COMP9444 Neural Networks and Deep Learning

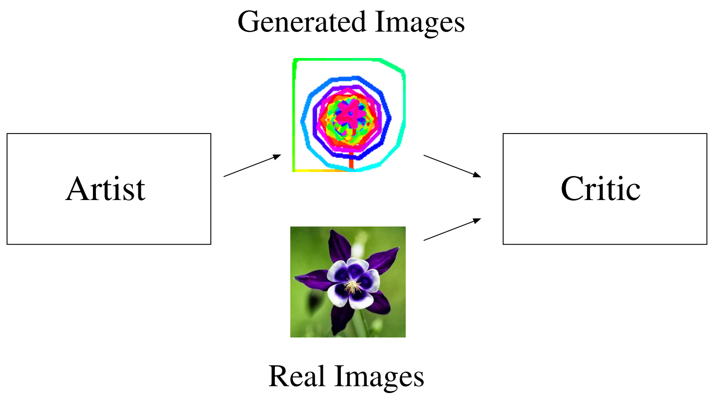
14: Adversarial Training and GANs

Alan Blair, UNSW, 2017

Outline

- Artist-Critic Co-Evolution
- Co-Evolution Paradigms
- Blind Watchmaker (GP Artist, Human Critic)
- Evolutonary Art (GP Artist, GP or NN Critic)
- Generative Adversarial Networks (CNN Artist, CNN Critic)

Artist-Critic Co-Evolution

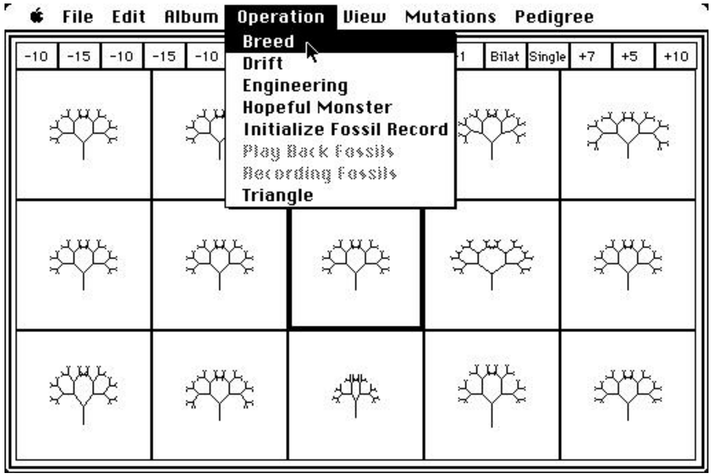


- Critic is rewarded for distinguishing real images from those generated by the artist.
- Artist is rewarded for fooling the critic into thinking that generated images are real.

Co-Evolution Paradigms

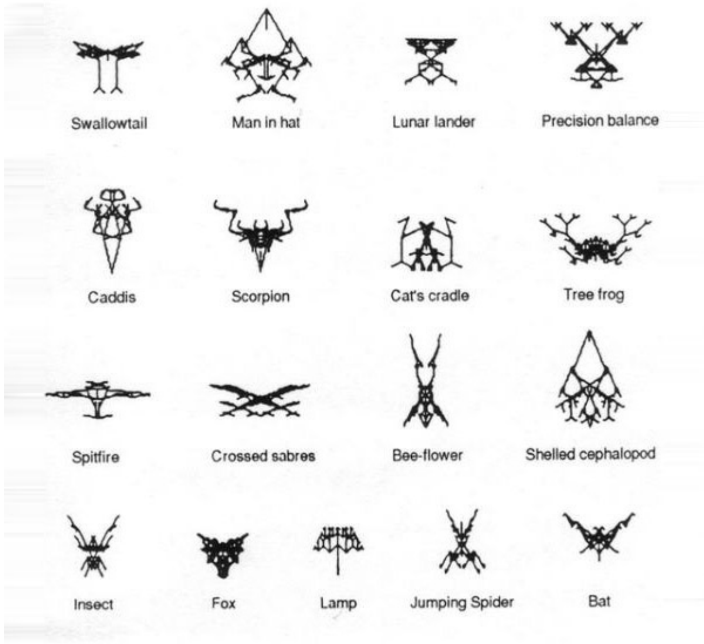
Artist	Critic	Method	Reference
Biomorph	Human	Blind Watchmaker	(Dawkins, 1986)
GP	Human	Blind Watchmaker	(Sims, 1991)
CPPN	Human	PicBreeder	(Secretan, 2011)
CA	Human	EvoEco	(Kowaliw, 2012)
GP	SOM	Artificial Creativity	(Saunders, 2001)
Photo	NN	Computational Aesthetics	(Datta, 2006)
GP	NN	Computational Aesthetics	(Machado, 2008)
Agents	NN	Evolutionary Art	(Greenfield, 2009)
GP	NN	Aesthetic Learning	(Li & Hu, 2010)
HERCL	HERCL	Co-Evolving Line Drawings	(Vickers, 2017)
HERCL	DCNN	HERCL Function/CNN	(Soderlund)
DCNN	DCNN	Generative Adversarial Nets	(Goodfellow, 2014)
DCNN	DCNN	Plug & Play Generative Nets	(Nguyen, 2016)

