NO.
Date

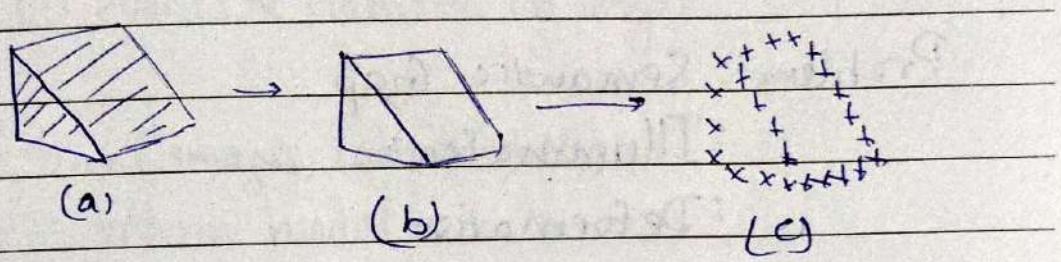
"Lecture 1"

Computer Vision - interpreting visual data.

Majority of data (70%) is visual
Problem: hard to analyze.

Computer Vision → useful in all fields

Block world (Larry Roberts)



Input Image → 2D sketch → 2.5D sketch
↓
3D diagram

Overview

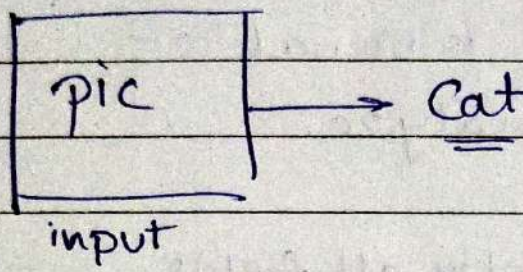
Primary focus → image classification
Other " → object detection, action classification, image captioning

Start with CNN

NO.
Date

"Lecture 2"

Image Classification



computer sees a grid of numbers $\in [0, 255]$

$W \times h \times RGB \longrightarrow 800 \times 600 \times 3$

Problem : Semantic Gap
: Illumination
: Deformation
: Occlusion
: Background Clutter
: Intra class variation.

Image classifier :

```
def classify_image(image):
    return class_label
```

There is no obvious way to recog a cat

NO.
Date

Attempts

Pic → Edges → corners → classify
(Poor at classifying)

Data-driven approach:

1. collect dataset of images & labels
2. Use ML to ~~eta~~ train a classifier
3. Evaluate classifier on images

```
def train(images, ~~tatos~~ labels):
    return model
def ~~eta~~ predict(model, test_img):
    return test_labels
```

First Classifier: Nearest Neighbour

~~to~~ train → memorise all data/label
predict → predict label of most similar training image.

NO.
Date

Distance Metric for comparison:

L1 distance $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test_img			training img			d_1		
32	2	−	14	2	=	18	0	⇒ 37
-7	91		17	82		10	9	

K nearest neighbor
↳ instead of copying label from nearest neighbor we take majority vote for K pts

L2 distance $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$
↳ use for random coordinate systems.

Hyperparameters
↳ best value of K to use
↳ best distance to use
These are choices we set before,
try all, see which is best.

NO.
Date

Setting hyperparameters

X #1 : choose hyperparameters that work best on data

X #2 : divide into train & test and choose best which works on test

✓ #3 : Split into train/test and validation, choos hyperparameters on val.

✓ #4 : Cross validation — split data into fold. (only for small data)

f1	f2	f3	f4	~~f5~~	test
f1	f2	f3	~~f4~~	f5	test
f1	f2	~~f3~~	f4	f5	test.

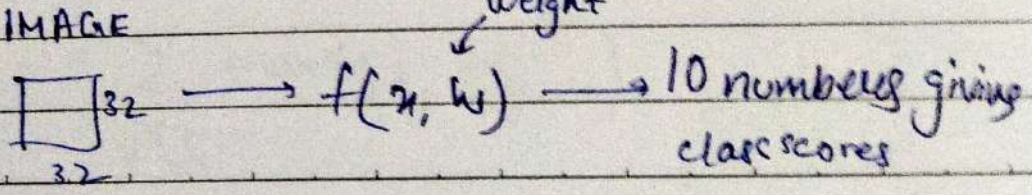
NOTE: WE DONT USE K NEAREST NEIGHBORS

i) Curse of dimensionality

ii) Pics may have diff colors → large dist, but same pic.

Linear Classification

Parametric Approach:



NO.
Date

"Lecture 3"

To: Do

① Define a loss function that quantifies unhappiness
② Come up with a way of efficiently finding parameters that minimize loss fx^n (optimization)

Suppose: 3 classes (cat/car/frog)

A loss function tells us how good our classifier is

Given a dataset,

$$\{x_i, y_i\}_{i=1 \text{ to } N}$$

Loss over the dataset is a sum of loss over examples

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and y_i is the label.

and let scores be $s = f(x^i, W)$

$$L_i = \begin{cases} 0 & s_{y_i} > s_{j+1} \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

NO.

Date

foreg, cat 3.2 [Correct] cat 1.3

car 5.1 car 4.9 [Correct]

frog -1.7 frog 2.0

$L_{cat} = \sum_{j \neq y_i} \ldots$

$= \max(5.1 - 3.2 + 1, 0) + \max(0, -1.7 - 3.2 + 1)$

$= 2.9 + 0$

$= 2.9$

$L_{car} = \max(1.3 - 4.9 + 1, 0) + \max(0, 2.0 - 4.9 + 1)$

$= 0 + 0$

$= 0$

NOTE: Mean also works

Can't use square as it is non-linear

Supervised formula.

$$L(W) = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i) + \lambda R(W)$$

Data Loss — Regularization

NO.

Date

Regularization (Penalty)

$\lambda \rightarrow$ Regularisation strength

L2 Regularisation: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 " : $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic Net ($L_1 + L_2$): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

L2 Regularization (Wt Decay)

$x = [1, 1, 1, 1]$

$w_1 = [1, 0, 0, 0]$

$w_2 = [.25, .25, .25, .25]$

$w_1^T x = w_2^T x = 1$

$R(W) = \sum_k \sum_l W_{k,l}^2$

SoftMax Classification (Multinomial Logistic Regression)

scores = unnormalized log probalities

$P(y = k | x = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$, where $s = f(x_i, W)$

NO.
Date

Want to maximise the log likelihood to minimise the negative log log likelihood.

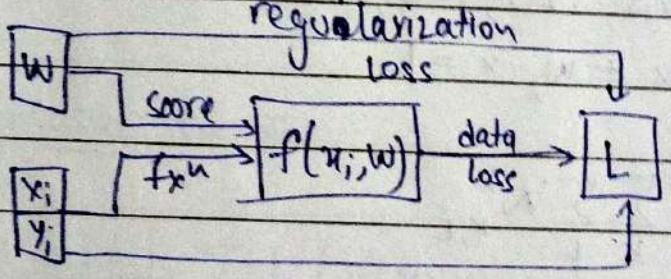
$$L_i = -\log P(Y = y_i | X = x^i)$$

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

foreg,

cat	[3.2]	24.5		0.13
car	5.1	exp → 164.0	normalize →	0.87
car	-1.7	0.18		0.00

$L_i = -\log(0.13)$
$= 0.87$



Strategy #1 : Random Search for W

BAD

NO.
Date

Strategy #2: Follow the slope $\left(\frac{df(x)}{dx}\right)$

Current W	(W + h)	grad dW
0.34	0.34 + 0.0001	-2.5
-1.11	-1.11 + "	0.6
0.78	0.78 + "	0

BAD

We need $\nabla_W L$

then compute the analytic gradient (this is faster)

~~Gch~~.

