Train Deep Learning Models Using H2O Deep Water

Arno Candel, Magnus Stensmo, Wen Phan

GTC 2017

May 11, 2017, San Jose, CA

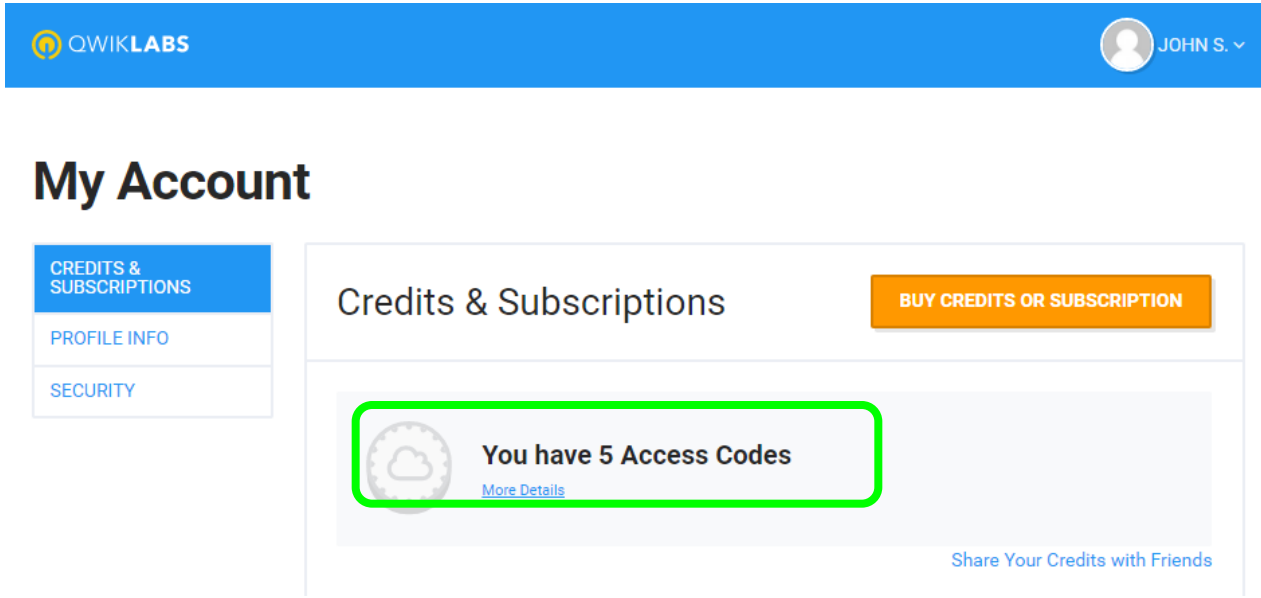
LAB CONNECTION INSTRUCTIONS - Part 1

Go to nvlabs.qwiklab.com
Sign in or create an account
Check for Access Codes (each day):

- Click My Account
- Click Credits & Subscriptions

If no Access Codes, ask for paper one from TA.
Please tear in half once used

An Access Code is needed to start the lab



WIFI SSID: **GTC_Hands_On**	Password: **HandsOnGpu**

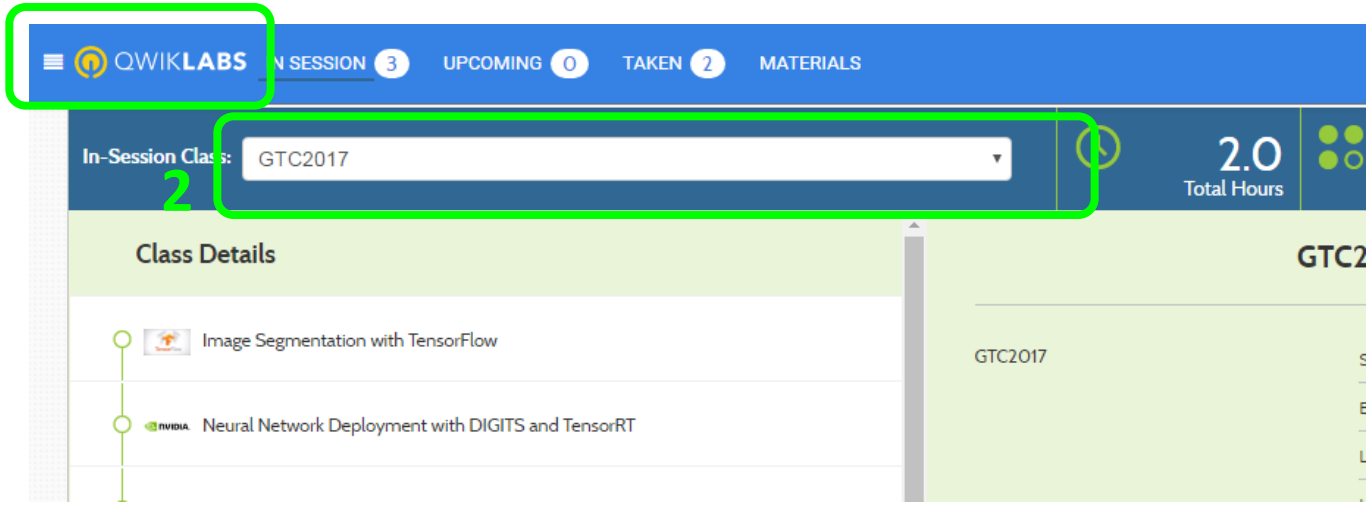
H2O.ai

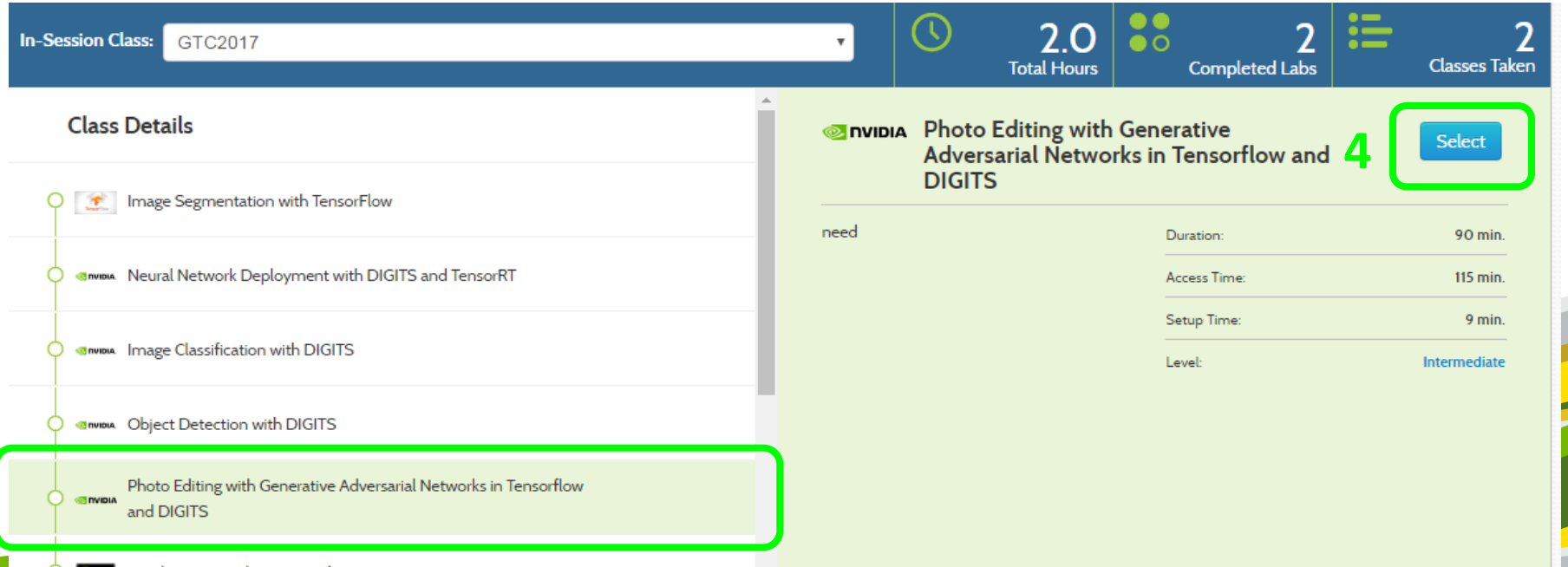
LAB CONNECTION INSTRUCTIONS - Part 2

1. Click Qwiklabs in upper-left
2. Select GTC2017 Class
3. Find lab and click on it
4. Click on Select
5. Click Start Lab

WIFI SSID:
GTC_Hands_On

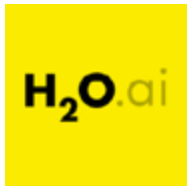
Password:
HandsOnGpu





Agenda

- Introduction to H2O
- H2O Deep Learning and Deep Water
- Image Classification
- Custom and User-Defined Networks
- Pre-Trained Networks
- Ensembles
- Deep Features
- Deployment



Introduction to H2O

Company Overview

Founded	2011 Venture-backed, debuted in 2012
Products	• H2O Open Source In-Memory AI Prediction Engine • Sparkling Water • Enterprise Steam • Deep Water
Mission	Operationalize data science and provide a platform for users to build beautiful data products
Team	70 employees • Distributed systems engineers doing machine learning • World-class visualization designers
Headquarters	Mountain View, CA

Leading Users Across Industries

H2O.ai

Gartner MQ: Data Science Platform

"Overall customer satisfaction is very high"

"H2O is especially suited to IoT edge and device scenarios"

"H2O had the highest reference customer analytics support score of all the vendors"