Blind Watchmaker (Dawkins, 1986)



- the User is presented with 15 images
- the chosen individual is used to breed the next generation

Blind Watchmaker Biomorphs



Blind Watchmaker (Sims, 1991)

- Artist = Genetic Program (GP)
  - used as function to compute R,G,B values for each pixel x, y
- Critic = Human

PicBreeder Examples

PicBreeder (Secretan, 2011)

- Artist = Convolutional Pattern Producing Neural Network (CPPN)
- Critic = Human
- interactive Web site (`picbreeder.org`) where you can choose existing individual and use it for further breeding

---

- Blind Watchmaker paradigm is cool, but it may require a lot of work from the Human
- Can the Human be replaced by an automated Critic?

Evolutionary Art

- Artist = Genetic Program (GP or HERCL)
  - artist used as a function to compute R,G,B values for each pixel location x, y
  - alternatively, artist issues a series of drawing instructions
- Critic = GP (evolution) or Neural Network (backpropagation)
- Critic is presented with "real" images from a training set, and "fake" images generated by the Artist
- Critic is trained to produce output close to 1 for real images and close to 0 for generated images (or vice-versa)
- inputs to Critic
  - small number of statistical features extracted from the image
  - more recently, raw image, fed to DCNN

Statistical Image Features

Feature	Abbreviation	Source
Mean	M^H, M^S, M^V	D, M
Standard deviation	S^H, S^S, S^V	D, M
Greyscale entropy	H	M
Mean edge weight	M_E	M
Standard deviation of edge weight	S_E	M
Number of homogenous patches	N_P	D
Mean of largest patch	P_1^H, P_1^S, P_1^V	D
Mean of 2nd-largest patch	P_2^H, P_2^S, P_2^V	D
Mean of 3rd-largest patch	P_3^H, P_3^S, P_3^V	D
Mean of 4th-largest patch	P_4^H, P_4^S, P_4^V	D
Mean of 5th-largest patch	P_5^H, P_5^S, P_5^V	D

[D = Datta et al., 2006]   [M = Machado et al., 2008]

Image Features

Feature	Abbreviation	Source
Size of largest patch	A_1	D
Size of 2nd-largest patch	A_2	D
Size of 3rd-largest patch	A_3	D
Size of 4th-largest patch	A_4	D
Size of 5th-largest patch	A_5	D
Convexity factor	C	D
Mean corner weight	M_C	-
Number of corners	N_C	-

[D = Datta et al., 2006]   [M = Machado et al., 2008]

Hierarchical Evolutionary Re-Combination Language (HERCL)

```
     INPUT:   ickey
    OUTPUT:
    MEMORY:   Minnie...............................
 REGISTERS:   .....[6]..[1].  [7]
     STACK:   MM
      CODE:   0[is|.<sy^5>};i|8{^s-~:+7=;wo8|-wo]
                                      ^
```

- combines elements from Linear GP and Stack-based GP.
- programs have access to a stack, registers and memory.
- each instruction is a single character, possibly preceded by a numerical (or dot) argument.

HERCL Commands

Input and Output

`i`	fetch INPUT to input buffer	
`s`	SCAN item from input buffer to stack	
`w`	WRITE item from stack to output buffer	
`o`	flush OUTPUT buffer	

Stack Manipulation and Arithmetic

`#`	PUSH new item to stack	$\ldots\ldots \mapsto \ldots\ldots x$
`!`	POP top item from stack	$\ldots\ldots x \mapsto \ldots\ldots$
`c`	COPY top item on stack	$\ldots\ldots x \mapsto \ldots\ldots x, x$
`x`	SWAP top two items	$\ldots y, x \mapsto \ldots x, y$
`y`	ROTATE top three items	$z, y, x \mapsto x, z, y$
`-`	NEGATE top item	$\ldots\ldots x \mapsto \ldots..(-x)$
`+`	ADD top two items	$\ldots y, x \mapsto \ldots(y+x)$
`*`	MULTIPLY top two items	$\ldots y, x \mapsto \ldots(y*x)$