Stochastic Grad Descent

$$L(W) = \frac{1}{N}\sum_i L_i(x_i, y_i, W) + \lambda R(W)$$

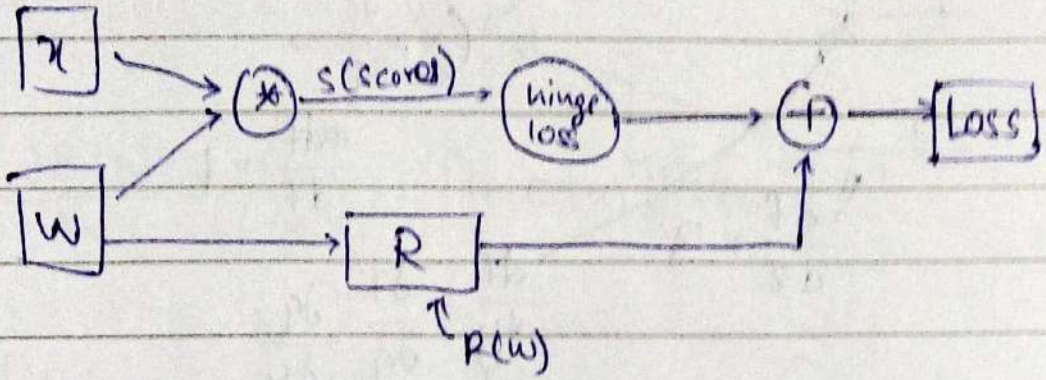
$$\nabla_W L(W) = \frac{1}{N}\sum_i \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

NO.
Date

"Lecture 4"

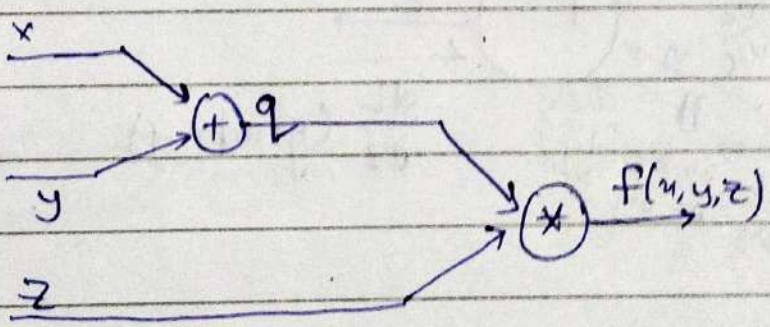
Computational Graphs

$f = Wx \qquad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$



Backpropagation

$f(x, y, z) = (x + y)z$



$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

NO.

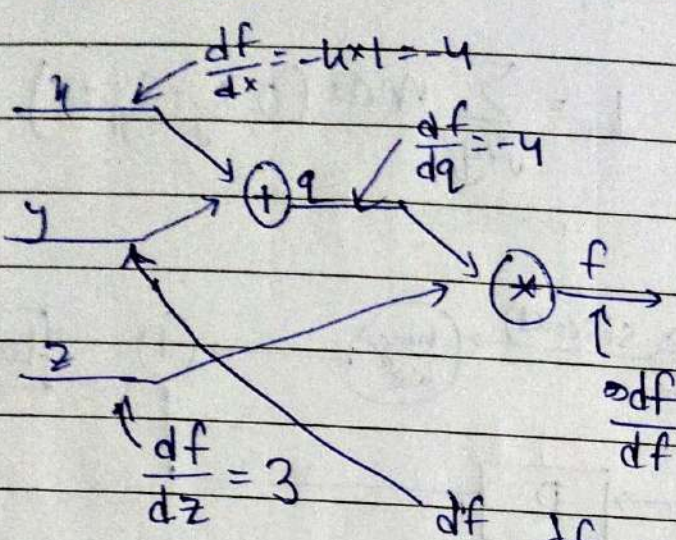
Date

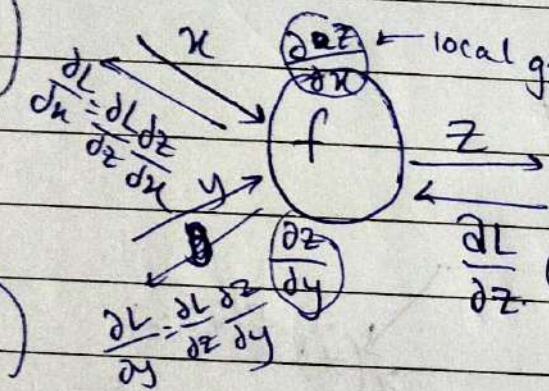
Start from back

$x = -2$

$y = 5$

$z = -4$



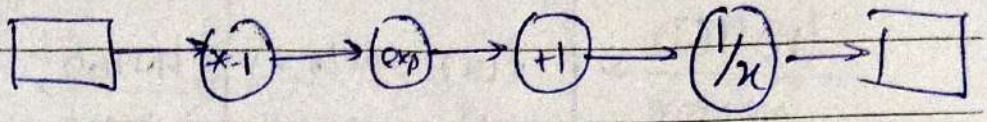


Another example

$$f(x) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

NO.
Date

Sigmoid gate



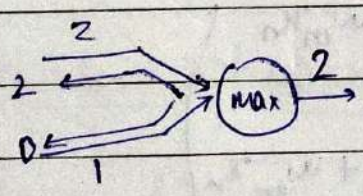
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = [1-\sigma(x)][\sigma(x)]$$

Patterns in backward flow:-

add gate: gradient distributor

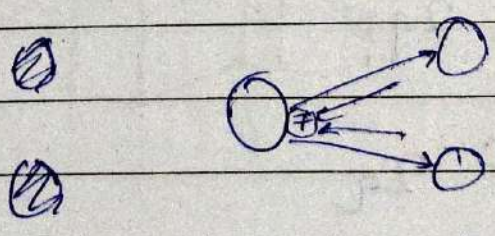
max gate:



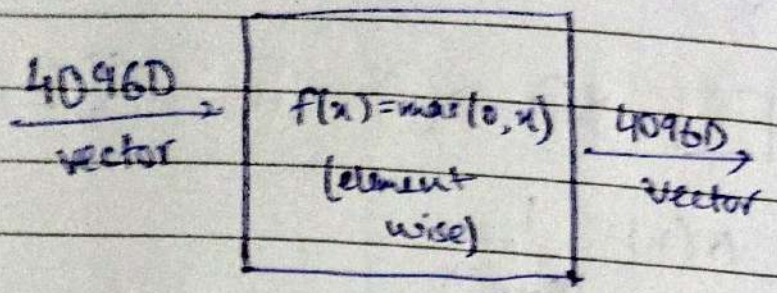
max gets full rest gets 0.

mul gate: gradient switcher

gradients add at branches:

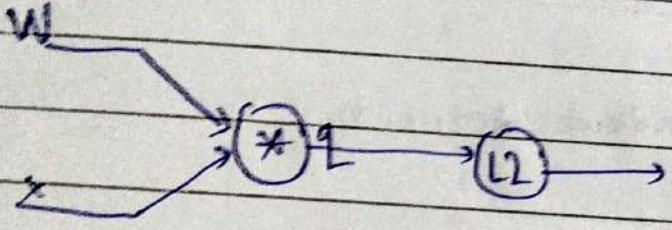


Vectorised operations



Example

↳ $f(x, W) = \|W.x\|^2 = \sum_i (W.x)_i^2$



$$q = W.x = \begin{pmatrix} w_{11}x_1 + \dots + w_{1n}x_n \\ \vdots \\ w_{n1}x_1 + \dots + w_{nn}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \dots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\nabla_q f = 2q$$

NO.

Date

$$\frac{\partial q_k}{\partial w_{ij}} = \mathbb{1}_{k=i} x_j$$

$$\frac{\partial f}{\partial w_{ij}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial w_{ij}}$$

$$= \sum_k 2 q_k \mathbb{1}_{k=i} x_j$$

$$= 2 q_k x_j$$

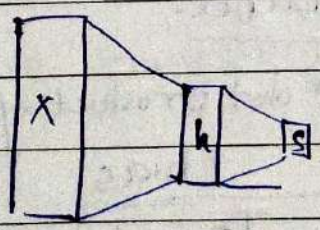
Similarly, $\frac{\partial f}{\partial x_i} = 2 q_k w_{k,i}$

$$\nabla_x f = 2 W^T \cdot q$$

Neural Networks

Before: $f = Wx$ [single layer]

After: $f = W_2 \max(0, W_1 x)$ [double layered]



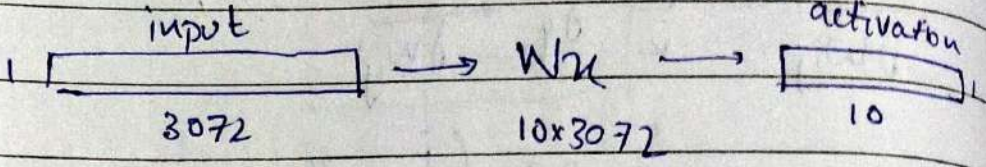
NO.

