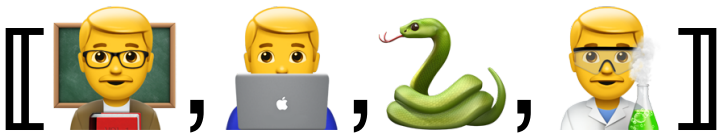
Lecture Notes for

Machine Learning in Python



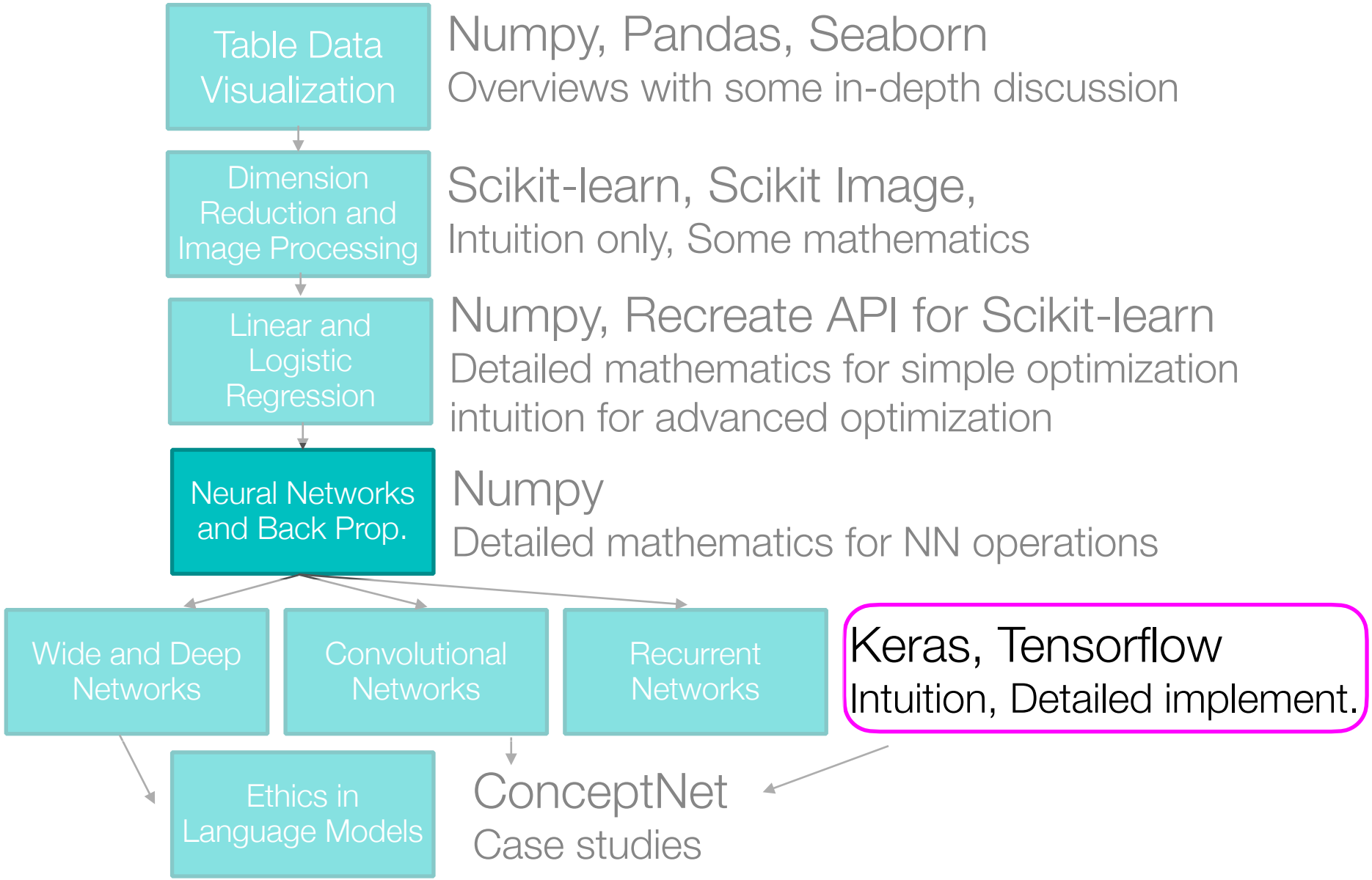
Revisiting Cross Validation

A “Not so Early” History of Deep Learning

Logistics and Agenda

- Logistics
  - Grading update
- Agenda
  - Revisiting Cross Validation
  - Town Hall
  - “Deep Learning” History

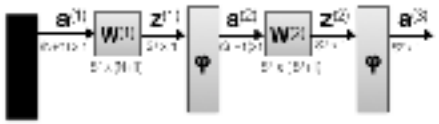
Class Overview, by topic



Last time:

Back propagation summary

$$J(\mathbf{W}) = \sum_{k}^{M} (\mathbf{y}^{(k)} - \mathbf{a}^{(L)})^2$$



$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} a_j^{(l)}$$

1. Forward propagate to get $\mathbf{z}$, $\mathbf{a}$ for all layers
2. Get final layer gradient
3. Update back propagation variables
4. Update each $\mathbf{W}^{(l)}$

for each $\mathbf{y}^{(i)}$

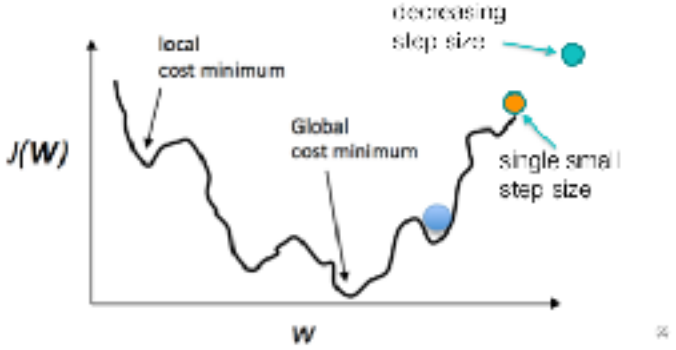
$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = -2(\mathbf{y}^{(i)} - \mathbf{a}^{(3)}) * \mathbf{a}^{(3)} * (1 - \mathbf{a}^{(3)})$$

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} = \text{diag}[\mathbf{a}^{(l+1)} * (1 - \mathbf{a}^{(l+1)})] \cdot \mathbf{W}^{(l+1)} \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l+1)}}$$

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial J(\mathbf{W}^{(l)})}{\partial \mathbf{z}^{(l)}} \cdot \mathbf{a}^{(l)}$$

Problems with Advanced Architectures

- Space is no longer convex
  - **One solution:**
    - start with large step size
    - "cool down" by decreasing step size for higher iterations



Practical Implementation of Architectures

- A new cost function: **Cross entropy**

$$J(\mathbf{W}) = -[\mathbf{y}^{(i)} \ln \mathbf{a}^{(L)} + (1 - \mathbf{y}^{(i)}) \ln(1 - \mathbf{a}^{(L)})]$$

speeds up initial training

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(L)}} = (\mathbf{a}^{(L+1)} - \mathbf{y}^{(i)})$$

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = (\mathbf{a}^{(3)} - \mathbf{y}^{(i)})$$

```
# vectorized backpropagation
sigma3 = (A3-Y_enc) # <- this is only line
sigma2 = (W2.T @ sigma3)*A2*(1-A2)

grad1 = sigma2[1:,:] @ A1
grad2 = sigma3 @ A2.T
```

new update

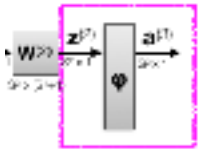
```
# vectorized backpropagation
sigma3 = -2*(Y_enc-A3)*A3*(1-A3)
sigma2 = (W2.T @ sigma3)*A2*(1-A2)

grad1 = sigma2[1:,:] @ A1
grad2 = sigma3 @ A2.T
```

