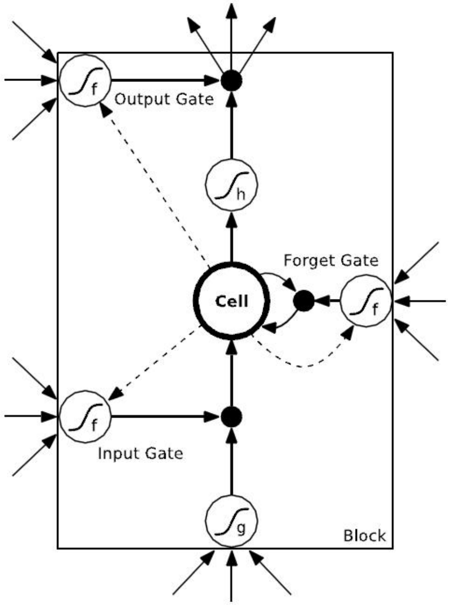
- For simplicity, all gates are either entirely open (o) or closed (-)
- The memory cell ‘remembers’ the first input as long as the forget gate is open and the input gate is closed
- The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell

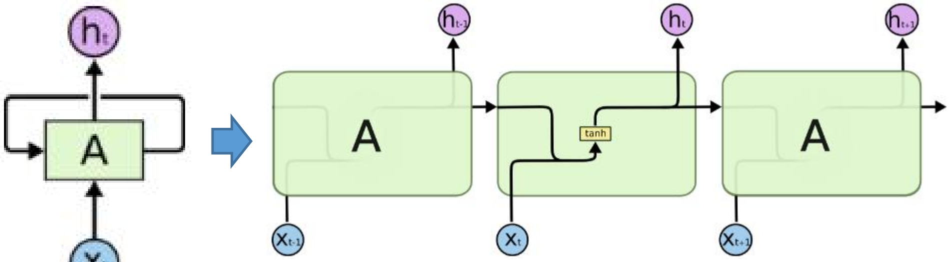
Fig from Graves (2012)

Recurrent Neural Network

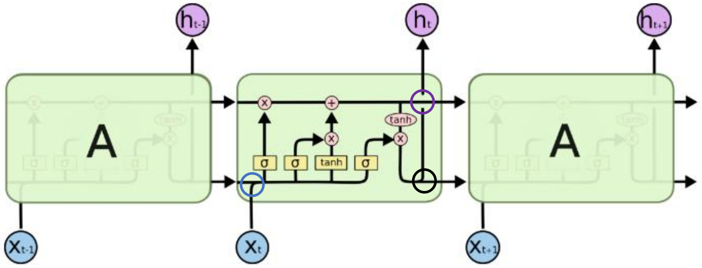
Long Shot-Term Memory

- LSTM cell architecture (left side)
  - Input gate adjust the influence from input to cell
  - Forget gate adjust the influence from cell to cell over time
  - Output gate adjust the influence from cell to output
- Comparison between standard RNN and LSTM (right side)





The repeating module in a standard RNN contains a single layer



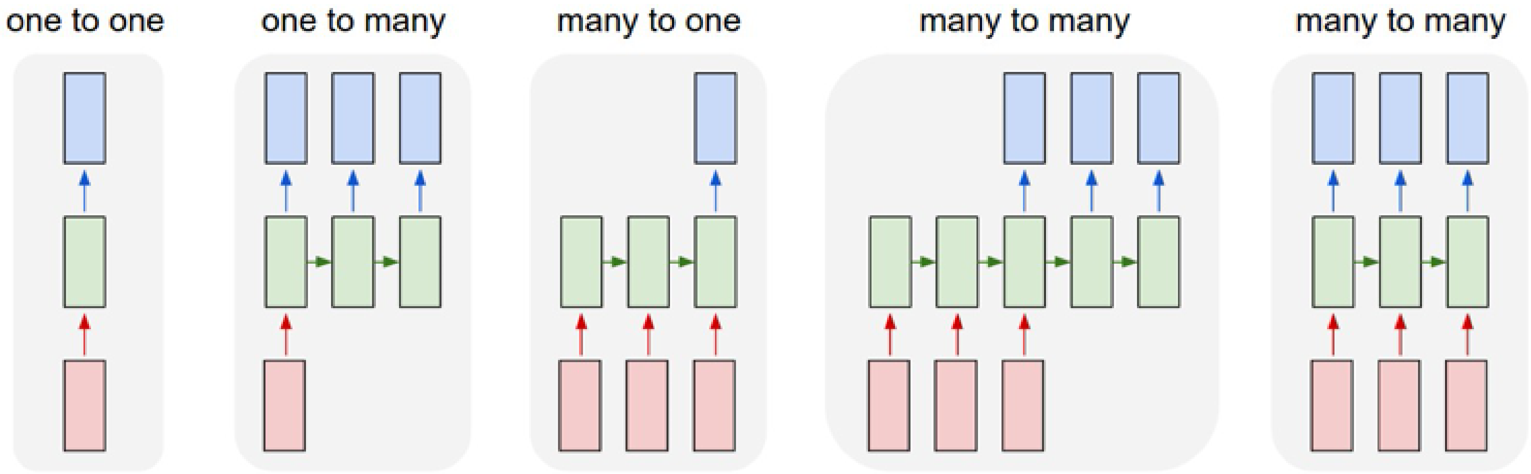
The repeating module in an LSTM contains four interacting layers

○ Input gate ○ Output gate ○ Forget gate

Fig from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network

Applications



- From left to right:
  - Without RNN, from fixed-sized input to fixed-sized output (e.g. image classification)
  - Sequence output (e.g. image captioning: takes a image and outputs a sentence of words)
  - Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment)
  - Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French)
  - Synced sequence input and output (e.g. video classification where we wish to label each frame of the video)

Fig from karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network

Applications (Image captioning)

- Generate or predict a text sequence from a given image
  - Image embedding is done by a CNN
  - Word embedding is done by an LSTM-RNN
  - Decoding to a text sequence is done by another LSTM-RNN
- Demo available at
  - http://www.cs.toronto.edu/~rkiros/lstm_scnlm.html

A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

Fig from LeCun, Bengio, & Hinton (2015)

Reinforcement Learning

Introduction

- RL is a general-purpose framework for decision-making
  - RL is for an agent with the capacity to act
  - Each action (a_t) influences the agent's future state (s)
  - Success is measured by a scalar reward (r_t) signal
  - Goal: optimal policy (π) to maximize future reward

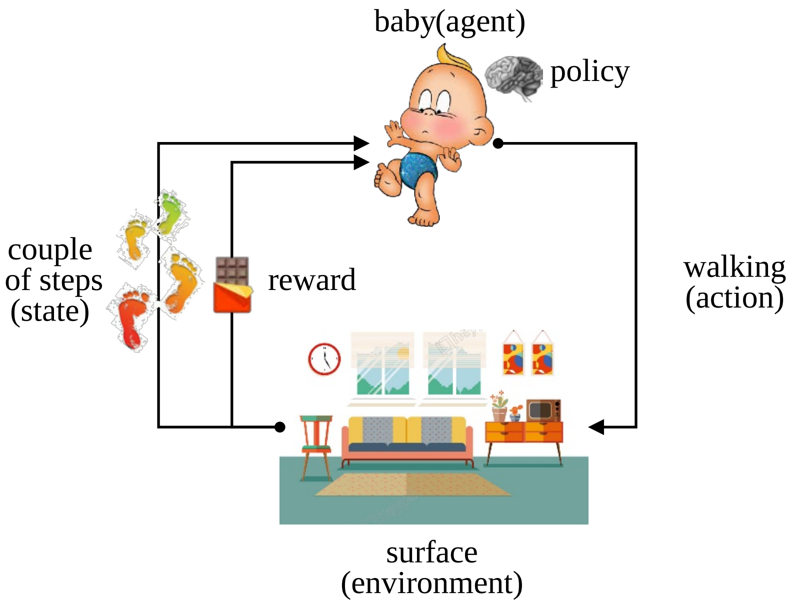
State

- State is the information used to determine what happens next

Policy

- A policy is the agent's behavior
  - It is a function from state to action:

$$a = \pi(s)$$



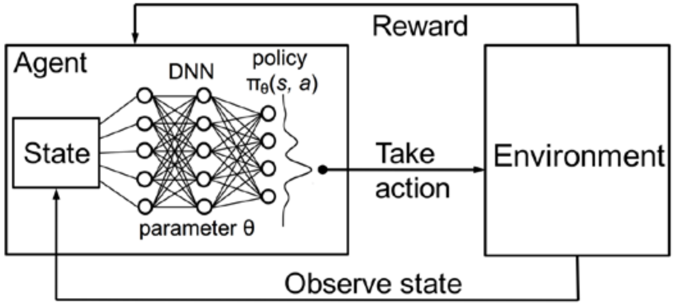
Reinforcement Learning

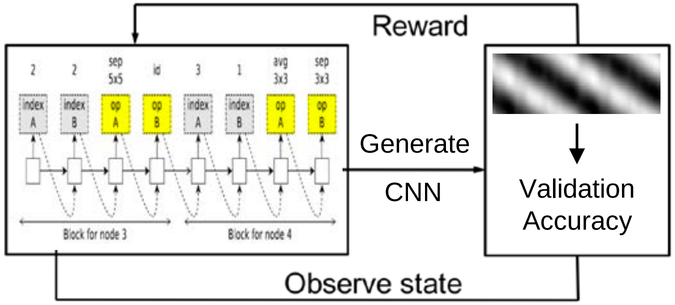
Policy Gradient

- Monte-Carlo Policy Gradient (REINFORCE)
  - Update parameters by stochastic gradient ascent
  - Update parameters to maximize rewards

Applications

- **Reinforcement Learning + RNN**
  - Objective : Find Optimized Convolutional Neural Network using ODR
  - Network : One-to-many RNN to generate Convolutional Neural Network
  - Objective Function : Maximize Validation Accuracy using Generated Neural Network





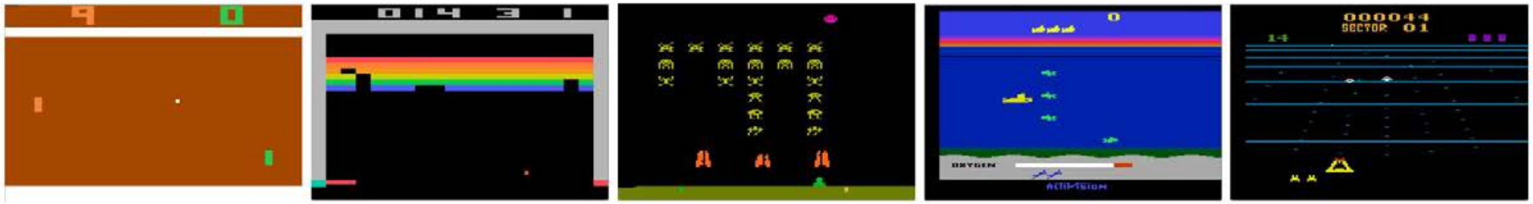
Deep Reinforcement Learning

Introduction

- Can we apply deep learning to RL?
- Use deep network to represent value function / policy / model
- Optimize value function / policy / model end-to-end using stochastic gradient descent
- If the function approximator is a deep neural network ➔ Deep Q-learning

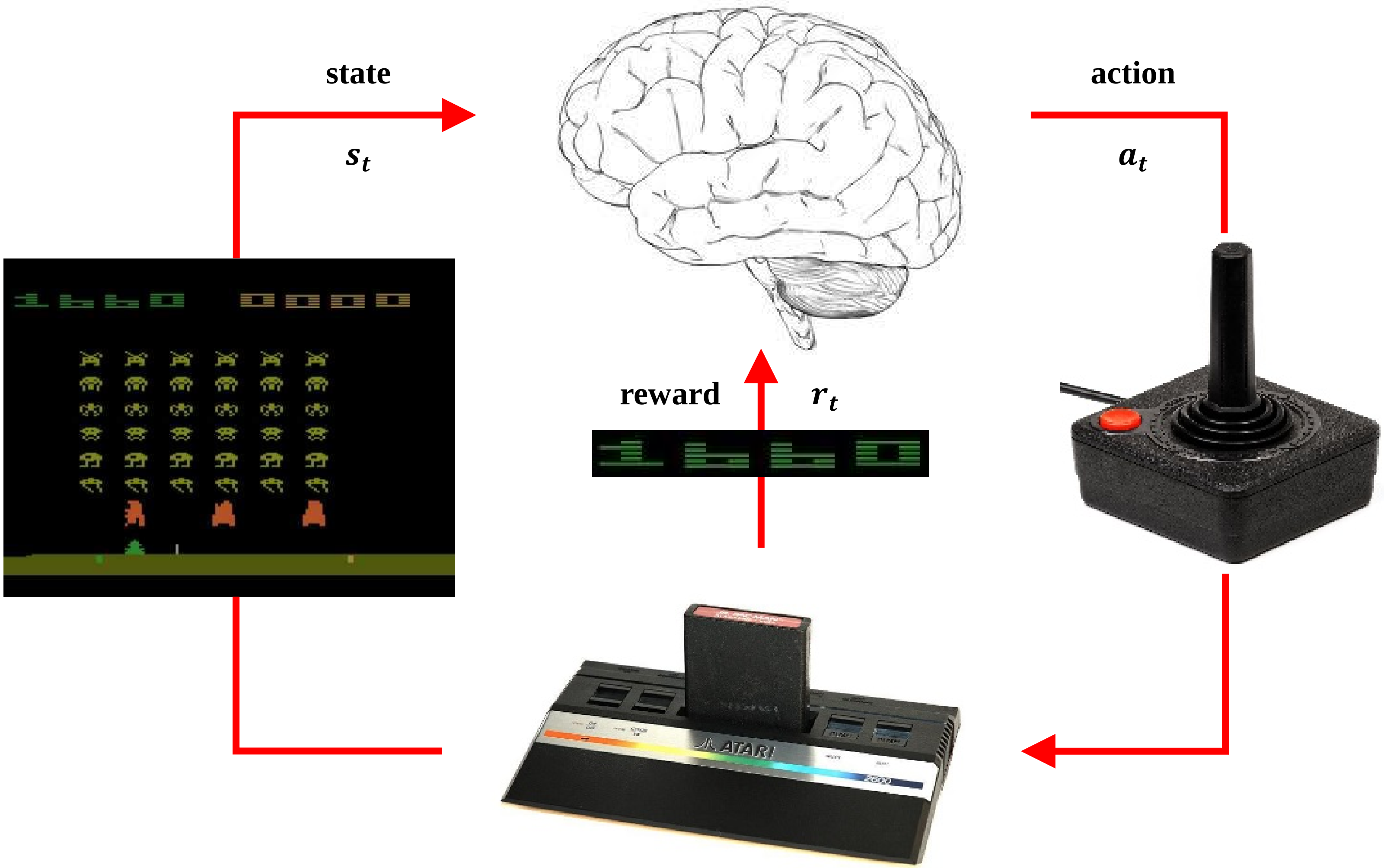
Deep Q-learning in Atari (video games)