HERCL Commands

Mathematical Functions

`r`	RECIPROCAL	$..x \to ..1/x$
`q`	SQUARE ROOT	$..x \to ..\sqrt{x}$
`e`	EXPONENTIAL	$..x \mapsto ..e^x$
`n`	(natural) LOGARITHM	$..x \mapsto ..\log_e(x)$
`a`	ARCSINE	$..x \mapsto ..\sin^{-1}(x)$
`h`	TANH	$..x \mapsto ..\tanh(x)$
`z`	ROUND to nearest integer	
`?`	push RANDOM value to stack	

Double-Item Functions

`%`	DIVIDE/MODULO	$..y, x \mapsto ..(y/x), (y \bmod x)$
`t`	TRIG functions	$..\theta, r \mapsto ..r\sin\theta, r\cos\theta$
`p`	POLAR coords	$..y, x \mapsto ..\mathrm{atan2}(y,x), \sqrt{x^2+y^2}$

HERCL Commands

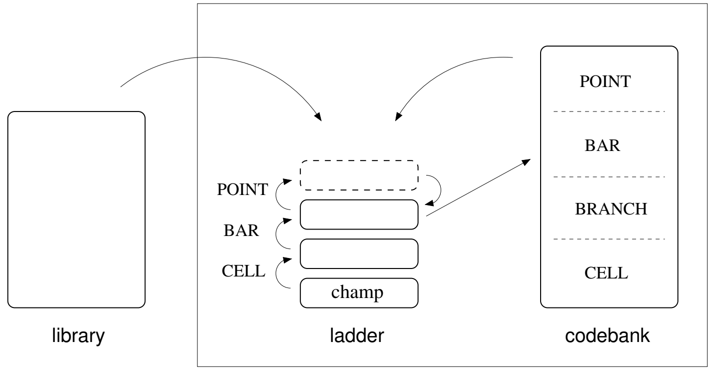
Registers and Memory

- `<` GET value from register
- `>` PUT value into register
- `^` INCREMENT register
- `v` DECREMENT register
- `{` LOAD from memory location
- `}` STORE to memory location

Jump, Test, Branch and Logic

- `j` JUMP to specified cell (subroutine)
- `|` BAR line (RETURN on `.|` HALT on `8|`)
- `=` register is EQUAL to top of stack
- `g` register is GREATER than top of stack
- `:` if TRUE, branch FORWARD
- `;` if TRUE, branch BACK
- `&` logical AND `/` logical OR `~` logical NOT

Hierarchical Evolutionary Re-Combination



- large crossover/mutation can be followed up by smaller ones.
- if top agent becomes fitter, it moves down to replace the one below it (which is moved to the codebank).
- if top agent exceeds max number of offspring, it is removed.
- good for co-evolution because it keeps the number of competing agents small while preserving diversity.

Line Drawing Commands

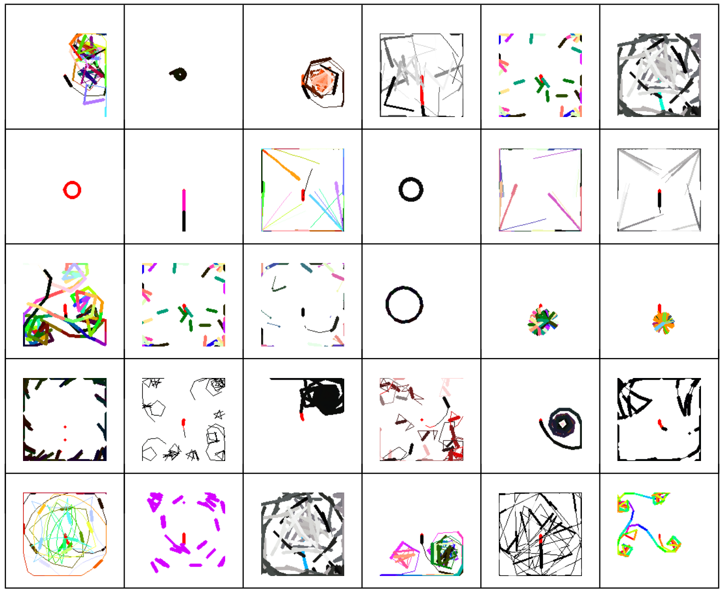
0	TOGGLE		lift pen on/off page
1	MOVE	x	move pen forward by x pixels ($0 \leq x \leq 15$)
2	TURN	x	turn x degrees clockwise
3	SIZE	p	set pen radius to p pixels ($1 \leq p \leq 4$)
4	COLOUR	v	set greyscale value [greyscale mode]
	COLOUR	$l\ h\ s$	set colour in HSV colour space [colour mode]