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = -2(\mathbf{y}^{(i)} - \mathbf{a}^{(3)}) * \mathbf{a}^{(3)} * (1 - \mathbf{a}^{(3)})$$ old update

tp=5

Practical Implementation of Architectures

- A new nonlinearity: **recitifed linear units**

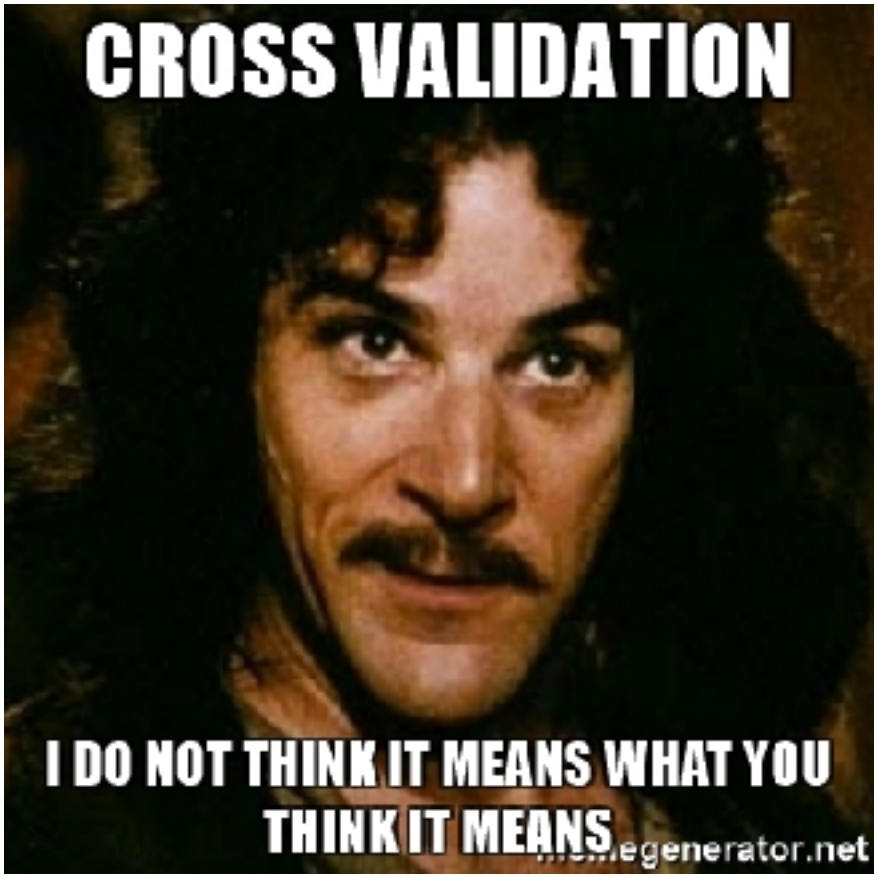


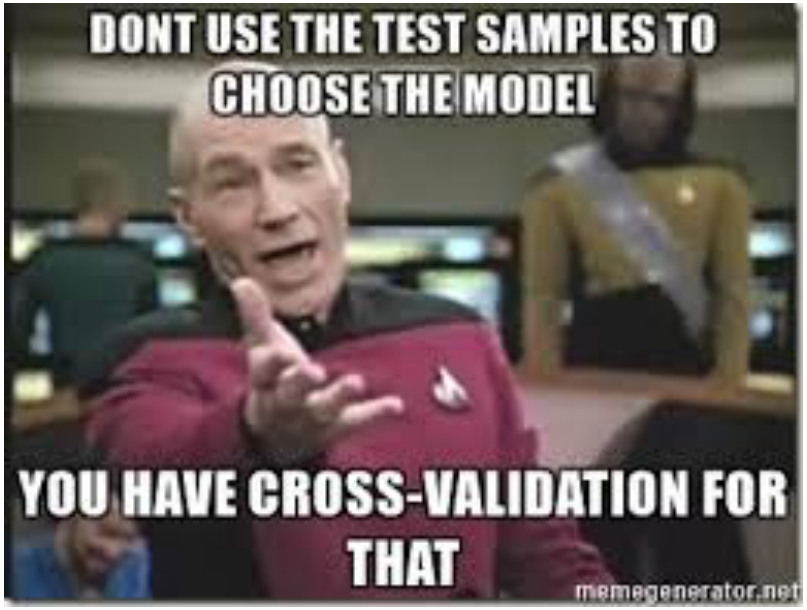
$$\phi(\mathbf{z}^{(i)}) = \begin{cases} \mathbf{z}^{(i)}, \text{ if } \mathbf{z}^{(i)} > 0 \\ \mathbf{0}, \text{ else} \end{cases}$$

it has the advantage of **large gradients** and **extremely simple** derivative

$$\frac{\partial \phi(\mathbf{z}^{(i)})}{\partial \mathbf{z}^{(i)}} = \begin{cases} \mathbf{1}, \text{ if } \mathbf{z}^{(i)} > 0 \\ \mathbf{0}, \text{ else} \end{cases}$$

