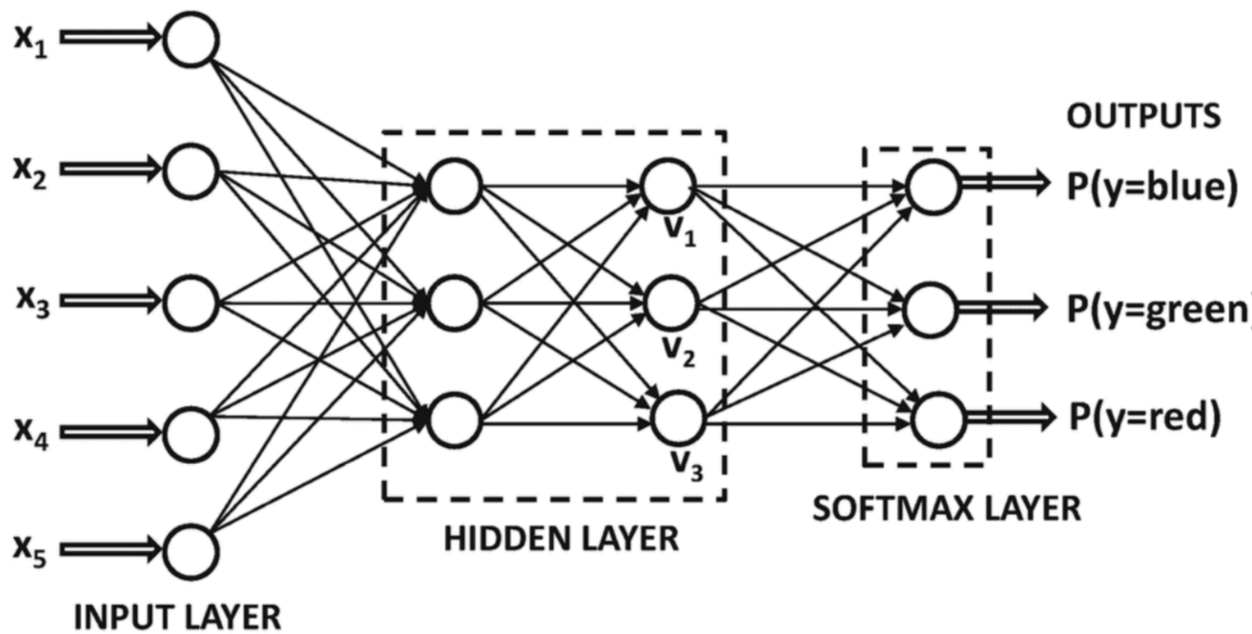
CSCE 636 Neural Networks (Deep Learning)

Lecture 20: Summary

Anxiao (Andrew) Jiang


What a journey it has been

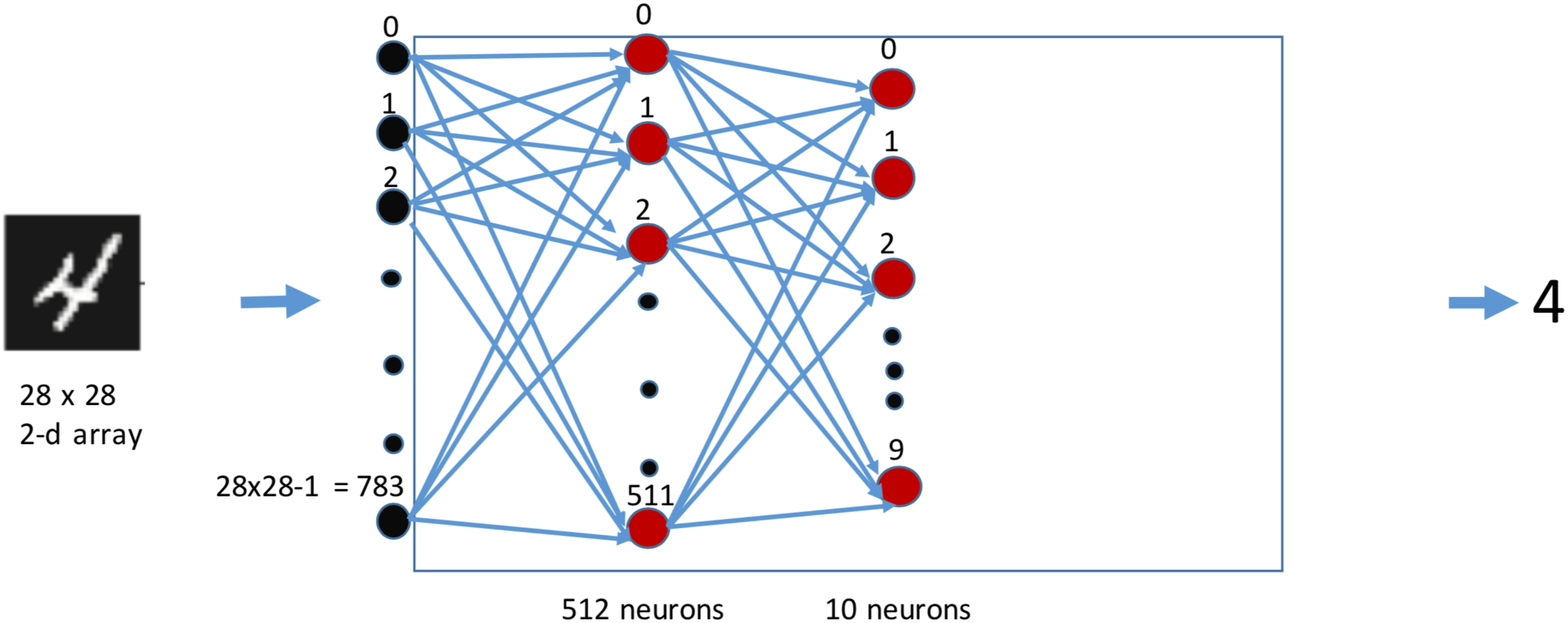
What is a neural network



Step 2: Build neural network architecture

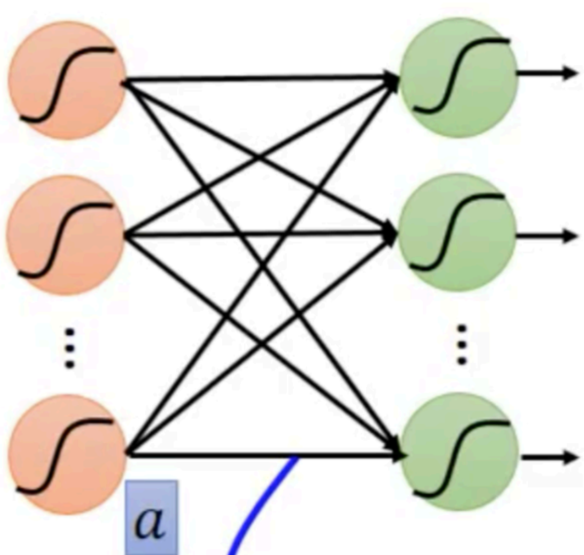
```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```

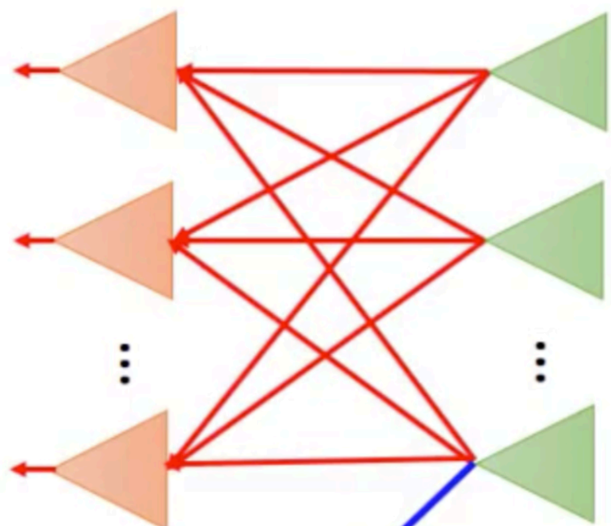


Backpropagation – Summary

Forward Pass



Backward Pass



$$\frac{\partial z}{\partial w} = a \quad \times \quad \frac{\partial C}{\partial z} \quad = \frac{\partial C}{\partial w}$$

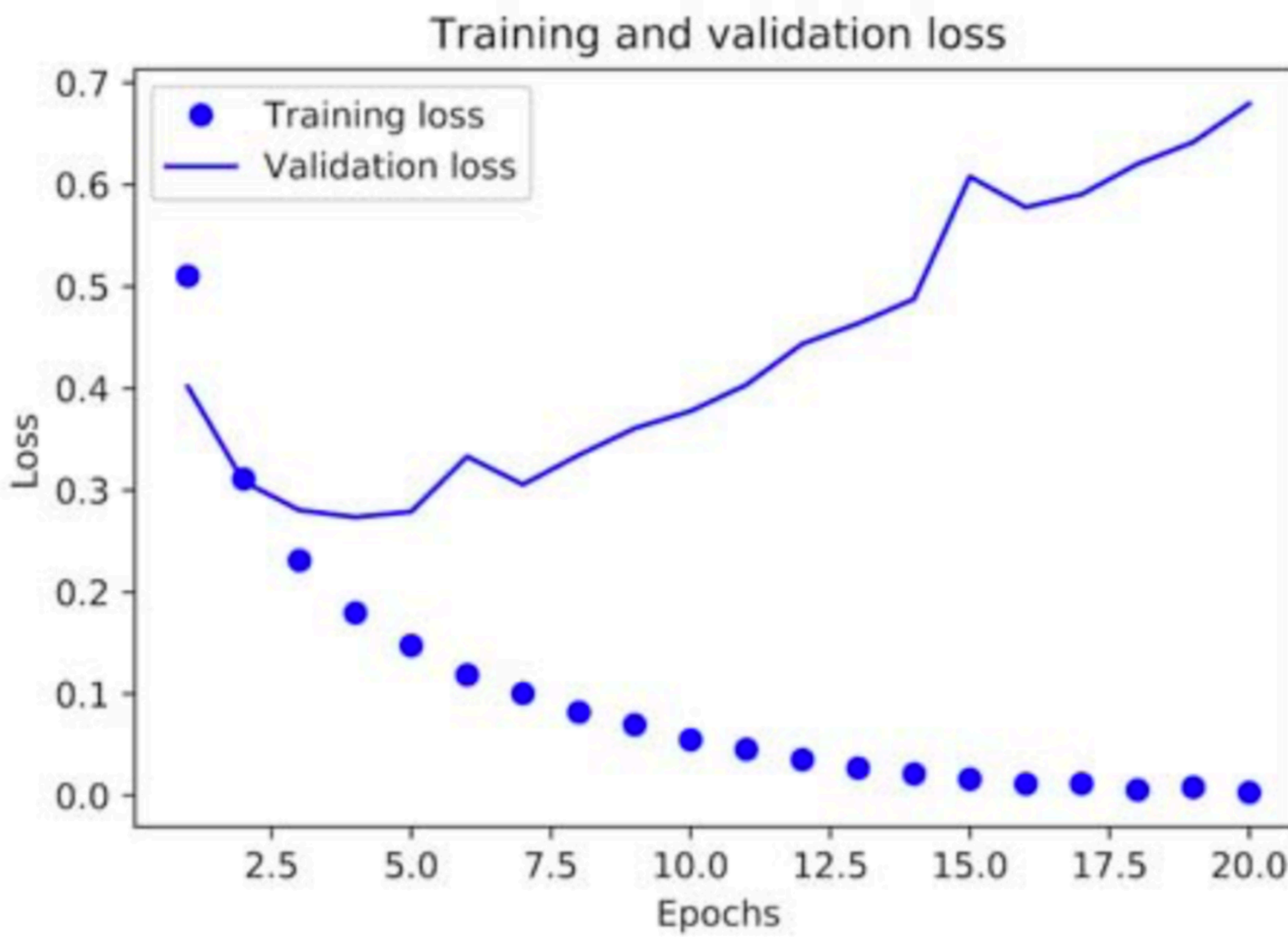
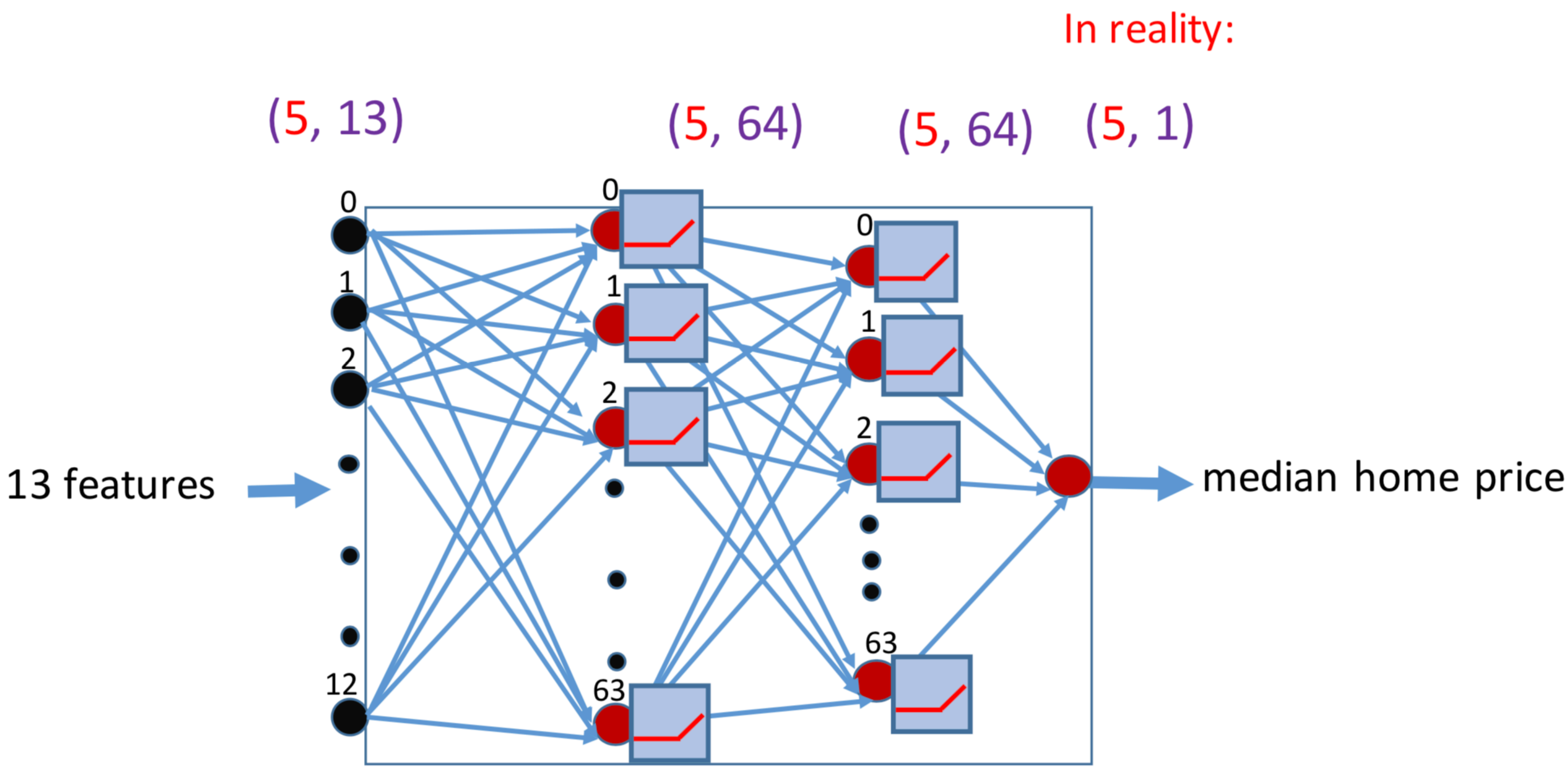


Figure 3.7 Training and validation loss

Say that the mini-batch size is 5 during training.