Date

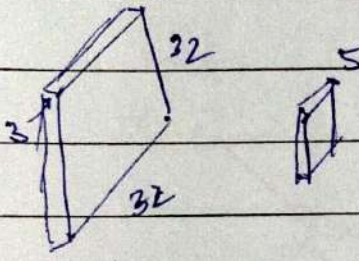
"Lecture 5"

Fully Connected Layer:

32×32×3 img → stretch 3072×1



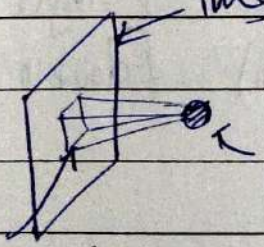
Convolution Layer



convolve the filter with the image

depth will be same

but, spatial area goes down



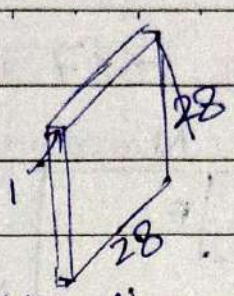
1 number:

↳ dot product of filter & img + bias

$$w^T x + b$$

NO.
Date

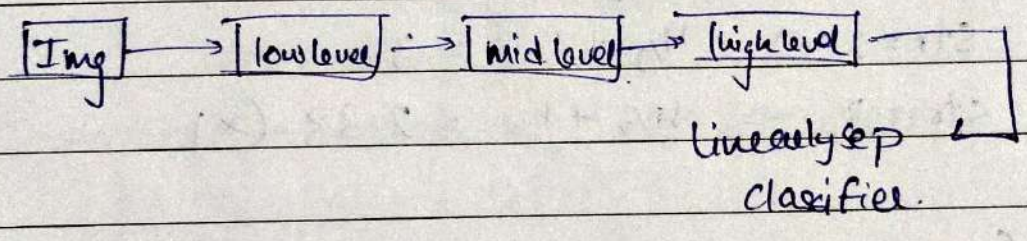
convolve over all loc



activation map

As many filters, that many act maps.

ConvNet: sequence of conv layers, interspersed with activation functions.



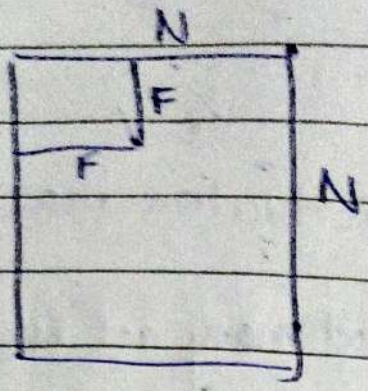
We call layer convolutional coz it is the conv. of two signals.

$$f[x,y] * g[x,y] = \sum_{n_1} \sum_{n_2} f[n_1, n_2] \cdot g[x-n_1, y-n_2]$$

NN will have conv, RELU, conv, RELU, Pool structure.

No
Date

Let their be 7×7 input and 3×3 filter



Output size: $\frac{N-F}{\text{stride}} + 1$

str = 1 ⇒ 4/1 + 1 = 5

str = 2 ⇒ 4/2 + 1 = 3

str = 3 ⇒ 4/3 + 1 = 2.33 (x)

Commonly, we have a padding of 0s so its 8×8 in reality!

Convnet, shrinks slowly as fast shrinking doesn't work much well

Q Input: 32×32×3

10 5×5 ~~input~~ filters with stride 1, pad 2

Ans Output size: N − F/str + 1

= (32 + 2×2 − 5)/1 + 1

= 32

NO.
Date

# of parameters = 5*5*3 + 1
= 76 params

760

ConvNet

→ accepts a vol of $W_1 \times H_1 \times D_1$

→ hyperparameters
- No of filters K
- spatial Extent F
- stride S
- zero padding P

→ produces a vol of $W_2 \times H_2 \times D_2$ where;
- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$
- $D_2 = K$

→ parameter sharing, so
- wts per filter = $F \times F \times D_1$
- Total wts = F.F.D_1.K and K biases

Common

$K = 2^n$

→ F=3, S=1, P=1

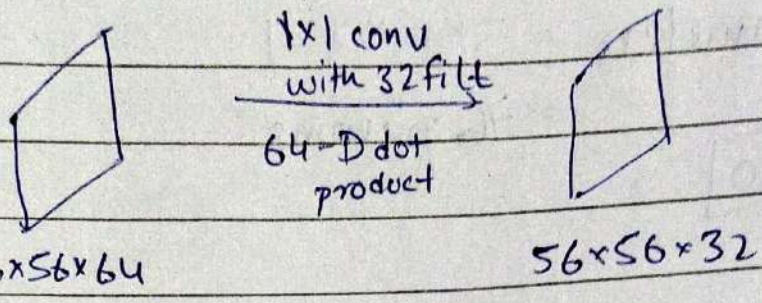
→ F=5, S=1, P=2

→ F=5, S=2, P=?

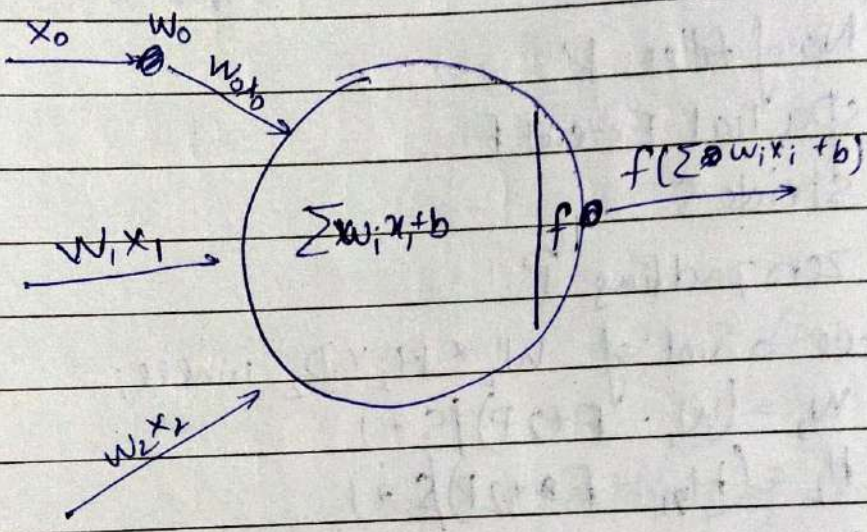
→ F=1, S=1, P=1

NO.

Date



Brain/Neuron View



Pooling Layer

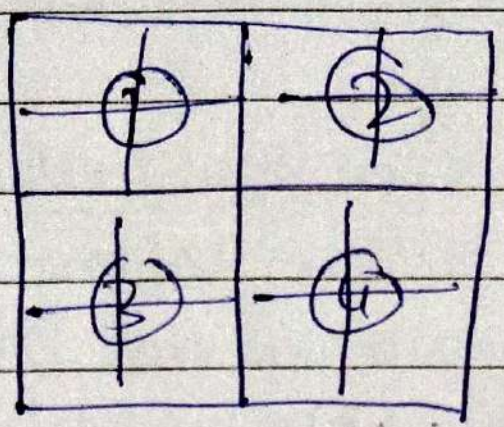
→ makes the representations smaller and more managable

→ operates over each activation map independently

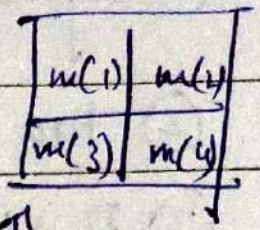
NO.

Date

Max pooling



maxpool
2×2 filter
stride 2



min max()

"Lecture 6"

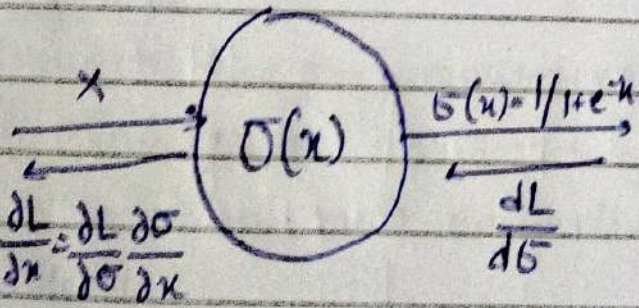
Activation function

some examples

1. Sigmoid $\Rightarrow \sigma(x) = \frac{1}{1+e^{-x}}$
2. tanh x
3. ReLU $\Rightarrow \max(0, x)$
4. Leaky ReLU $\Rightarrow \max(0.1x, x)$
5. Maxout $\Rightarrow \max(w_1^T x + b_1, w_2^T x + b_2)$
6. ELU $\Rightarrow \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

Sigmoid

- Squashes numbers in range [0,1]



NO.
Date

gradients on w?
↳ all positive or all negative

Problems with sigmoid:
① saturated neurons kill gradient
② not zero centered
③ exp() compute expensive.

tanh(x)
- squashes no from [-1, 1]
- zero centered
- but still kills gradients

ReLU
- doesn't saturate
- efficient
- converges faster than earlier
- more plausible than sigmoid.
- but not zero centered.

Leaky ReLU
↳ max(0.01x, x)
→ will not die

NO.

Date

PReLU

↳ $\max(\alpha x, x)$

α ↑ parameter.

ELU

↳ $\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

→ adds robustness to noise

Maxout Neuron

→ generalizes ReLU and Leaky ReLU

$\max(w_1^T x + b_1, w_2^T x + b_2)$

problem: double the parameters

In Practice:

① Use ReLU

② Use Leaky ReLU/maxout/ELU

③ use tanh (bad)

④ never use sigmoid.

NO.
Date

Preprocessing of Data

① zero center them
② normalise them. (std deviation)

Step1:
① PCA the data
② Whiten the data

for images:
- subtract the mean image, or
- subtract per channel mean.