H2O.ai

Market Validation

Gartner

Gartner Magic Quadrant for Data Science Platforms

H2O.ai Selected in Top 16 MQ Vendors for 2017

FORRESTER®

Forrester Wave for Predictive Analytics

H2O.ai Selected in Top Wave Vendor for 2017

CB INSIGHTS

H2O.ai Selected in AI Top 100

46 PRESS MENTIONS

WSJ

THE HUFFINGTON POST

TechCrunch

107 OF THE FORTUNE 500

H2O

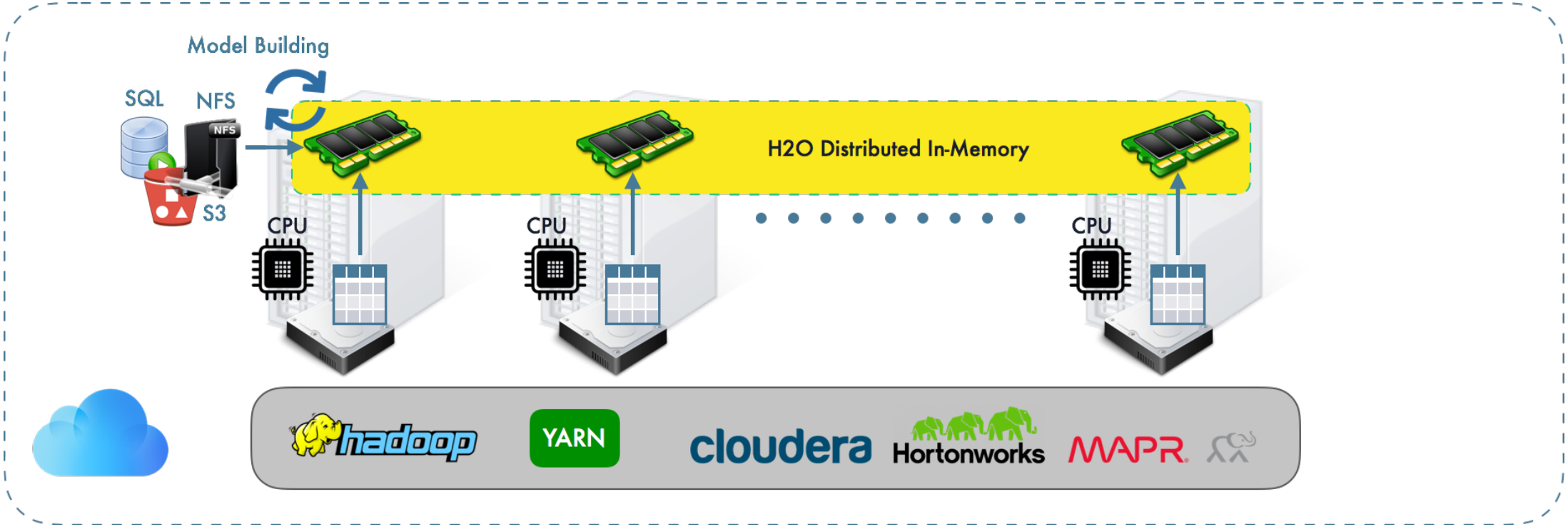
8 OF TOP 10 BANKS

7 OF TOP 10 INSURANCE COMPANIES

4 OF TOP 10 HEALTHCARE COMPANIES

H2O.ai

H2O Core


Model Building
SQL
NFS
NFS
S3
H2O Distributed In-Memory
CPU
CPU
CPU
hadoop
YARN
cloudera
Hortonworks
MAPR

Algorithm Overview

Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- Stacking / Super Learner

Deep Neural Networks

- MLP
- Autoencoder
  - Anomaly Detection
  - Deep Features
- CNN, RNN (Deep Water)

Clustering

- K-Means (Auto-*K*)

Dimension Reduction

- Principal Component Analysis
- Generalized Low Rank Models

Word Embedding

- Word2Vec

Time Series

- iSAX

Machine Learning Tuning

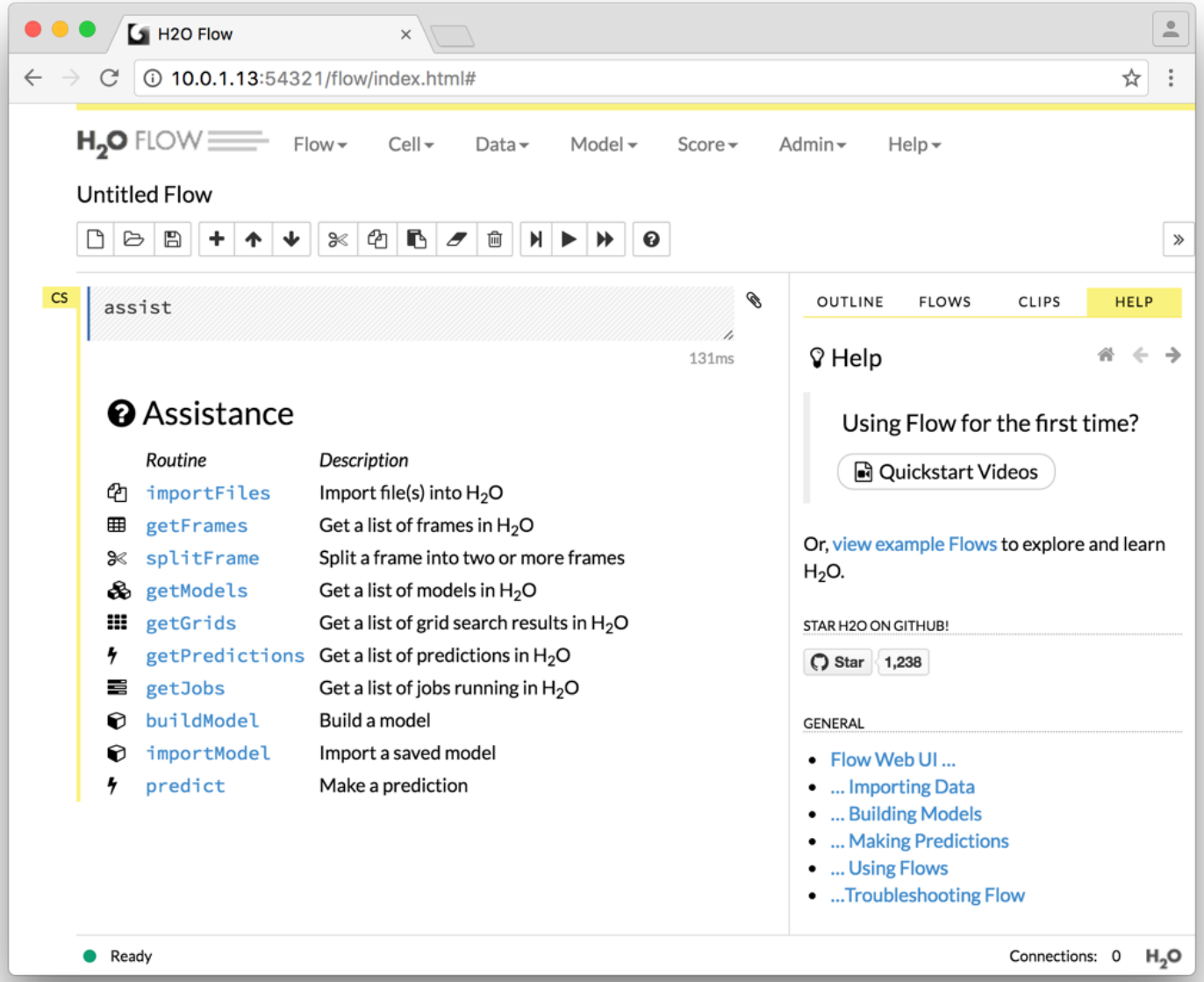
- Hyperparameter Search
- Early Stopping