- the output from the HERCL program is interpreted as a series of line drawing commands

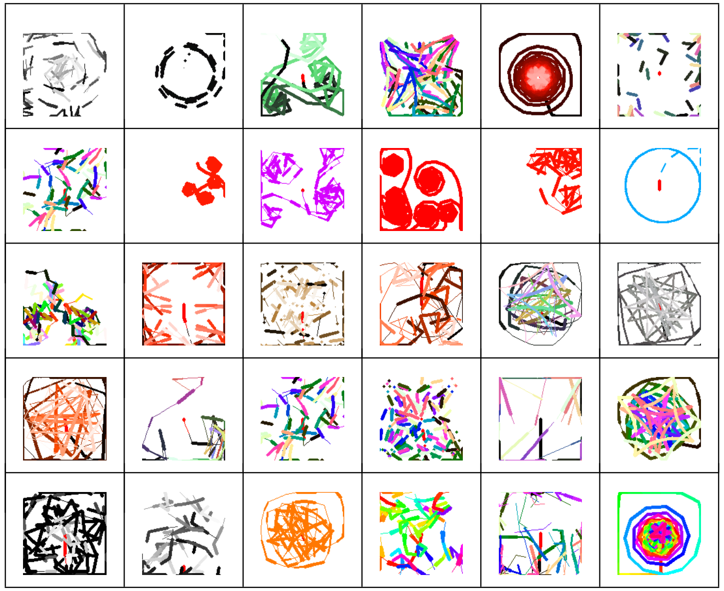
Experimental Details

- 10 artists and 3 critics per iteration (evolved independently)
- features are extracted from the image and fed to the critic
- target value is 1 for real images and 0 for generated images
- cost for the critic is cross-entropy error
- critic is successful when cost < 0.1 per image
- successful artist code from previous generations goes to library
- cost for artist is weighted sum of critics from all previous generations, with older critics weighted less
- artist is successful when cost < 0.1
- "real" images obtained from Google search for "Circle"

Evolved Images (Generation 3,5,7,8,9)



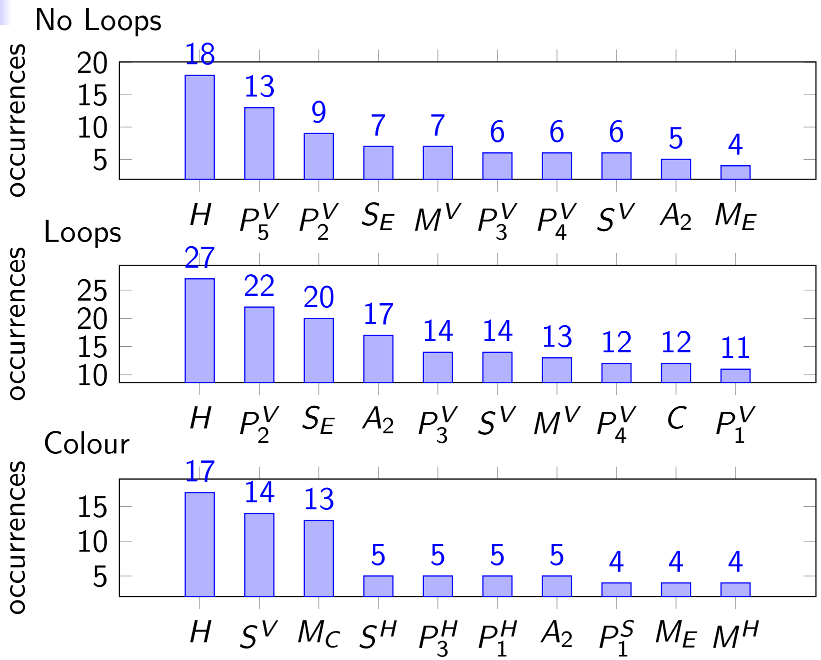
Evolved Images (Generation 11,13,15,17,18)