Revisiting Cross Validation

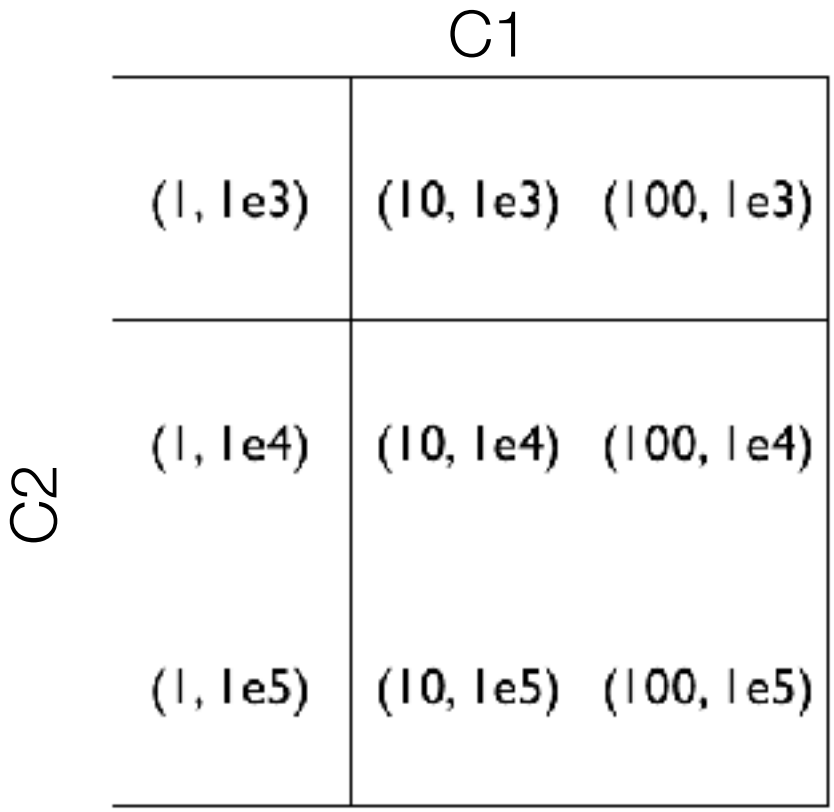




Review: Grid Searching

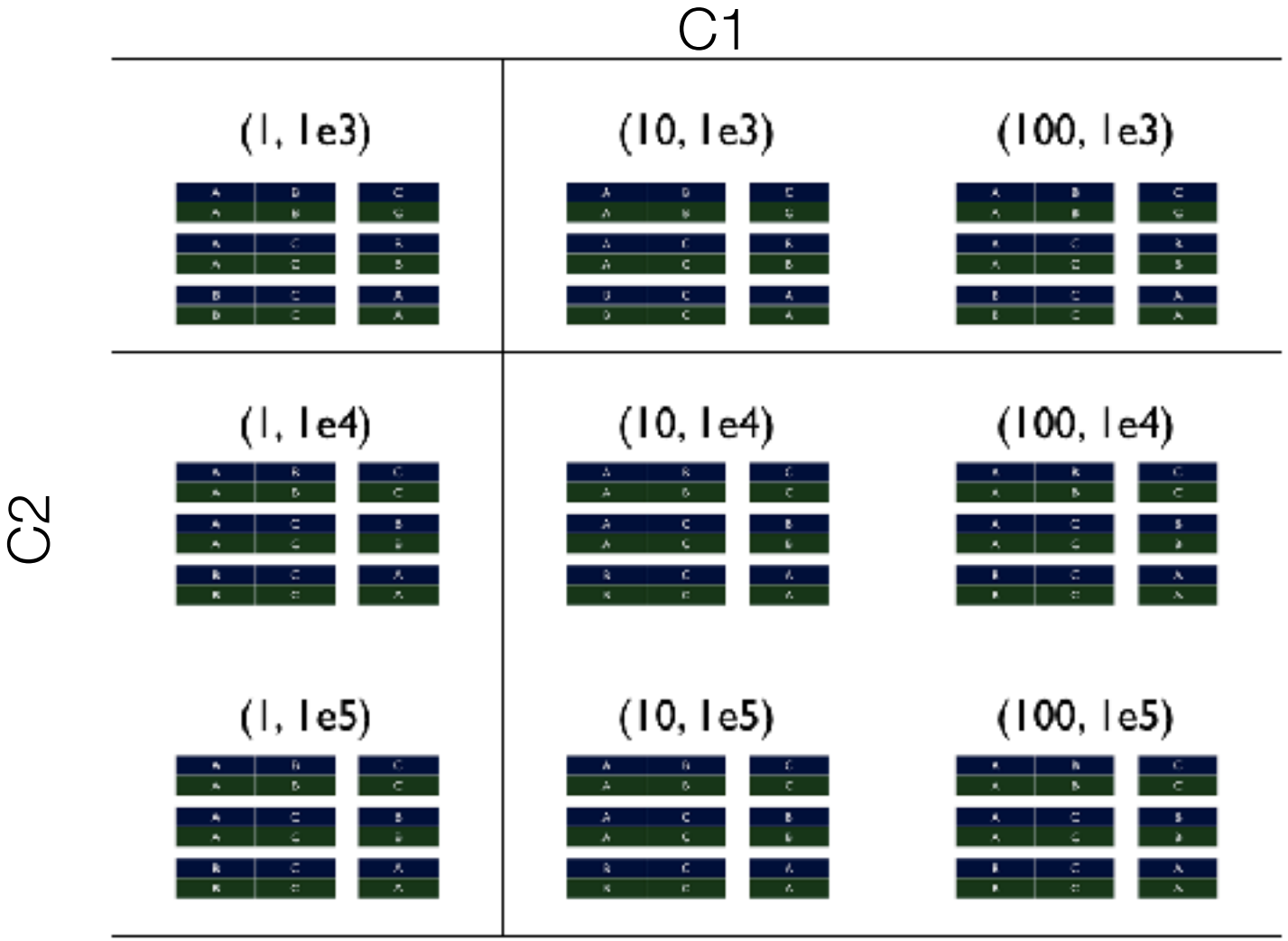
Trying to find the best parameters

```
NN: C1=[1, 10, 100] C2=[1e3,1e4,1e5]
```