What is the shape of data in each layer?



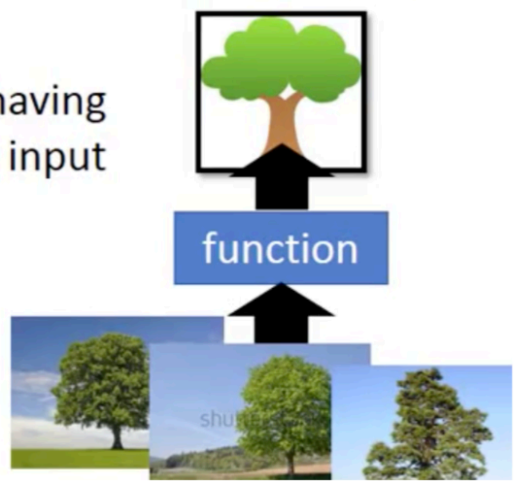
Supervised Learning

- Input and output are both known. Just learn the function.
- The four applications introduced so far in our class are all supervised learning.

Unsupervised Learning

- Output is unknown. Learn the relationship between data.
  - Dimension Reduction

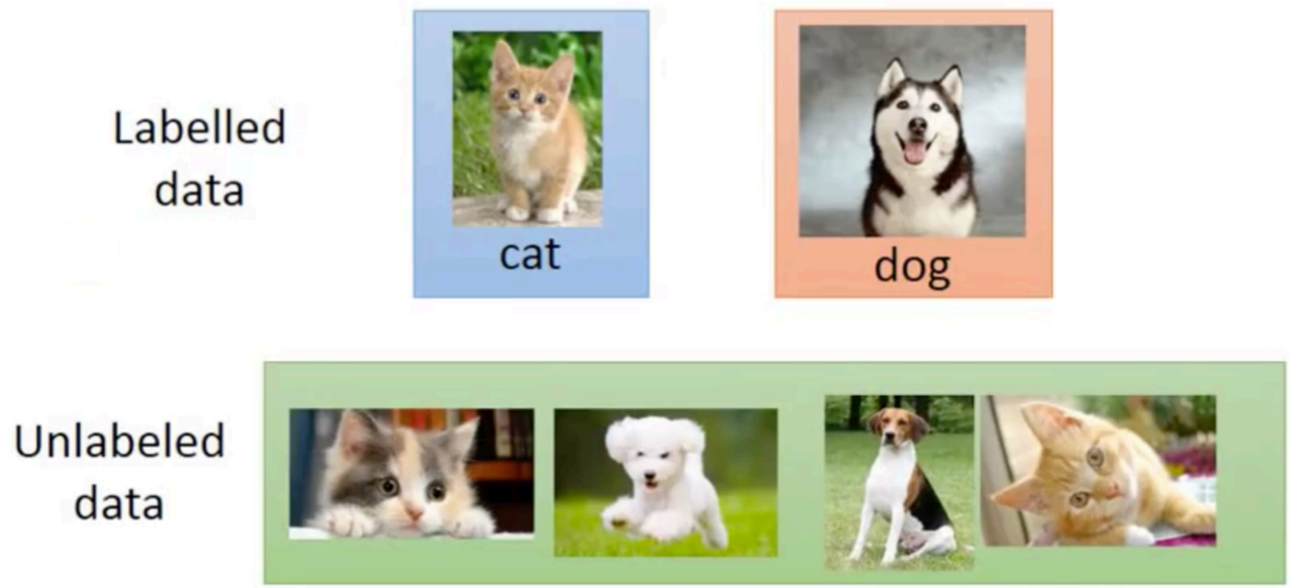
only having function input



Clustering

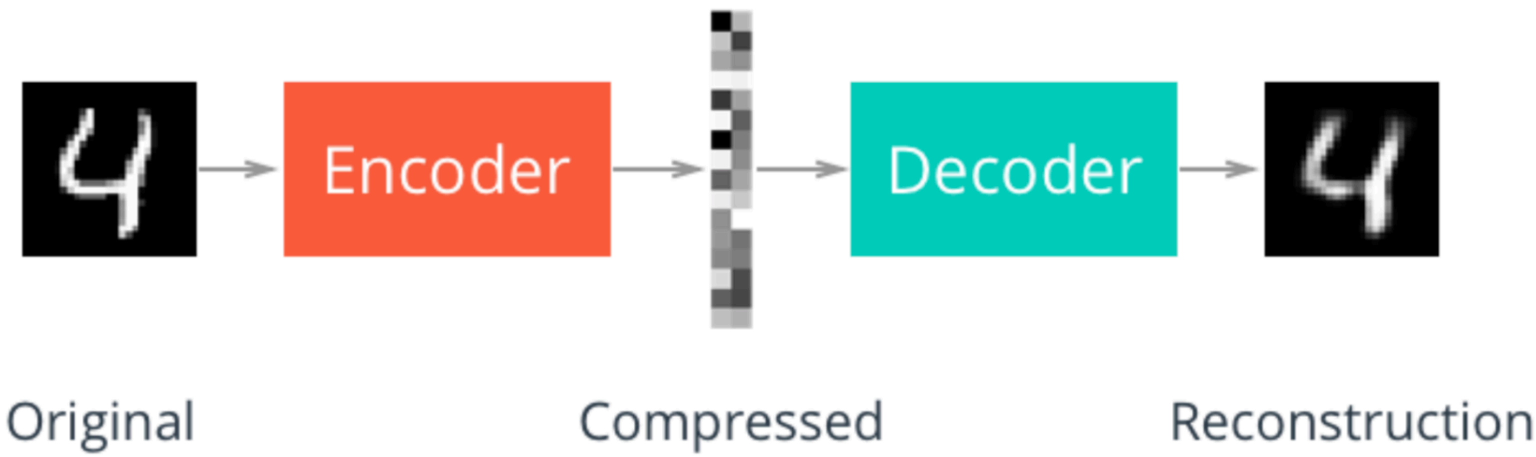
Semi-supervised Learning

- Some outputs are known, but not all. (Most data are unlabeled.)



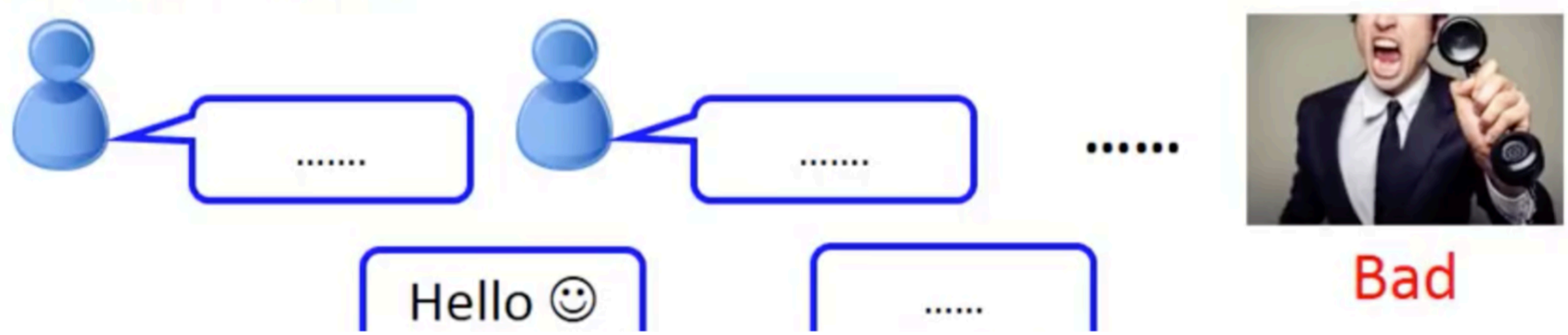
Self-supervised Learning

- Output is generated from input data, without human help.
- Example: auto-encoder



Reinforcement Learning

- Learn from feedback (penalty or reward) from environment.
- But the environment does not tell what to do.



Regularization techniques

- Weight regularization: add a function of weights to the loss function, to prevent the weights from becoming too large.

L2 regularization $$\text{new loss function} = \text{old loss function} + \lambda \sum_i w_i^2$$

L1 regularization $$\text{new loss function} = \text{old loss function} + \lambda \sum_i |w_i|$$

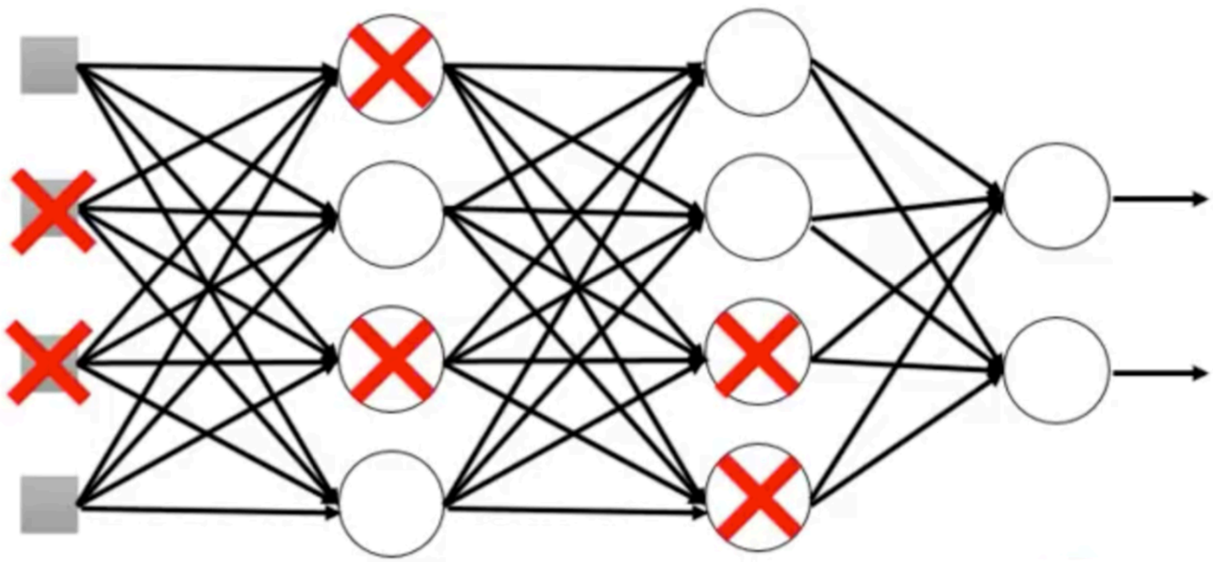
A reason for weight regularization: large weight can make the model more sensitive to noise/variance in data.

L2 regularization: it tends to make all weights small.