Weight initialization:

Q What happens W=0?

First idea: small random nos

(W = 0.01 * np.random.randn(D,H))

This is okay for small networks

Xavier initialization

(W = np.random.randn(fi, fo)/np.sqrt(fi)

NO.

Date

He et al

↳ divide by 2

$(W = np.random.randn(fi, fo) / np.sqrt(fi/2))$

Batch Normalization

To make each dimension unit gaussian, apply:

$$\hat{x}_k = \frac{x^k - E(x^k)}{\sqrt{Var(x^k)}}$$

add after Fully connected or ConvNet layer.

first find mean and var for each dimension and then normalize

Then allow the network to do this:

$$y^k = \gamma^k \hat{x}^k + \beta^k$$

NO.
Date

Batch Normalization:

i/p : values of x over mini-batch $\mathcal{B} = \{x_1, \ldots m\}$
parameters : γ, β.

o/p : $\{y_i = BN_{\gamma,\beta}(x_i)\}$

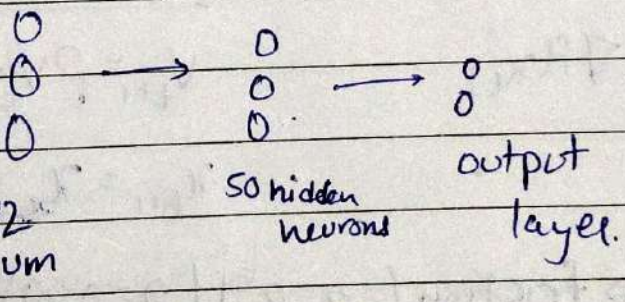
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_i x_i$$

$$\sigma^2_{\mathcal{B}} \rightarrow \frac{1}{m} \sum_i (x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma x_i + \beta \equiv BN_{\gamma,\beta}(x_i)$$

(2) Chose an architecture
Let's choose 50.



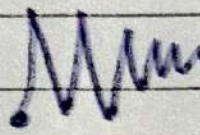
double-check if the loss is reasonable.

✓ do cross validation

NO.
Date

"Lecture 7"

Loss function has high cond^n number.
very slow progress along shallow dimension, jitter along steep direction.

 (slow progress).

Q What if loss fx^n has a local minima or saddle point?
A: Grad descent gets stuck.

Gradients come in minibatches, so they can be noisy

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

SGD + momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

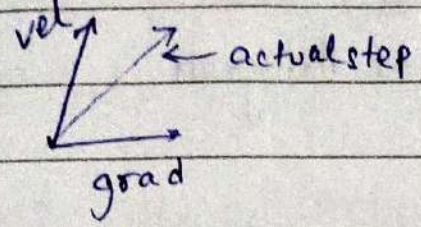
$\rho \Rightarrow$ friction ($\rho = 0.9$ / $\rho = 0.99$)

- no noise
- better progress

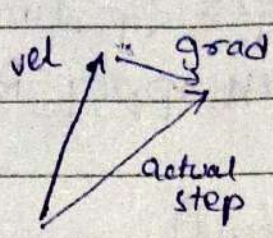
NO.

Date

Nesterov Momentum



Normal



Nesterov

$$V_{t+1} = \rho V_t - \alpha \nabla f(x_t + \rho V_t)$$

$$x_{t+1} = x_t + V_{t+1}$$

change,

$$x_{t+1} = \rho V_t - \alpha \nabla f(\tilde{x}_t)$$

$$\tilde{x}_{t+1} = x_t - \rho V_t + (1+\rho) V_{t+1}$$

$$= \tilde{x}_t + V_{t+1} + \rho(V_{t+1} - V_{\partial t})$$

Adagrad

while True:

dx ⟶

gradsq = dx * dx

$x_{\partial} = x - \text{learningrate} * dx / \text{sqrt}(\text{gradsq}) + 1e\text{-}7$

RMS prop

gradsq = decay_rate * gradsq + (1 - decay_rate) * dx * dx

NO.
Date

Adam

firstp = β_1 * firstp + (1 - β_1) * dx
then, apply RMSProp or Adagrad

Adam(full)

Adam half

second moment
first unbias = firstp / $(1-\beta_1^t)$
second " = secondp / $(1-\beta_2^t)$
do adagrad / RMSprop on unbias

Use

$\beta_1 = 0.9$ $\beta_2 = 0.999$
$\alpha = 1e^{-3}$ / $1e^{-5}$

Learning Rate Decay

① Step decay
half per few epoch

② Exp decay
$\alpha = \alpha_0 e^{-kt}$

③ 1/t decay
$\alpha = \alpha_0 / (1+kt)$

NO.
Date

First Order optimization

(1) Use gradient from linear approx.
(2) Step to minimize the approx.

Second order optimization

(1) Use grad & hessian to form quadratic approx.
(2) step to the minima of approximation.

In practice,
→ Use ADAM
→ try out LBFGS

Model Ensemblee

1) Train multiple independent models
2) At test time, average their results
3) Use snapshots of single model during training.

Dropout Regularison

→ we drop random neurons to 0
→ forces the network to have redundant rep.
→ prevents co-adaption of features.

NO
Date

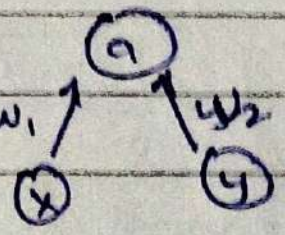
Basically it is training a large ensemble of models.
Each binary mask is one model!

Dropout makes our output random

$$y = f_w(x, z) \quad \text{(z: random mask)}$$

$$y = f(x) = E_z[f(x,z)] = \int p(z) f(x,z)\, dz$$

Consider a single neuron,



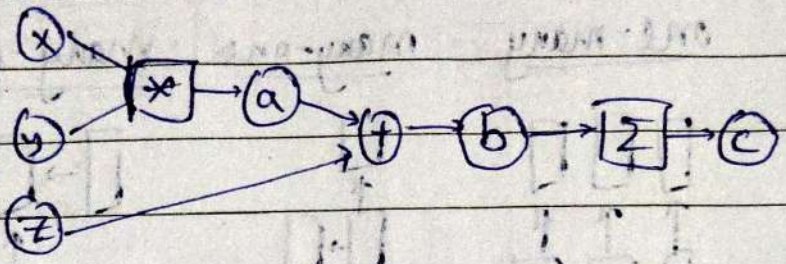
$$E[a] = w_1 x + w_2 y$$
$$= \frac{1}{4}(w_1 x + w_2 y) + \frac{1}{4}(w_1 x + 0y) + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2 y)$$
$$= \frac{1}{2}(xw_1 + yw_2)$$

Data Augmentation

NO.
Date

"Lecture 8"

Comp Graph Example



N, D = 3, 4

x = randn(N, D)
y = "
z = "
a = x + y
b = a + z
c = sum(b)

numpy

→ no GPU.
→ compute grad on own.

dc = 1.0
db = dc * ones(N, D)
da = db
dz = db
dx = da * x
dy = da * x

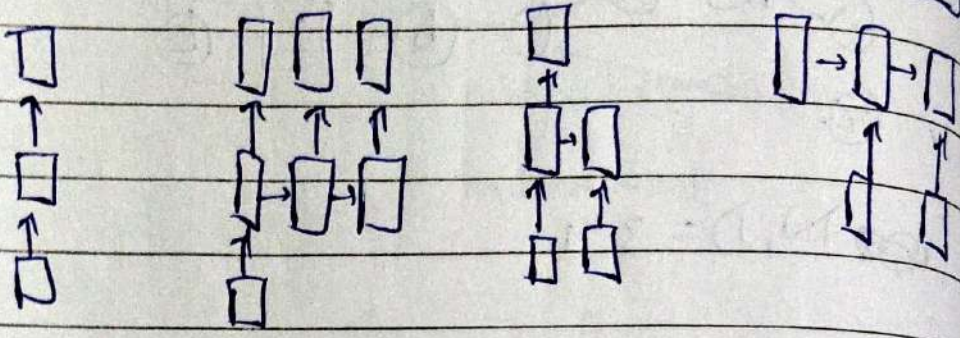
NO.