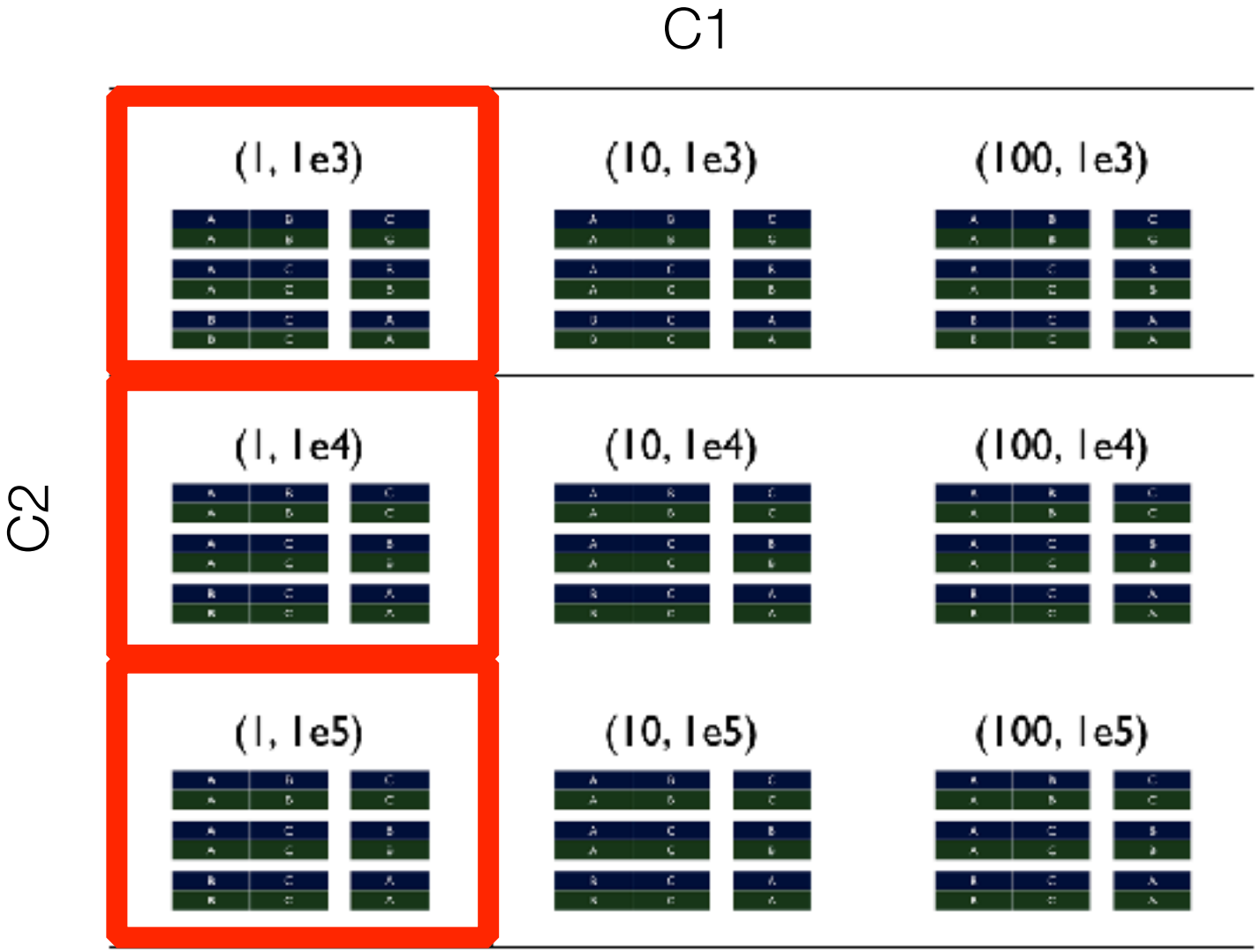
Review: Grid Searching

For each value, want to run cross validation…



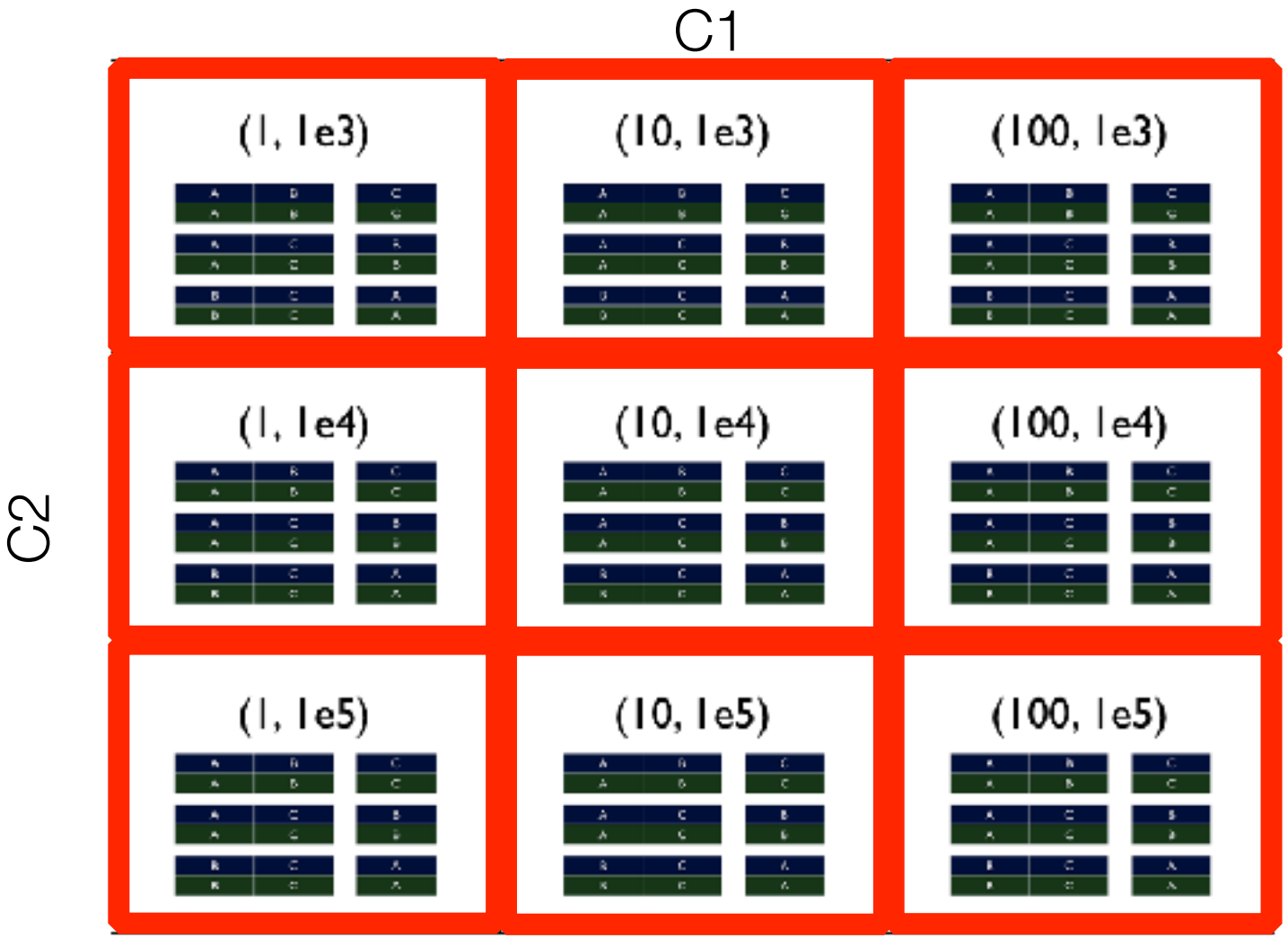
Review: Grid Searching

Could perform iteratively



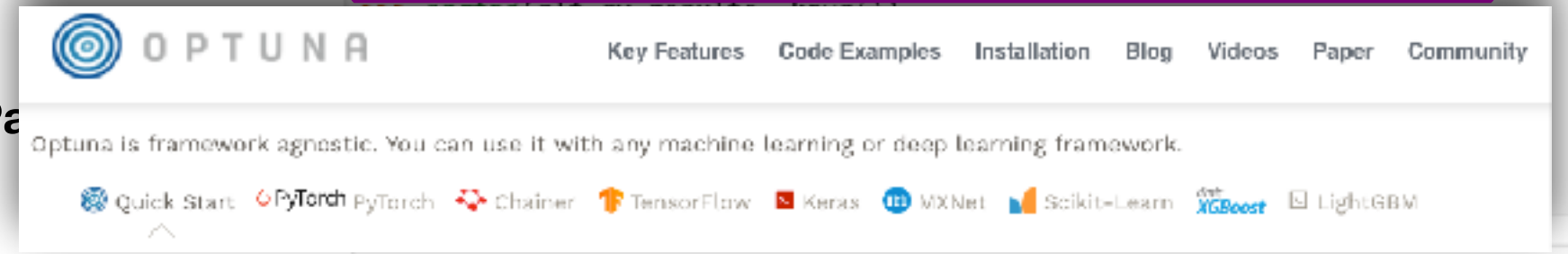
Review: Grid Searching

or at random…



Review: Grid Searches in Scikit-learn

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC()
>>> clf = GridSearchCV(svc, parameters)
>>> clf.fit(iris.data, iris.target)
GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 10], 'kernel': ('linear', 'rbf')})
```



Pa

-
-

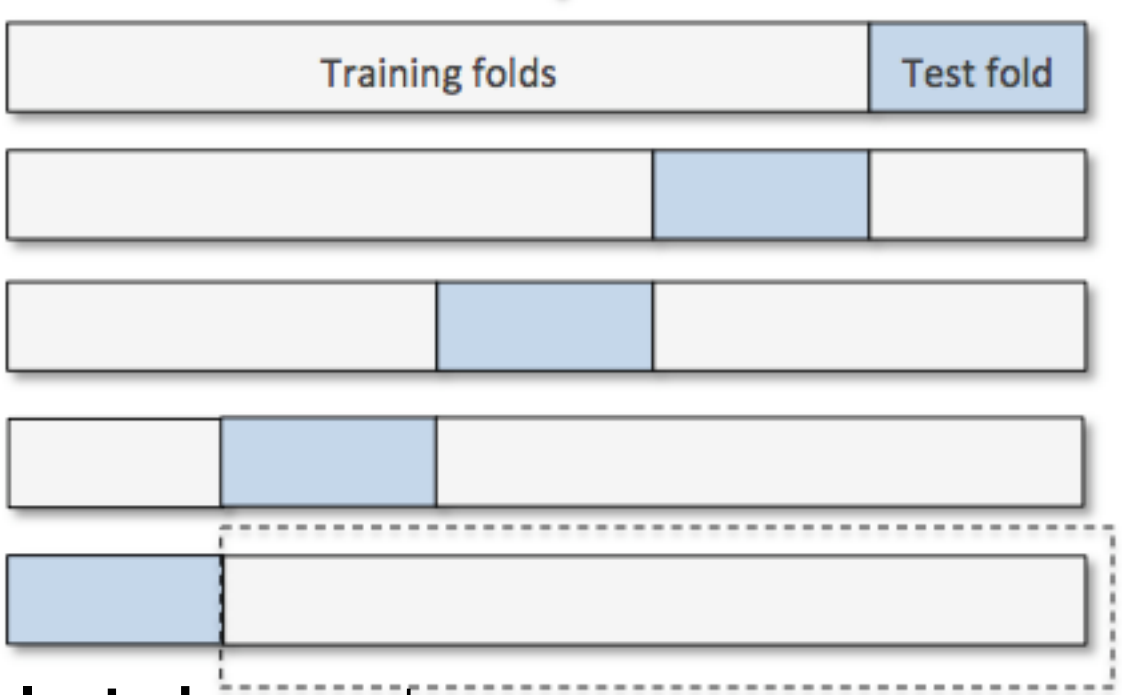
values, sampled

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
...                               random_state=0)
>>> distributions = dict(C=uniform(loc=0, scale=4),
...                      penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions, random_state=0)
>>> search = clf.fit(iris.data, iris.target)
>>> search.best_params_
{'C': 2..., 'penalty': 'l1'}
```