H$_2$O.ai

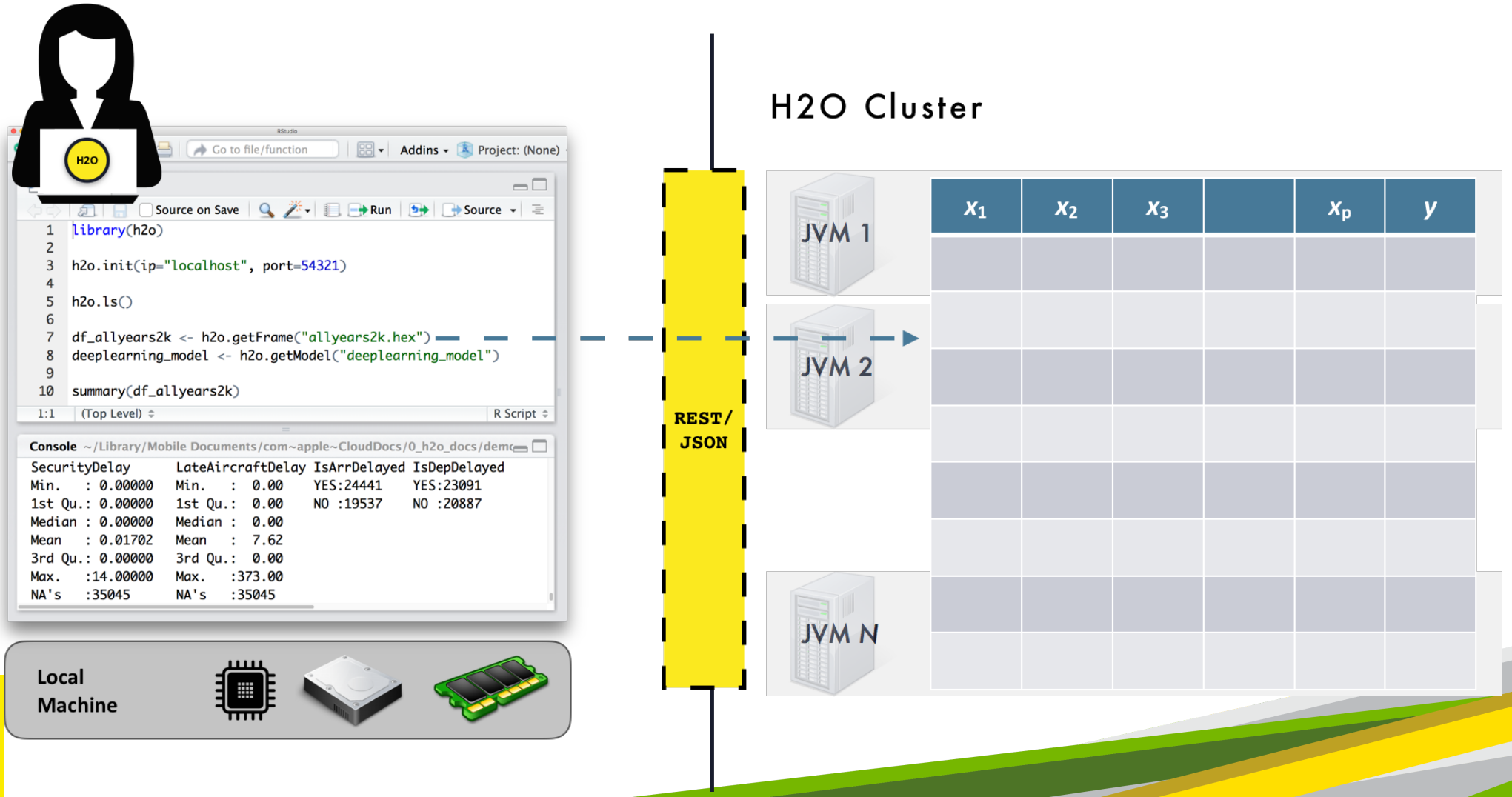
User Interfaces and APIs

H2O.ai

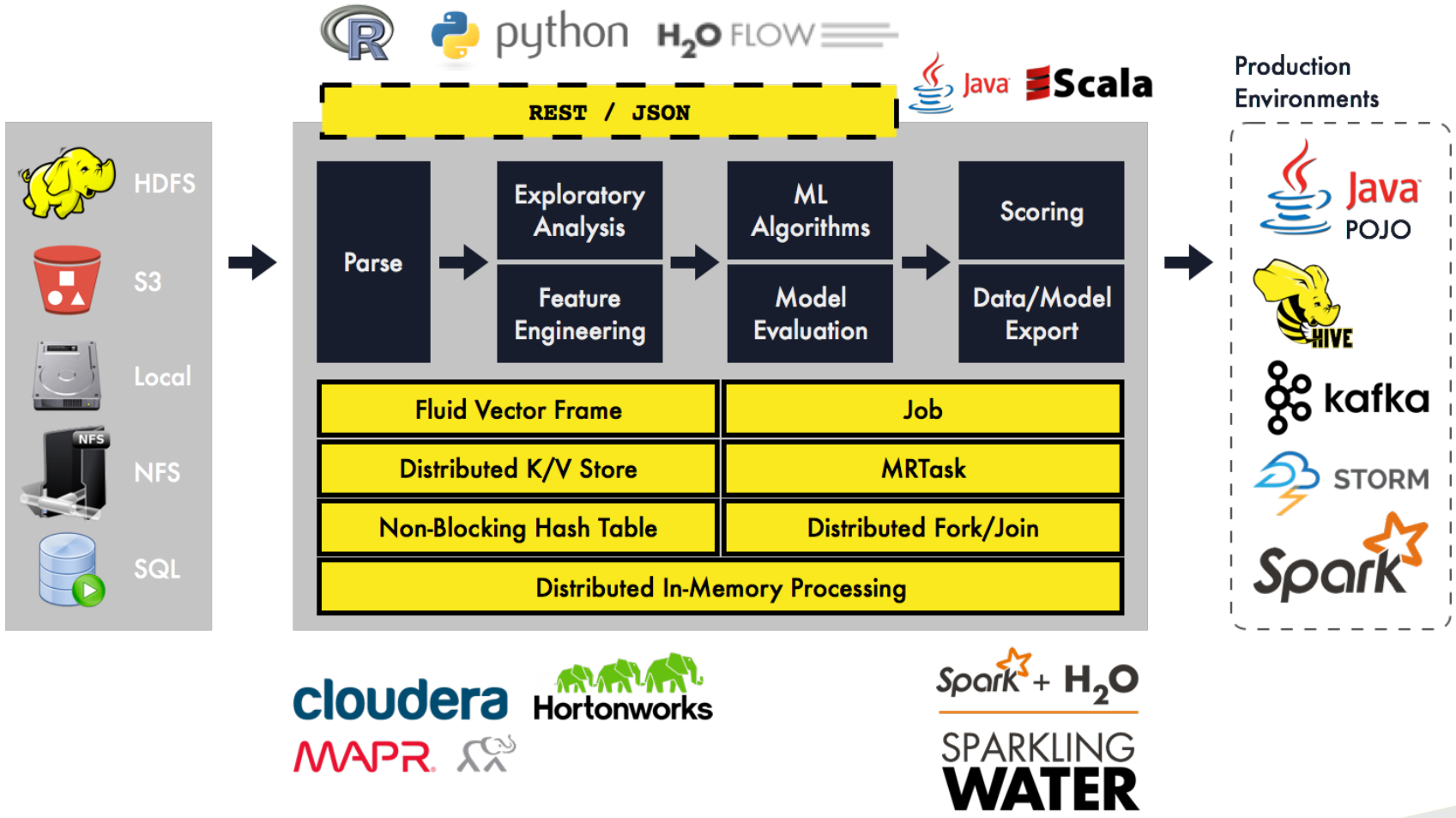
H2O Flow


H2O Flow
10.0.1.13:54321/flow/index.html#
H2O FLOW
Flow
Cell
Data
Model
Score
Admin
Help
Untitled Flow
CS
assist
131ms
Assistance
Routine
Description
importFiles
Import file(s) into H2O
getFrames
Get a list of frames in H2O
splitFrame
Split a frame into two or more frames
getModels
Get a list of models in H2O
getGrids
Get a list of grid search results in H2O
getPredictions
Get a list of predictions in H2O
getJobs
Get a list of jobs running in H2O
buildModel
Build a model
importModel
Import a saved model
predict
Make a prediction
OUTLINE
FLOWS
CLIPS
HELP
Help
Using Flow for the first time?
Quickstart Videos
Or, view example Flows to explore and learn H2O.
STAR H2O ON GITHUB!
Star
1,238
GENERAL
Flow Web UI ...
... Importing Data
... Building Models
... Making Predictions
... Using Flows
...Troubleshooting Flow
Ready
Connections: 0
H2O

H2O Clients and API



Core H2O Architecture



H2O.ai

Other Solutions

- Sparkling Water (H2O + Spark)
- Enterprise Steam

Upcoming and Ongoing

- Automatic Machine Learning (AutoML)
- Machine Learning Interpretability

H2O Deep Learning and Deep Water

H2O Deep Learning

- **“Best in Class” Multi-Layer Feed-Forward Deep Neural Networks**
- **Modern training options:** specifications for distributions (Bernoulli, Multinomial, Poisson, Gamma, Tweedie, Laplace, Huber, Quantile, Gaus- sian), loss functions (cross entropy, quadratic, absolute, Huber), learning rate, annealing, momentum, mini-batch size, and initialization
- **Deep autoencoders for unsupervised learning**: deep features and anomaly detection

H2O Deep Learning (cont.)

- **Automatic and flexible data handling to maximize productivity:** standardization, one-hot encoding, observation weights and offsets, class balancing, sampling factors, ignoring constant columns, sparse data handling, and input layer constraints

- **Tuning parameters to prevent model overfitting and efficient model development**: cross-validation, regularization, drop out, early stopping, model checkpointing, and hyperparameter search

World-Record MNIST Performance

```
> library(h2o)
> h2oServer <- h2o.init(ip="mr-0xd1", port=53322)
> train_hex <- h2o.importFile(h2oServer, "mnist/train.csv.gz")
> test_hex  <- h2o.importFile(h2oServer, "mnist/test.csv.gz")
> train_hex[,785] <- as.factor(train_hex[,785])
> test_hex[,785] <- as.factor(test_hex[,785])
> dl_model <- h2o.deeplearning(x=c(1:784), y=785, training_frame=train_hex, l1=1e-5,
                               activation="RectifierWithDropout", input_dropout_ratio=0.2,
                               classification_stop=-1,hidden=c(1024,1024,2048), epochs=8000)
|=================================================================================================
=================================| 100%
> h2o.confusionMatrix(dl_model, test_hex)
Confusion Matrix - (vertical: actual; across: predicted):
         0    1    2    3   4   5   6    7   8    9   Error          Rate
0      974    1    1    0   0   0   2    1   1    0 0.00612 =     6 / 980
1        0 1135    0    1   0   0   0    0   0    0 0.00088 =    1 /1,135
2        0    0 1028    0   1   0   0    3   0    0 0.00388 =   4 / 1,032
3        0    0    1 1003   0   0   0    3   2    1 0.00693 =   7 / 1,010
4        0    0    1    0 971   0   4    0   0    6 0.01120 =    11 / 982
5        2    0    0    5   0 882   1    1   1    0 0.01121 =    10 / 892
6        2    3    0    1   1   2 949    0   0    0 0.00939 =     9 / 958
7        1    2    6    0   0   0   0 1019   0    0 0.00875 =   9 / 1,028
8        1    0    1    3   0   4   0    2 960    3 0.01437 =    14 / 974
9        1    2    0    0   4   3   0    2   0  997 0.01189 =  12 / 1,009
Totals 981 1142 1038 1013 977 891 956 1031 964 1007 0.00830 = 83 / 10,000
```

- No data augmentation
- No distortions
- No convolutions
- No pre-training
- No ensemble

Table 1: *Classification error rate comparison: DBN vs. DCN*

DBN [3] (Hinton's)	DBN (MSR's)	DCN (Fine-tuning)	DCN (no Fine-tuning)	Shallow (D)CN (Fine-tuned single layer)
1.20%	1.06%	0.83%	0.95%	1.10%

Current World Record:
H2O Deep Learning
0.83% Test Set Error

Full Run: 10 Hours on 10-Node Cluster
Desktop: 2 Hours to 0.9% Test Set Error

H2O.ai

H2O Deep Water

One Interface - GPU Enabled - Significant Performance Gains

Inherits All H_2O Properties in Scalability, Ease of Use and Deployment

H_2O.ai

TensorFlow

Caffe mxnet

H_2O Deep Learning Algo

H_2O integrates with existing GPU backends for significant performance gains

Convolutional Neural Networks enabling **Image, video, speech recognition**

Hybrid Neural Network Architectures enabling **speech to text translation, image captioning, scene parsing** and more

Recurrent Neural Networks enabling **natural language processing, sequences, time series,** and more

H_2O.ai

H2O Deep Water (cont.)