Date

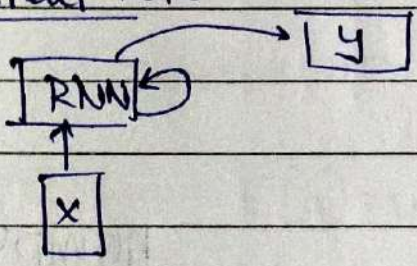
"Lecture 10"

Process sequences of recurrent NNs

one-one one-many many-one many-many



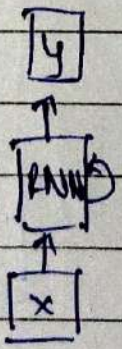
Recurrent NN



Recurrance relation:

$$h_t = f_w(h_{t-1}, x_t)$$

Vanilla RNN



~~$h_t = h_{wt}$~~

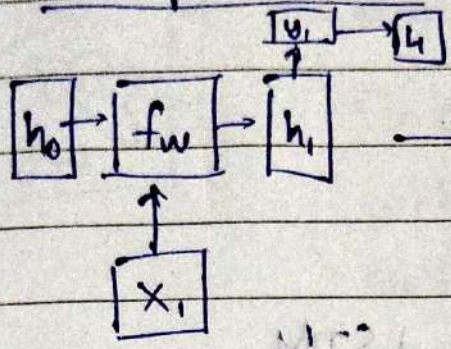
$$h_t = f_w(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} x_t$$

NO.
Date

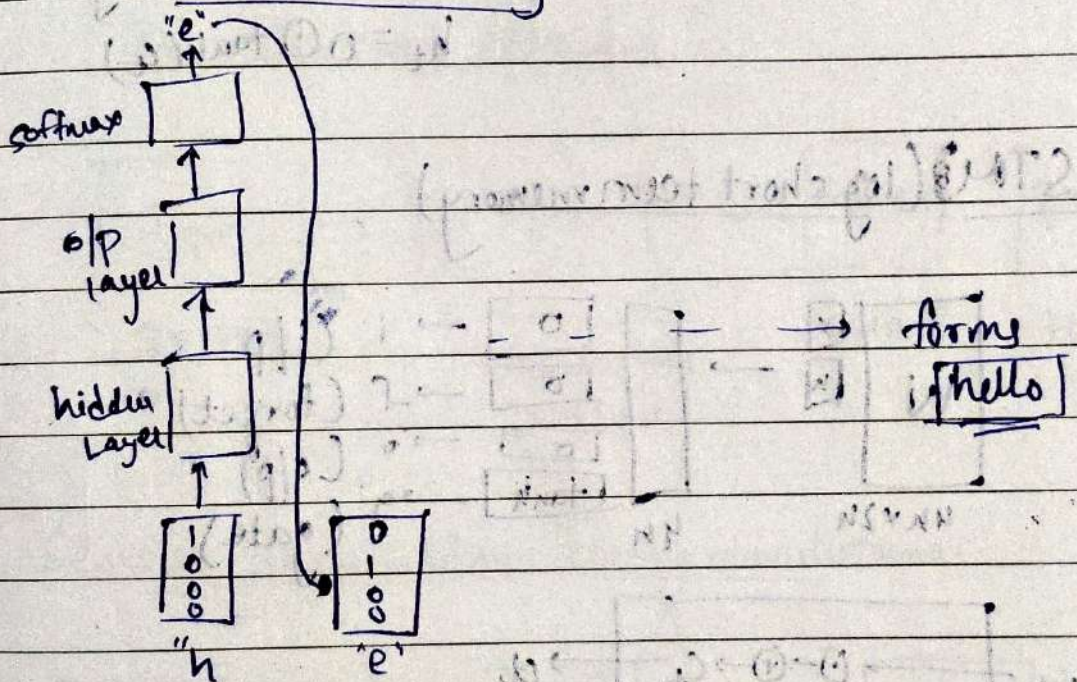
RNN: Computational Graph



Use same W for every case for x_0

$$L = \sum_i L_i$$

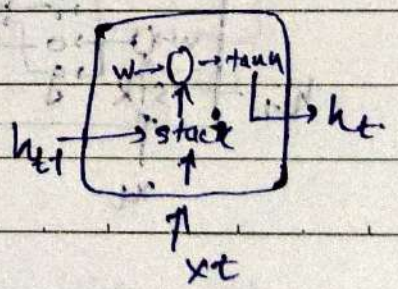
Character Level Lang Model



Vanilla RNN gradient flow

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$



NO

Date

Largest singular value > 1 :
exploding gradient
Largest singular value < 1 :
vanishing gradient

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

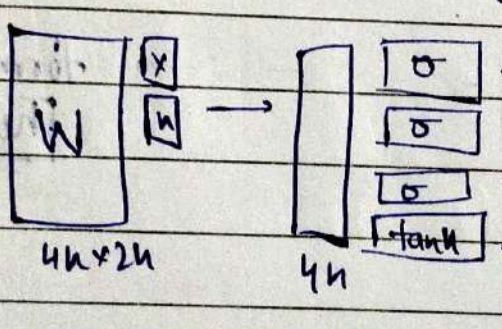
LSTM

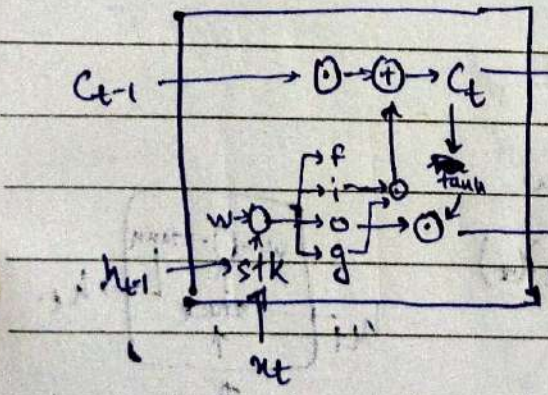
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM (long short term memory)





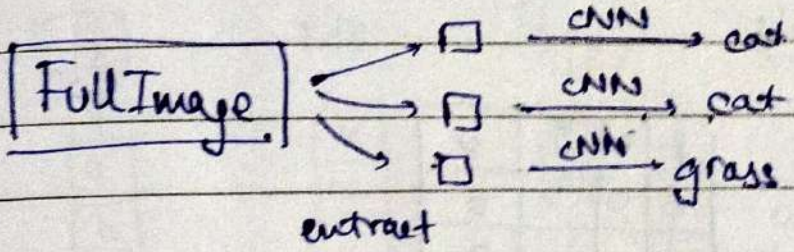
structure

"Lecture 911"

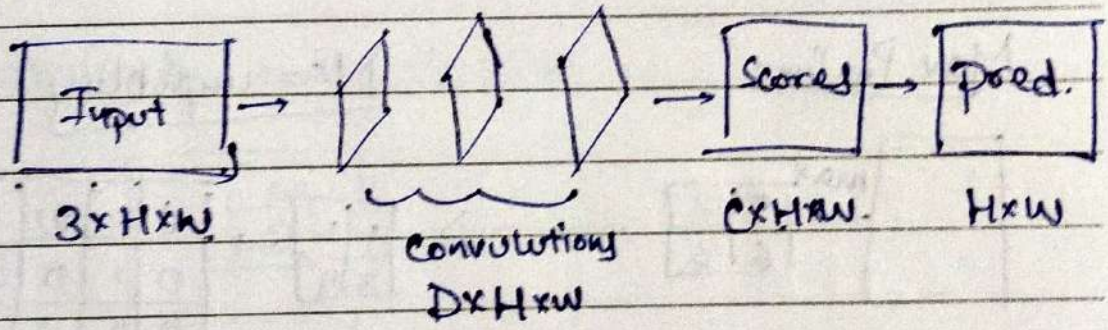
NO
Date

← Semantic Segmentation

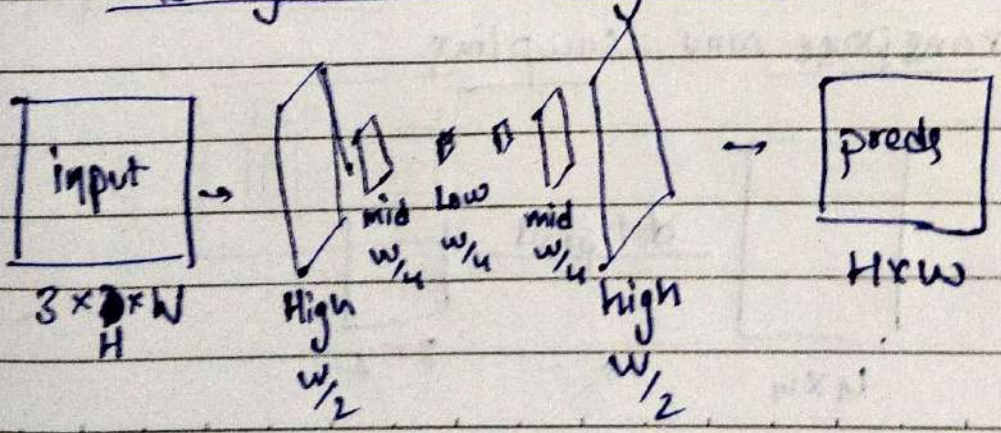
- label each pixel in the image with a category email
- don't differentiate instances, only care about pixels.