- Objective: Human-level controller manipulation using image pixels of video games
- End-to-end learning of values $Q(s, a)$ from pixels
- Input state s is stack of raw pixels from last 4 frames (84×84 each)
- Reward is change in score for that step
- Network architecture and hyperparameters are fixed across all games



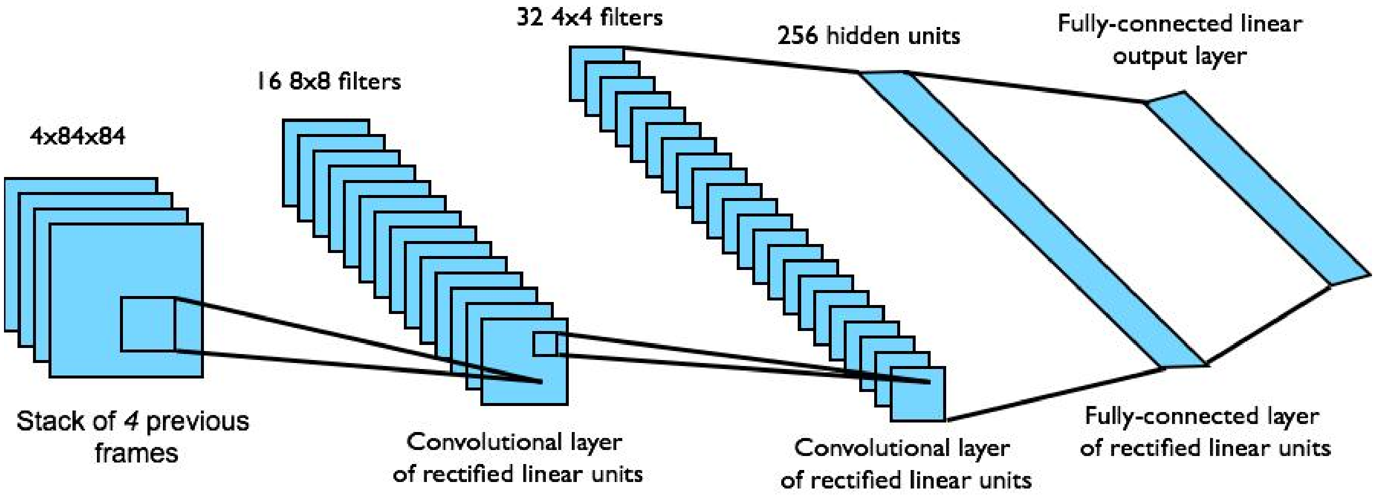
Deep Reinforcement Learning

Deep Q-learning in Atari (video games)



Deep Reinforcement Learning

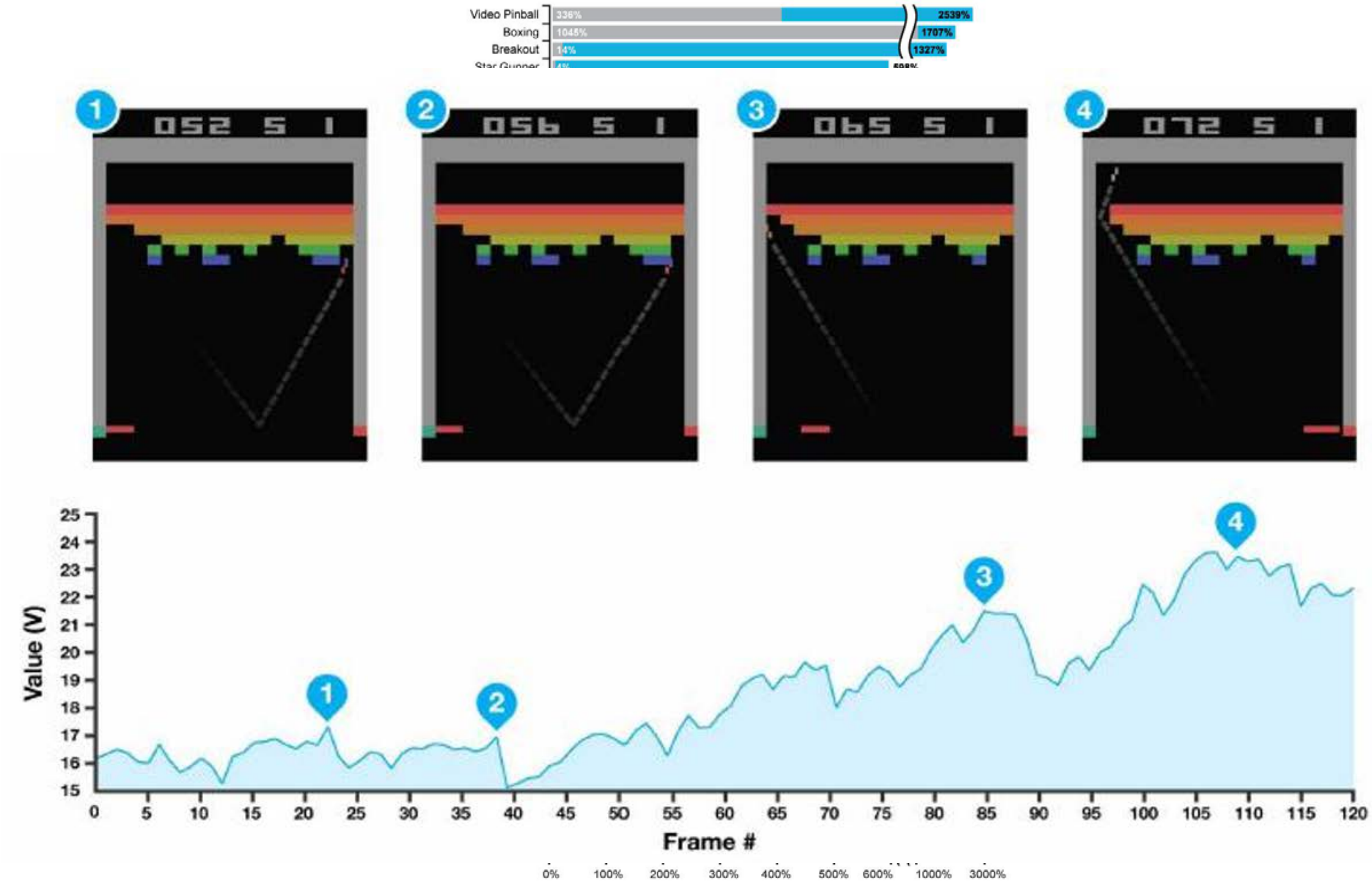
Deep Q-network (DQN) architecture



Layer	Input	Filter size	Stride	Num_filters	Activation	Output
conv1	84×84×4	8×8	4	16	ReLU	20×20×16
conv2	20×20×16	4×4	2	32	ReLU	9×9×32
fc3	9×9×32				ReLU	256
fc4	256				Linear	18

Deep Reinforcement Learning

Deep Q-network (DQN) Results in Atari



- Superhuman performance on over half of the games in Atari

Case Study - Autoencoder

Autoencoder

- Is a neural network operating in unsupervised learning mode
- The output and the input are set equal to each other
- Learns an identity mapping from the input to the input
- Applications
  - Dimensionality reduction (Efficient, Non-linear)
  - Representation learning (discovering interesting structures)
  - Alternative to RBM for layer-wise pre-training of DNN

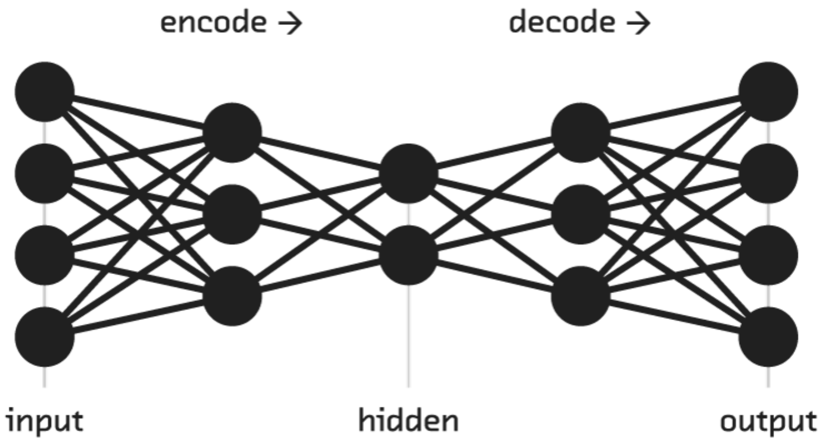
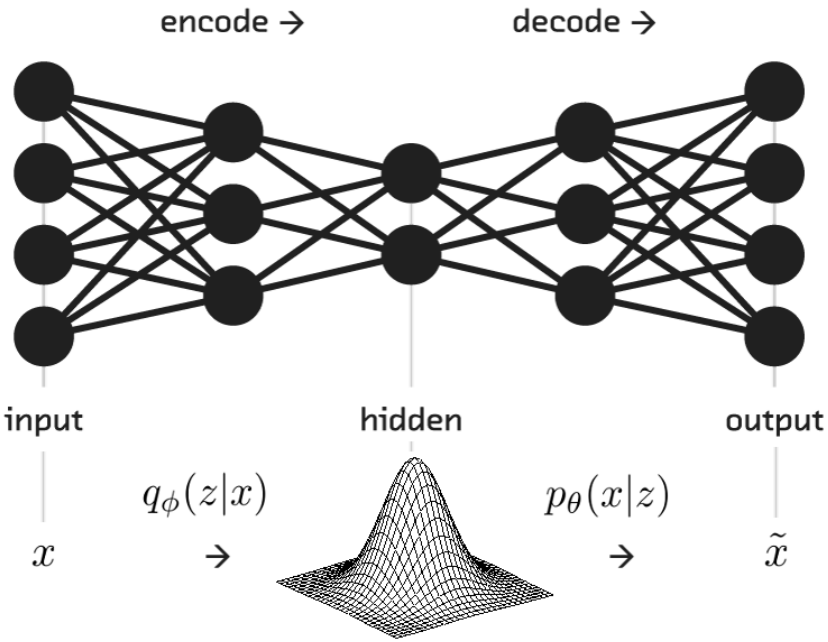


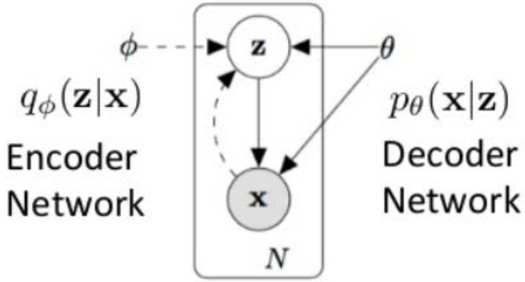
Fig from https://www.jianshu.com/p/298ad3d531f7

Case Study - Autoencoder

Variational Autoencoder

- Minimize reconstruction loss
- Minimize distance between encoded latent distribution and prior distribution





Minimize: $D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})]$

Intractable: $p_\theta(\mathbf{z}|\mathbf{x}) = \dfrac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}$

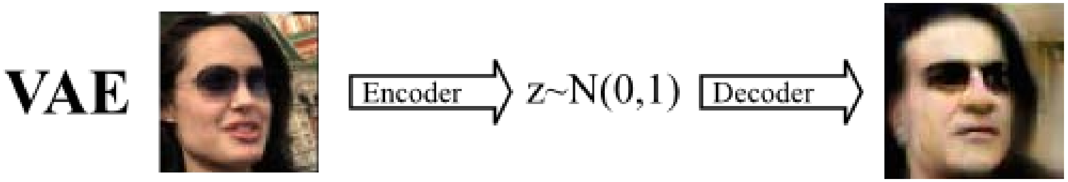
Fig from https://www.jianshu.com/p/298ad3d531f7

Fig from https://www.slideshare.net/ckmarkohchang/variational-autoencoder

Case Study - Autoencoder

Image Generation using Variational Autoencoder

- Applications (faces)