Review: Self Test

- Using the grid search parameters and testing on the same set…
- Is this **data snooping**?
  - A. True, this is snooping because it uses test set to define parameters
  - B. True, this is snooping because we can no longer reliably define the expected performance on new data
  - C. False, this is not snooping because we still separated train and test data
  - D. False, this is not snooping because hyper parameters are not trainable

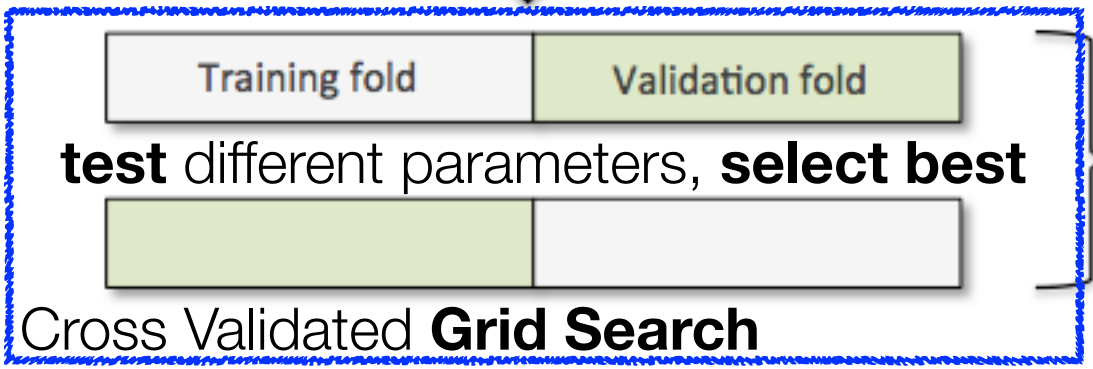
A Costly Solution: Nested Cross Validation