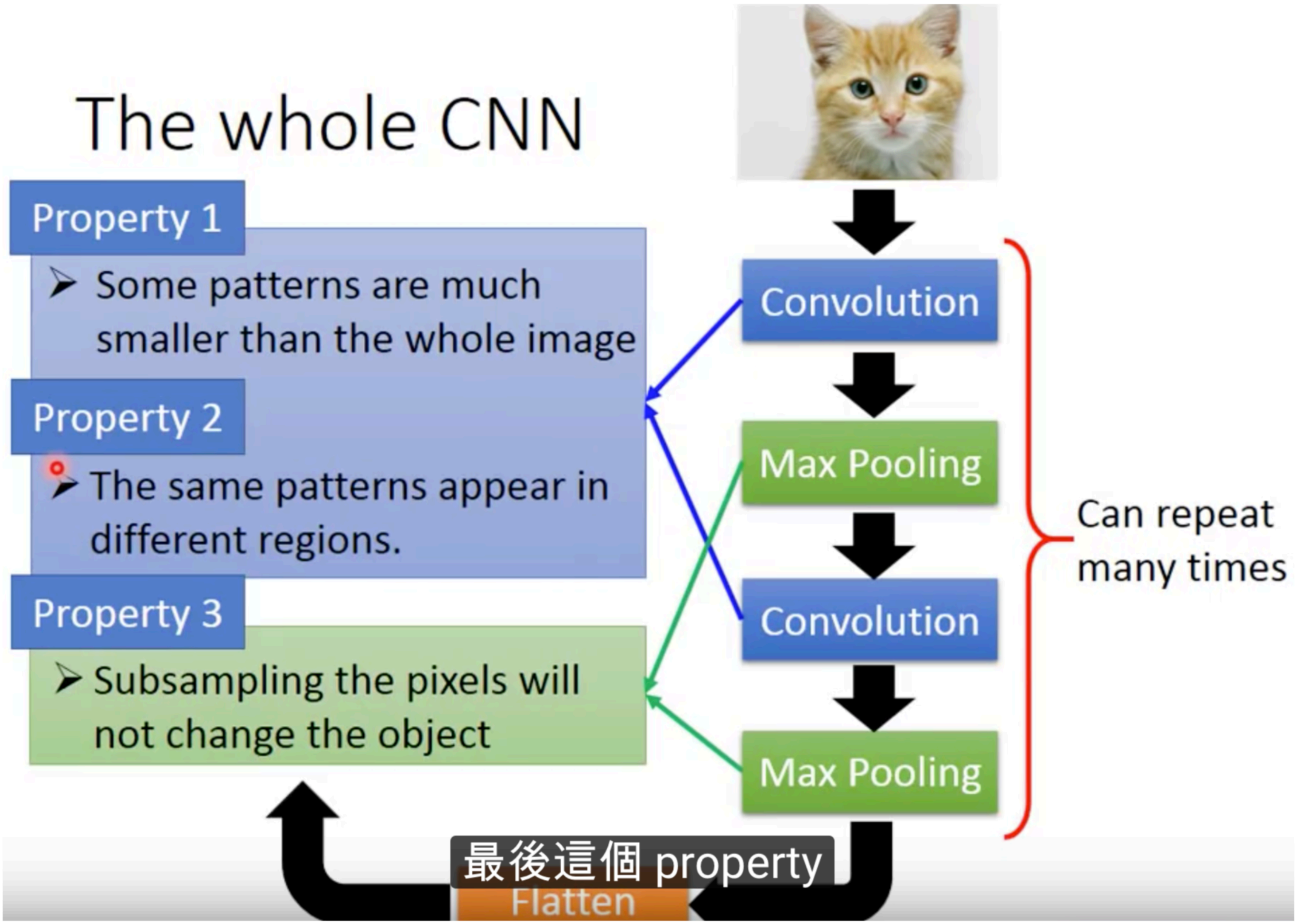
L1 regularization: it tends to make weights sparser (namely, more 0s).

Dropout

Training:

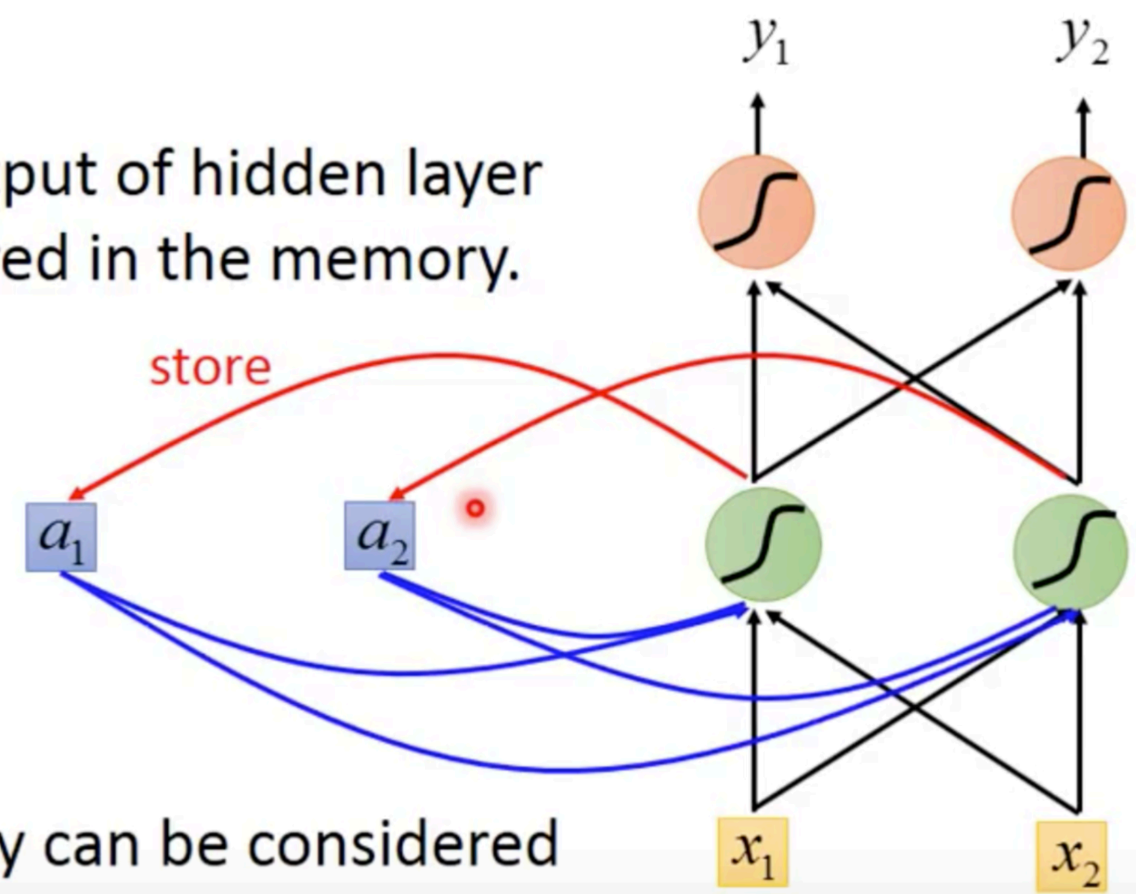


- **Each time before updating the parameters**
  - Each neuron has p% to dropout


The whole CNN
Property 1
Some patterns are much smaller than the whole image
Property 2
The same patterns appear in different regions.
Property 3
Subsampling the pixels will not change the object
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten
最後這個 property

Recurrent Neural Network (RNN)

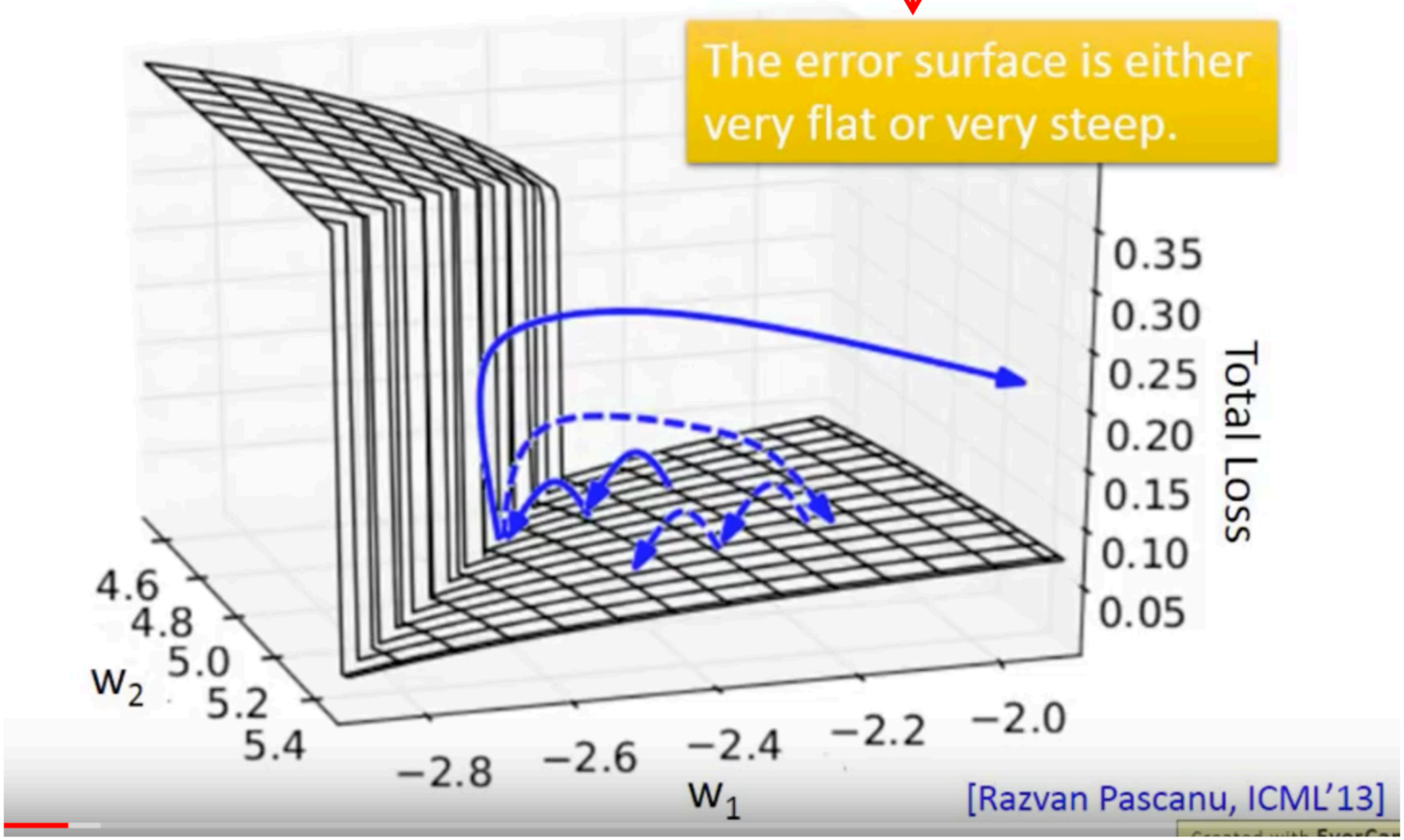
The output of hidden layer are stored in the memory.



Memory can be considered as another input.

The error surface is rough.

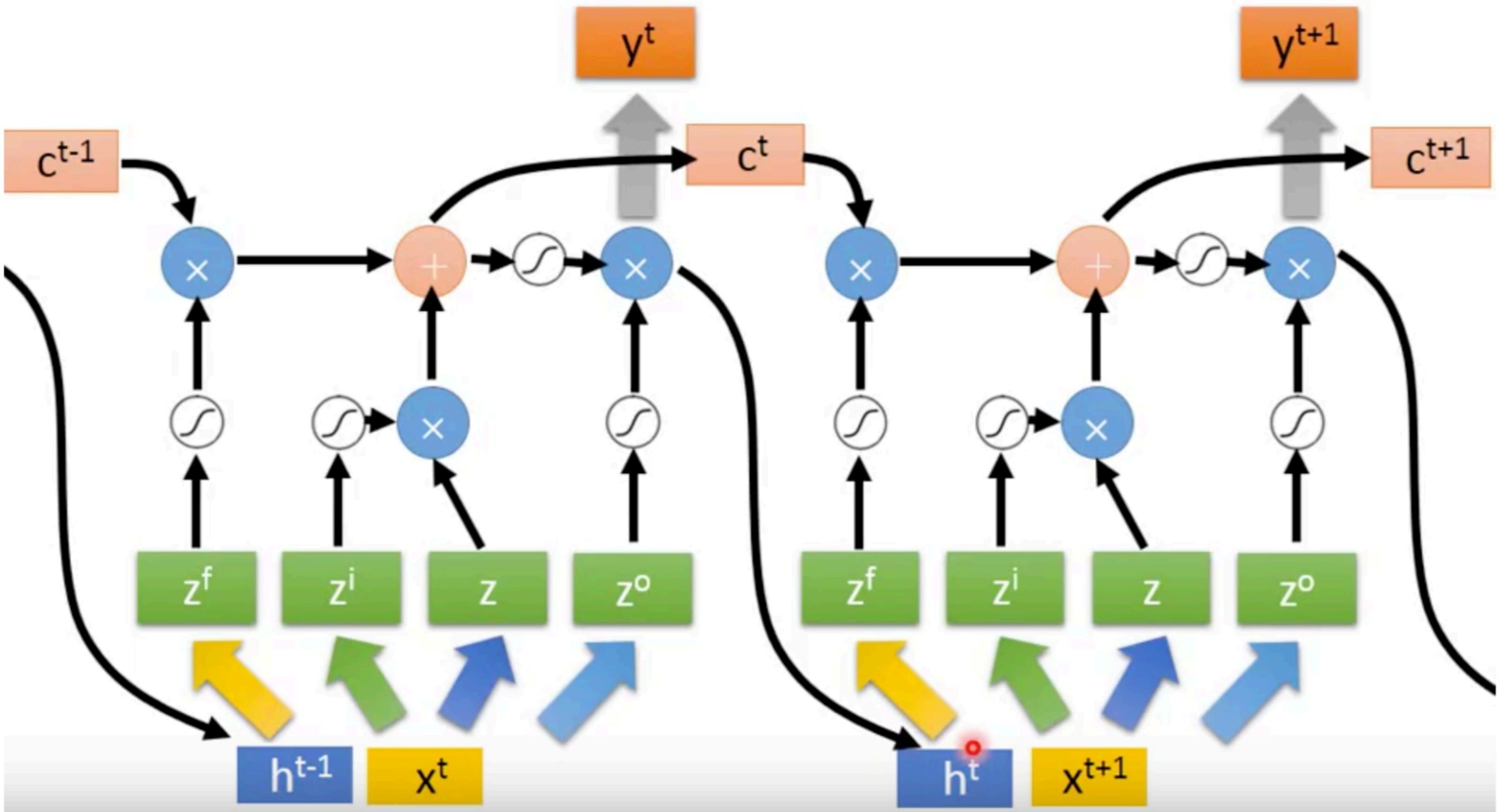
The error surface is either very flat or very steep.