Problem: Not efficient



Semantic Segmentation: Fully convolutional

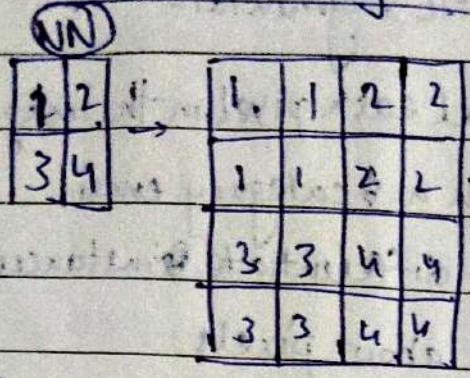


downsample then upsample

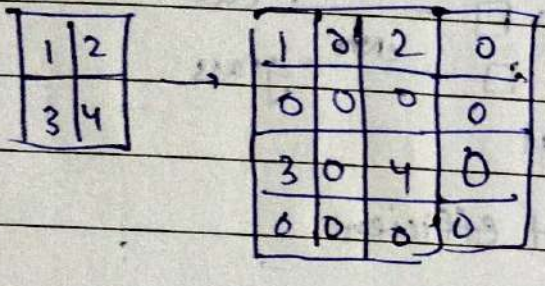
NO.

Date

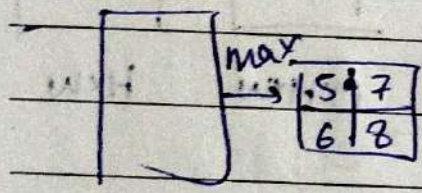
In-Network upsampling : Unpooling



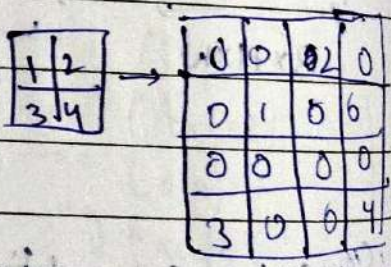
Bed of Nails



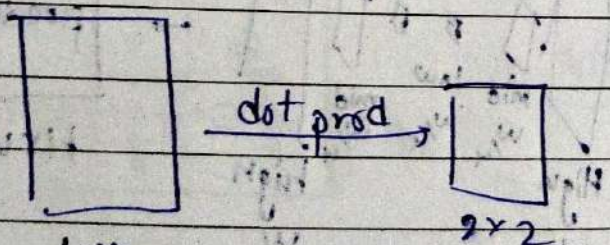
Max Pooling



Max unpooling

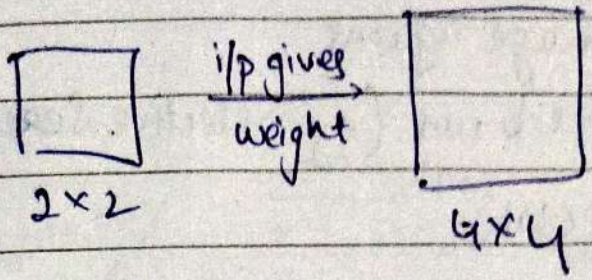


Transpose conv : sampling

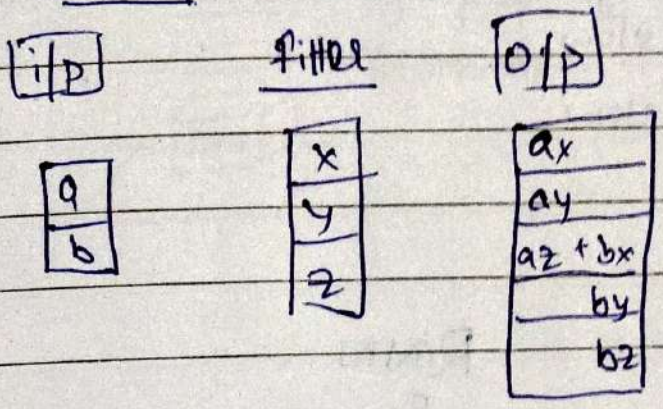


NO.
Date

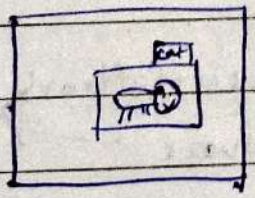
unsampling



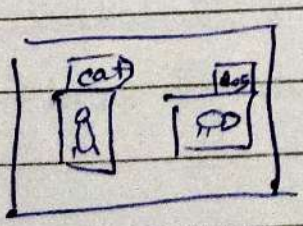
1-D example



Classification and localization



Object Detection



NO.

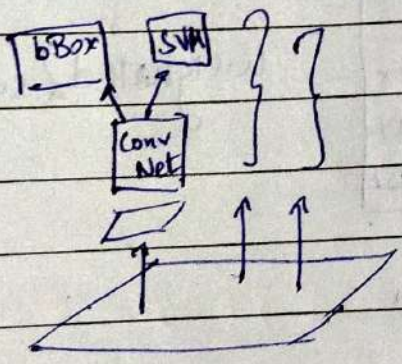
Date

Region Proposal

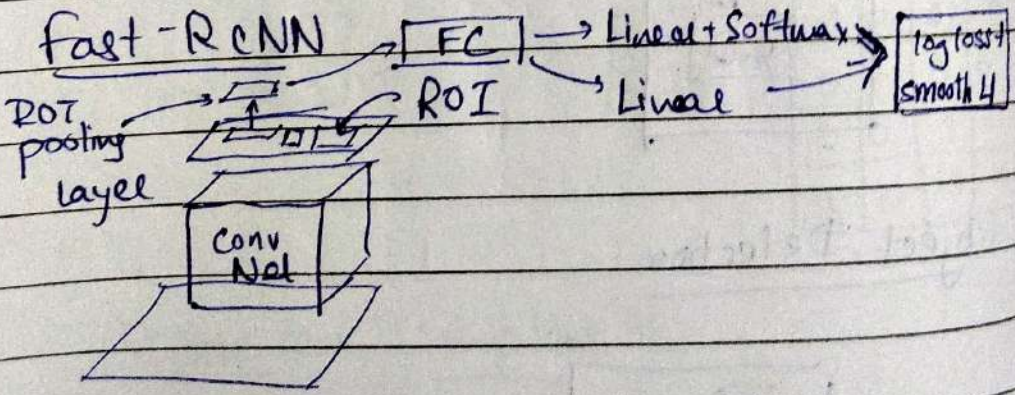
- find blobby image regions
- relatively faster to run (eg. Selective Search,

Problems w RCNN

- Ad hoc training objectives
- Training is slow
- detection is slow

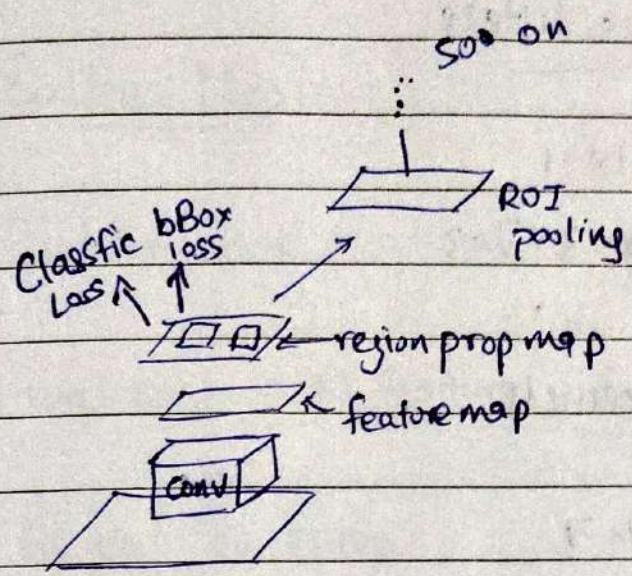


RCNN Structure



NO.
Date

Faster RCNN



jointly train 4 losses

1. RPN classify obj/or not
2. RPN regress box coordinates
3. Final classfication score
4. final box coordinates.

Classification w/o proposals (YOLO)

→ ~~within each grid cell~~
↳ makes a ~~final box with~~ 5no (dy, dx, dh, dw, conf)

→ predict scores for each class

Instance segmentation

↳ Mask RCNN

NO.

Date

"Lecture 12"

ConvNet Structure

First Layer: Visualize filters

AlexNet: 3 x 11 x 11

ResNet 101: 64 x 3 x 7 x 3

Visualize filters / kernels

layer 1: 16 x 3 x 7 x 7

layer 2: 20 x 16 x 7 x 7

layer 3: 20 x 20 x 7 x 7

Last Layer: FC7 layer

- 4096-dim feature vector for an image.
- applies NN.
- dimensionality reduction.

visualize 13 x 13 x 128 map as 128 13 x 13 grayscale images

Maximally Activating patches

NO.
Date

Occlusion Experiment

Saliency Maps

Intermediate features via guided backprop

Visualizing CNN features : Gradient Ascent

Guided Backprop:
find part of image that neuron responds to

Use grad ascent.

$$I^* = \arg\max_I f(I) + R(I)$$

neuron value

natural image regularizer.