Outer loop

Train with optimal parameters

use **selected** parameters

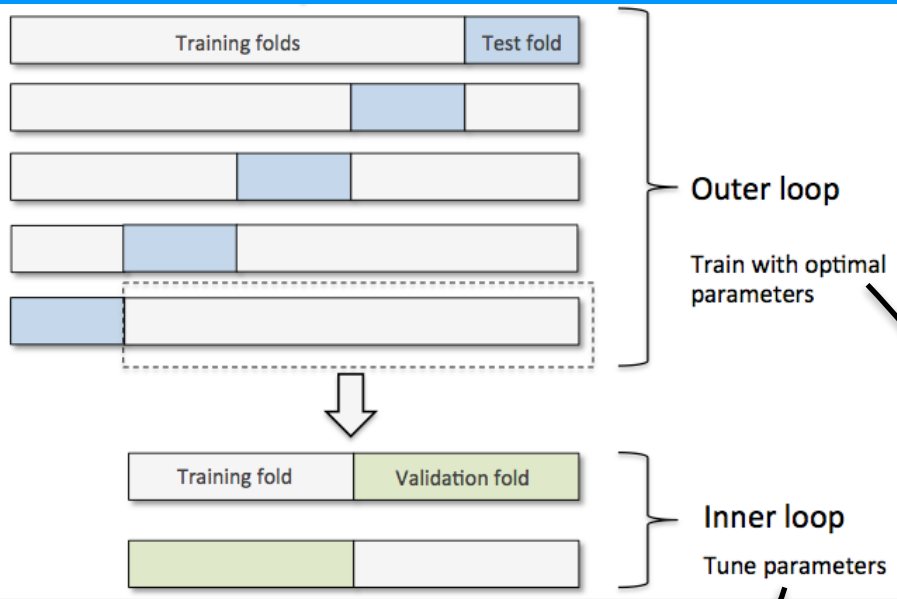


Inner loop

Tune parameters

https://github.com/rasbt/python-machine-learning-book/

Review: Nested Cross Validation: Hyper-parameters




```
gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=2)

# Note: Optionally, you could use cv=2
# in the GridSearchCV above to produce
# the 5 x 2 nested CV that is shown in the figure.

scores = cross_val_score(gs, X_train, y_train, scoring='accuracy', cv=5)
print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

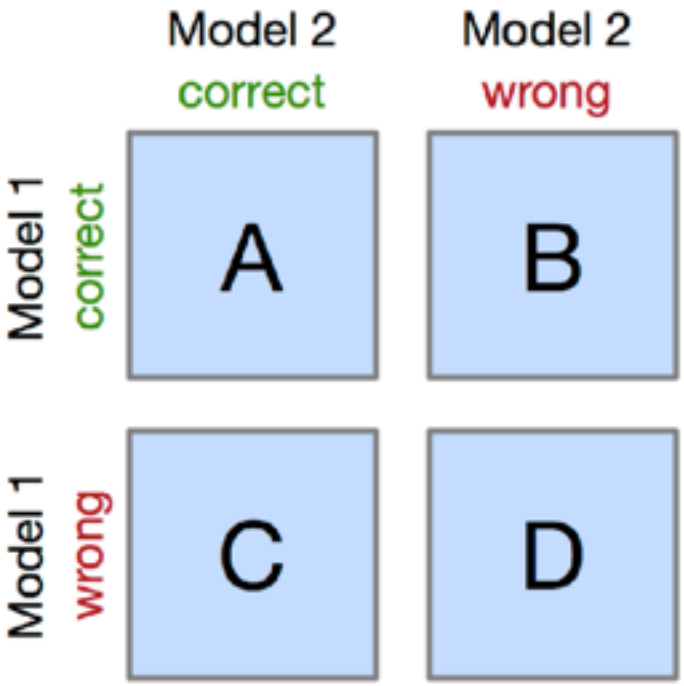
https://github.com/rasbt/python-machine-learning-book/

Self Test

- **What is the end goal of nested cross-validation?**
  - A. To determine hyper parameters
  - B. To estimate generalization performance
  - C. To estimate generalization performance when performing hyper parameter tuning
  - D. To estimate the variation in tuned hyper parameters

McNemar Testing for Comparing Performance

Few assumptions, **Null hypothesis**: predictions are not different!



McNemar and Edwards, 1948

$$\chi^2 \approx \frac{(|B - C| - 1)^2}{B + C}$$

If predictions are drawn from the same distributions, then this equation follows χ **squared statistic with one DOF**

Steps:

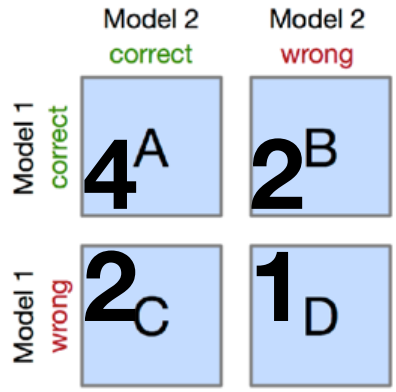
1. Compare each model's predictions on the same test data (2x2 matrix)
2. Calculate χ^2 statistic
3. Look up *critical value* associated with χ^2 statistic for given confidence
4. Are you confident enough to **reject the null hypothesis** that the performance is the same (p<0.05)?

One caveat: Statistical power depends upon B+C, which might be small, even with lots of test data.

https://sebastianraschka.com/blog/2018/model-evaluation-selection-part4.html

McNemar Example

Model 1	Model 2	Label	Matrix
T-shirt	T-shirt	T-shirt	A
Sneaker	T-shirt	Sneaker	B
T-shirt	Pullover	Pullover	C
Sneaker	Sneaker	Sneaker	A
T-shirt	Sneaker	Sneaker	C
Pullover	Pullover	T-shirt	D
Pullover	T-shirt	Pullover	B
Sneaker	Sneaker	Sneaker	A
Sneaker	Sneaker	Sneaker	A



McNemar and Edwards, 1948

$$\chi^2 \approx \frac{(|B - C| - 1)^2}{B + C}$$

$$\chi^2 = \frac{(|2 - 2| - 1)^2}{2 + 2} = 0.25$$

Confidence	0.90	0.95	0.99
1 DOF, Critical Value	2.706	3.841	6.635

https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm

Since 0.25 < 3.841, we cannot reject the null hypothesis. This means **we should not say the models' performance are different** based on the evidence.

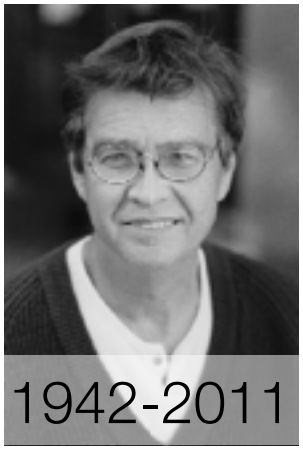
Town Hall

Some History of Deep Learning

Neural Networks: Where we left it

- Before 1986: AI Winter
- 1986: *Rumelhart*, *Hinton*, and *Williams* popularize gradient calculation for multi-layer network