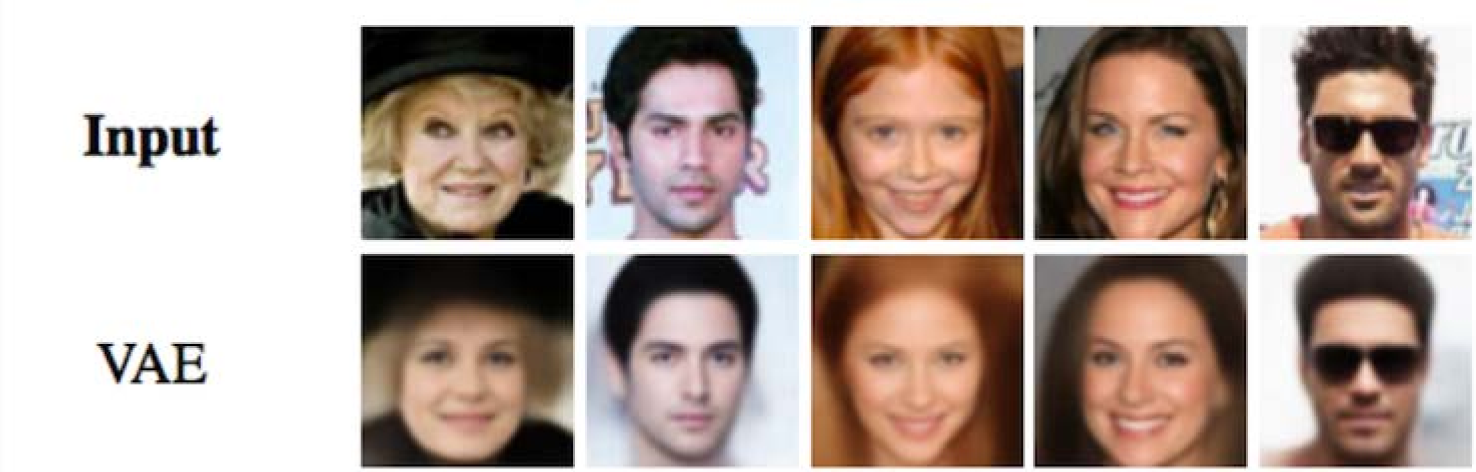


Fig from http://torch.ch/blog/2015/11/13/gan.html

Case Study - Convolutional Neural Networks

Convolutional Neural Networks

- CNN have become an important tool for object recognition
  - Image Classification
  - Object Detection
  - Instance Segmentation

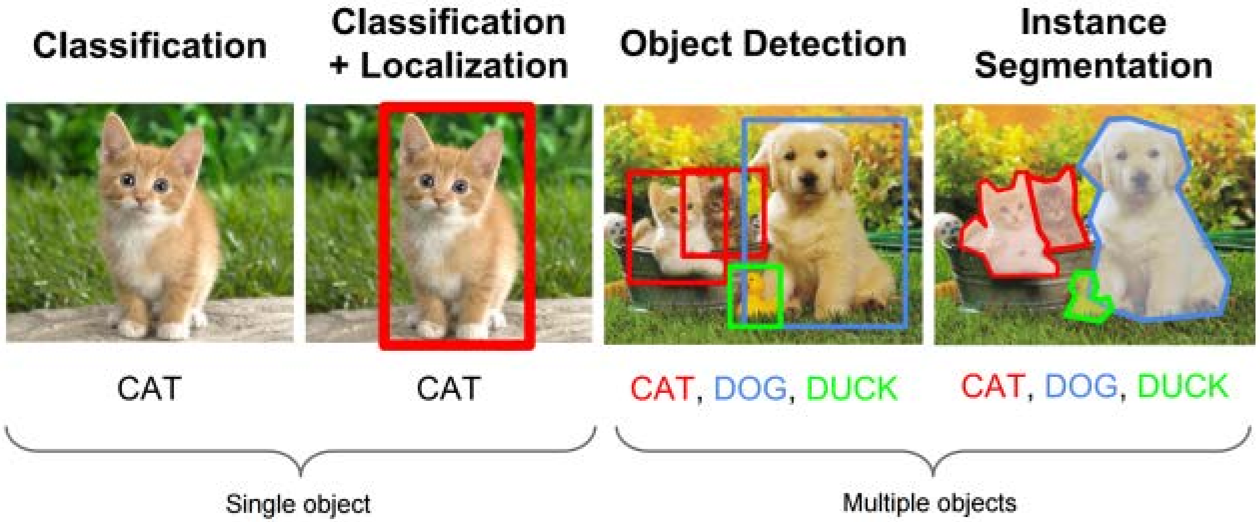


Fig from http://man-about-town.tistory.com/51

Case Study - Convolutional Neural Networks

2D Image Classification

- What’s the problem with the value iteration algorithm?
  - C1: filtered by 4×5×5×1 (kernel×w×h×color) convolutional kernels which create 4 feature maps
  - S1: Feature maps are subsampled by maxpooling
  - C2: filtered 10×5×5×1 kernels
  - S2: Feature maps are subsampled by maxpooling
  - FC: Fully connected layer where all generated features are combined and used in the classifier.

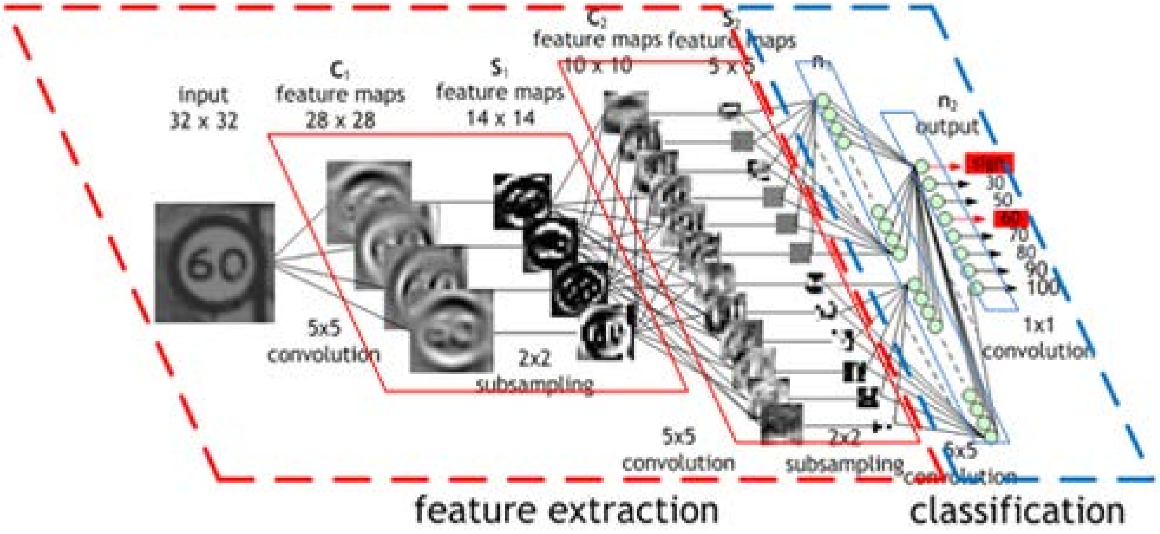


Fig from https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/

Case Study - Convolutional Neural Networks

Multi-View Convolutional Neural Network for 3D Shape

- Multi-View CNN (MVCNN) which:
  - Generates compact shape descriptors
  - Leverages both image and 3D Shape datasets
  - Can be fine-tuned end-to-end for improved performance

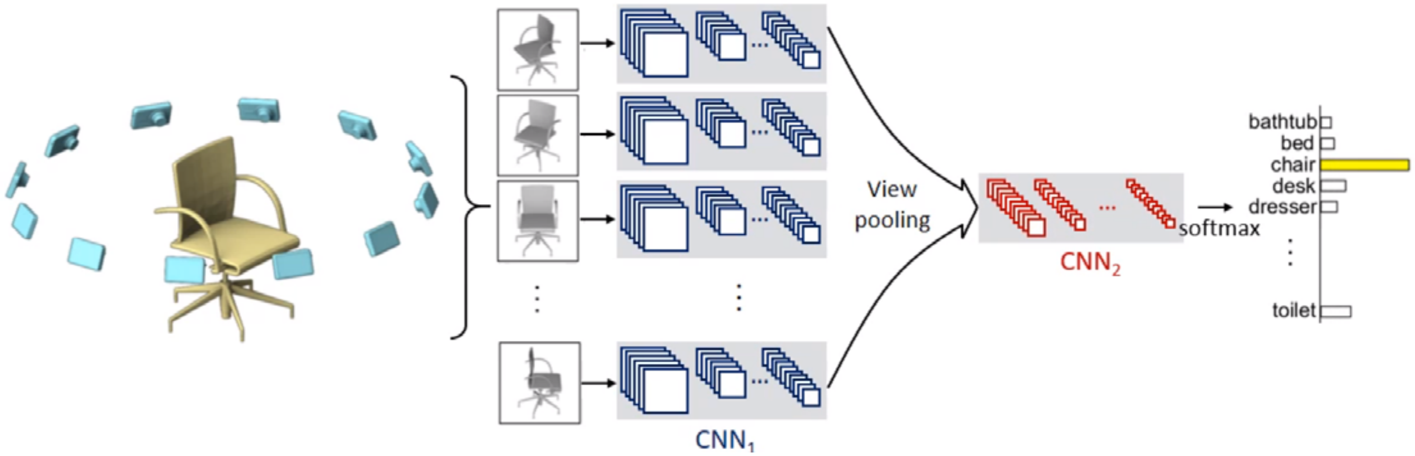


Fig from http://vis-www.cs.umass.edu/mvcnn/

Case Study - Convolutional Neural Networks

Voxel based 3D Convolutional Neural Networks

- Voxelize high dimension mesh
  - Volumetric representation
  - High-level deformation intentions

3D Convolutional Neural Networks

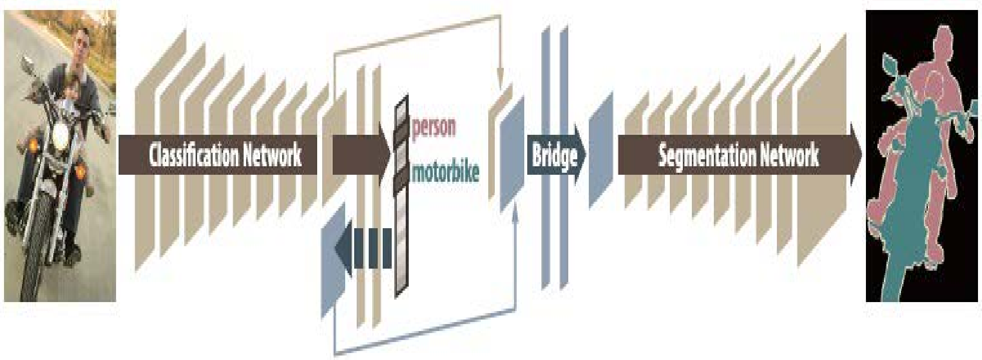
Fig from http://geometry.cs.ucl.ac.uk/projects/2016/semantic_learning/

Fig from https://www.mathworks.com/discovery/convolutional-neural-network.html

Case Study - Convolutional Neural Networks

Deconvolution Neural Network for Segmentation

- Overall architecture of the proposed network
  - On top of the convolution network based on VGG 16-layer net
  - The deconvolution network is composed of deconvolution and unpooling layers



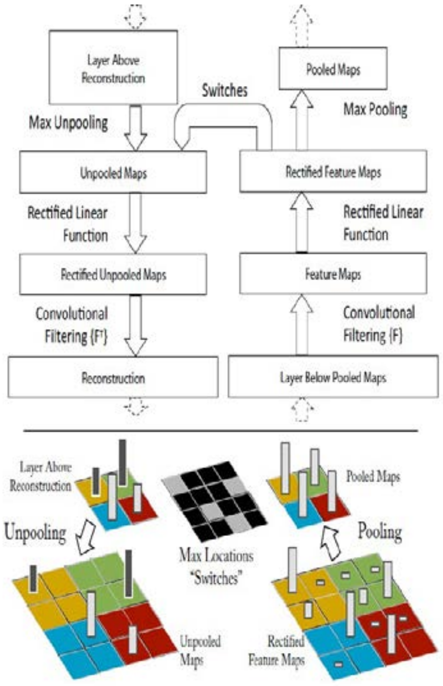


Fig from http://developers-club.com/posts/253859/

Fig from http://blog.csdn.net/yihaizhiyan/article/details/46991147

Case Study - Convolutional Neural Networks

CNN for Sentence Classification

- New: Multi-Channel
  – A model with two sets of word vectors. Each set of vectors is treated as a ‘channel’ and each filter is applied to both channels