[Razvan Pascanu, ICML'13]

LSTM

Real LSTM

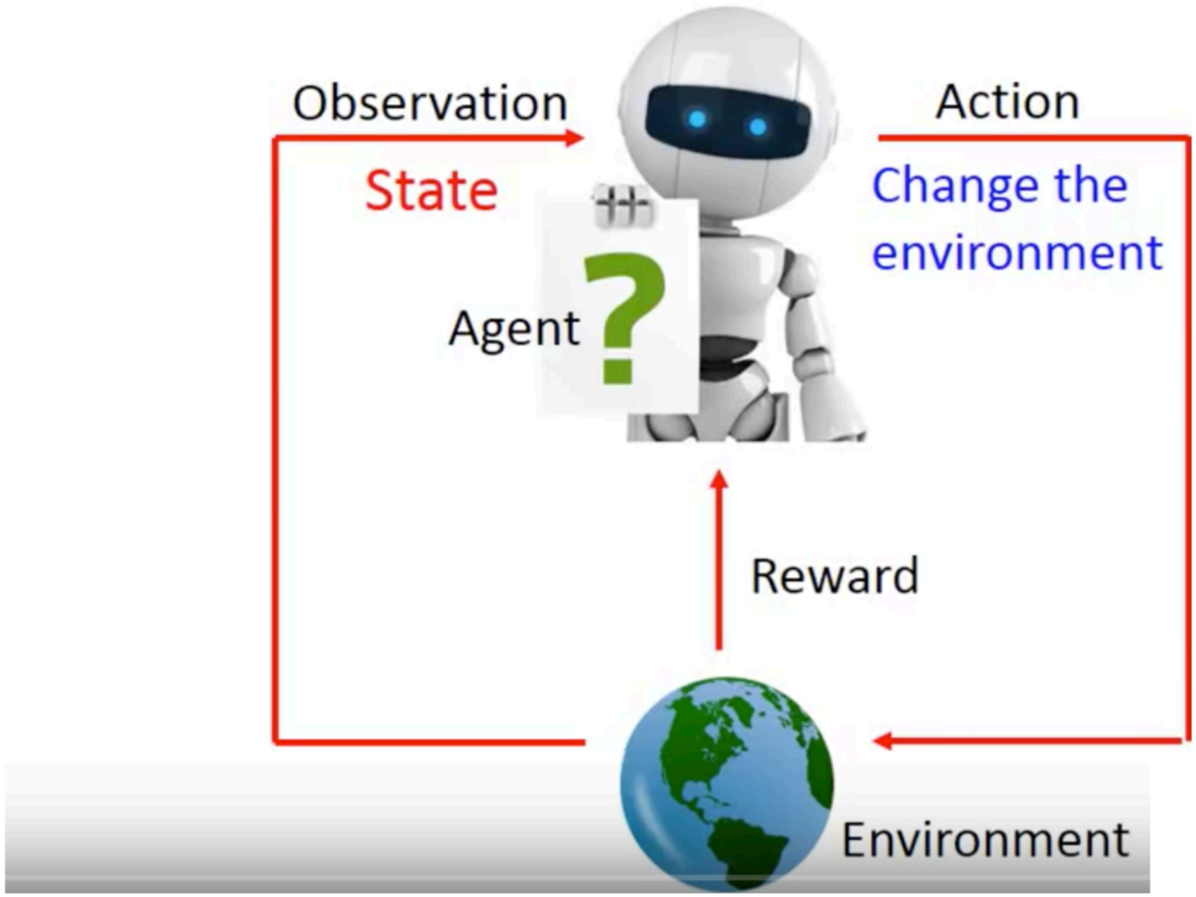


Keras Functional API

Directed acyclic graphs of layers

Inspecting and monitoring deep-learning models using Keras callbacks and TensorBoard

Scenario of Reinforcement Learning



Policy-based Approach

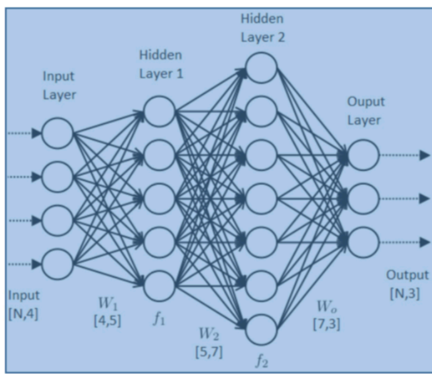
$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n,\theta)$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\underline{\nabla \log p(a_t^n|s_t^n,\theta)}$$

negative Cross entropy

Target output

hurt leg



Actor

train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

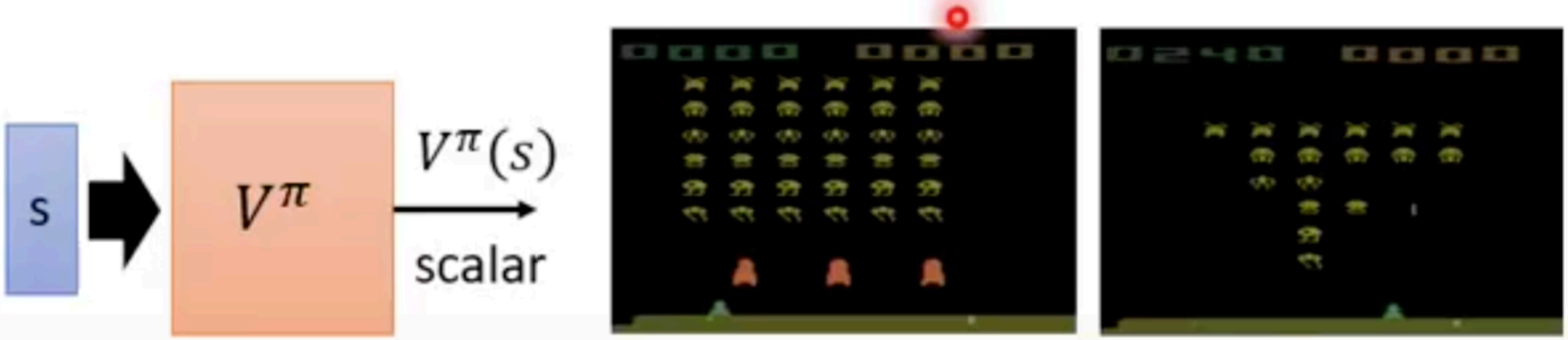
From on-policy to off-policy

Using the experience more than once

Q-Learning

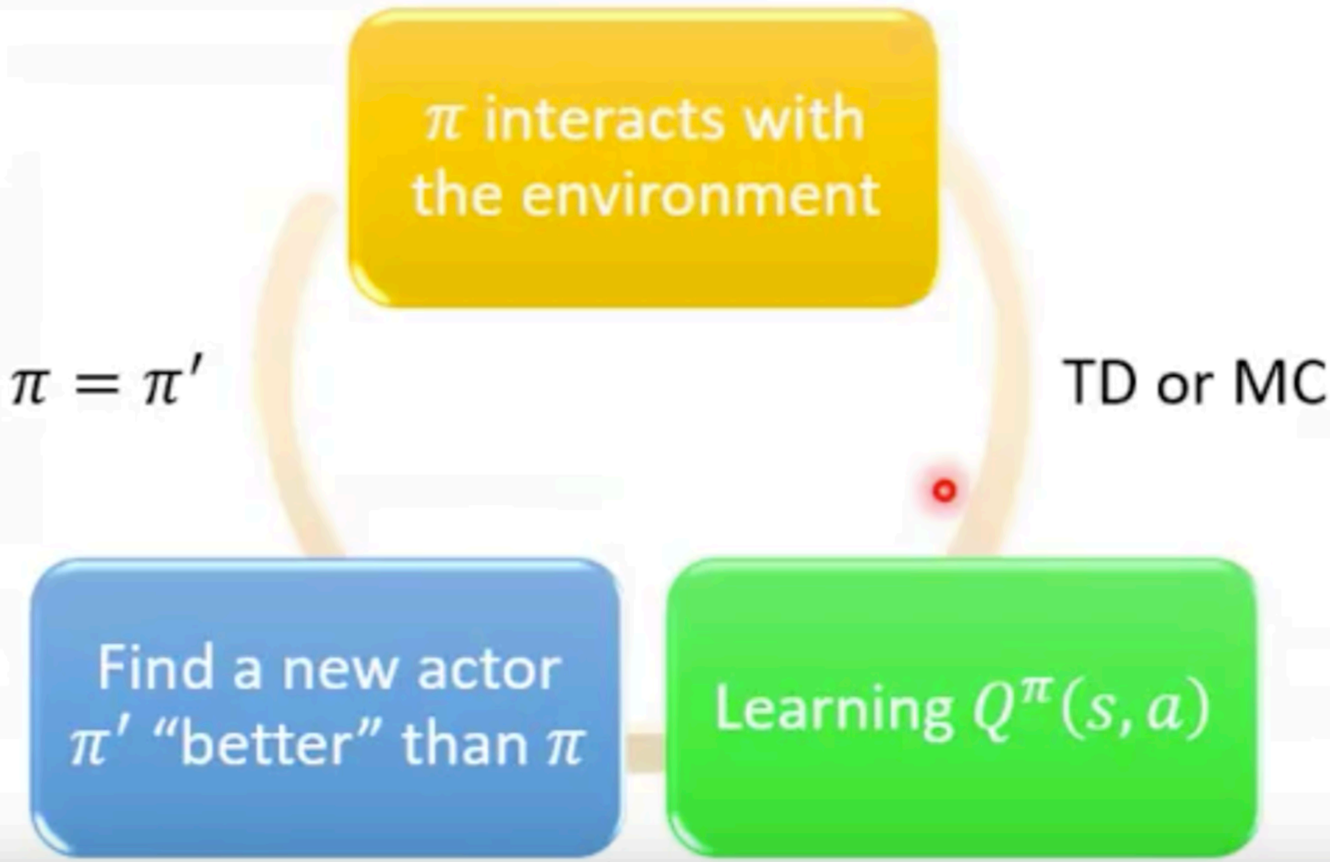
Critic

- A critic does not directly determine the action.