  - *technically* introduced by Werbos in 1982
- **difference**: Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm is popularly called **Back-Propagation**
- wins pattern recognition prize in 1993, becomes de-facto machine learning algorithm in the 90's

David Rumelhart



1942-2011

Geoffrey Hinton

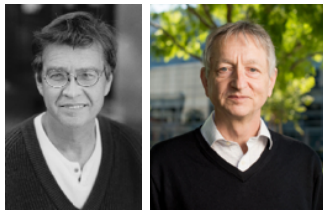
Machine Learning Timeline (Neural Nets)

- Up to this point: back propagation saved AI winter
- 80's, 90's, 2000's: neural networks for image processing start to get deeper
  - but back propagation no longer efficient for training
  - Back propagation gradient **stagnates** research—can't train **deeper** networks

- Second AI winter begins, research in NN plummets
- Funding for and accepted papers with Neural Networks asymptotically approaches zero

1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

1940 — 1960 — 1980 — 2000 — 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

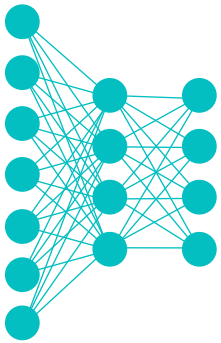
1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

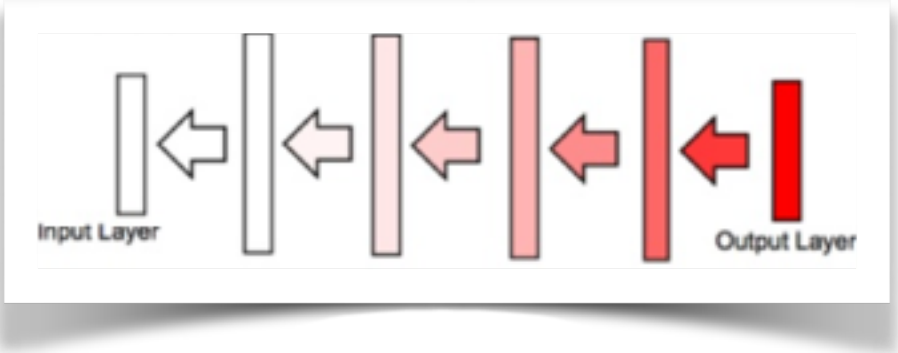
Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

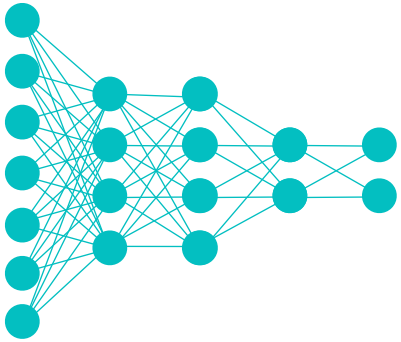
History of Deep Learning: Winter

- AI Winter is coming:



Easy to train, performs on par with other methods





Hard to train, performs worse than other methods
~chance (untrainable)

Researcher have difficulty reconciling expressiveness with performance

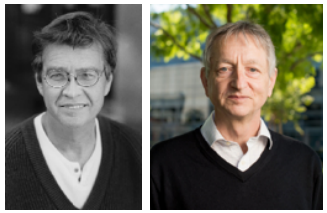
Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Machine Learning Timeline (Neural Nets)

- 2004: Hinton secures funding from CIFAR based on his reputation
  - *eventually*: Canada would be savior for neural networks
  - Hinton rebrands: **Deep Learning**
- 2006: Hinton publishes paper on using pre-training and Restricted Boltzmann Machines
- 2007: Another paper: Deep networks are more efficient when pre-trained
  - RBMs not really the important part

1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

1940 | 1960 | 1980 | 2000 | 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

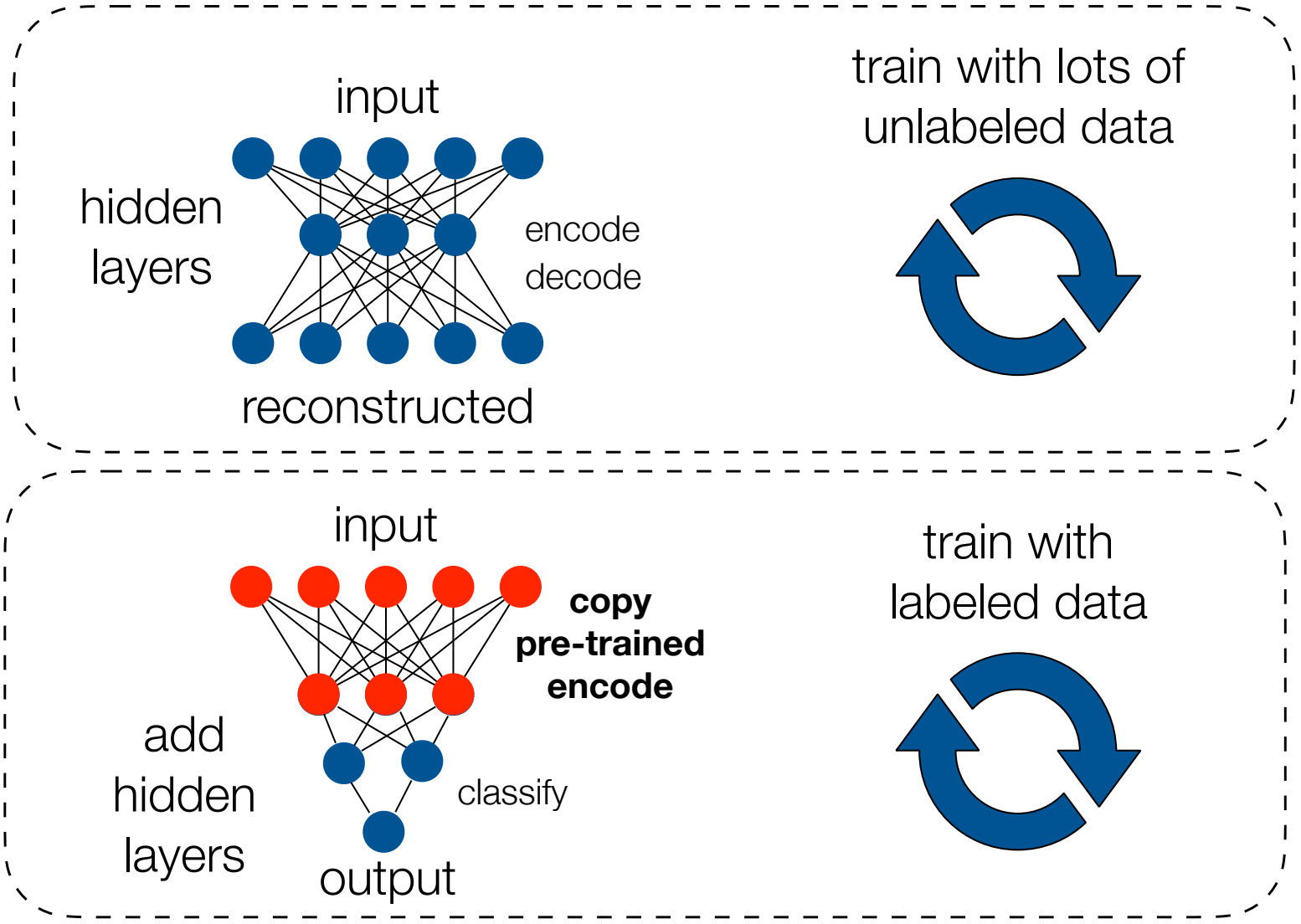
1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

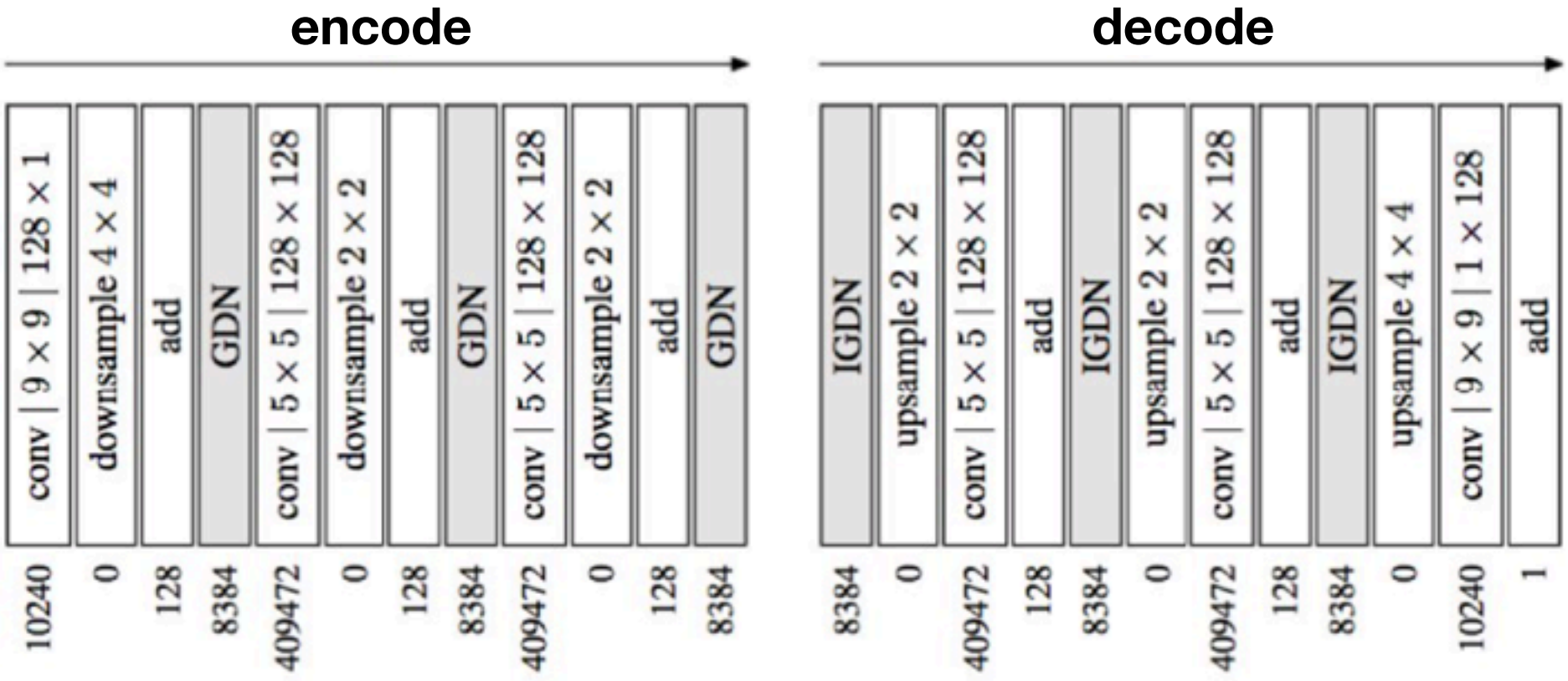
Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Pre-training: still in the long winter

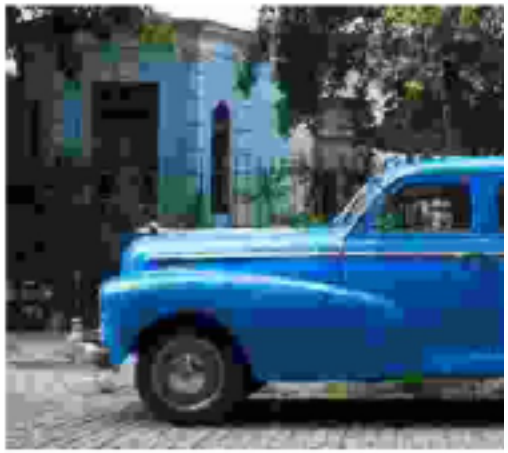
- auto-encoding (a form of pre-training)



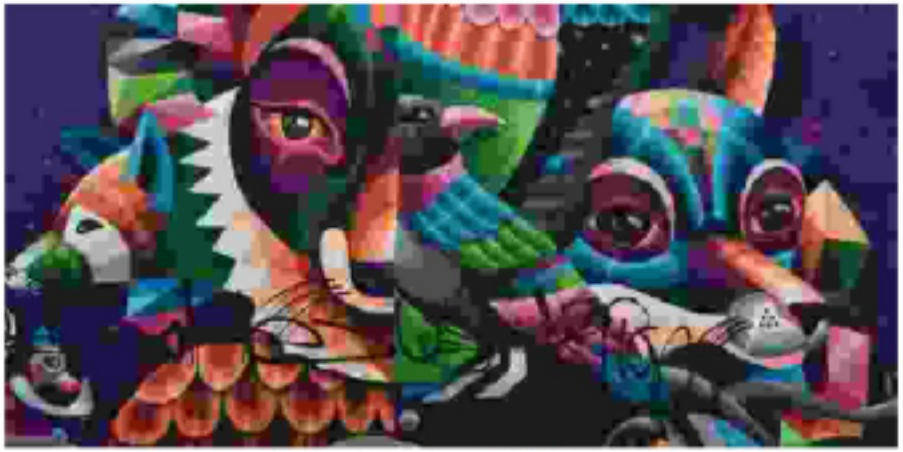
Pre-training: modern example



https://arxiv.org/abs/1611.01704



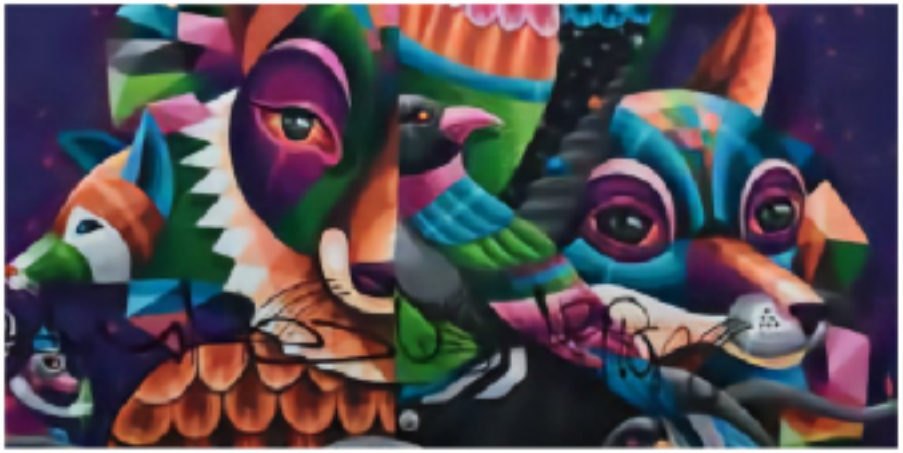
JPEG, 6006 bytes (0.170 bit/px), RMSE: 19.75



JPEG, 5928 bytes (0.168 bit/px), RMSE: 15.44/12.40, PSNR: 24.36 dB/26.26 dB

RMSE: 11.07/10.60, PSNR: 27.25 dB/27.63 dB

Proposed method, 5910 bytes (0.167 bit/px), RMSE