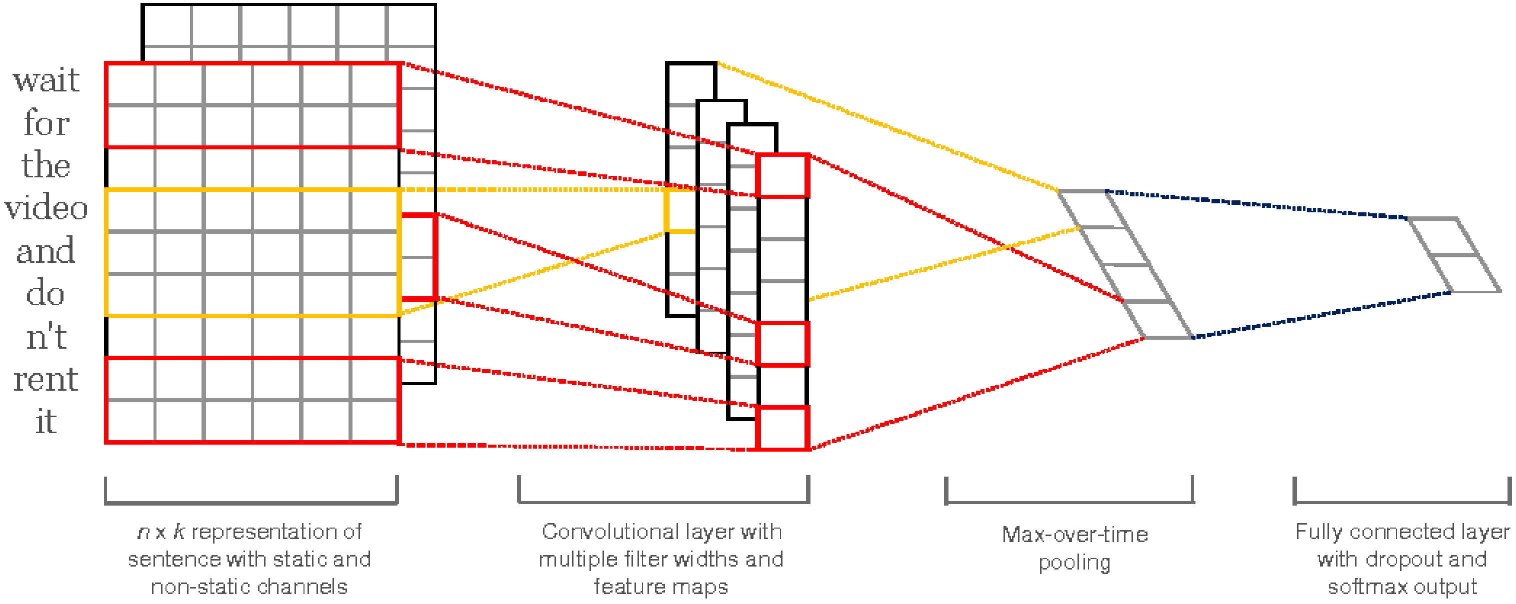
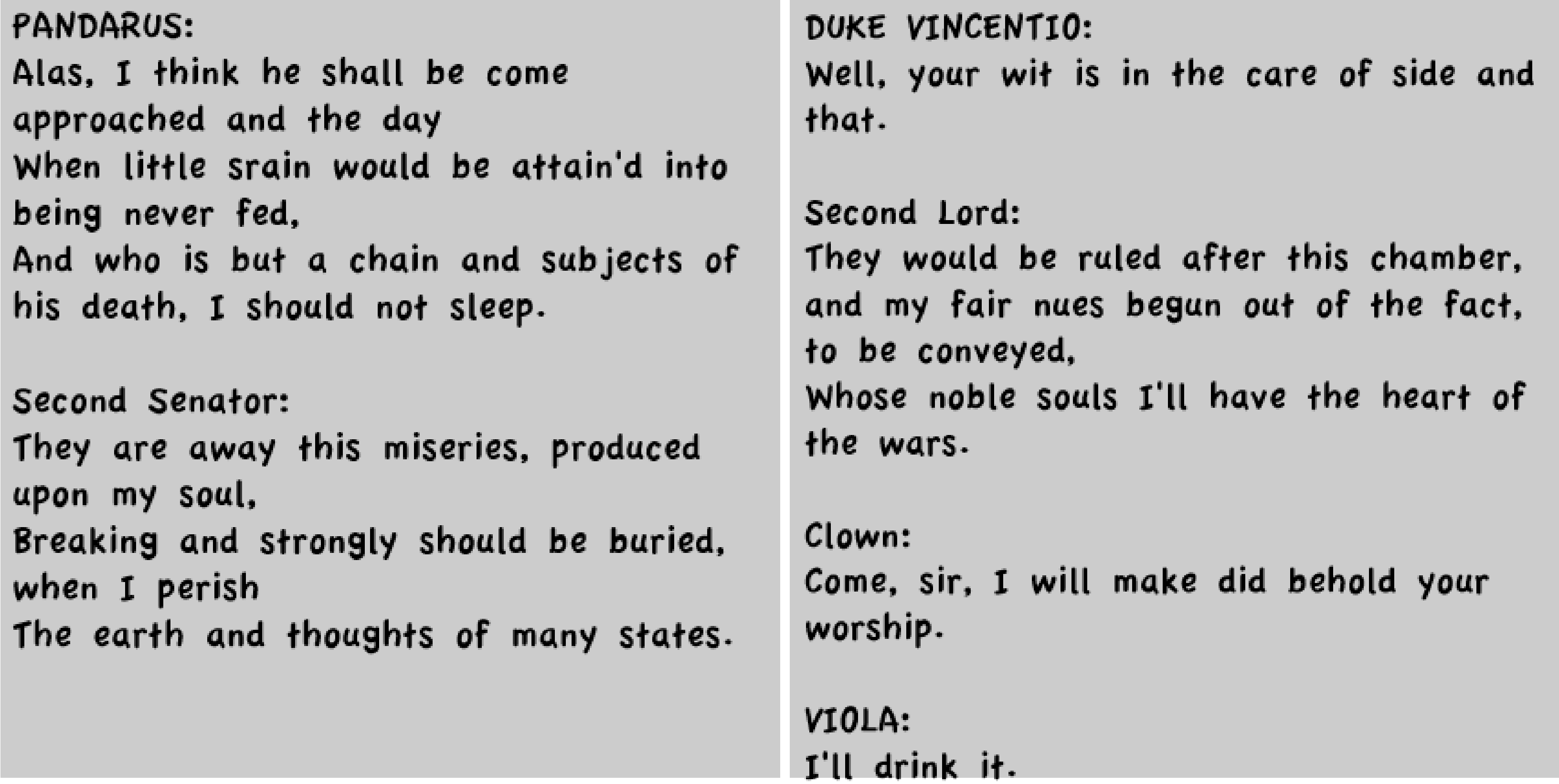


Fig from https://jamiekang.github.io/2017/06/12/cnn-for-sentence-classification/

Case Study - Recurrent Neural Network

Applications (Text generation)

- Download all the works of Shakespeare and concatenated them into a single file
- 3-layer RNN with 512 nodes on each layer
- Output: a sequence of characters (not words)



PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death, I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

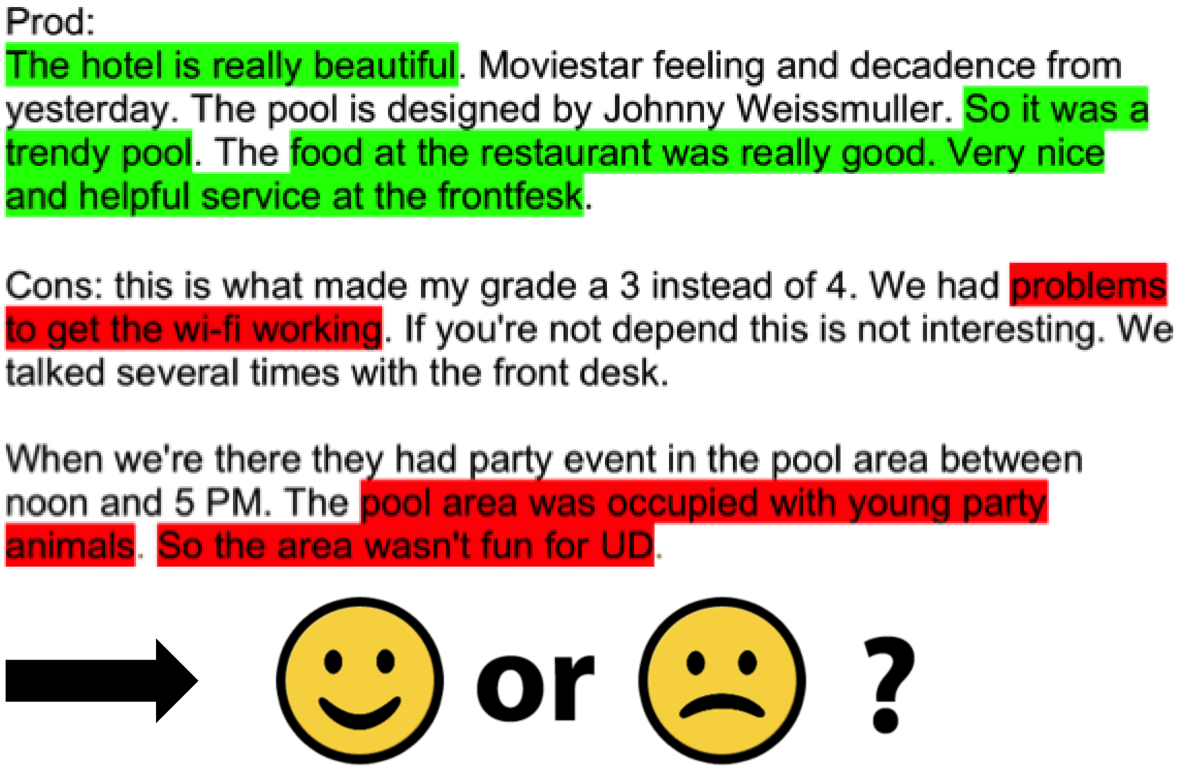
VIOLA:
I'll drink it.

Fig from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Case Study - Recurrent Neural Network

Applications (Sentimental analysis)

- Input: a sequence of characters
- Identify the orientation of opinion in a piece of text
- Determine whether a sentence or a document expresses positive or negative



Prod:
The hotel is really beautiful. Moviestar feeling and decadence from yesterday. The pool is designed by Johnny Weissmuller. So it was a trendy pool. The food at the restaurant was really good. Very nice and helpful service at the frontfesk.

Cons: this is what made my grade a 3 instead of 4. We had problems to get the wi-fi working. If you're not depend this is not interesting. We talked several times with the front desk.

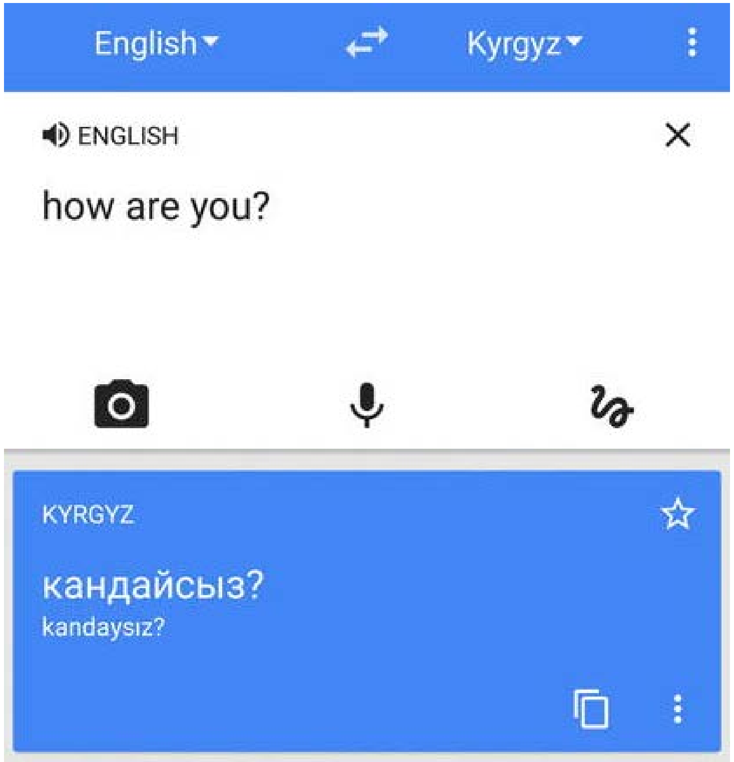
When we're there they had party event in the pool area between noon and 5 PM. The pool area was occupied with young party animals. So the area wasn't fun for UD.

Fig from https://www.cloudbeds.com/articles/perform-sentiment-analysis-reviews/

Case Study - Recurrent Neural Network

Applications (Machine Translation)

- Input: a sequence of characters
- Output: a sequence of characters
- Application of computers to the task of translating texts from one language to another



how are you?

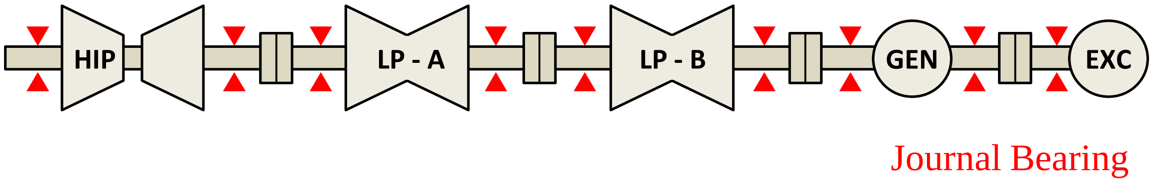
кандайсыз?
kandaysız?