1. Initialize your image to zeros

Repeat

2. Forward image to compute current scores
3. Backprop to get grad
4. make small image

score for class $c \Rightarrow \arg\max S_c(I) - \lambda \|I\|_2^2$

Simple regularizer $\Rightarrow \lambda \|I\|_2^2$

Better regularizer $\Rightarrow$ Gaussian $\rightarrow$ clip pixel to 0
$\hookrightarrow$ clip small grad to 0.

NO.

Date

Adversarial Examples

(1) Start from arbitrary image.

(2) Pick arbitrary class

(3) Modify image to maximise class

(4) Repeat till they are fooled

Feature Inversion

Given a CNN feature find new image:-

1) Matches given feature vector

2) looks natural

$$x^* = \underset{x \in \mathbb{R}}{\operatorname{argmin}}\ l(\phi(x), \phi_0) + \lambda R(x)$$

where, $l(\phi(x), \phi_0) = \| \phi(x) - \phi_0 \|^2$

$$R(x) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\beta}$$

- Texture Synthesis : Nearest Neighbour.

NO.

Date

- Neural Texture Synthesis: Gram Matrix.
- Neural Style Transfer.
- Fast Style Transfer

NO.
Date

Lecture 13"

Unsupervised Learning

Data: x (no label)

Goal: Learn some hidden structure of the data

Eg: Clustering, dimension reduction, feature learning, density estimation.

Generative Models

from training data, generate new samples

$$P_{data}(x) \longrightarrow P_{model}(x)$$

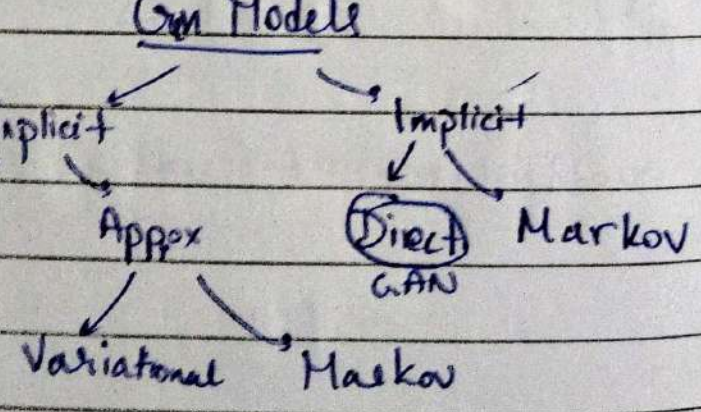
Addresses density estimation, a core problem.

Flavors:

i) explicit density estimation

ii) Implicit density estimation

Taxonomy



NO.
Date

Pixel RNN & CNN

fully visible belief network:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, x_2 \dots, x_{i-1})$$

PixelRNN

Generate image pixels starting from corner.
use RNN(LSTM)

PixelCNN

Still generate image pixel starting from corner.
↳ train using max likelihood.

Cons → Sequential gen → slow
Pros → can compute likelihood/expected likelihood
→ good sample.

Variable Autoencoders (VAE)

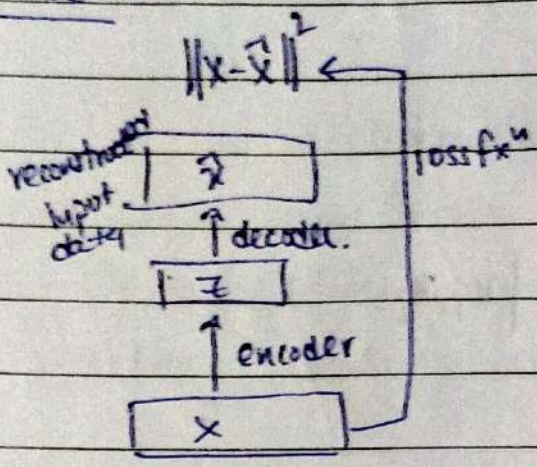
$$P_\theta(x) = \int P_\theta(z) P_\theta(x|z) dz$$

↳ can't optimize directly.

NO.

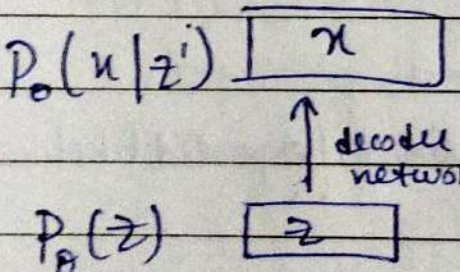
Date

Autoencoder



Encoder can be used as supervised model

Variational autoencoder



$x \rightarrow$ image

$z \Rightarrow$ factors that form x

How to represent this model?

- Choose $p(z)$ to be simple
- ~~cons~~ $p(x|z)$ is complex $\Rightarrow$ neural network

NO.

Date

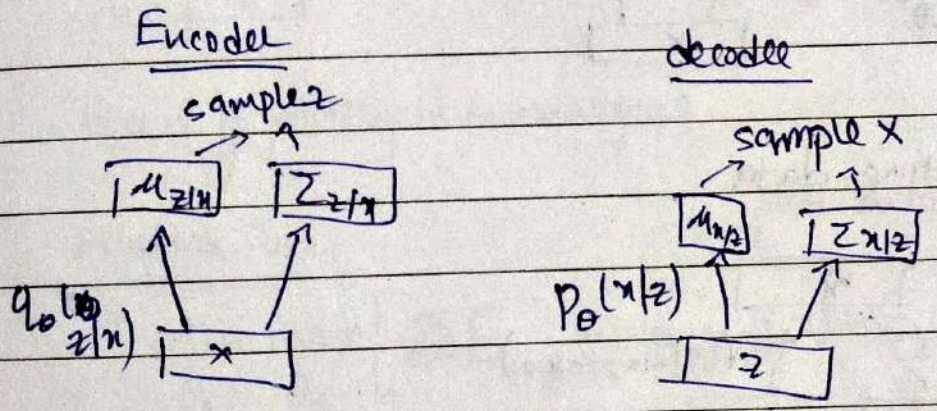
How to train this model?

Learn model parameters to maximise likelihood of training data.

$$P_\theta(x) = \int P_\theta(z)\, P_\theta(x|z)\, dz$$

Data likelihood $\Rightarrow$ $P_\theta(x) = \int P_\theta(z)\, P_\theta(x|z)\, dz$ (intractable)

Post. density $\Rightarrow$ $P_\theta(x|z) = P_\theta(x|z)\, P_\theta(z) \,/\, P_\theta(z)$ (intractable)



$$\log P_\theta(x^i) = E_{z \sim q(z|x^i)}\left[\log P_\theta(x^i)\right]$$

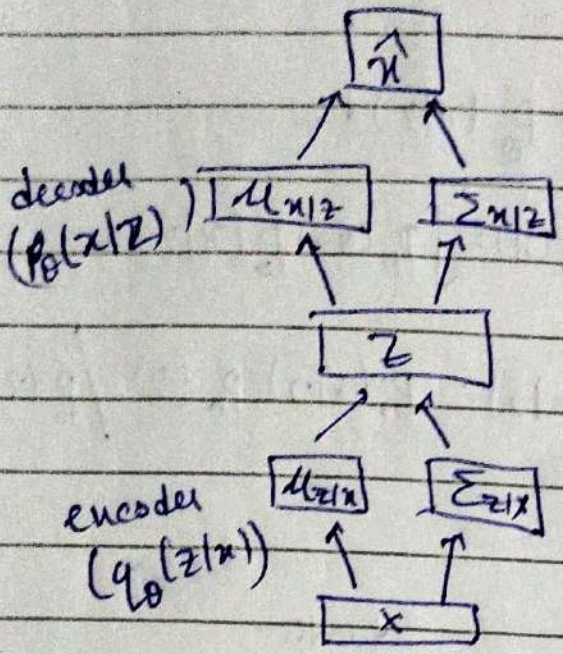
$$= E_z\left[\log \frac{P_\theta(x^i|z)\, P_\theta(z)}{P_\theta(z|x^i)}\right] \quad D_{KL}(q_\phi(z|x^i) \,||\, P_\theta(z))$$

$$= E_z[\log P_\theta(x^i|z)] - E_z\left[\log \frac{q_\phi(z|x^i)}{P_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|x^i)}{P_\theta(z|x^i)}\right]$$

$$D_{KL}(q_\phi(z|x^i) \,||\, P_\theta(z|x^i))$$

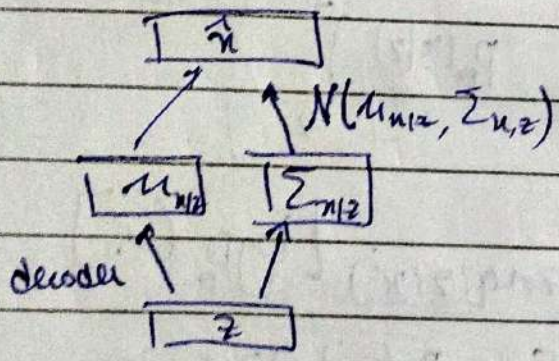
NO.
Date

$$L(x^i, \theta, \phi) = E_z[\log p_\theta(x^i|z)] - D_{KL}(q_\phi(z|x^i) \| p_\theta(z))$$



VAE structure

Generating data



Vary z_1 / z_2 for degree of smile.