Proposed method, 5685 bytes (0.161 bit/px), RMSE: 10.41/5.98, PSNR: 27.78 dB/32.60 dB

it/px), RMSE: 6.10/5.09, PSNR: 32.43 dB/34.00 dB

JPEG 2000, 5918 bytes (0.167 bit/px), RMSE: 1

JPEG 2000, 5724 bytes (0.162 bit/px), RMSE: 13.75/7.02, PSNR: 25.36 dB/31.20 dB

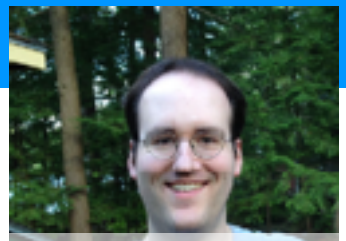
x), RMSE: 8.56/5.71, PSNR: 29.49 dB/32.99 dB

Still in the Long Winter

- 2009: Hinton's lab starts using GPUs, Also Andrew Ng
  - GPUs decrease training time by 70 fold…
- 2010: Hinton's and Ng's students go to internships with Microsoft, Google, IBM, and Facebook

Research at Google

Navdeep Jaitly
Google Brain Team

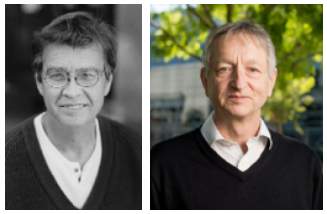


George Dahl
Google Brain Team

Abdel-rahman Mohamed
Microsoft Research
Redmond, Washington | Computer Software
Current Microsoft
Previous University of Toronto, IBM, Microsoft
Education University of Toronto

1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

- Xbox Voice
- Android Speech Recognition
- IBM Watson
- DeepFace
- All of Baidu

1940 | 1960 | 1980 | 2000

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

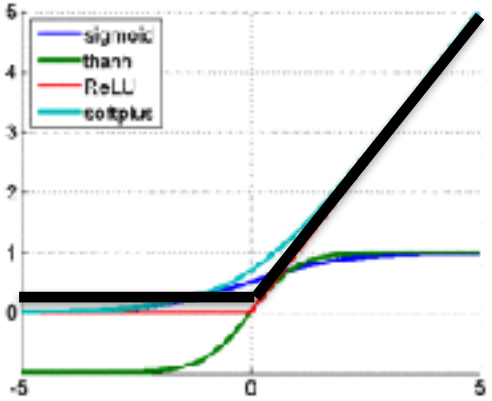
1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Getting out of the long Winter

- 2011: Glorot and Bengio investigate more systematic methods for why past deep architectures did not work
  - **discover some interesting, simple fixes**: the type of neurons chosen and the selection of initial weights
  - do not require pre-training to get deep networks properly trained, just sparser representations and less complicated derivatives



ReLU: $f(x) = \max(0,x)$
$f'(x) = 1$ if $x>0$ else 0

1949, Hebb's Law
Close neuron fire together


1960, Widrow & Hoff
Adaline Network


1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

1940 — 1960 — 1980 — 2000 — 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

2011, Bengio
Init and ReLU

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

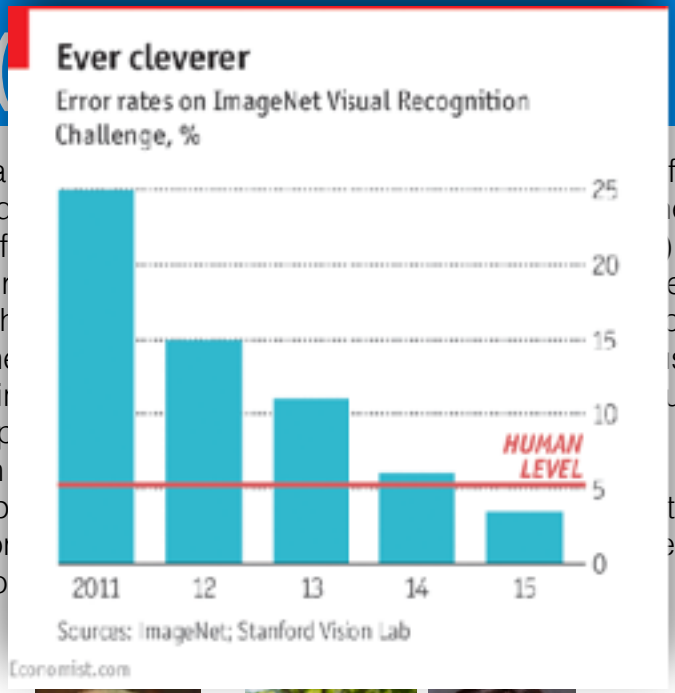
Machine Learning Timeline (

- **ImageNet competition occurs**
- **Second place**: 26.2% error rate
- **First place**:
  - From Hinton's lab, uses convolutional network with ReLU and dropout
  - 15.2% error rate
- Computer vision adopts deep learning with convolutional neural networks en mass

"I ha
hanc
last f
so I
as sh
come
pacir
happ
from
skep
a cor
Visio

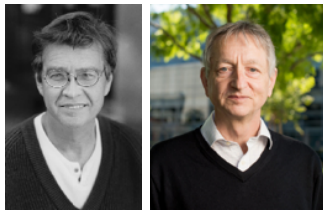
first
e
),
ef
on
s
ust

t as
er



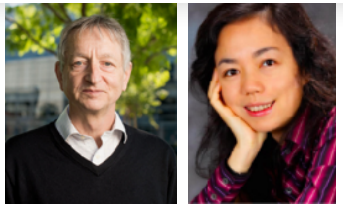
1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs



2012, Hinton, Fei-Fei Li
CNNs win ImageNet

1940 — 1960 — 1980 — 2000 — 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

2011, Bengio
Init and ReLU

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/