Fig from http://www.transperfect.com/blog/machine-translation-solution-you-can-bank-on

Journal Bearing Fault Diagnosis

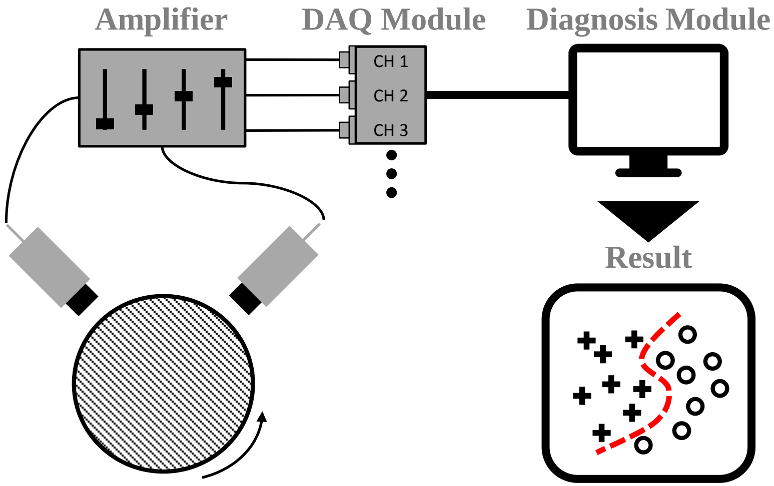
Journal Bearings in Steam Turbine

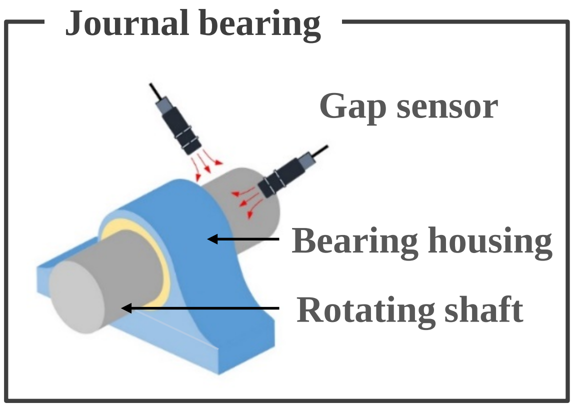
- 5 tilting pad, 4 tilting pad, elliptical bearings



Sensor and Data Acquisition

- Optimized settings for accurate diagnosis & prognosis

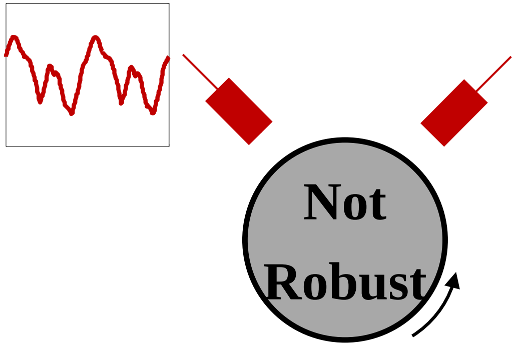




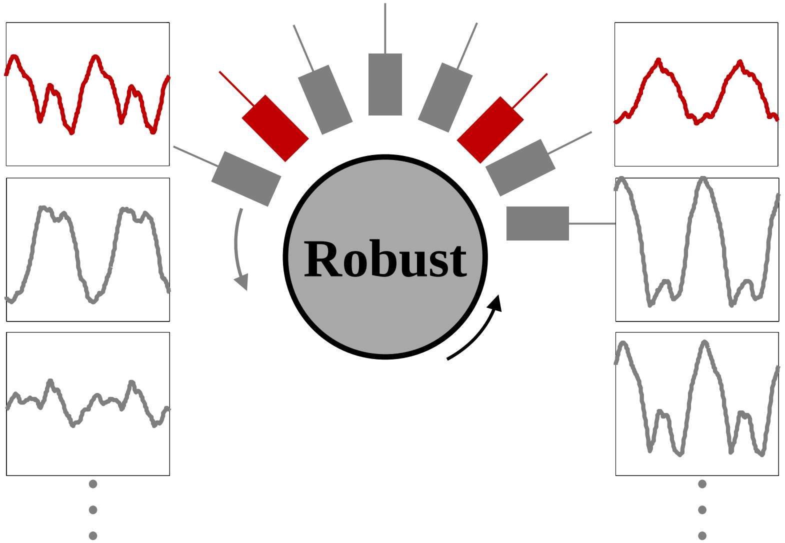
Journal Bearing Fault Diagnosis

Vibrational Signal from Rotating Machine

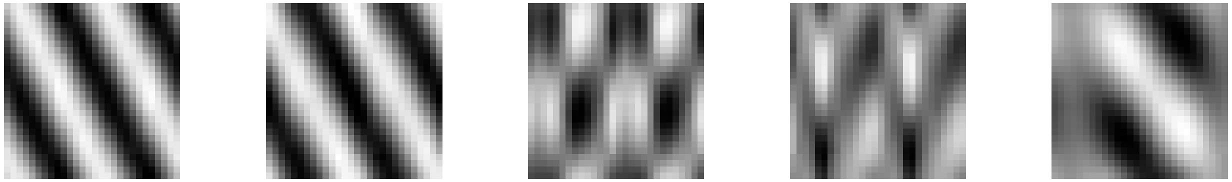
- Signals from two gap sensors



- Vibrational signals



- Vibrational signals

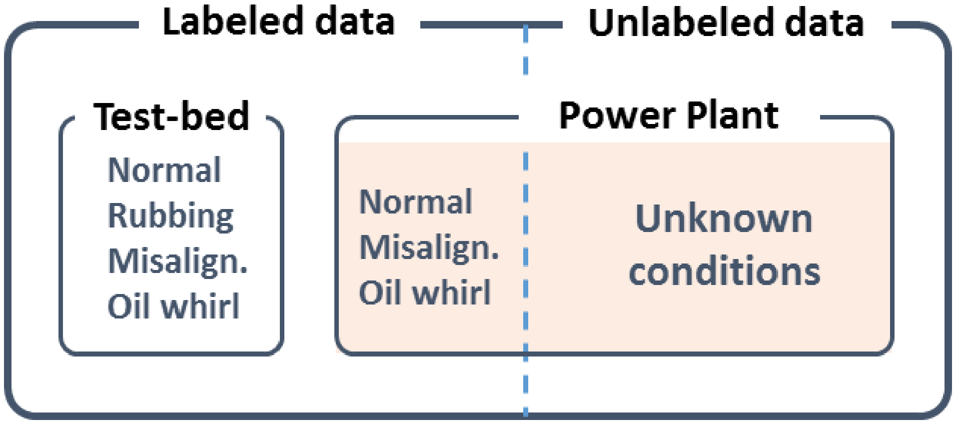


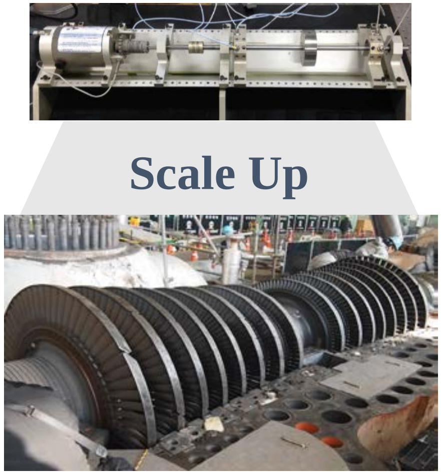
Journal Bearing Fault Diagnosis

Vibrational Signal from Rotating Machine

- Challenge 1. To minimize the time and effort to extract optimal health features
- Challenge 2. To use the algorithm trained from the testbed for real power plants

Labeled data from testbed and real power plants





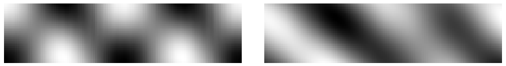
Journal Bearing Fault Diagnosis

Deep Learning Algorithms (DBN vs CNN)

Vibration Image Generation

RK4 test-bed (Labeled)
Power plant (Labeled)

Power plant (Unlabeled)



Vibration images using ODR

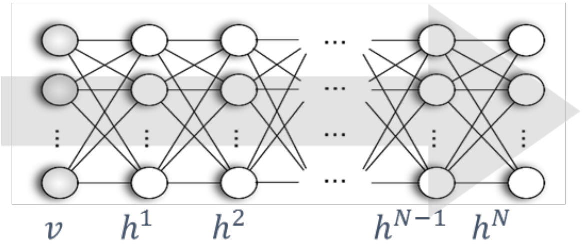
High-level Feature Abstraction

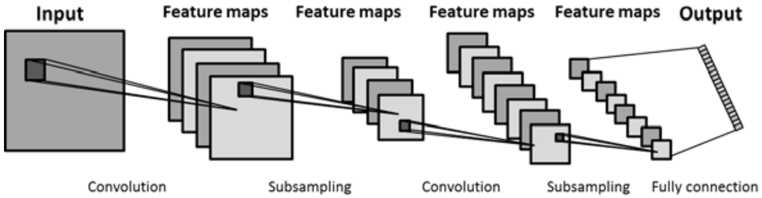
- Greedy Layer-wise unsupervised pretraining: restricted Boltzmann machine in deep belief network (DBN) + histogram of gradient (HOG)

- Convolutional Neural Network (CNN)
  - Convolutional layer
  - ReLu/Pooling layer
  - Fully connected layer

Health Reasoning

- Classification: multi-layer perceptron (MLP)
- Clustering: self-organizing map



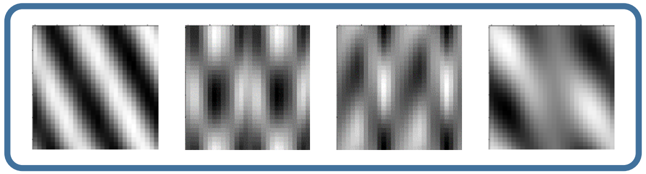


Journal Bearing Fault Diagnosis

Motivation – **To develop autonomous feature engineering for journal bearing rotor systems without label information**

Unsupervised Feature Extraction

Vibration Image Generation1)



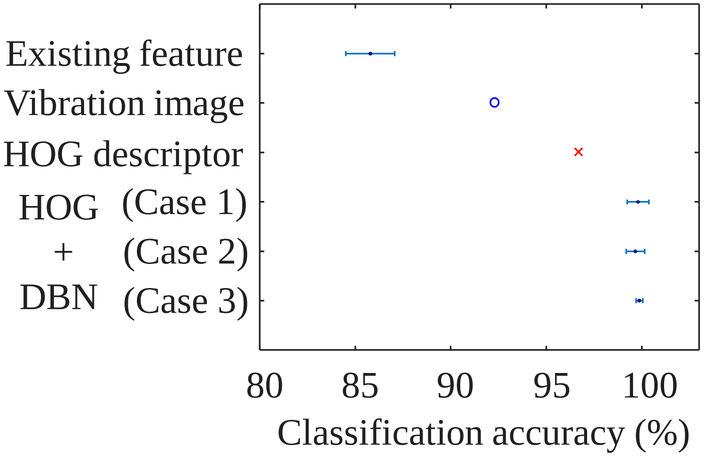
High-level Feature Abstraction2)

- Unsupervised feature extraction by three-layer deep belief network (DBN)

Health Reasoning2)

- Classification: multi-layer perceptron (MLP)
- Clustering: self-organizing map (SOM)

Class Prediction Results

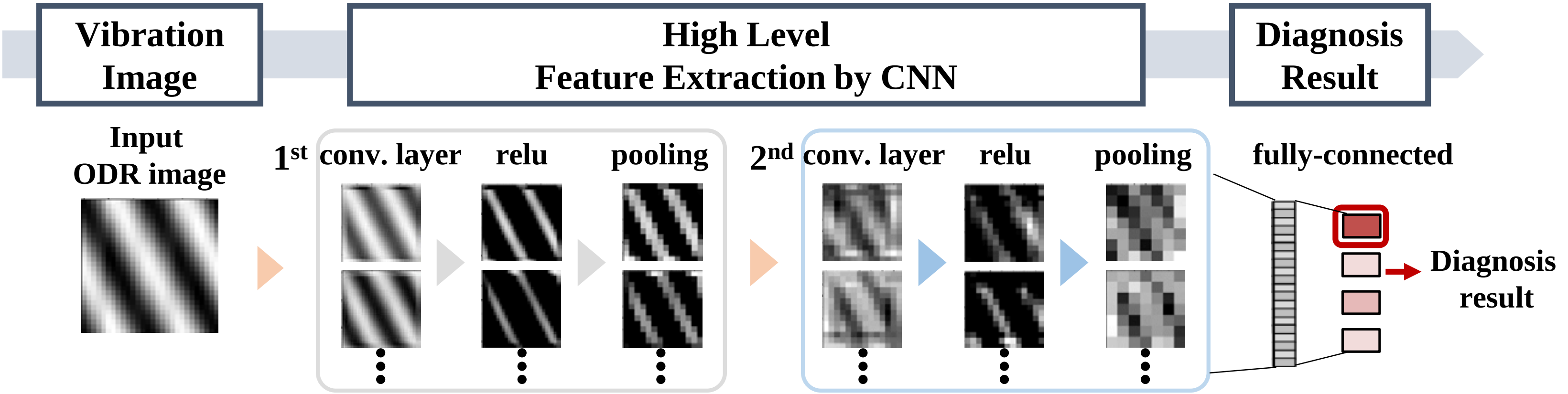


	(Case 1)	(Case 2)	(Case 3)
Node combination	512, 1024, 2048	540, 1080, 1620	500, 1000, 2000