- **Deep learning framework integration**: TensorFlow, MXNet, and Caffe
  - GPU-accelerated training, modern deep learning acrchitectures
- **Machine learning platform**: Other algos. Ensemble.
- **Ease of use and APIs:** Web UI (Flow), R, Python, Java
- **Deployment**

MXNet

dmlc
mxnet *for Deep Learning*

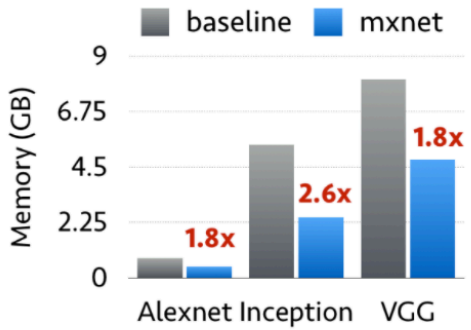
THE APACHE SOFTWARE FOUNDATION

amazon

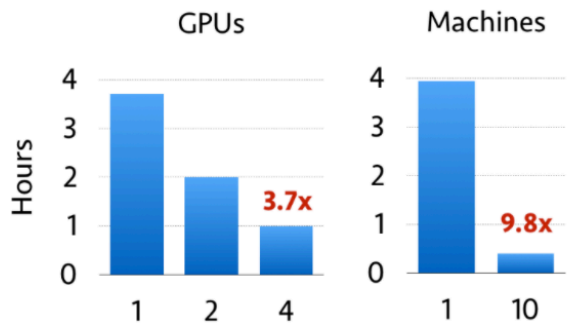
build passing | docs latest | license Apache 2.0

Portable

Efficient

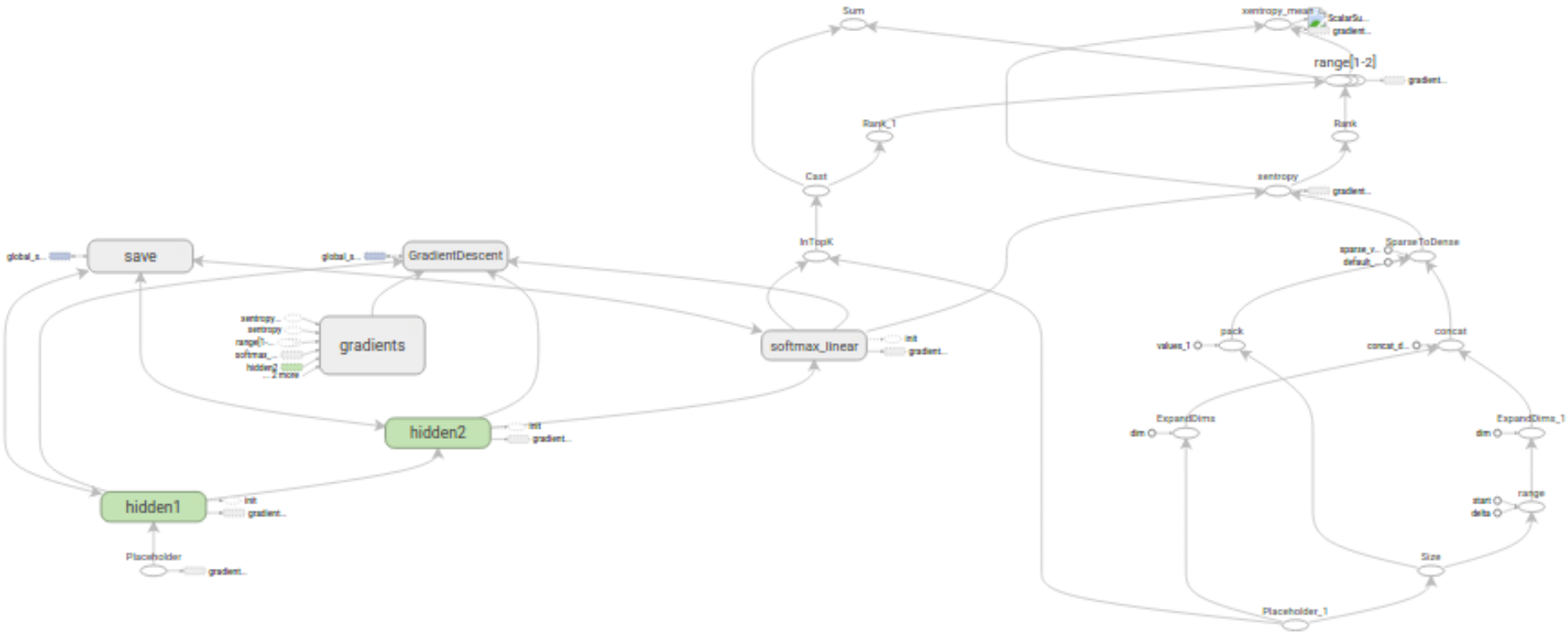


Scalable



H2O.ai

TensorFlow


Sum
range[1-2]
Rank_1
Rank
Cast
InTopK
softmax_linear
save
GradientDescent
gradients
hidden2
hidden1
Placeholder
SparseToDense
pack
concat
ExpandDims
ExpandDims_1
range
Size
Placeholder_1

Caffe

Caffe

BAIR

BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Caffe2

facebook

H$_2$O.ai

Setup

Setup

- Extract data.
  - cd /gtc-2017/data
  - tar zxvf mnist_png.tar.gz

- Start H2O
  - cd /gtc-2017
  - java –Xmx10g –jar /opt/h2o.jar &

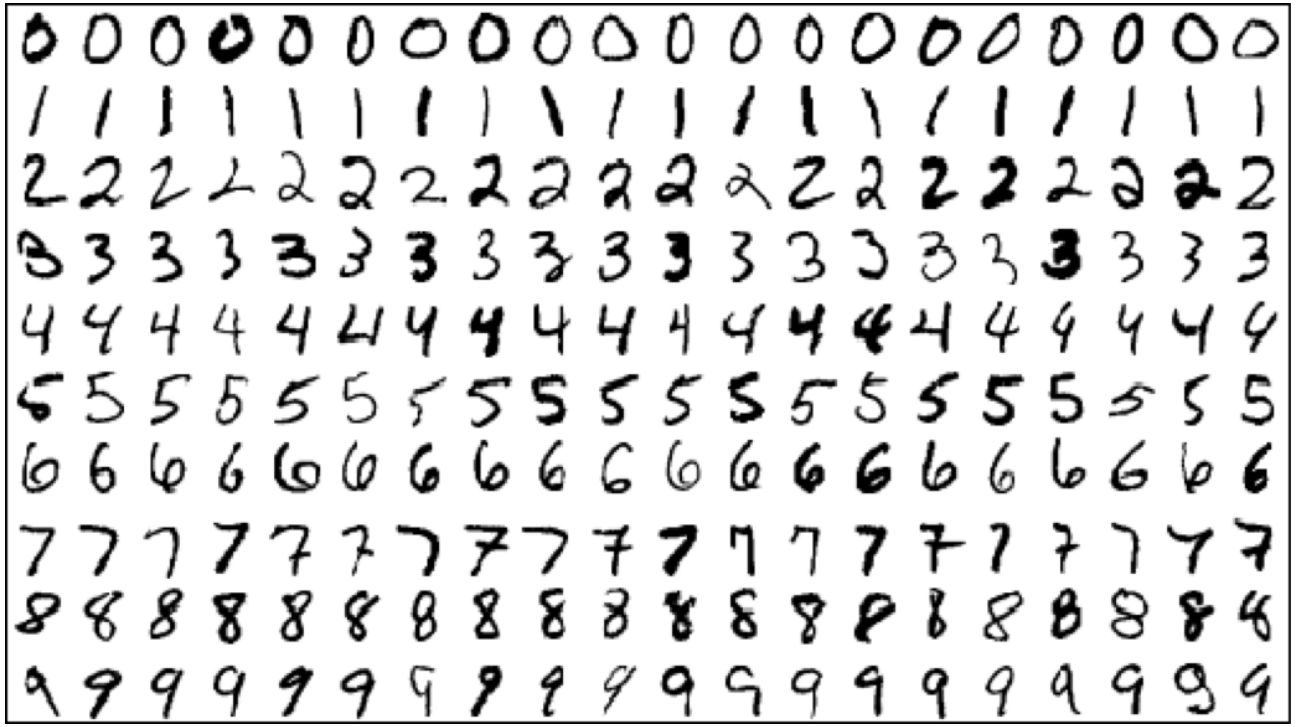
Setup (cont.)