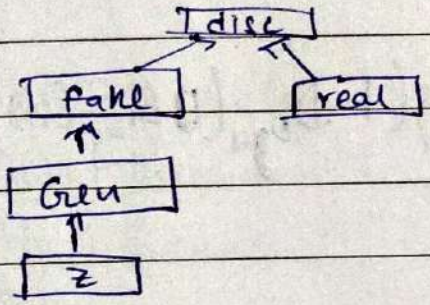
Cons: maximised lower bound of likelihood
samples blurry

NO.
Date

GANs

→ Training (2 player game)

- Generator: try to fool disc by making real img
- discriminator: check if img is real or no.



GAN structure

Train g jointly in minmax game.

MinMax func

$$\min_{\theta_g} \max_{\theta_d} \left[E_{x \sim data} \log D_{\theta_d}(x) + E_{z \sim p(z)} \log \left(1 - D_{\theta_d}(G_{\theta_g}(z))\right) \right]$$

real Data

Generated Data.

→ θ_d wants to maximize obj s.t $D(x) \sim 1$ & $D(G(x)) \sim 0$

→ θ_g wants minimize obj s.t $D(G(z)) \sim 1$ [fooling]

NO
Date

Alternate b/w

① Grad asc on disc

$$\max_{\theta_d} [E_{x \sim p(data)} \log D_{\theta_d}(x) + E_{z \sim p(z)} \log(1 - D_{\theta_g}(G_{\theta_g}(z)))]$$

② Grad descent on gtr

$$\min_{\theta_g} E_{z \sim p(z)} \log(1 - D_{\theta_g}(G_{\theta_g}(z)))$$

↳ updated obj

$$\min_{\theta_g} E_{z \sim p(z)} \log D_{\theta_d}(G_{\theta_g}(z))$$

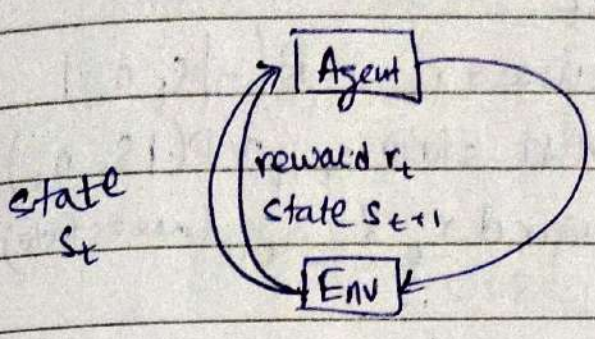
Algorithm

for no of training do
for k steps do
• Sample minibatch of m sample from noise $P_g(z)$
• Sample minibatch of m from $P_{data}(x)$
• Update the disc by ascending its stochastic descent
• Sample m noise samples from $P_g(z)$
• Update gen by stochastic descent
end for.

NO.
Date

"Lecture 14"

Reinforcement Learning



Eg

1. Cart Pole problem
2. Robot Locomotion
3. Atari Games
4. Go

How to write RL in math?

Ans Markov Decision Process

defined by, (S, A, R, P, γ)

$S \rightarrow$ set of possible states

$A \rightarrow$ set of " actions

$R \rightarrow$ dist of reward to (S, A) pair

$P \rightarrow$ dist to next state of (S, A) pair

$\gamma \rightarrow$ discount factor.

At $t=0$, env samples $s_0 \sim p(s_0)$

NO.

Date

Then, for t=0 until done:

- Agent selects a_t
- Env samples reward $r_t \sim R(\cdot|s_t, a_t)$
- " " next state $s_{t+1} \sim P(\cdot|s_t, a_t)$
- Agent gets reward r_t & s_{t+1} (next state)

A policy π is a func from $S \to A$ that tells the action to take

Obj: find π^* that max cum. disc. reward

↳ $\sum_{t \geq 0} \gamma^t r_t$

Formally,

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi\right]$$ where, $s_0 \sim p(s_0)$

$a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

Value fx^n

↳ expected cum. reward from policy from s.

$$V^\pi(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r^t | s_0 = s, \pi\right]$$

Q value fx^n

↳ expected cum. ~~function~~ reward from a in state s.

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r^t | s_0 = s, a_0 = a, \pi\right]$$

NO.

Date

Bellman Eqⁿ

Optimal Q value fxⁿ $\longrightarrow Q^*$

$$Q^*(s,a) = \max_{\pi} E\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = 0, a_0 = a, \pi\right]$$

$$Q^*(s,a) = E_{s' \sim \mathcal{E}}\left[r + \gamma \max Q^*(s', a') | s, a\right]$$

optimal policy π^* corresponds to ~~taking~~ best action acc to Q^*.

Value iteration algorithm

$$Q_{i+1}(s,a) = E\left[r + \gamma \max_{a'} Q_i(s', a') | s, a\right]$$

$Q_i \rightarrow Q^*$ as $i \rightarrow$ inf.

Q-learning

→ use func approximator for a-v func.

$$Q(s,a;\theta) \approx Q(s,a)$$

→ deep q learning (if we use DNN)

We need to find Q^* satisfying Bellman eqⁿ.

NO.

Date

Forward Pass

Loss fx^n : $L_i(\theta_i) = E_{s,a \sim p(\cdot)} [(y_i - Q(s,a;\theta_i))^2]$

where, $y_i = E_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})]$

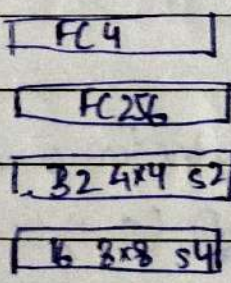
Backward Pass

$\nabla_{\theta_i} L_i(\theta_i)$

$= E_{\theta s, a \sim p(\cdot); s' \sim E} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s,a;\theta_i)) \nabla_{\theta_i} \theta(s,a;\theta_i)]$

Net architecture (Atari)

$Q(s,a;\theta)$



$S_t \Rightarrow 84 \times 84 \times 4$

NO.

Date

Train the Q network by exp relay

- continually update memory table of transitions (s_t, a_t, r_t, s_{t+1}) as eps are played
- train Q network on minibatches of transitions (random) instead of consecutive samples.

Policy gradients

$$\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$$

Let $J(\theta) = E\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$

$$\theta^* = \arg\max_\theta J(\theta) = ?$$

Reinforce Algo

$$J(\theta) = E_{\tau \sim p(\tau;\theta)}[r(\tau)]$$

$$= \int_\tau r(\tau) p(\tau;\theta) d\tau$$

where $r(\tau)$ is reward of trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

$$\nabla_\theta J(\theta) = \int_\tau r(\tau) \nabla_\theta p(\tau;\theta) d\tau$$

$$\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla \log(\tau;\theta)$$

NO.
Date

So, $\nabla_\theta J(\theta) = E_{T\sim p(T;\theta)}\left[r(T)\nabla_\theta \log p(T;\theta)\right]$

We know, $p(T;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\,\pi_\theta(a_t|s_t)$

$$\log p(T;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(T)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

→ if $r(T)$ high, push up prob
→ " " low, pull down "

Variance Reduction

① $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0}\left(\sum_{t'\geq t} r_{t'}\right)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

② $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0}\left(\sum_{t'\geq t} \gamma^{t'-t} r_{t'}\right)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

③ Baseline

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0}\left(\sum_{t'\geq t} \gamma^{t'-t} r_{t'} - b(s_t)\right)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

NO.
Date

Actor-Critic Algorithm

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

for it = 1, 2, ... do

$\Delta\theta \leftarrow \theta$

for i = 1, 2, .. m do

for t = 1, 2, .. T do

$$A_t = \sum_{t'>t} \gamma^{t'} r_{t'}^i - V_\phi(s_t^i)$$

$$\Delta\theta \leftarrow \theta + A_t \nabla_\theta \log(a_t^i | s_t^i)$$

$$\Delta\phi \leftarrow \sum_i \sum_t \nabla_\phi \|A_t^i\|^2$$

$\theta \leftarrow \alpha \Delta\theta$

$\phi \leftarrow \beta \Delta\phi$

end for

Recurrent Attention Model (RNN)

↳ uses Reinforce Algo