Result – **Powerful unsupervised feature engineering without label information**

Journal Bearing Fault Diagnosis

Motivation – To extract high level features from vibration images automatically

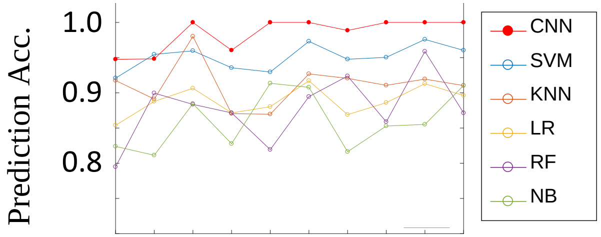


Vibration Image

High Level Feature Extraction by CNN

Diagnosis Result

Example) Journal Bearing Rotor System Diagnosis



- **CNN method has the best average performance.**
- **CNN method does not require human resources for feature engineering.**

*Training: 3 sets
Testing: 2 sets → 10 set combinations

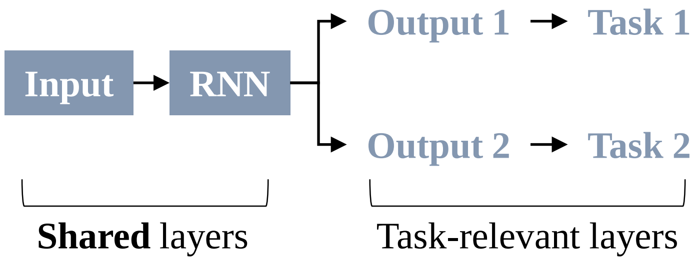
Result

- Autonomous feature engineering using CNN based on vibration images
- Computational tractable with GPU

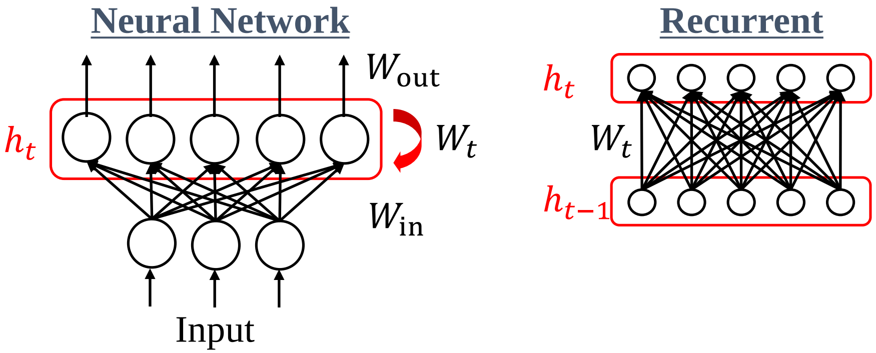
Journal Bearing Fault Diagnosis

Motivation – To improve generalization of a classifier by learning more than a task

Multi-task Learning (diagnosis & prognosis)

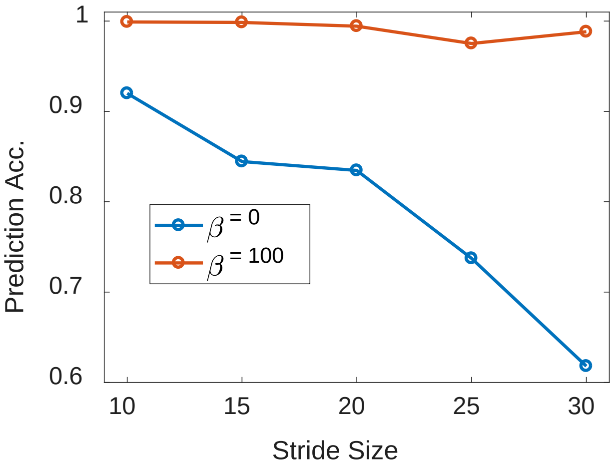


Recurrent Neural Network (RNN)



Example) Journal Bearing Rotor Systems

- Task 1: classification, Task 2: regression
- $L = L_{classification} + \beta \times L_{regression}$
  (L: cost value, β: hyper-parameter)

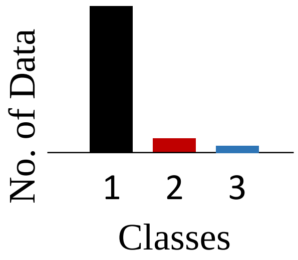


Result – Multi-task learning for generalization (diagnosis & prognosis) with good performance regardless of stride size

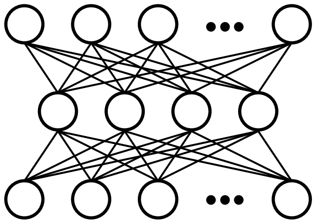
Journal Bearing Fault Diagnosis

Motivation – To improve the feature learning for class imbalanced data

Class Imbalanced Data

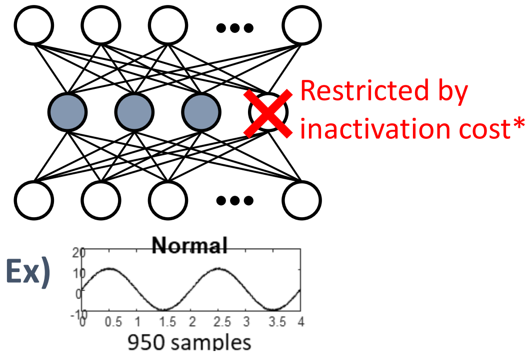


Auto Encoder

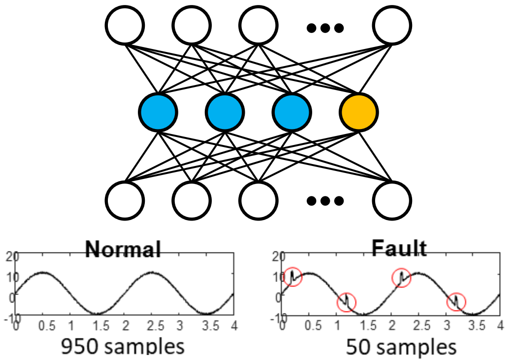


Class-selective Pretraining with Class Imbalanced Data

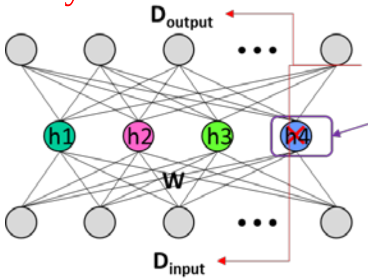
1) Train with major class samples



2) Train with balanced classes



*Restrict by inactivation cost



Reconstruction error: $\mathbf{J1} = \sqrt{(D_{input} - D_{output})^2}$

Inactivation cost: $\mathbf{J2} = \sqrt{(\mathbf{1} - \boldsymbol{h4})^2}$

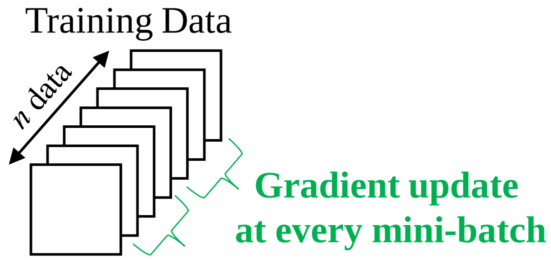
Total cost function: $\mathbf{J} = \mathbf{J1} + \boldsymbol{\alpha} \cdot \boldsymbol{J2}$

Result – Drastic improvement (Accuracy: 57.6% → 95.1%) of diagnosis performance with minor fault state data (5%)

Journal Bearing Fault Diagnosis

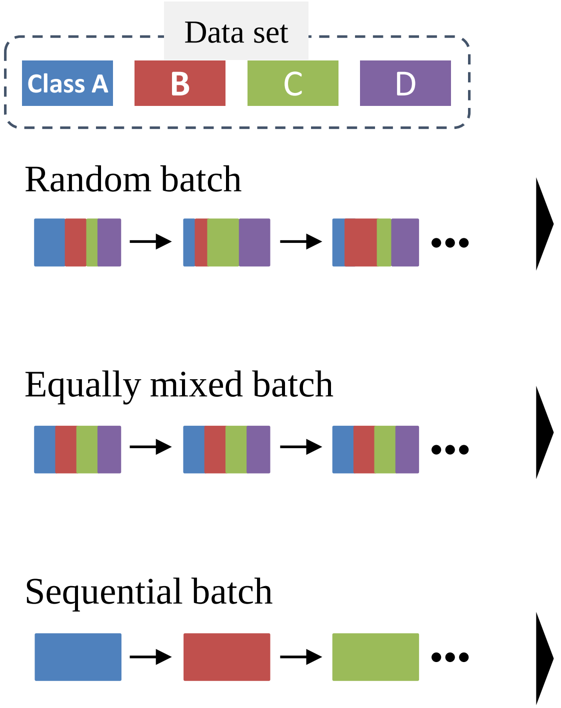
Motivation – To enhance performance and efficiency through batch order optimization in mini-batch learning

Mini-batch Learning



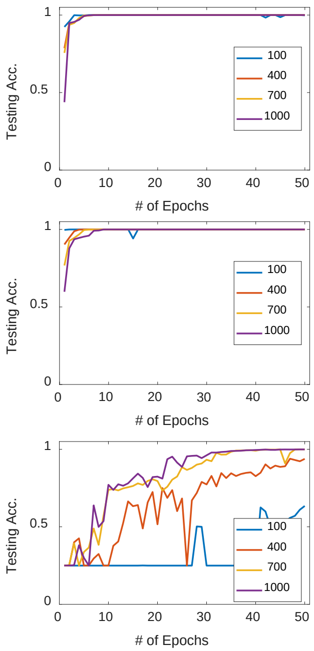
	Batch Size	Computation Cost	Convergence
Stochastic	1	Low	Unstable
Mini-batch	m (<n)	Med.	Med.
Batch	n	High	Stable

Batch Order of Mini-batch



Performance

Prediction Accuracy



Result – Batch optimization in mini-batch training for deep learning

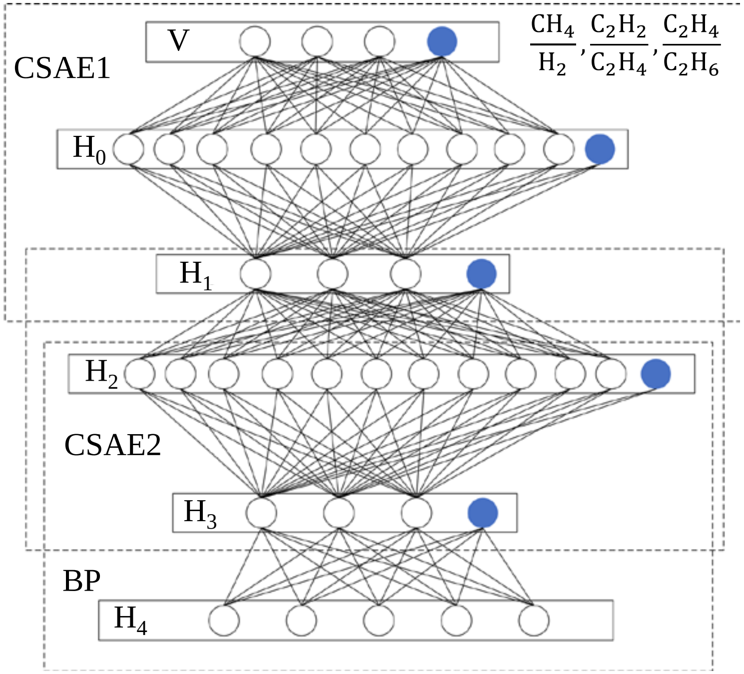
Power Transformer Fault Diagnosis

Motivation – To improve diagnosis accuracy with unsupervised feature learning

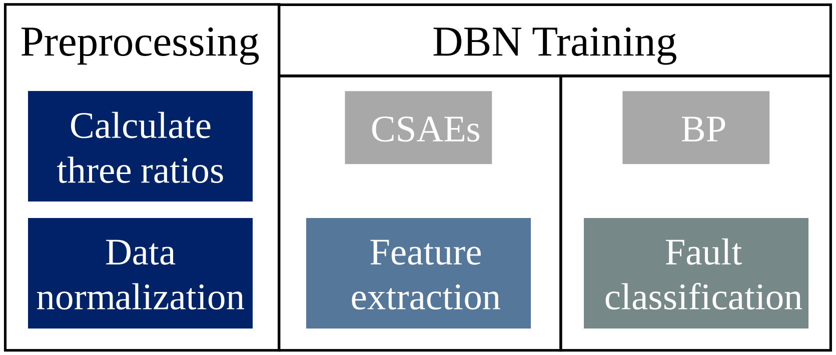
<Network structure>

<Experiment & Accuracy>

CSAE and BP architecture



Flowchart of proposed method



Preprocessing

Calculate three ratios

Data normalization

DBN Training

CSAEs

Feature extraction

BP

Fault classification

Classification accuracy of IEC TC 10 database

	K-NN	RBF-SVM	CSAE	BP
Accuracy (%)	90	79.9	93.6	84.1

Result

- Unsupervised learning three gas ratios of representations
- Better than the traditional algorithm (ex. SVM, K-NN)