Evolved Artist Code

```
[1.165#w8.#ewo7.#.vw1.#<.6#-g!/:w2.#.gaw17#.|ww9.#
  0vw8.#g21.#w15.#o<ww8vo<v0<a15.#aw17#ww.vo<<v7<+w;]
[5^^^<w3=!/:.>g:w:|5^^^<w<ocw^wo9^.g<w<w;]
[^p<+wc<w<o11.#-w<wopc+w5.#-8.#wc<o2.#-w<ow;]
[gv<7g:~1.#-2.#%w<<w^ow^<^w.^7.#~.#2.#%yw<w^ow^<^w
  4^o<6v;w8.#w7.#|ww<ww<ow<7g6.#-o!w15.#vwo<-w;:]
[x9.#-1.#-|ww9.#0vw8.#g21.#w15.#o<ww8vo<v0<a15.#
  aw17#ww.vo<<v7<+w;]
[<v<wg<wo.#w15.#vww<w7v<%<7g6.#-o!w15.#vwo<-w;o]
[ww9.#0vw8.#g21.#w15.#o<ww8vo<v0<a15.#aw17#ww.vo<<v7<+w;]
[5^^^2<w<ocw^wo9^<w<w;gw5.#-7.#w]
[<c<wvwo<-<wcw<o10.#-p7.#eww<w&p+w9};]
[3.#a1^6<woww.<!ow^11.#-9=g~<7=iw{^<ccwa1^<ww7<!w
  ^<c<w0<!oww^&c</^<^w15.#;]
```

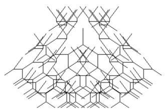
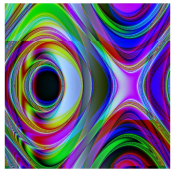
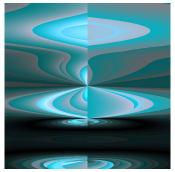
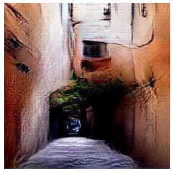

Image Features most used by Critics



Evolved Critic Code

```
[29<5g:wo8|1<6=hg~<:r1:|cttt>88.#-.164#g~!:<31<+a%|awo]
[6<qaawo]
[6<.168#4>g-1:q5g~:31<6<!z<+-p1g:<|c=!:2}
  !1{h2g~q:31<|yywo]
[6<0g:ewo.|31<%q21g~:8<25gr:-|5<x4g~:n|a+awo]
[8<18g~:22<qqazwo8|4<7g:8<|31<}%{t+a++1{+wo]
[1<h6g13<+:5<t31<a+wo.|ewo]
[20<19<>g2=/:q<+qzwo.|.56#-23<+<+31<29gx:6<+
  r+wo.|6<+22<+pwo]
[1g:{wo8|<6<xhg%:26<6<az21g:-wo.|++15<pwo]
[6<nr<aa>z<1ga%~:<zx31g:{26<|pwo]
[6<aa1g~:31<5<19gx:xa12g:zwo.|1g=/~:qn|zewo]
```

Image Generating Paradigms

 Biomorph	 GP	 Picbreeder	 CA
 HERCL(draw)	 HERCL(func)	 HERCL(func)	 GAN

Generative Adversarial Networks

Generator (Artist) G_θ and Discriminator (Critic) D_ψ are both Deep Convolutional Neural Networks.

Generator $G_\theta : z \mapsto x$, with parameters θ, generates an image x from latent variables z (sampled from a Normal distribution).

Discriminator $D_\psi : x \mapsto D_\psi(x) \in (0,1)$, with parameters ψ, takes an image x and estimates the probability of the image being real.

Generator and Discriminator play a 2-player zero-sum game to compute:

$$\min_\theta \max_\psi \Big(\mathbf{E}_{x \sim p_{\text{data}}} \big[\log D_\psi(x) \big] + \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z))\big) \big] \Big)$$

Discriminator tries to maximize the bracketed expression, Generator tries to minimize it.

Generative Adversarial Networks

Alternate between:

Gradient ascent on Discriminator:

$$\max_\psi \Big(\mathbf{E}_{x \sim p_{\text{data}}} \big[\log D_\psi(x) \big] + \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z))\big) \big] \Big)$$

Gradient descent on Generator, using:

$$\min_\theta \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z))\big) \big]$$

Generative Adversarial Networks

Alternate between:

Gradient ascent on Discriminator:

$$\max_\psi \Big(\mathbf{E}_{x \sim p_{\text{data}}} \big[\log D_\psi(x) \big] + \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z))\big) \big] \Big)$$

Gradient descent on Generator, using:

$$\cancel{\min_\theta \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z))\big) \big]}$$

This formula puts too much emphasis on images that are correctly classified. Better to do gradient ascent on Generator, using:

$$\max_\theta \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(D_\psi(G_\theta(z))\big) \big]$$

This puts more emphasis on the images that are wrongly classified.