- Given an actor π, it evaluates how good the actor is
- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after visiting state s



$V^{\pi}(s)$ is large $\quad$ $V^{\pi}(s)$ is smaller

Another Way to use Critic: Q-Learning



Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", ICML, 2016

Actor-Critic

$$Q^{\pi_\theta}(s_t^n, a_t^n) - V^{\pi_\theta}(s_t^n)$$

$$V^{\pi_\theta}(s_t^n)$$

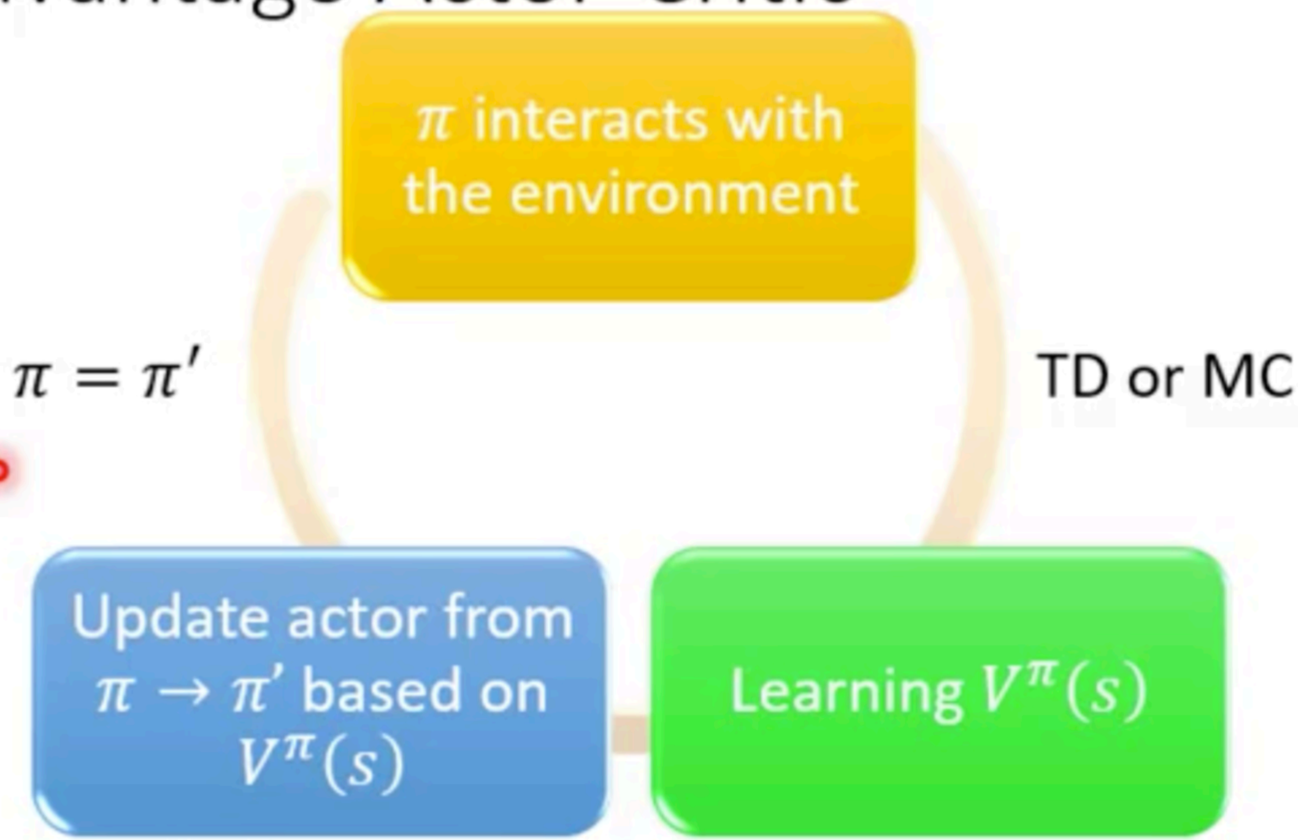
baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\boxed{\left(\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n - b\right)}\nabla log p_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

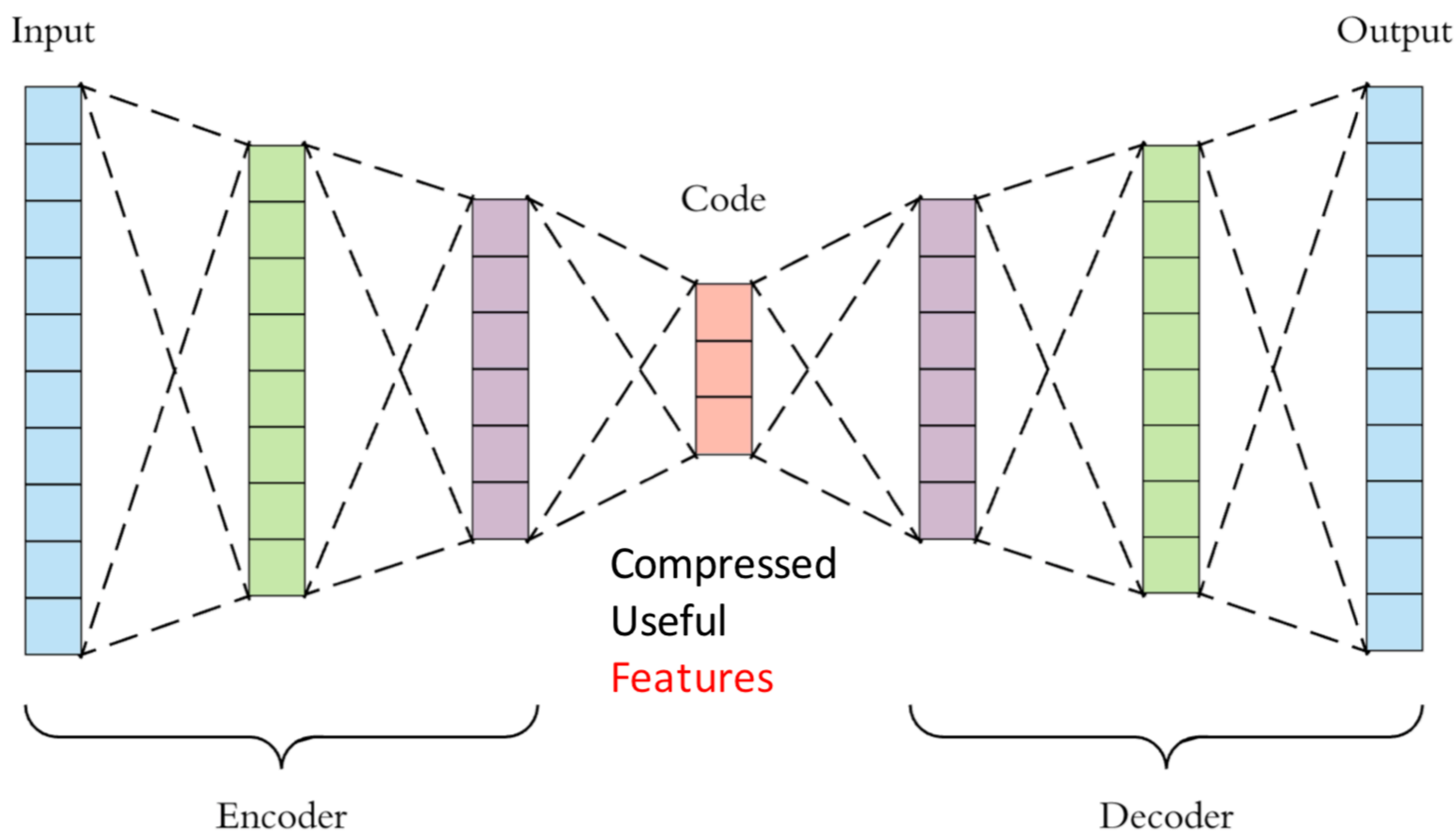
$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Advantage Actor-Critic

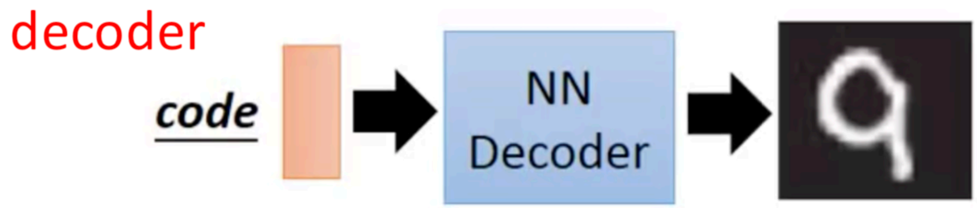


$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(r_t^n + V^{\pi}(s_{t+1}^n) - V^{\pi}(s_t^n) \right) \nabla log p_{\theta}(a_t^n | s_t^n)$$

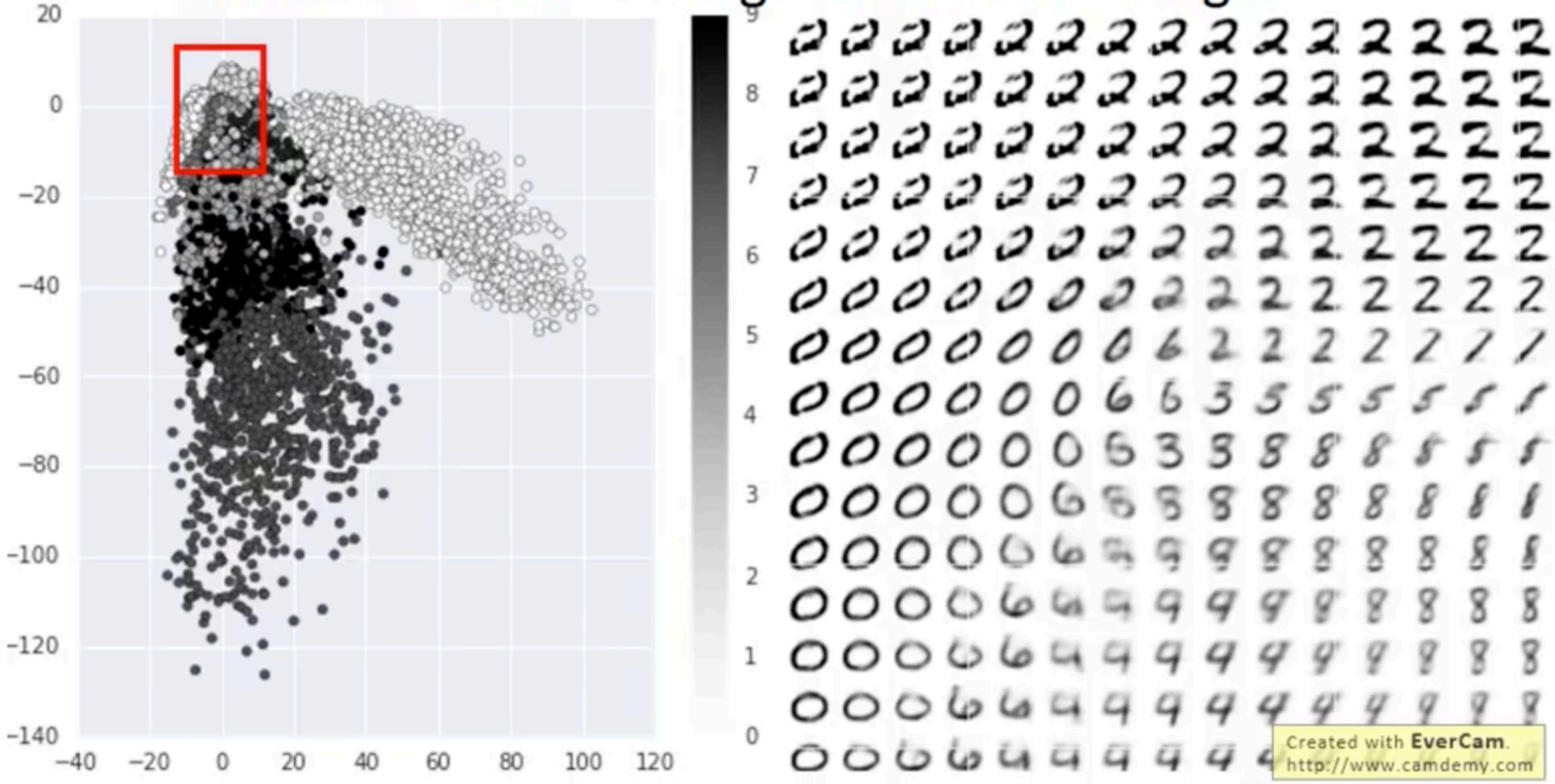
Auto-Encoder



Next .....



- Can we use decoder to generate something?