Power Transformer Fault Diagnosis

Motivation – Learning noisy labeled data with deep neural network

<Noisy labeled data>

Cat

Tiger

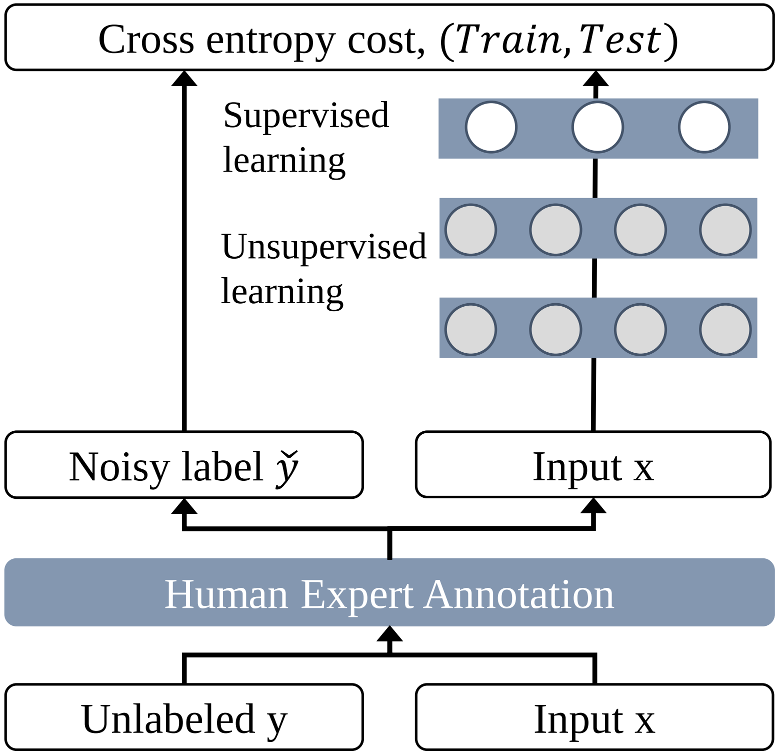
Dog

Wolf

Sea otters

Beaver

<DNN architecture for noisy labeled learning>



Cross entropy cost, $(Train, Test)$

Supervised learning

Unsupervised learning

Noisy label $\check{y}$

Input x

Human Expert Annotation

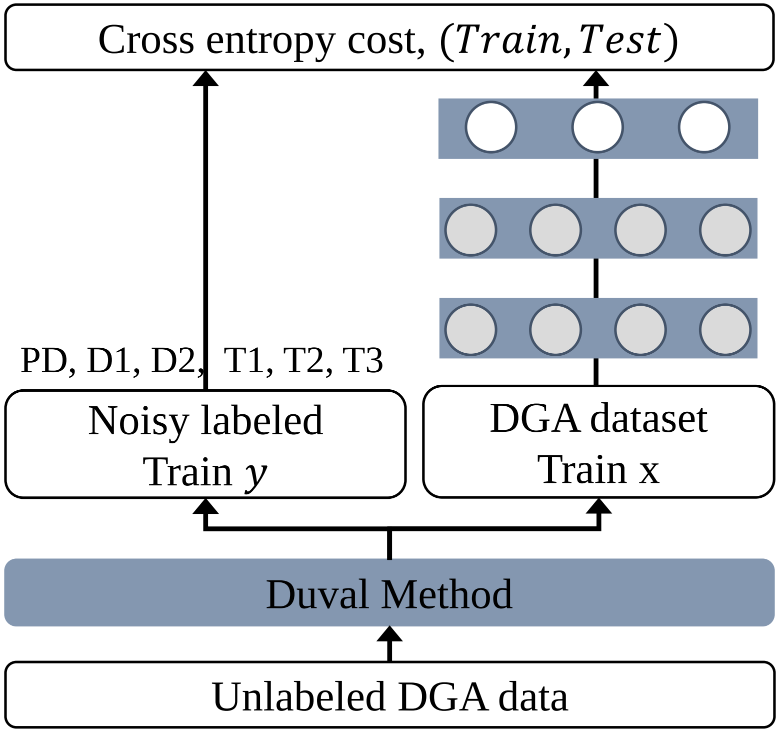
Unlabeled y

Input x

Power Transformer Fault Diagnosis

Motivation – Learning noisy labeled data with deep neural network

<DNN architecture for transformer diagnosis>



Cross entropy cost, $(Train, Test)$

PD, D1, D2, T1, T2, T3

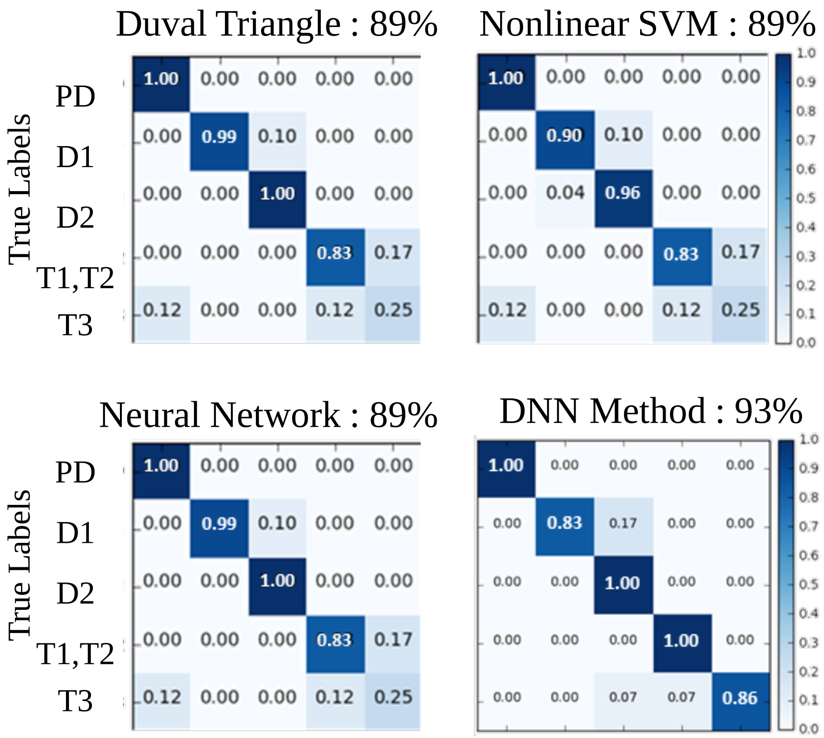
Noisy labeled Train y

DGA dataset Train x

Duval Method

Unlabeled DGA data

<Confusion Matrix of Fault Diagnosis>



Duval Triangle : 89%

PD	1.00	0.00	0.00	0.00	0.00
D1	0.00	0.99	0.10	0.00	0.00
D2	0.00	0.00	1.00	0.00	0.00
T1,T2	0.00	0.00	0.00	0.83	0.17
T3	0.12	0.00	0.00	0.12	0.25

Nonlinear SVM : 89%

PD	1.00	0.00	0.00	0.00	0.00
D1	0.00	0.90	0.10	0.00	0.00
D2	0.00	0.04	0.96	0.00	0.00
T1,T2	0.00	0.00	0.00	0.83	0.17
T3	0.12	0.00	0.00	0.12	0.25

Neural Network : 89%

PD	1.00	0.00	0.00	0.00	0.00
D1	0.00	0.99	0.10	0.00	0.00
D2	0.00	0.00	1.00	0.00	0.00
T1,T2	0.00	0.00	0.00	0.83	0.17
T3	0.12	0.00	0.00	0.12	0.25

DNN Method : 93%

PD	1.00	0.00	0.00	0.00	0.00
D1	0.00	0.83	0.17	0.00	0.00
D2	0.00	0.00	1.00	0.00	0.00
T1,T2	0.00	0.00	0.00	1.00	0.00
T3	0.00	0.00	0.07	0.07	0.86

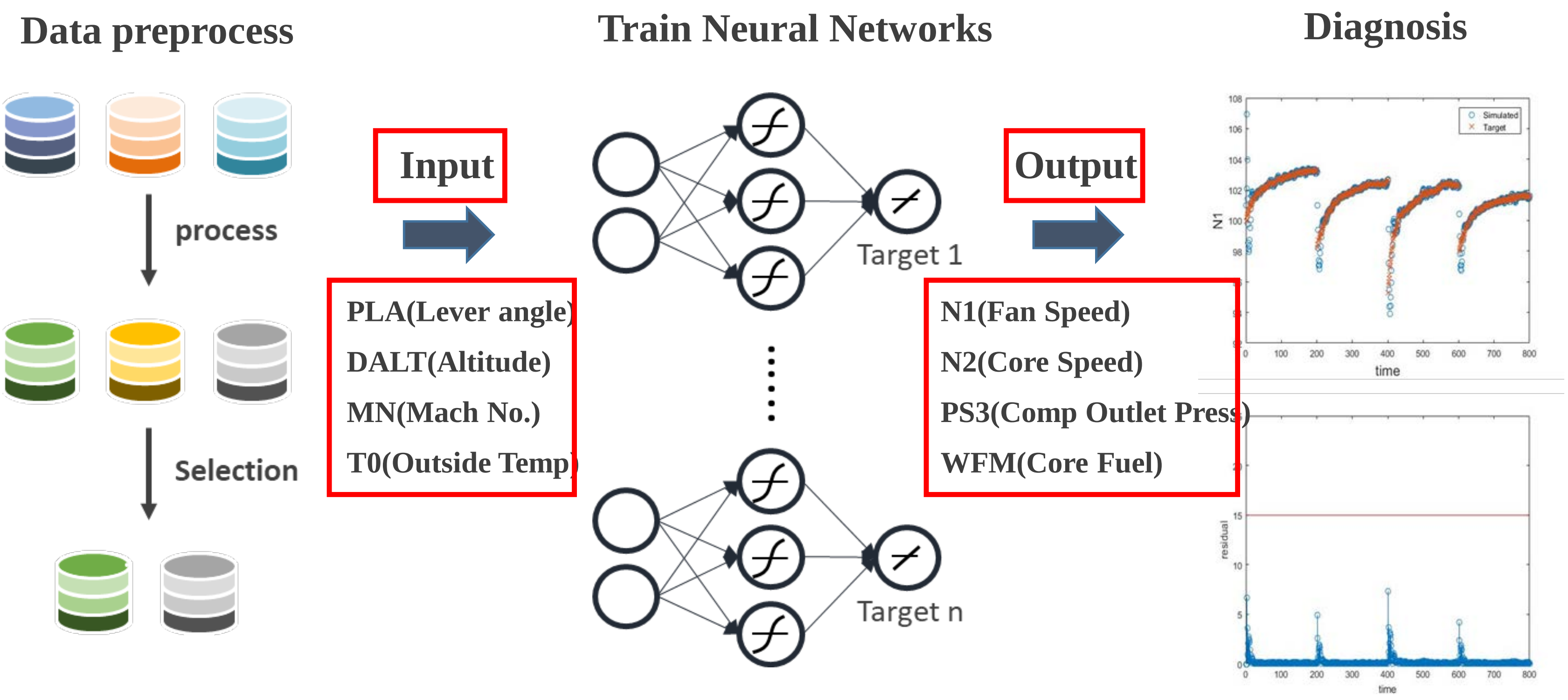
Result – Robust algorithm for fault diagnosis of noisy labeled data

- DNN with noisy labeled data by weak supervision learning
- Abstract representational features to achieve high performance by hierarchical feature learning

Engine Condition Diagnosis

Diagnosis based on artificial neural networks

- Selection of Input / Output parameters for artificial neural networks
  - There are factors affecting engine operation or performance
  - There are internal factors that explain engine behavior
  - Determine Input and Output parameters by two standards



Data preprocess

Train Neural Networks

Diagnosis

Input

PLA(Lever angle)
DALT(Altitude)
MN(Mach No.)
T0(Outside Temp)

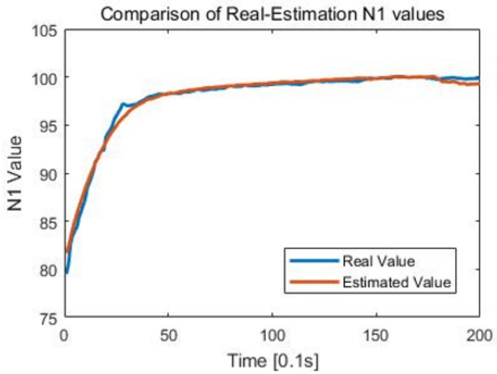
Output

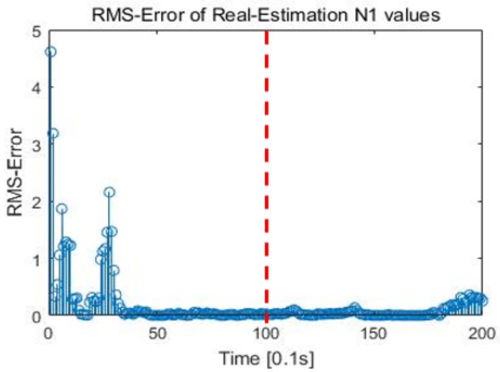
N1(Fan Speed)
N2(Core Speed)
PS3(Comp Outlet Press)
WFM(Core Fuel)

Engine Condition Diagnosis

Health Index Extraction

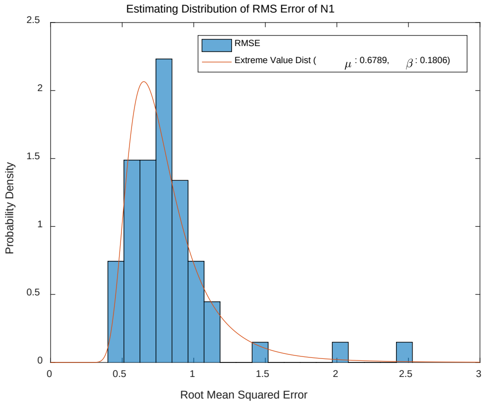
- Use RMSE(Root Mean Squared Error) after 10 seconds as a health index

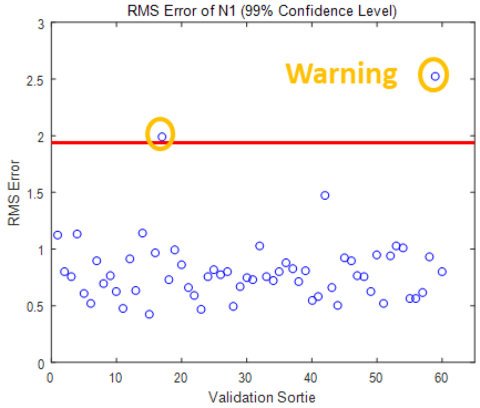




Implementation of condition diagnosis using confidence interval

- RMSE follows Extreme Value Distribution
- Set the 99% confidence level interval





Top 10 AI technology trends for 2018

Deep learning theory

The information bottleneck principle explains how a deep neural

Capsule networks

New type of deep neural network that learns with fewer errors and less data, by preserving key hierarchical relationships.