Generative Adversarial Networks

GAN's differ from previous approaches (Evolutionary Art) in that:

- there is no need for a population; one network produces the full range of images x, with different values for the latent variables z
- differentials are backpropagated through the Discriminator network and into the Generator network
- the images produced are much more realistic!

Generative Adversarial Networks

repeat:

  for k steps do

    sample minibatch of m latent samples $\{z^{(1)}, \dots, z^{(m)}\}$ from $p(z)$

    sample minibatch of m training items $\{x^{(1)}, \dots, x^{(m)}\}$

    update Discriminator by gradient ascent on ψ:

$$\nabla_\psi \frac{1}{m} \sum_{i=1}^{m} \left[\log D_\psi(x^{(i)}) + \log\left(1 - D_\psi(G_\theta(z^{(i)}))\right)\right]$$

  end for

  sample minibatch of m latent samples $\{z^{(1)}, \dots, z^{(m)}\}$ from $p(z)$

  update Generator by gradient ascent on θ:

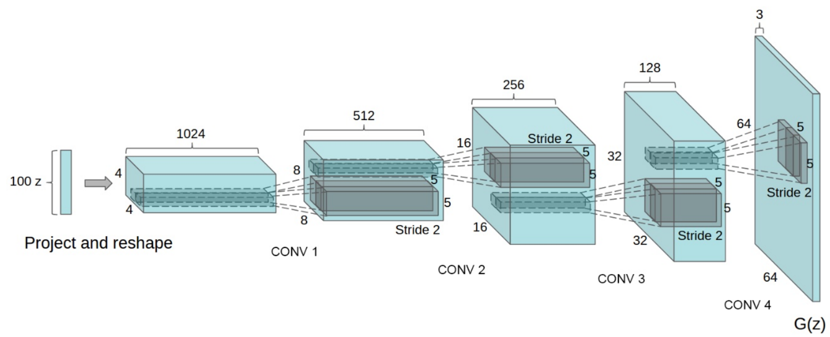
$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(D_\psi(G_\theta(z^{(i)}))\right)$$

end repeat

GAN Convolutional Architectures

- normalize images to between -1 and $+1$
- replace pooling layers with:
  - strided convolutions (Discriminator)
  - fractional-strided convolutions (Generator)
- use BatchNorm in both Generator and Discriminator
- remove fully connected hidden layers for deeper architectures
- use tanh at output layer of Generator, ReLU activation in all other layers
- use LeakyReLU activation for all layers of Discriminator

Generator Architecture

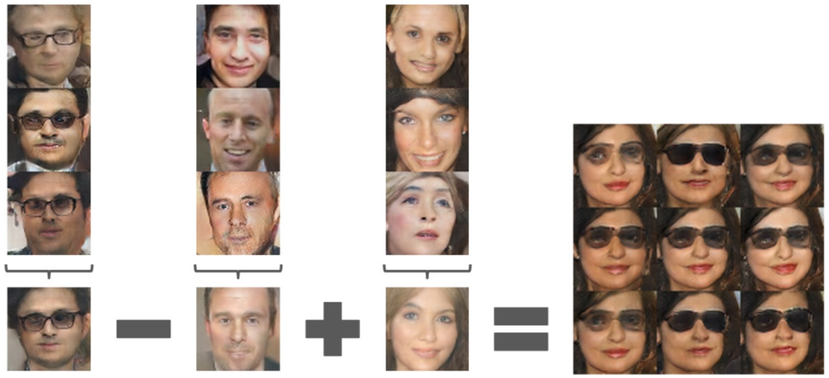


Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford et al., 2016)

GAN Generated Images

GAN Image Vector Arithmetic

GAN Image Vector Arithmetic



Mediocre Stable States

- Like any coevolution, GANs can sometimes oscillate or get stuck in a mediocre stable state.
- **oscillation**: GAN trains for a long time, generating a variety of images, but quality fails to improve (compare IPD)
- **mode collapse**: Generator produces only a small subset of the desired range of images, or converges to a single image (with minor variations)

Methods for avoiding mode collapse:

- Conditioning Augmentation
- Minibatch Features (Fitness Sharing)
- Unrolled GANs

The GAN Zoo

- Contex-Encoder for Image Inpainting
- Texture Synthesis with Patch-based GAN
- Conditional GAN
- Text-to-Image Synthesis
- StackGAN
- Patch-based Discriminator
- S^2-GAN
- Style-GAN
- Plug-and-Play Generative Networks