- Start Jupyter notebook
  - cd /gtc-2017
  - jupyter notebook --allow-root --ip=* &
    - This will give you a link with localhost. Copy that link.

- Go to browser
  - H2O: http://you-ip-address:54321
  - Jupyter: Paste link into browser. Replace localhost with your ip address.

MNIST Data

MNIST Database



H$_2$O.ai

MNIST Digit

- 28 x 28 Pixels

```
Shape: (28, 28)
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0  26 121 134 253 253 255 127  13   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0 121 242 252 252 252 252 253 252  82   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0  90 142 226 247 252 252 252 252 252 253 252 242  85   0   0   0   0   0   0   0   0   0   0
  0   0   0   0 102 250 252 252 252 235 172  88  39  39 211 252 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0 121 252 252 231 150  42   0   0   0   0  89 249 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0  95 198 149  45   0   0   0   0   0   0   0 240 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 240 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 240 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  70 247 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0 121 252 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0 121 253 253 173   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0 121 252 252 172   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0 240 252 252  53   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  43 253 252 252 107  81 109 192  81  34   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0  46 227 253 252 252 252 252 252 252 252 140   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   6 169 252 253 252 252 252 252 252 252 252 248  89   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  48 252 252 253 252 252 218 107  79 176 252 250 100   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 160 252 252 253 252 178  15   0   0  36 225 143   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 160 252 252 253 183  11   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 160 252 252 190   6   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

H$_2$O.ai

Image as Pixel Vector

- 784 features (28x28)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_P \end{bmatrix}$$