Deep reinforcement learning

This technique combines reinforcement learning with deep neural networks to learn by interacting with the environment

Generative adversarial networks*

A type of unsupervised deep learning system, implemented as two competing neural networks, enabling machine learning with less human intervention.

Lean and augmented data learning

Different techniques that enable a model to learn from less data or synthetic data.

Probabilistic programming

A high-level language that makes it easy for developers to define probability models.

Hybrid learning model

Approach that combines different types of deep neural networks with probabilistic approaches to model uncertainty.

Automated machine learning

Technique for automating the standard workflow of machine learning.

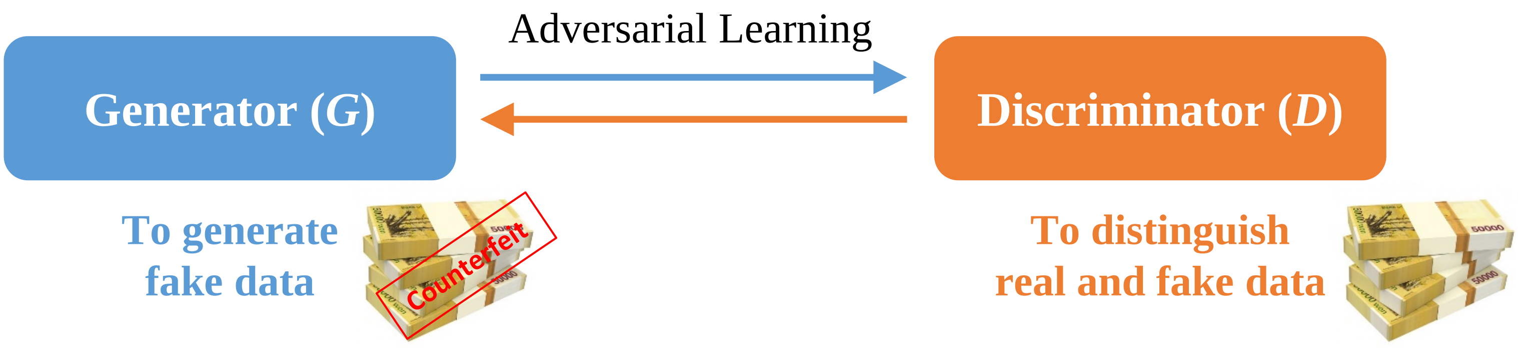
Digital twin

A virtual model used to facilitate detailed analysis and monitoring of physical or psychological systems.

Explainable artificial intelligence

Machine learning techniques that produce more explainable models while maintaining high performance.

Training GAN
(=Training Generator and Discriminator)



Adversarial Learning

Generator (*G*)

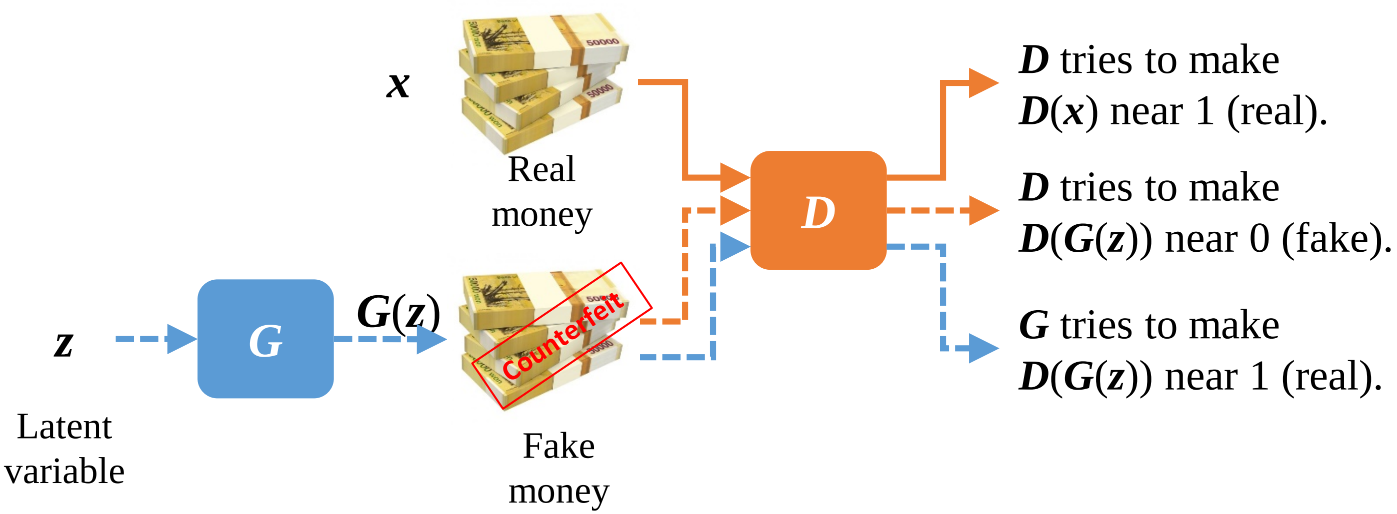
Discriminator (*D*)

To generate fake data

To distinguish real and fake data

Ex) Counterfeiter

Ex) Police



$\boldsymbol{x}$

Real money

$\boldsymbol{z}$

Latent variable

G

$\boldsymbol{G(z)}$

Fake money

D

D tries to make ***D***(***x***) near 1 (real).

D tries to make ***D***(***G***(***z***)) near 0 (fake).

G tries to make ***D***(***G***(***z***)) near 1 (real).

Training GAN

Mathematical Notation

- Discriminator cost function

x: real data (label: 1)
$G(z)$: fake data (label: 0)
p_{data}: Prob. distribution of the samples x
p_g: Prob. distribution of the samples $G(z)$

$\max(D(x))$ $\min(D(G(z)))$

$$J^{(D)} = -\frac{1}{2}\left(E_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(x)] + E_{z\sim p_z}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

- Generator

$$J^{(G)} = -J^{(D)}$$

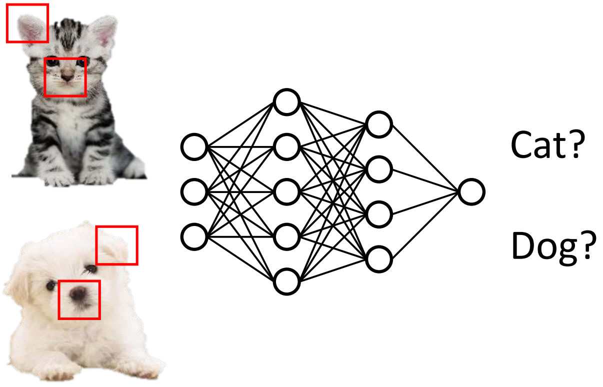
➔ Minimax Game $$\min_G \max_D V(D,G)$$

➔ Formulation (heuristically)

$$J^{(G)} = -\frac{1}{2}E_{z\sim p_z}[\log\left(D\left(G(z)\right)\right)]$$

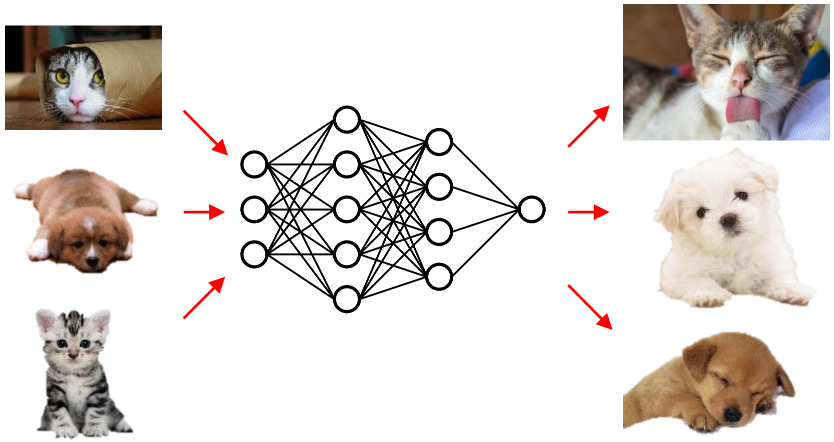
Why is GAN popular?

Conventional Model



Classification by learning features

Generative Model



Understands training data thoroughly

https://www.slideshare.net/carpedm20/pycon-korea-2016

Why is GAN popular?

Vector Arithmetic for Visual Concept

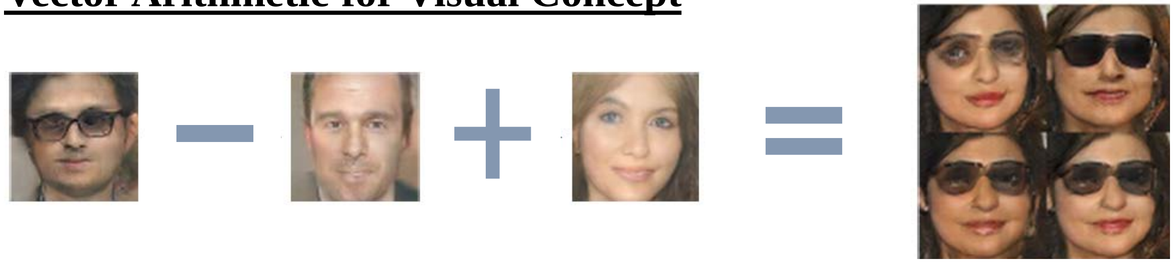
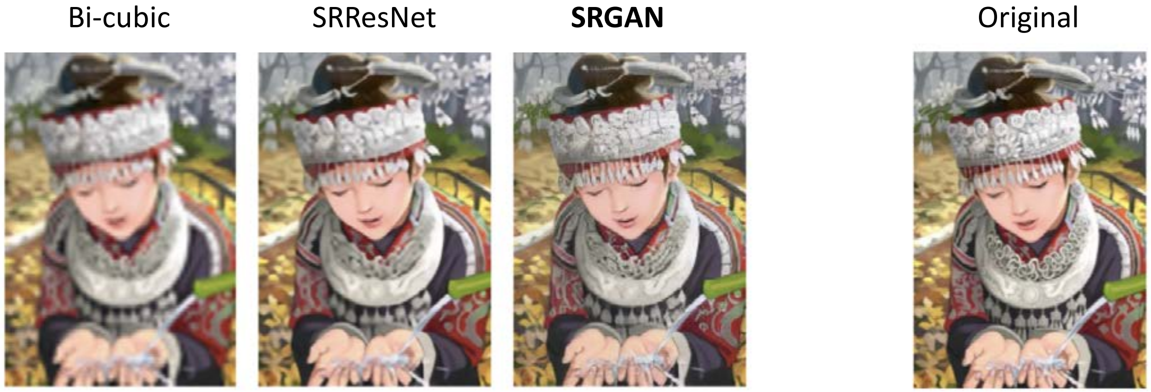
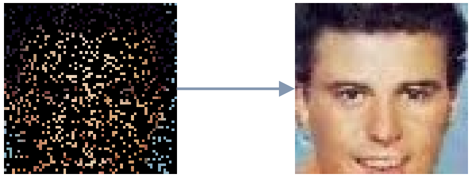


Image Super-Resolution



Semantic Inpainting*



Radford *et al.*, 2015, CVPR; Ledig *et al.*, 2016, CVPR

*refers to algorithms to replace lost parts of the image

GAN in PHM

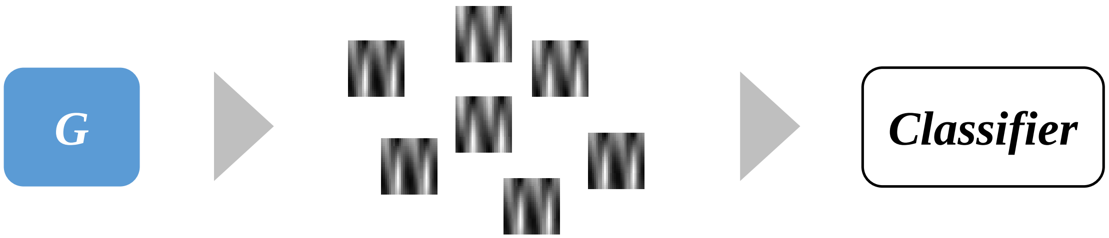
Motivation

Testbed	**Powerplant**
Normal	Normal
Rubbing	Rubbing
Misalignment	Misalignment
Oil Whirl	Oil Whirl
⋮	⋮

- Data not available
- Different scale & characteristics
- **Need to generate data based on testbed data information**

Data generation by GAN (Semi-supervised Learning)

GAN Data Usage in PHM



Generator Trained by GAN

Generated Images

Classifier (trained by real data)

➔ If ***D***(***G***(***z***)) is classified properly, ***G*** generated synthetic labeled data well!

THANK YOU FOR LISTENING