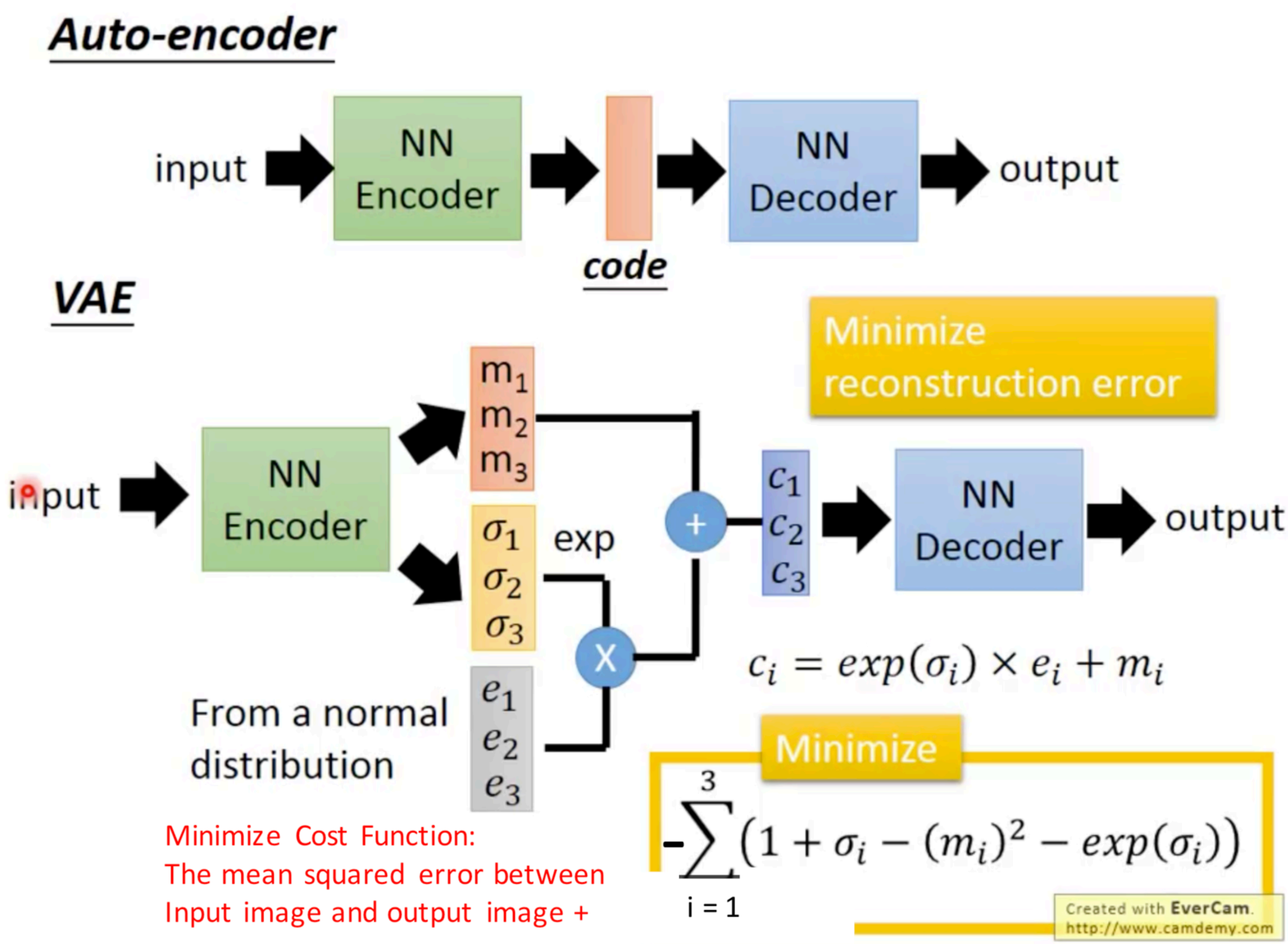
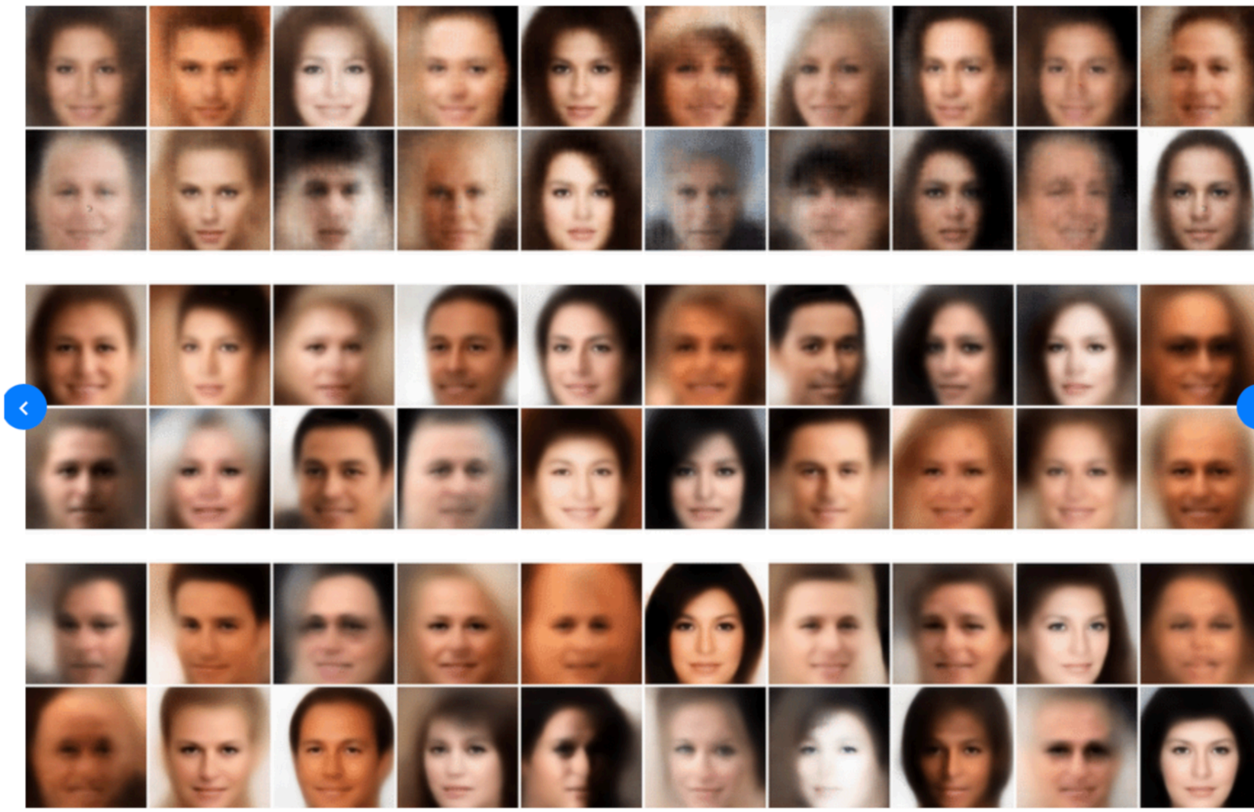

Auto-encoder
input
NN Encoder
code
NN Decoder
output
VAE
input
NN Encoder
m_1 m_2 m_3
σ_1 σ_2 σ_3
exp
e_1 e_2 e_3
From a normal distribution
c_1 c_2 c_3
NN Decoder
output
Minimize reconstruction error
$c_i = exp(\sigma_i) \times e_i + m_i$
Minimize
$-\sum_{i=1}^{3}(1 + \sigma_i - (m_i)^2 - exp(\sigma_i))$
Minimize Cost Function:
The mean squared error between
Input image and output image +
Created with EverCam.
http://www.camdemy.com

Figure 8 - uploaded by Hongyang Gao

Download

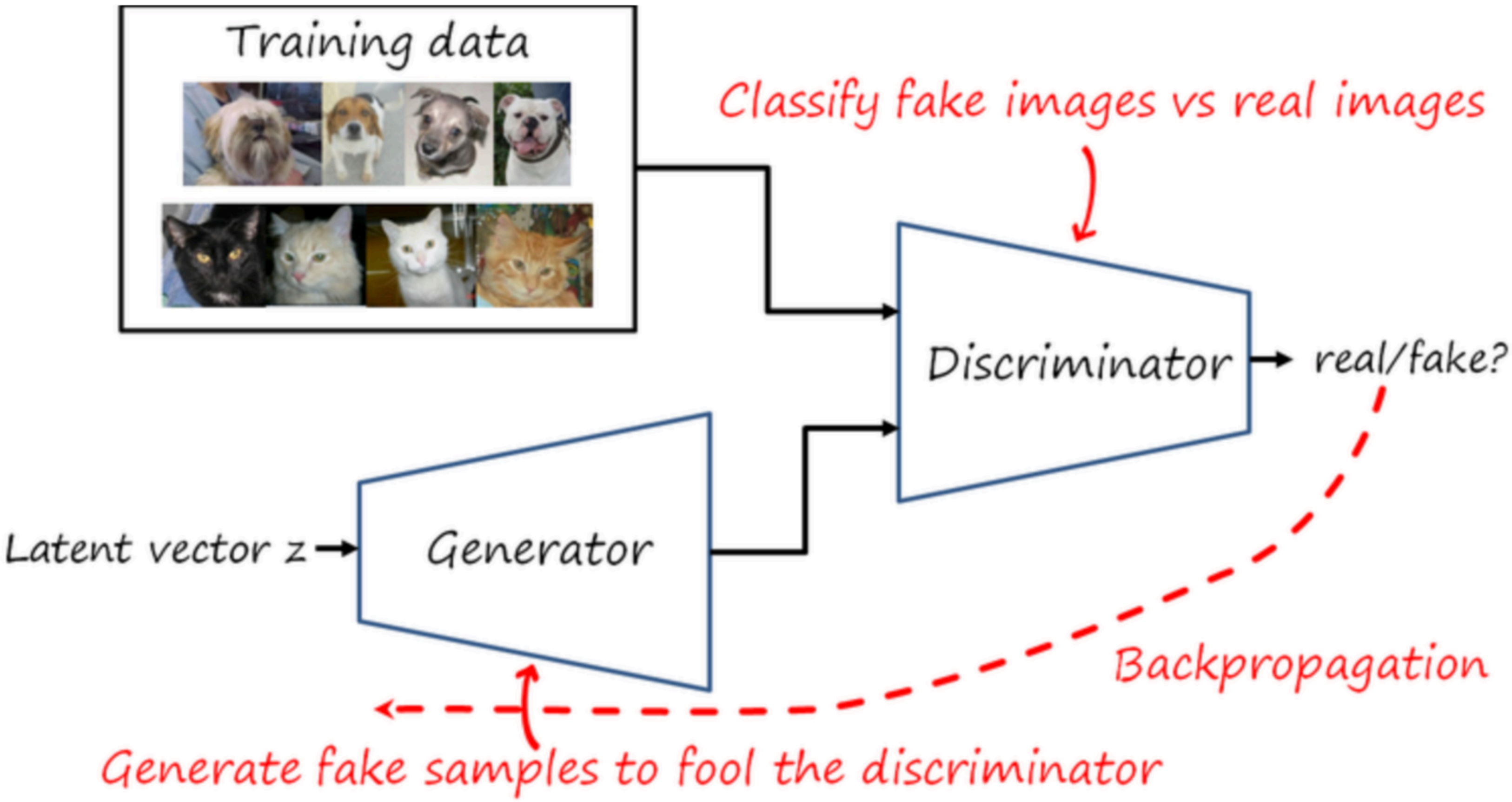
View publication



Sample images generated by different models when trained on the CelebA dataset. The first two rows are images generated by a standard VAE. The middle two rows are images generated by deep residual VAE. The last two rows are images generated by multi-stage VAE.

https://www.researchgate.net/figure/Sample-images-generated-by-different-models-when-trained-on-the-CelebA-dataset-The-first_fig5_317062169

GAN


Training data
Classify fake images vs real images
Discriminator
real/fake?
Latent vector z
Generator
Backpropagation
Generate fake samples to fool the discriminator

Images generated using Progressive GAN

(a) DeepFashion

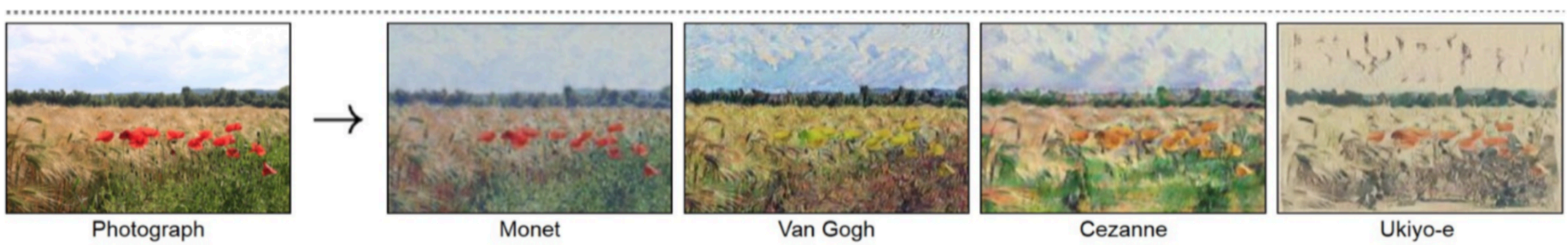
(b) Market-1501

(c) Generating from a sequence of poses

Pose Guided Person Image Generation

CycleGAN

Cross-domain transfer GANs will be likely the first batch of commercial applications. These GANs transform images from one domain (say real scenery) to another domain (Monet paintings or Van Gogh).