```
[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0  32 133  74   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0  50 159  40  15  80  80 139 238 245 253 198   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0  23 116 249 255 219 239
  254 254 244 228 158 115  55   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   4 167 253 243 192 142 142 123  63  20   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0  18 148 253 211  67   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 184 253 211   4   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0  50 254 253 210 158
  158 159 158 120  35   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0 125 255 254 254 240 238 239 238 246 234  71   0   0
    0   0   0   0   0   0   0   0   0   0   0  10   0   0   0   0 118  83
   63   8   0   0   0  88 253 154   0   0   0   0   0   0   0   0   0   0
    0   0   0 147 155   0   0   0   0   0   0   0   0  18  93 244 226  38
    0   0   0   0   0   0   0   0   0   0   0   0   0 189 249 182  74  24
    0   0   0   0  65 184 253 241  58   0   0   0   0   0   0   0   0   0
    0   0   0   0   0  11 129 238 253 243 184 158 173 238 247 254 208  27
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  40
  125 183 239 245 254 240 163  50   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0  28  63   8   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
```

MNIST Data Set

- Training Set: 60,000
- Testing/Validation: 10,000

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,P} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,P} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$$

$$\mathbf{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,P} \end{bmatrix}$$

where $\mathbf{x}_i$ is i-th data record

H$_2$O.ai

mnist_training.hex

Actions: View Data | Split... | Build Model... | Predict | Download | Export

	Rows	*Columns*	*Compressed Size*
	60000	2	5MB

▾ COLUMN SUMMARIES

label	*type*	*Missing*	*Zeros*	*+Inf*	*-Inf*	*min*	*max*	*mean*	*sigma*	*cardinality*	*Actions*
uri	string	0	0	0	0	·	·	·	·	·	Convert to enum
label	enum	0	5923	0	0	0	9.0	·	·	10	Convert to numeric

mnist_testing.hex

Actions: View Data | Split... | Build Model... | Predict | Download | Export

	Rows	*Columns*	*Compressed Size*
	10000	2	807KB

▾ COLUMN SUMMARIES

label	*type*	*Missing*	*Zeros*	*+Inf*	*-Inf*	*min*	*max*	*mean*	*sigma*	*cardinality*	*Actions*
uri	string	0	0	0	0	·	·	·	·	·	Convert to enum
label	enum	0	980	0	0	0	9.0	·	·	10	Convert to numeric

mnist_training.hex

▼ DATA

Previous 20 Columns | Next 20 Columns

Row	uri	label
1	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/6453.png	6
2	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/13970.png	6
3	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/21967.png	6
4	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/11277.png	6
5	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/27350.png	6
6	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/32119.png	6
7	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/44201.png	6
8	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/58770.png	6
9	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/39440.png	6
10	/home/ubuntu/sales-engineering/wen/gtc2017/data/mnist_png/training/6/41988.png	6

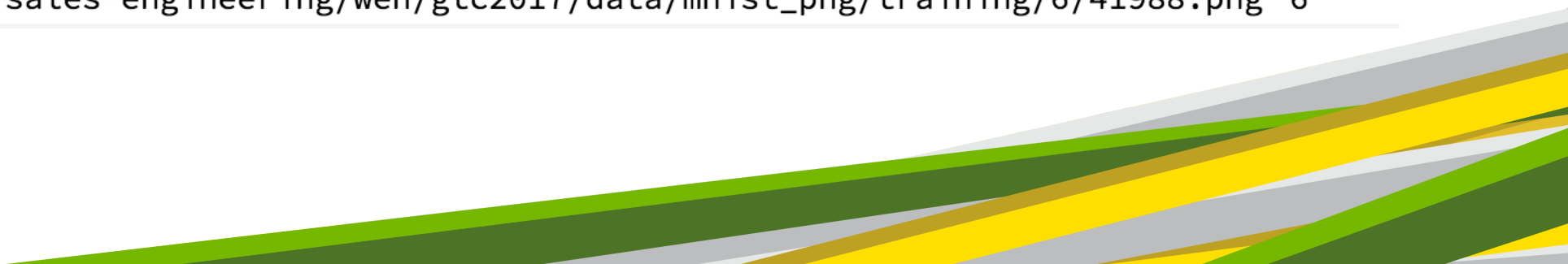


Image Classification

LeNet

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. **Gradient-Based Learning Applied to Document Recognition**. In *Proceedings of the IEEE*, Nov. 1998. URL `http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf`

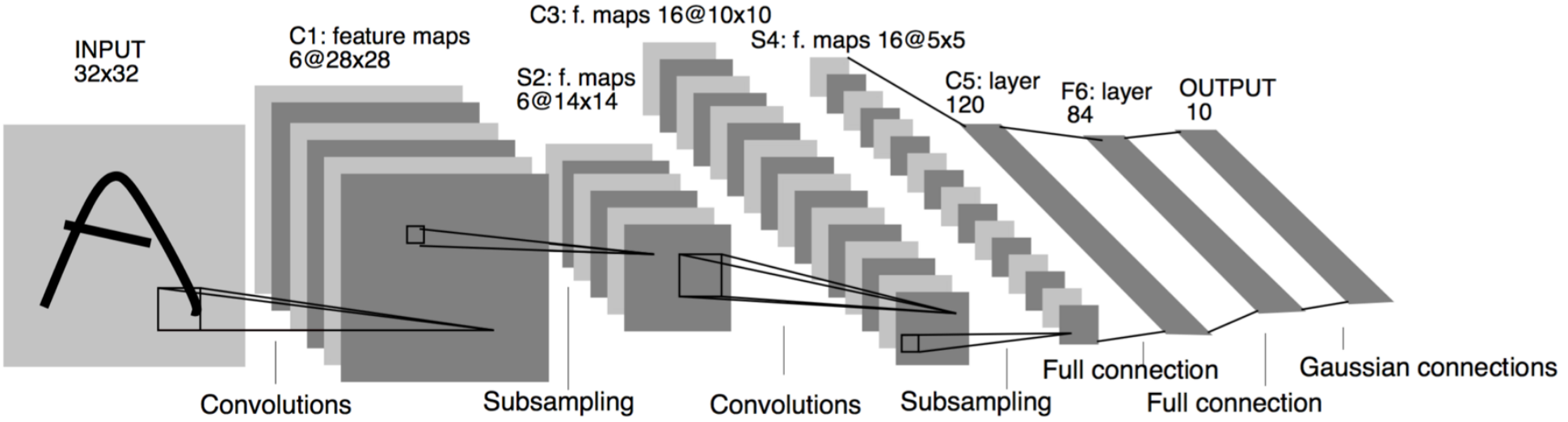
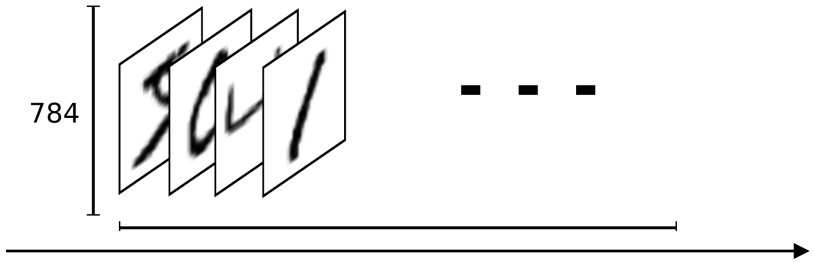


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

H2O.ai

Supervised Learning

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,P} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,P} \end{bmatrix}$$

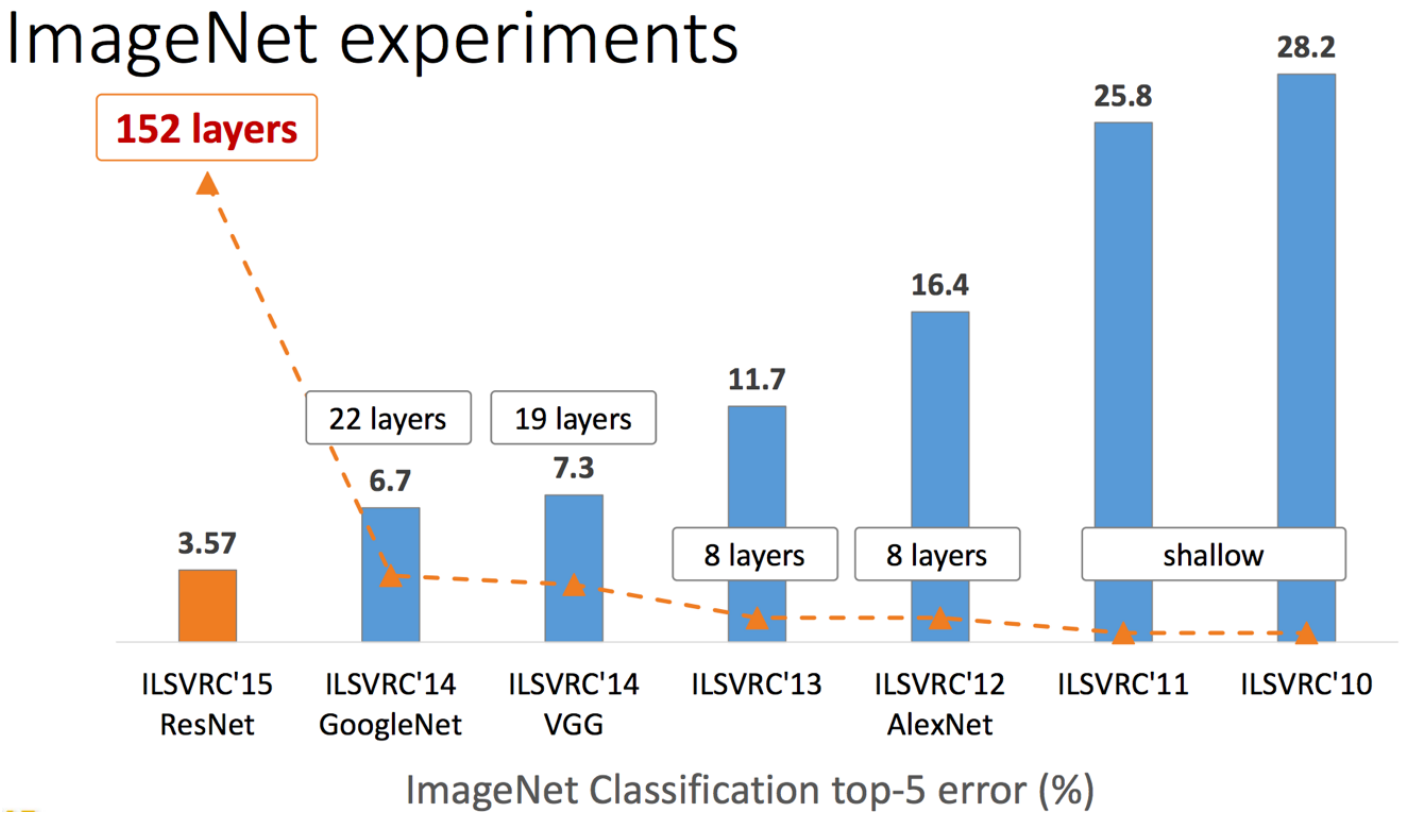


$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,C} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,C} \end{bmatrix}$$

H$_2$O.ai

ILSVRC Winners

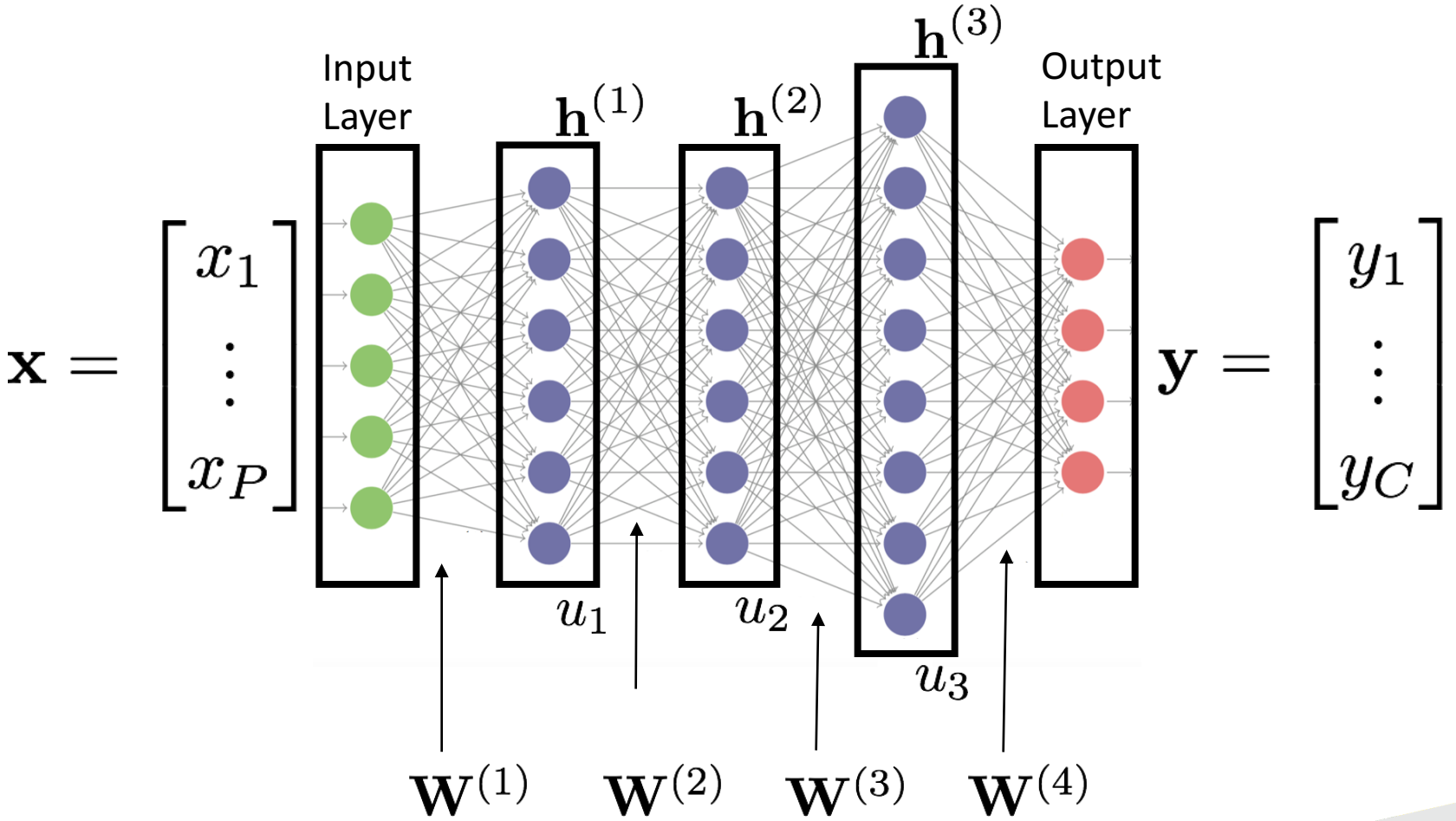
Microsoft Research



H2O.ai

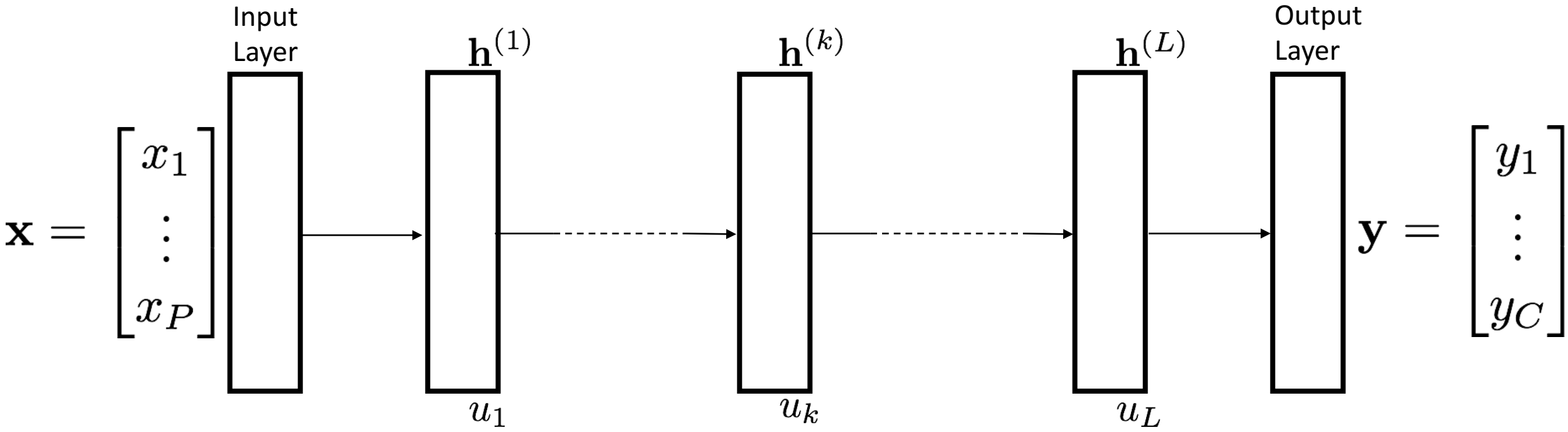
Custom and User-Defined Networks

Neural Network



H2O.ai

Graph


Input Layer
$\mathbf{h}^{(1)}$
$\mathbf{h}^{(k)}$
$\mathbf{h}^{(L)}$
Output Layer
$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_P \end{bmatrix}$
u_1
u_k
u_L
$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_C \end{bmatrix}$

H2O.ai

MXNet