GAN References

`http://dl.ee.cuhk.edu.hk/slides/gan.pdf`

`cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf`

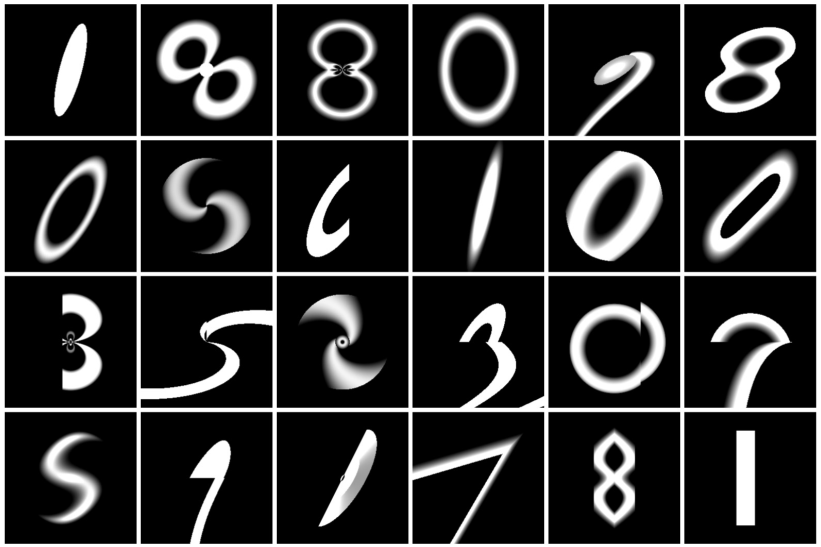
`http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf`

`https://arxiv.org/abs/1612.00005`

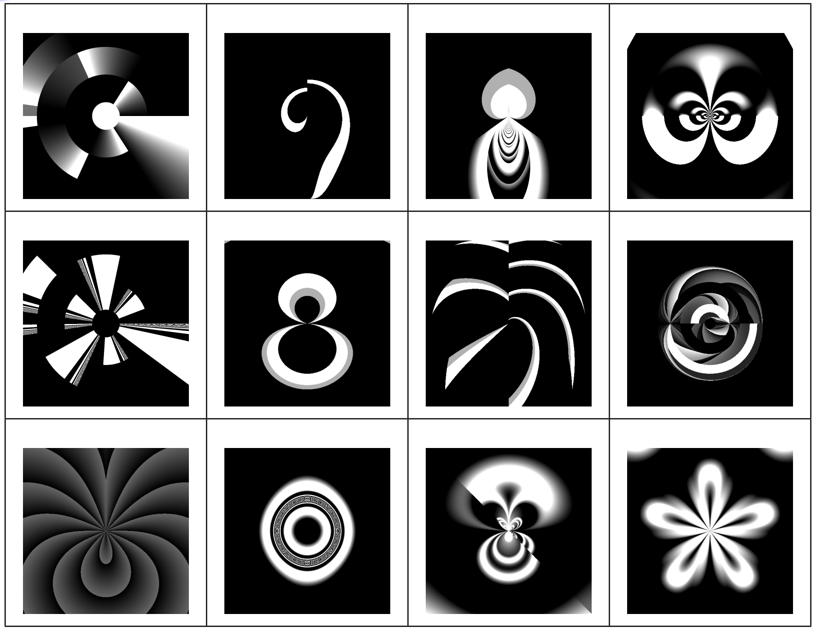
HERCL Artist, CNN Critic

- Artist = HERCL program, used as function to produce R,G,B values from pixel location x, y
- Critic = Deep CNN (LeNet or simplified AllConv Network)
- Alternate between evolution of Artist and gradient descent for Critic
- Artist evolved to fool current Critic
- Critic trained by backpropagation to distinguish real images from those produced by all previous Artists