CycleGAN

Transfer Learning

Transfer Learning - Overview

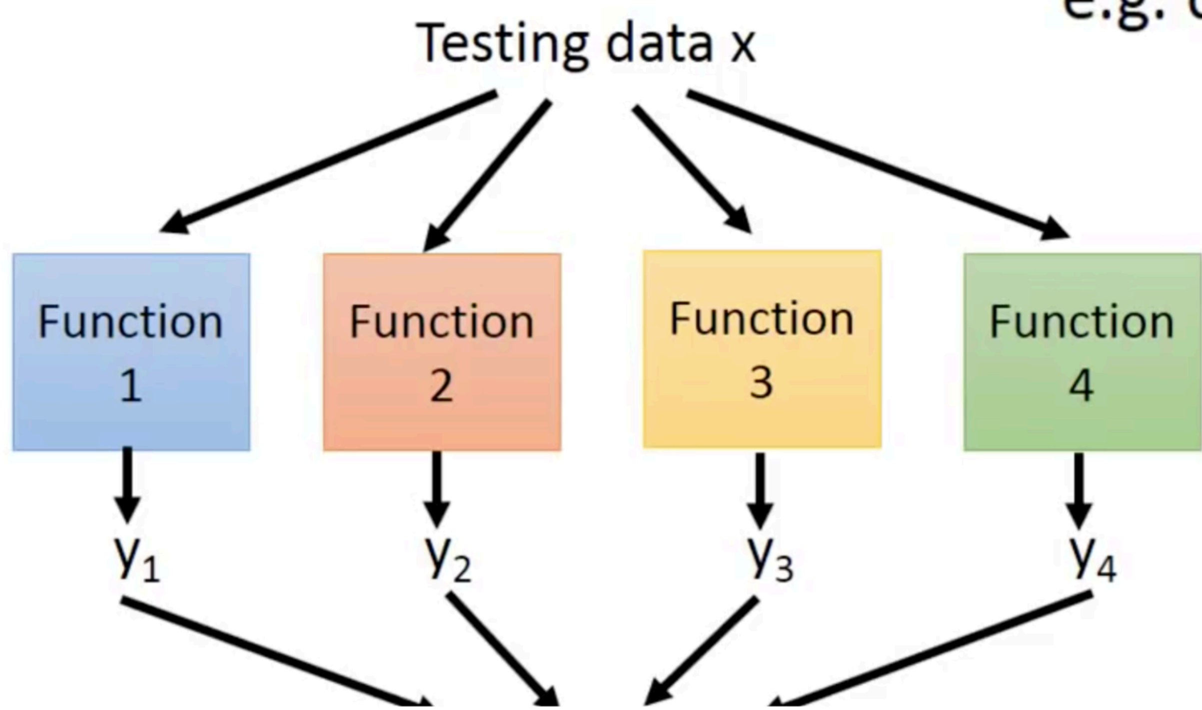
		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning Multitask Learning	Self-taught learning Rajat Raina , Alexis Battle , Honglak Lee , Benjamin Packer , Andrew Y. Ng, Self-taught learning: transfer learning from unlabeled data, ICML, 2007
	unlabeled	Domain-adversarial training Zero-shot learning	Self-taught Clustering Wenyuan Dai, Qiang Yang, Gui-Rong Xue, Yong Yu, "Self-taught clustering", ICML 2008

Ensemble Learning

Bagging

This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree



Use average or voting to get a final result

Ensemble: Boosting

Improving Weak Classifiers

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Algorithm for AdaBoost

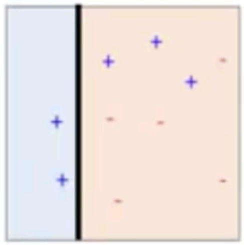
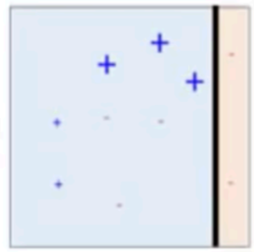
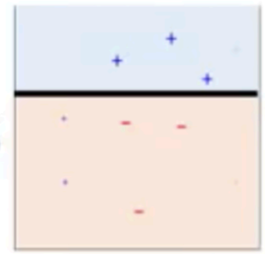
- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

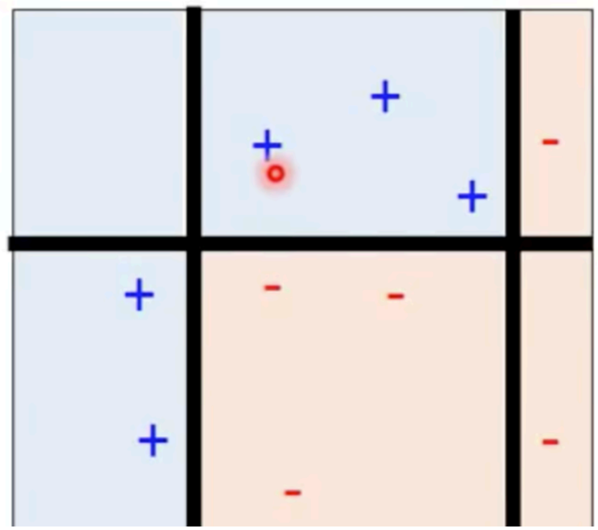
$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Toy Example

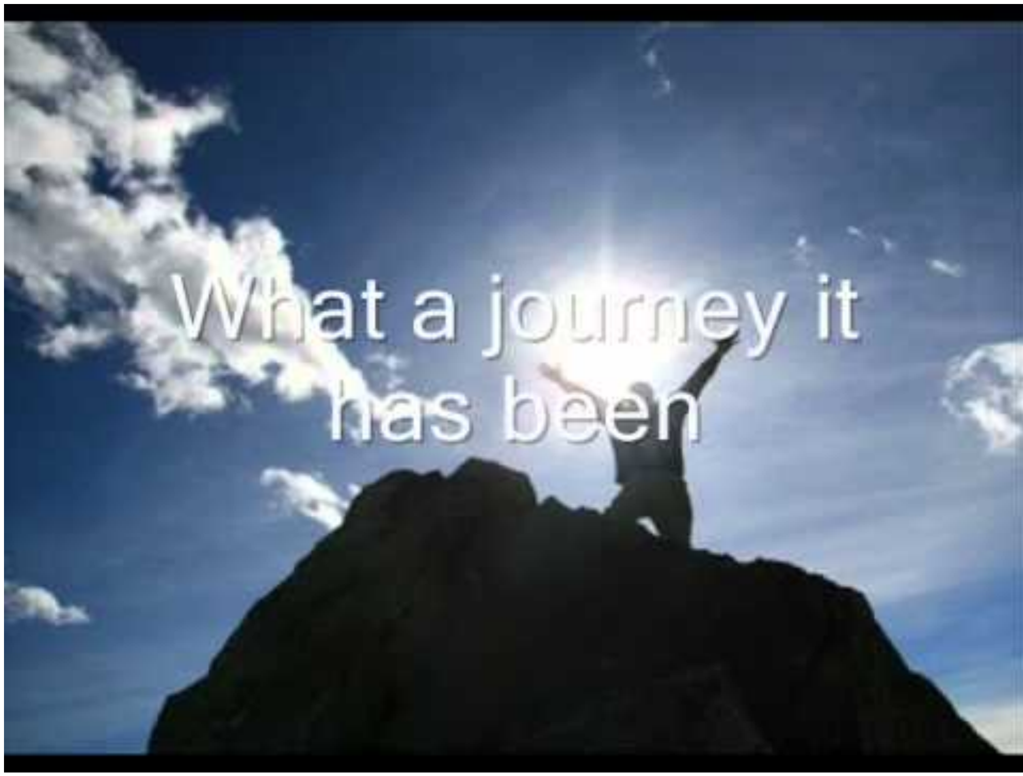
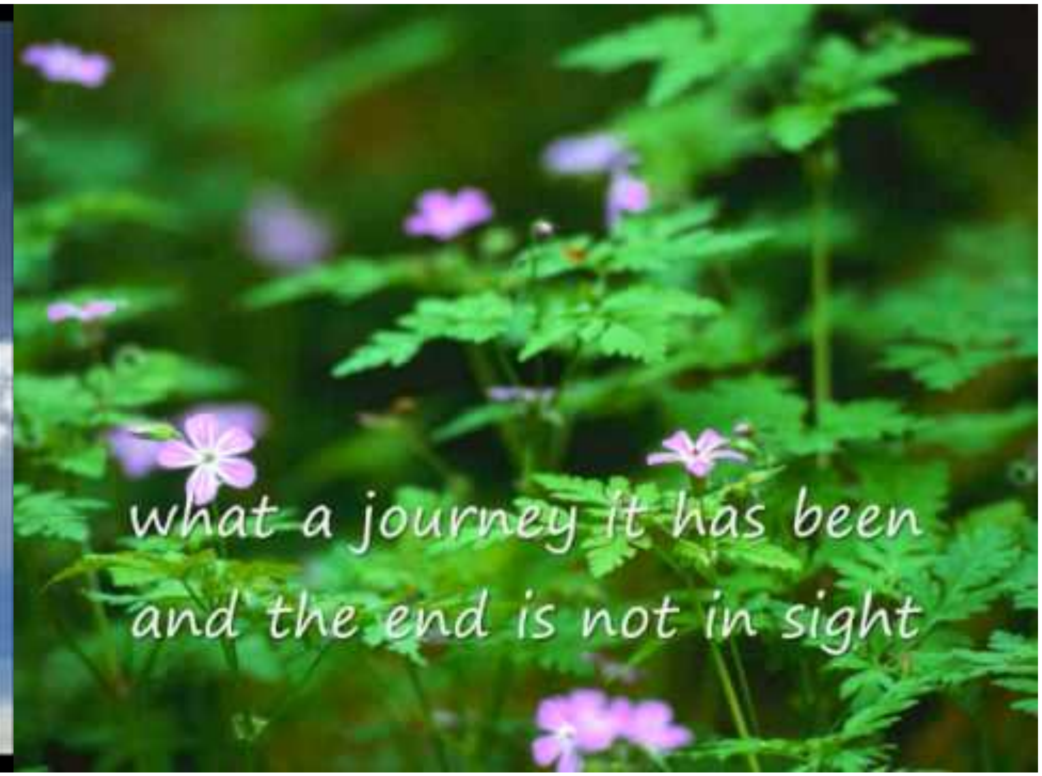
- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$sign($ 0.42  + 0.66  + 0.95  $)$



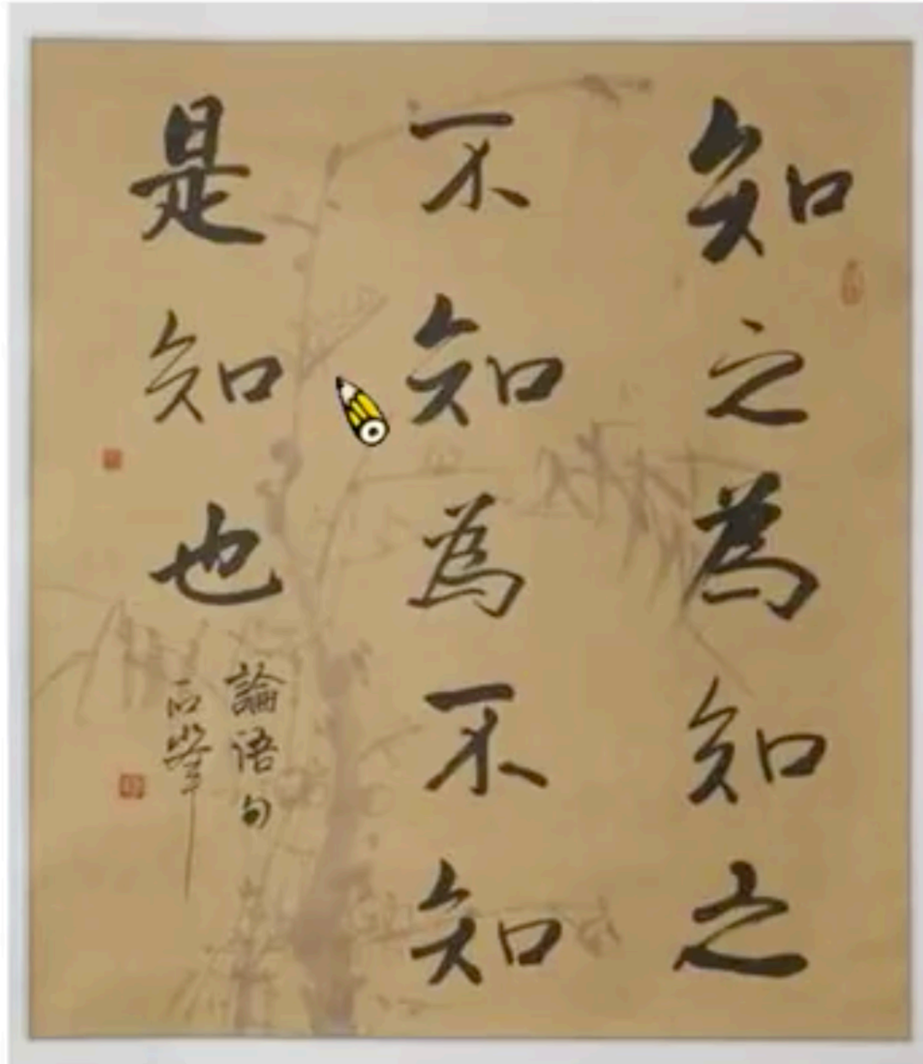
Final Error Rate = 0


What a journey it has been


What a journey it
has been

what a journey it has been
and the end is not in sight

Anomaly Detection

Hung-yi Lee
李宏毅



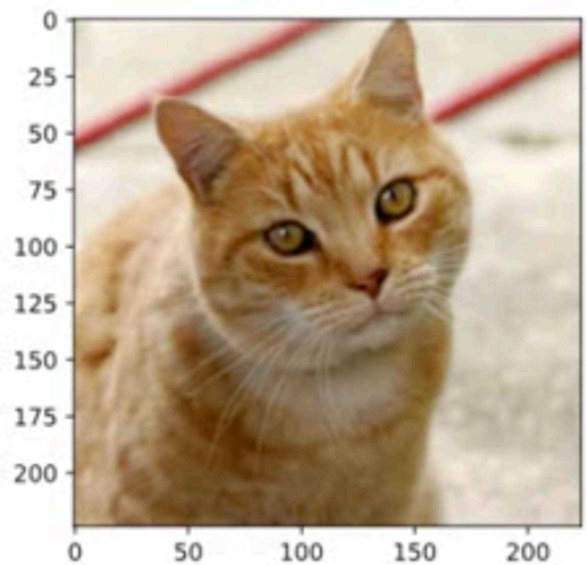
Example

$$L(x') = -C(y', y^{true}) + C\left(y', y^{false}\right)$$

True = Tiger cat
False = Star Fish

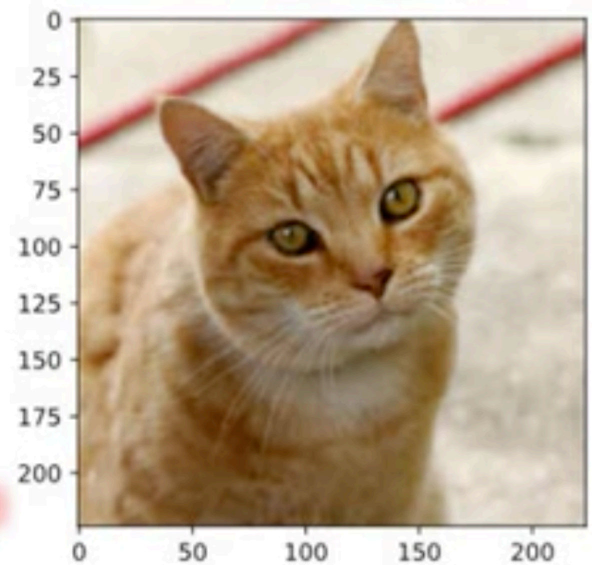
$f =$ ResNet-50

Original Image



Tiger Cat
0.64

Attacked Image



Star Fish
1.00

https://www.cs.cmu.edu/~sbhagava/papers/face-rec-ccs16.pdf

Attack in the Real World

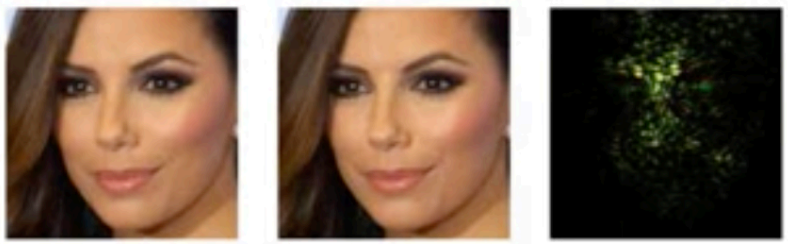
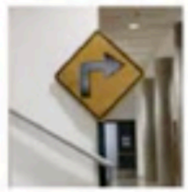
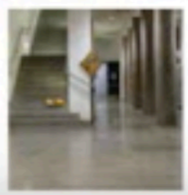
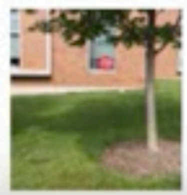
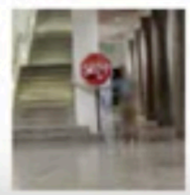
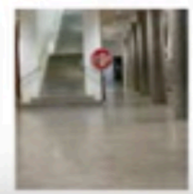


Figure 2: A dodging attack by perturbing an entire face. Left: an original image of actress Eva Longoria (by Richard Sandoval / CC BY-SA / cropped from https://goo.gl/7QUvRq). Middle: A perturbed image for dodging. Right: The applied perturbation, after multiplying the absolute value of pixels' channels ×20.