```
net = mx.sym.Variable('data')
net = mx.sym.FullyConnected(data=net, name='fc1', num_hidden=128)
net = mx.sym.Activation(data=net, name='relu1', act_type="relu")
net = mx.sym.FullyConnected(data=net, name='fc2', num_hidden=10)
net = mx.sym.SoftmaxOutput(data=net, name='out')
mx.viz.plot_network(net, shape={'data':(100,200)})
```

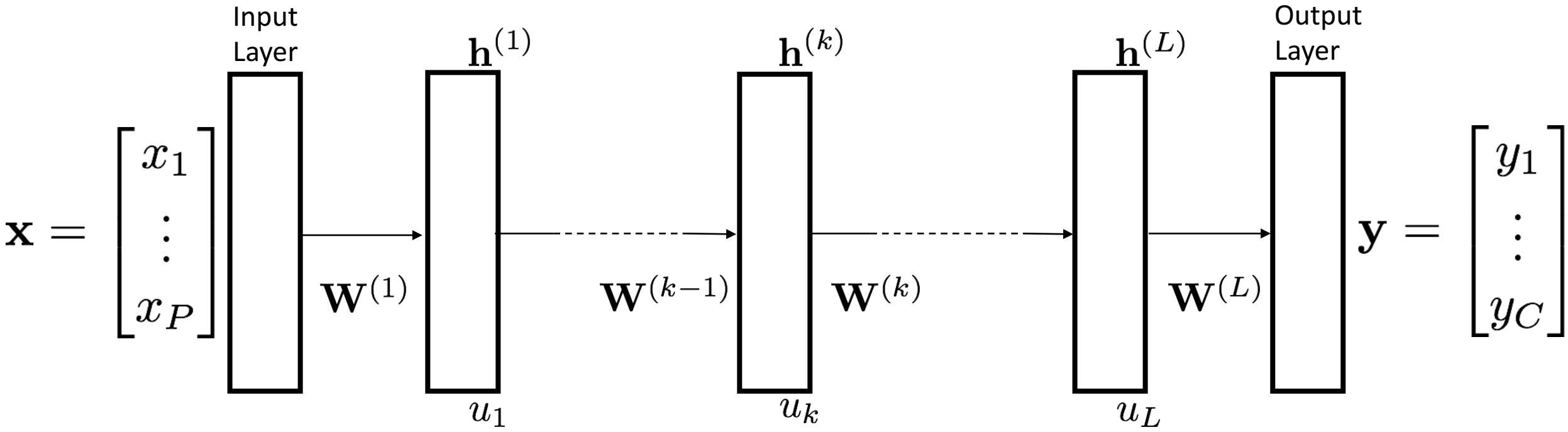
Symbol.save	Saves symbol to a file.
BaseModule.save_params	Save model parameters to file.

TensorFlow