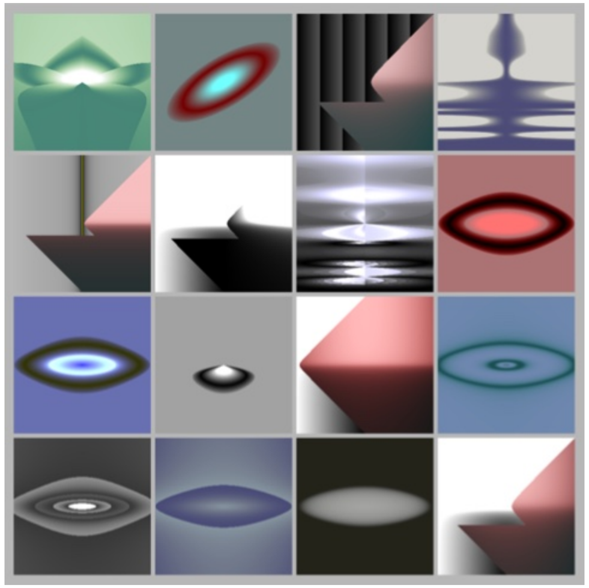
Images trained against MNIST



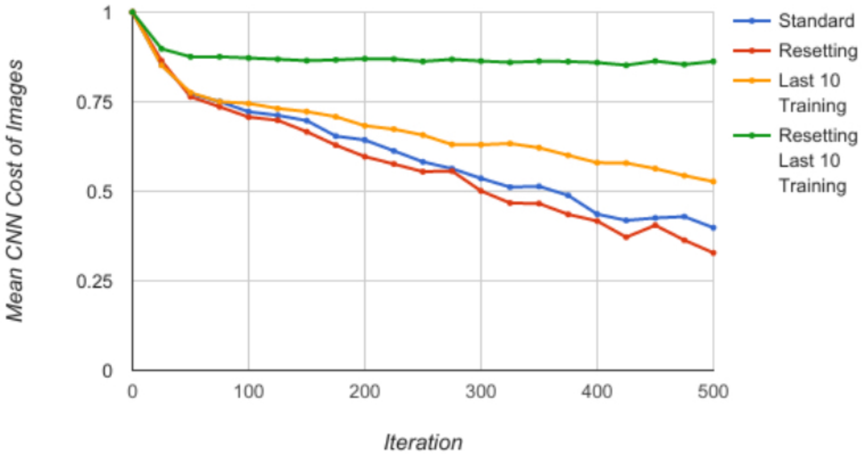
Images trained against MNIST



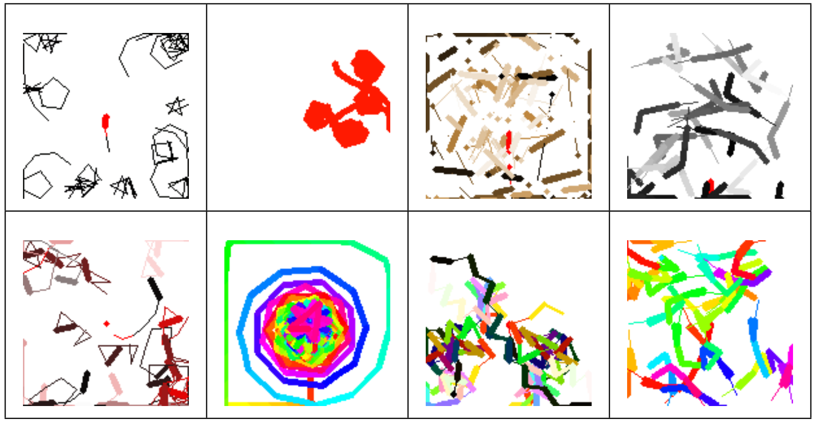
Images trained against CIFAR-10 Boats

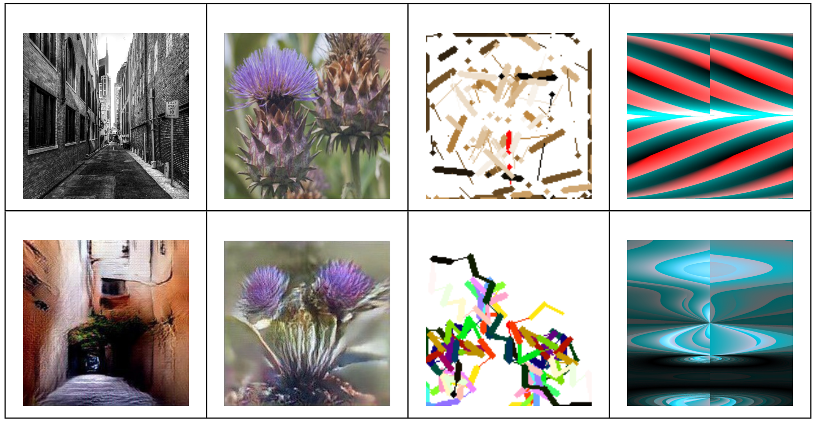


Coevolutionary Dynamics



Self-Similarity, Low Complexity Art





Questions ?