Figure 3: An impersonation using frames. Left: Actress Reese Witherspoon (by Eva Rinaldi / CC BY-SA / cropped from https://goo.gl/a2sCdc). Image classified correctly with probability 1. Middle: Perturbing frames to impersonate (actor) Russel Crowe. Right: The target (by Eva Rinaldi / CC BY-SA / cropped from https://goo.gl/AO7QYu).

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	10[illegible]	[illegible]

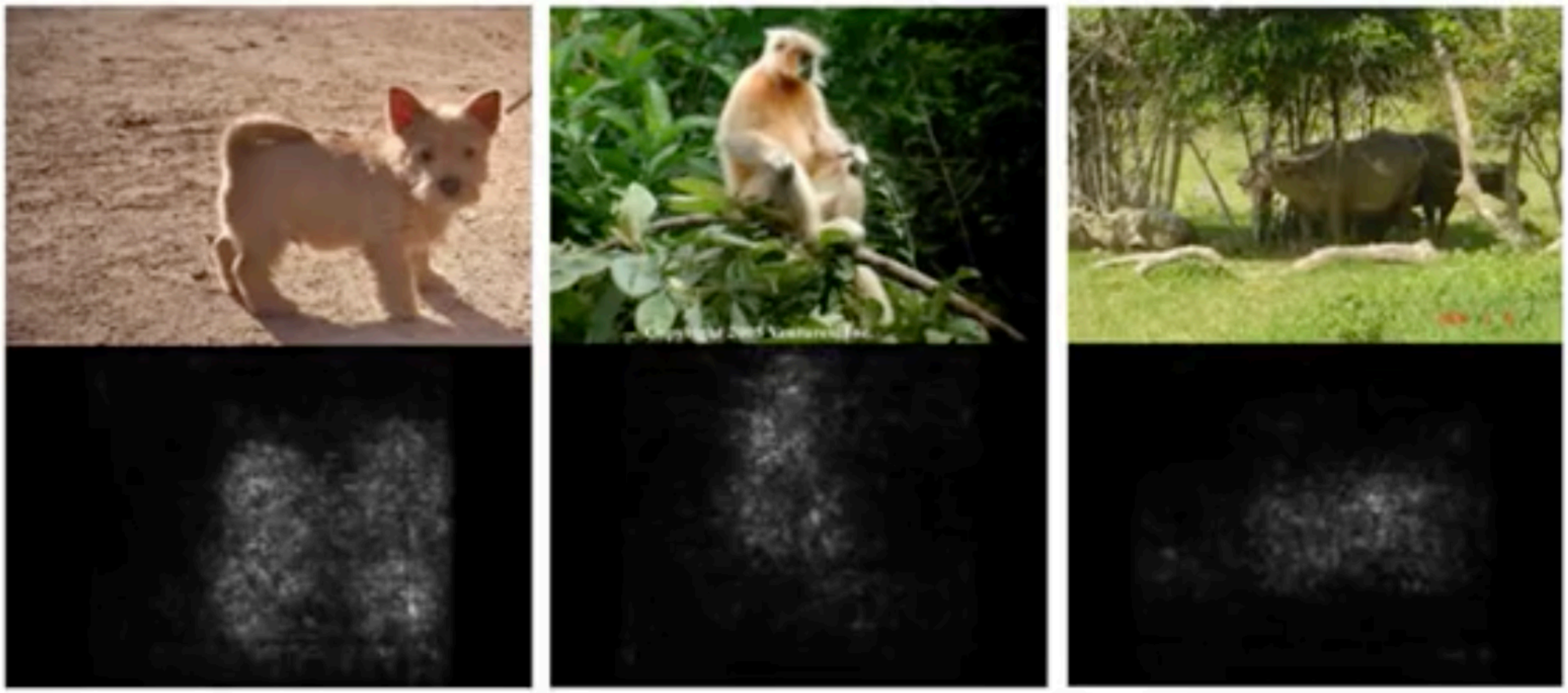
https://arxiv.org/abs/1707.08945

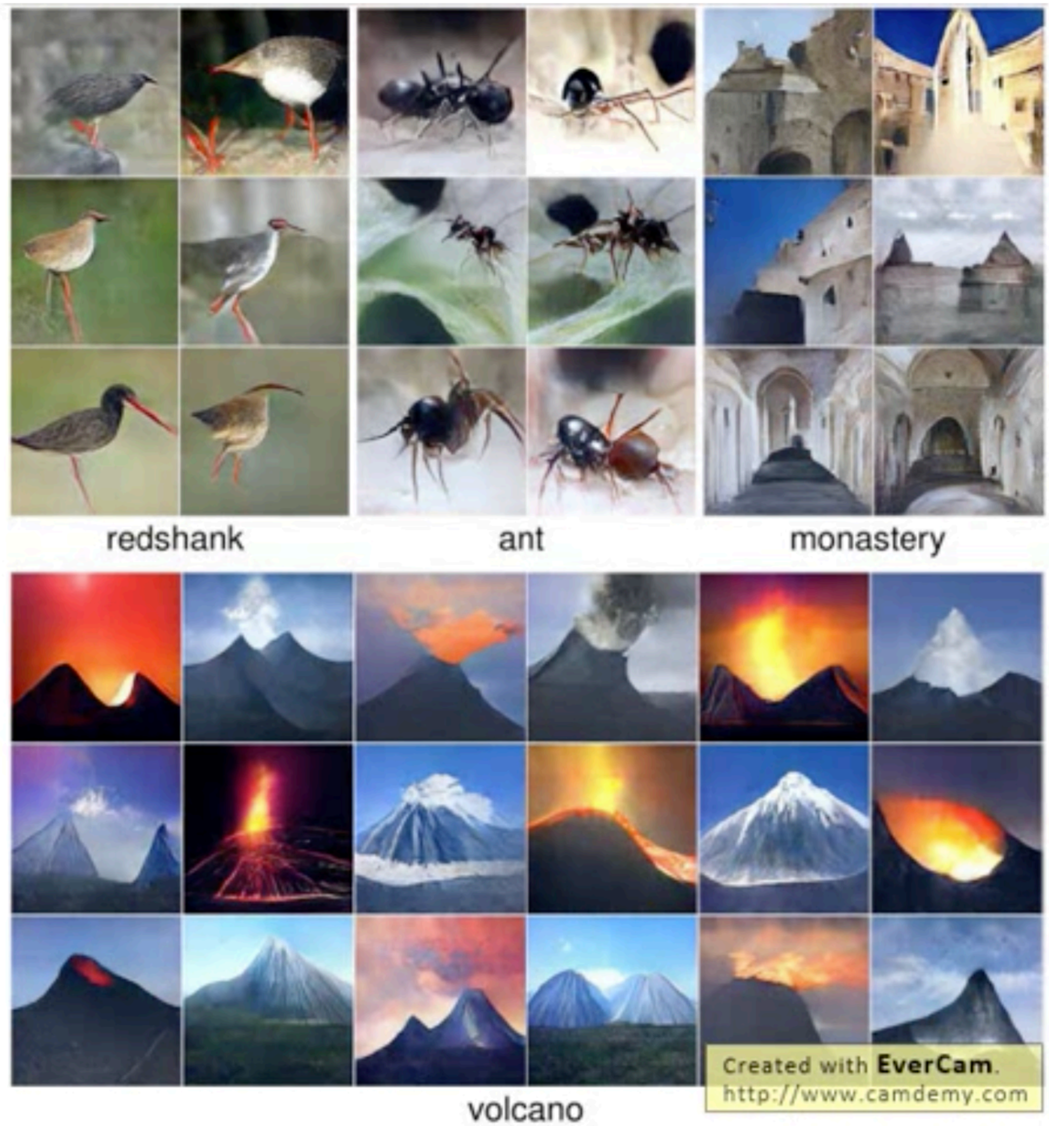
EXPLAINABLE MACHINE LEARNING

$$\{x_1, \cdots, x_n, \cdots, x_N\} \rightarrow \{x_1, \cdots, x_n + \Delta x, \cdots, x_N\}$$

$$y_k \rightarrow y_k + \Delta y \qquad \left|\frac{\Delta y}{\Delta x}\right| \rightarrow \left|\frac{\partial y_k}{\partial x_n}\right|$$

y_k: the prob. of the predicted class of the model





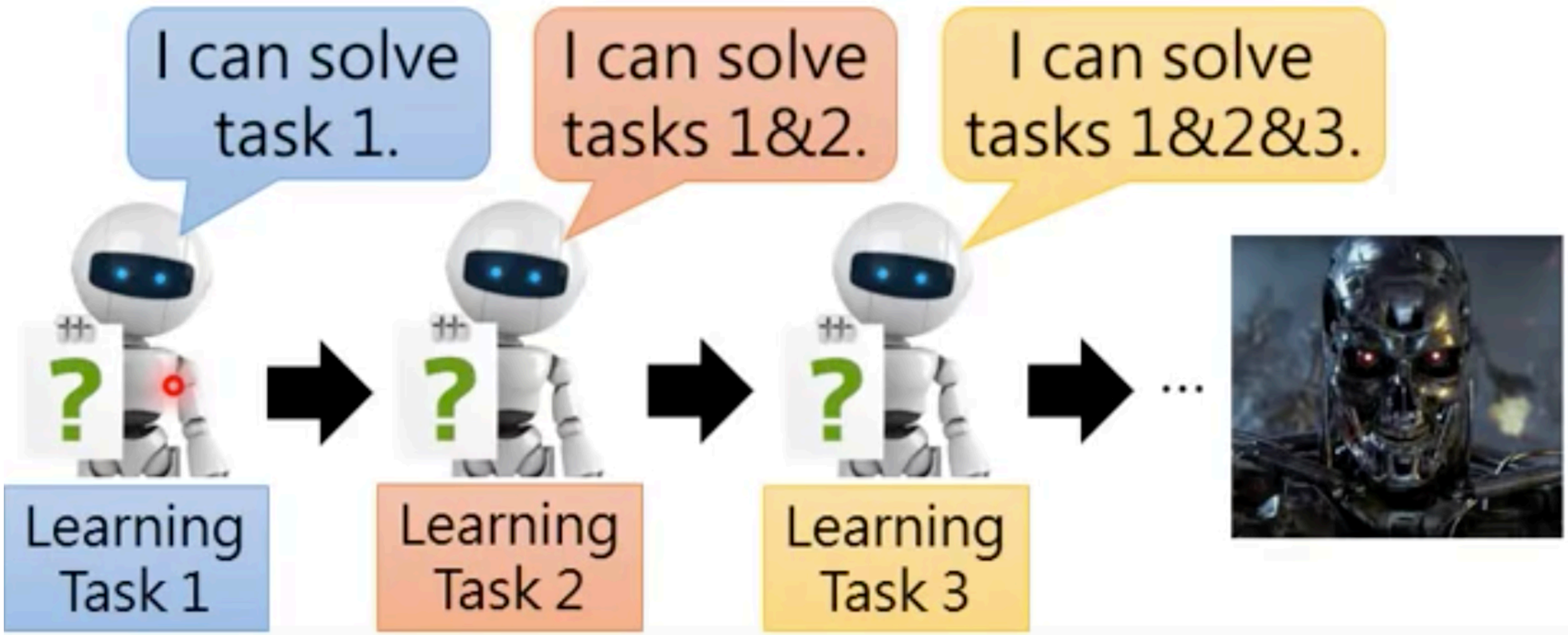
https://arxiv.org/abs/1612.00005

Life Long Learning

Hung-yi Lee
李宏毅

Life Long Learning (LLL)

Continuous Learning, Never Ending Learning, Incremental Learning



Catastrophic Forgetting

Life-long Learning

Knowledge Retention

- **but NOT Intransigence**

Knowledge Transfer

Model Expansion

- **but Parameter Efficiency**

Elastic Weight Consolidation (EWC)

Basic Idea: Some parameters in the model are important to the previous tasks. Only change the unimportant parameters.

Next Spring:
Advanced Topics in Deep Learning