```
with tf.Session() as sess:
  # Initializes all the variables.
  sess.run(init_all_op)
  # Runs to logit.
  sess.run(logits)
  # Creates a saver.
  saver0 = tf.train.Saver()
  saver0.save(sess, 'my-save-dir/my-model-10000')
  # Generates MetaGraphDef.
  saver0.export_meta_graph('my-save-dir/my-model-10000.meta')
```

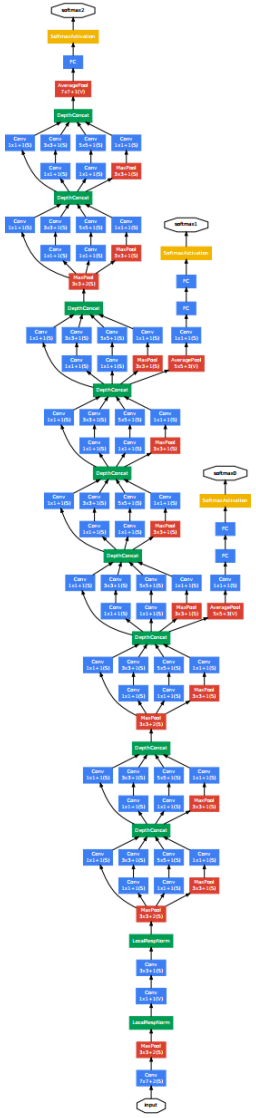
Pre-Trained Networks

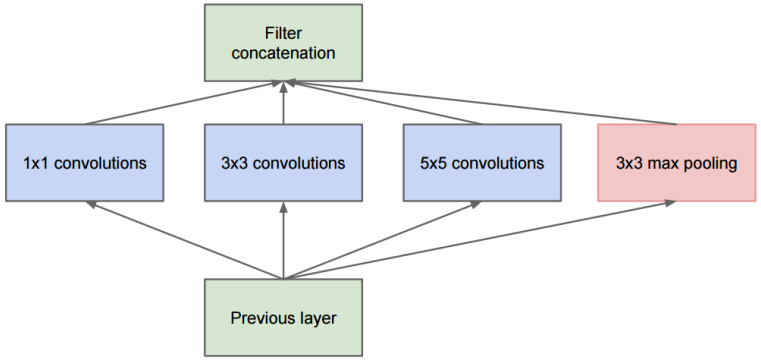
Graph and Weights



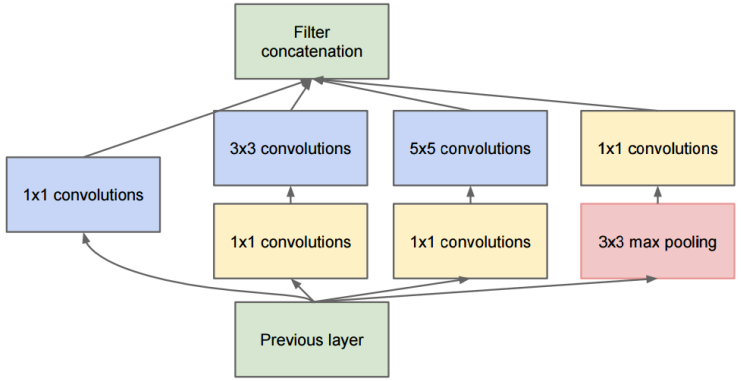
H2O.ai

GoogLeNet and the Inception Module





(a) Inception module, naïve version



(b) Inception module with dimension reductions

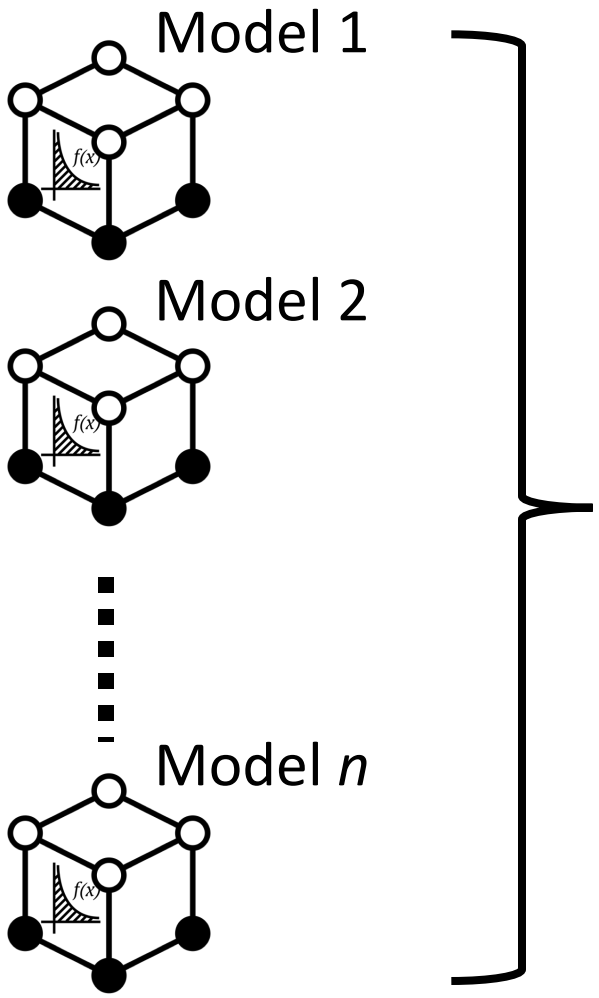
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. **Going Deeper with Convolutions**. 2014. URL `https://arxiv.org/pdf/1409.4842.pdf`

H2O.ai

Ensembles

Ensembles

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,P} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,P} \end{bmatrix}$$



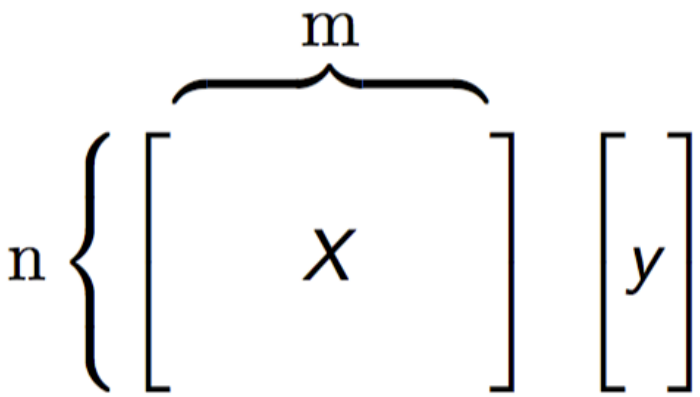
$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,C} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,C} \end{bmatrix}$$

H2O.ai

Stacking / Super Learner

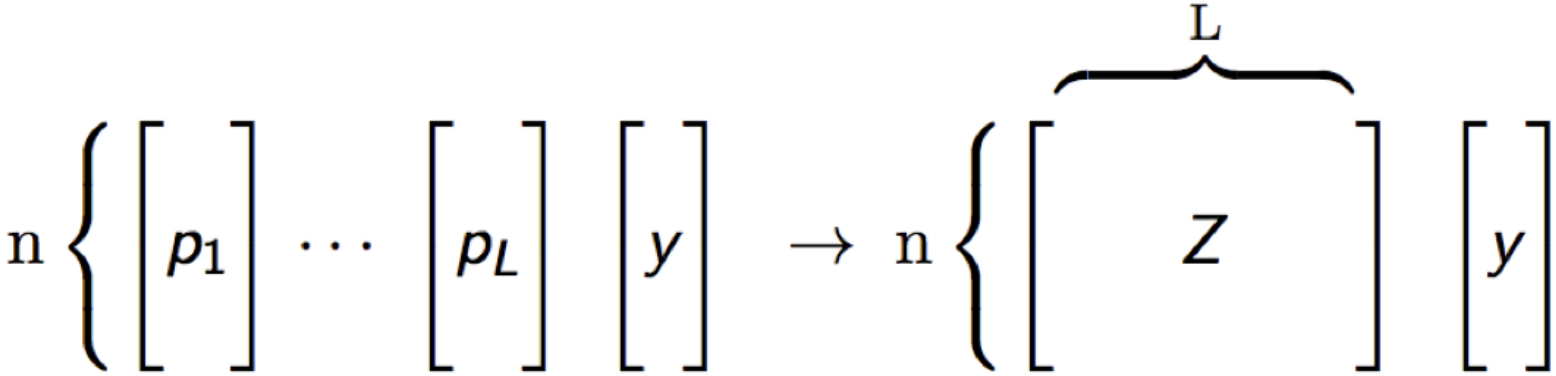
- Train a second machine learning model (called a “meta-learner”) to learn the optimal combination of the base learners

“Level-Zero” Data



1. Start with design matrix, X, and response, y
2. Specify *L* base learners
3. Specify a meta-learner
4. Perform *k*-fold cross-validation (CV) on each of the *L* learners

“Level-One” Data

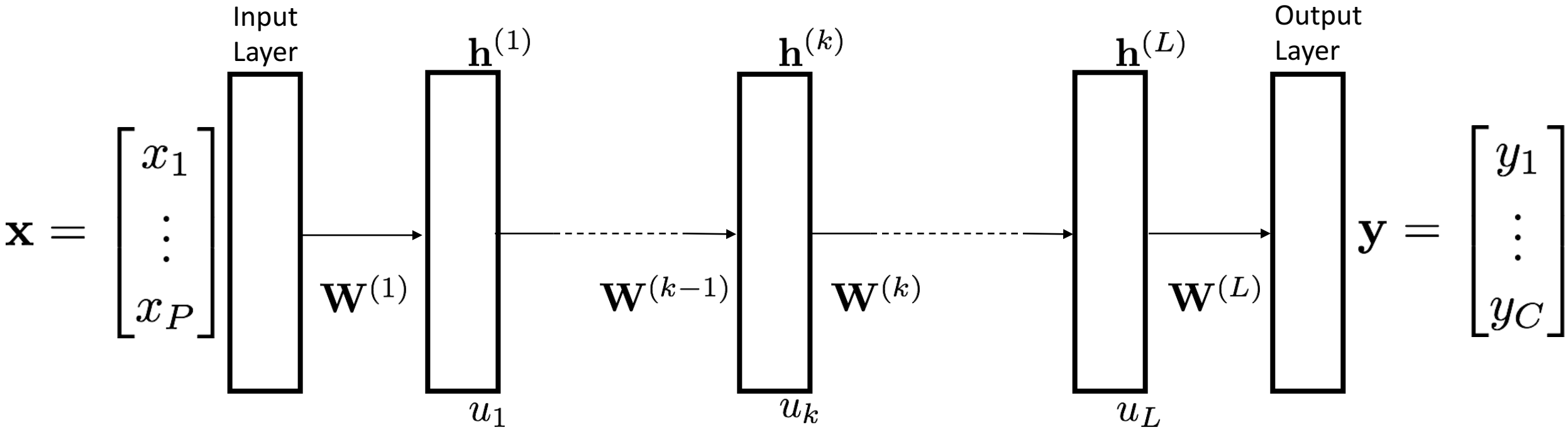


1. Collect the predicted values from *k*-fold CV that was performed on each of the *L* base learners
2. Column-bind these prediction vectors together to form a new design matrix, Z
3. Train the metalearner using Z, y

H$_2$O.ai

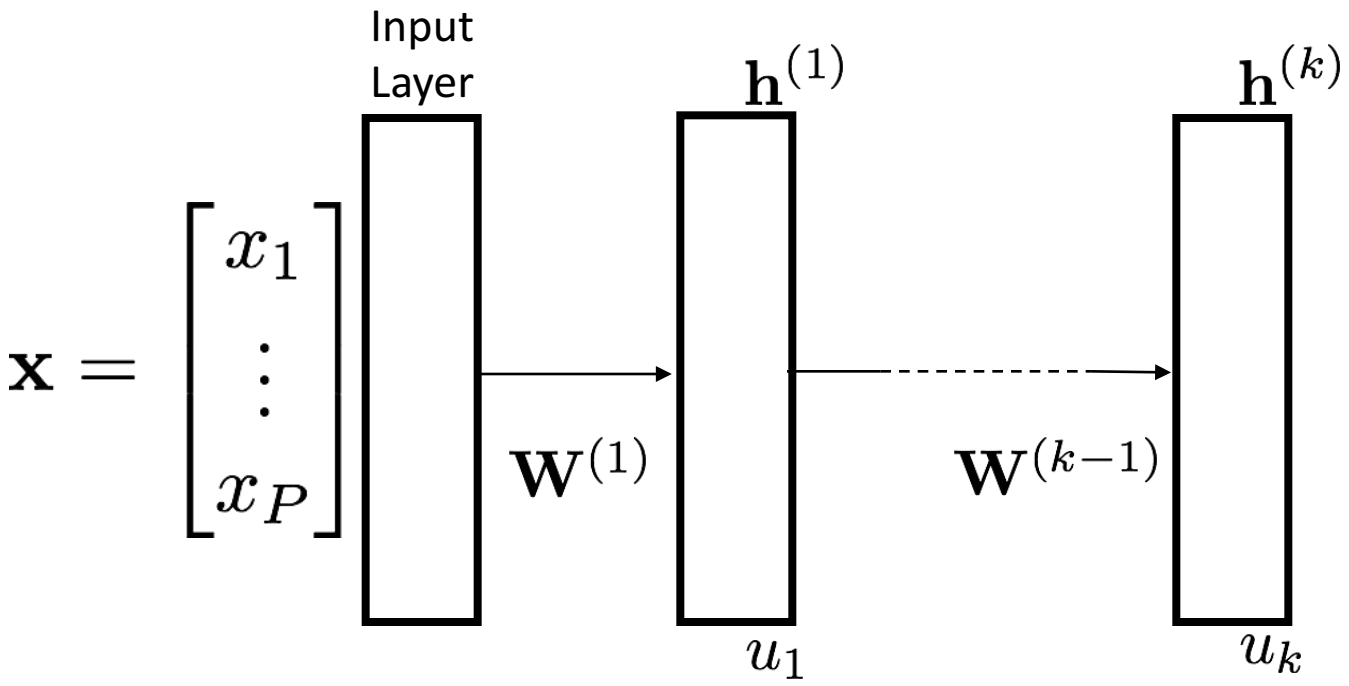
Deep Features

Trained Network



H2O.ai

Hidden Layer Features



$$\mathbf{h}^{(k)} = \begin{bmatrix} h_1^{(k)} \\ \vdots \\ h_{u_k}^{(k)} \end{bmatrix}$$

Matrix of Hidden Layer (Deep) Features

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,P} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,P} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$$

$$\mathbf{H}^{(k)} = \begin{bmatrix} h_1^{(k,1)} & \dots & h_{u_k}^{(k,1)} \\ \vdots & \ddots & \vdots \\ h_1^{(k,N)} & \dots & h_{u_k}^{(k,N)} \end{bmatrix} = \begin{bmatrix} [\mathbf{h}^{(k,1)}]^T \\ \vdots \\ [\mathbf{h}^{(k,N)}]^T \end{bmatrix}$$

H2O.ai

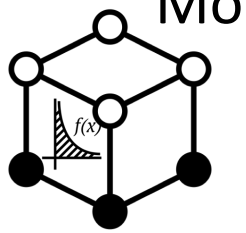
Deep Features

- Can use the deep features as input features for another model (algorithm)

- Can use the deep features as data representation for similarity analysis

Deep Features Classifier

$$\mathbf{H}^{(k)} = \begin{bmatrix} h_1^{(k,1)} & \dots & h_{u_k}^{(k,1)} \\ \vdots & \ddots & \vdots \\ h_1^{(k,N)} & \dots & h_{u_k}^{(k,N)} \end{bmatrix} \longrightarrow$$



$$\longrightarrow \mathbf{Y} = \begin{bmatrix} y_{1,1} & \dots & y_{1,C} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \dots & y_{N,C} \end{bmatrix}$$

H2O.ai

Distance

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,P} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \dots & a_{M,P} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_M^T \end{bmatrix} \quad \text{where } \mathbf{a}_i = \begin{bmatrix} a_{i,1} \\ \vdots \\ a_{i,P} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} b_{1,1} & \dots & b_{1,P} \\ \vdots & \ddots & \vdots \\ b_{O,1} & \dots & b_{O,P} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_O^T \end{bmatrix} \quad \text{where } \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ \vdots \\ b_{i,P} \end{bmatrix}$$

H2O.ai

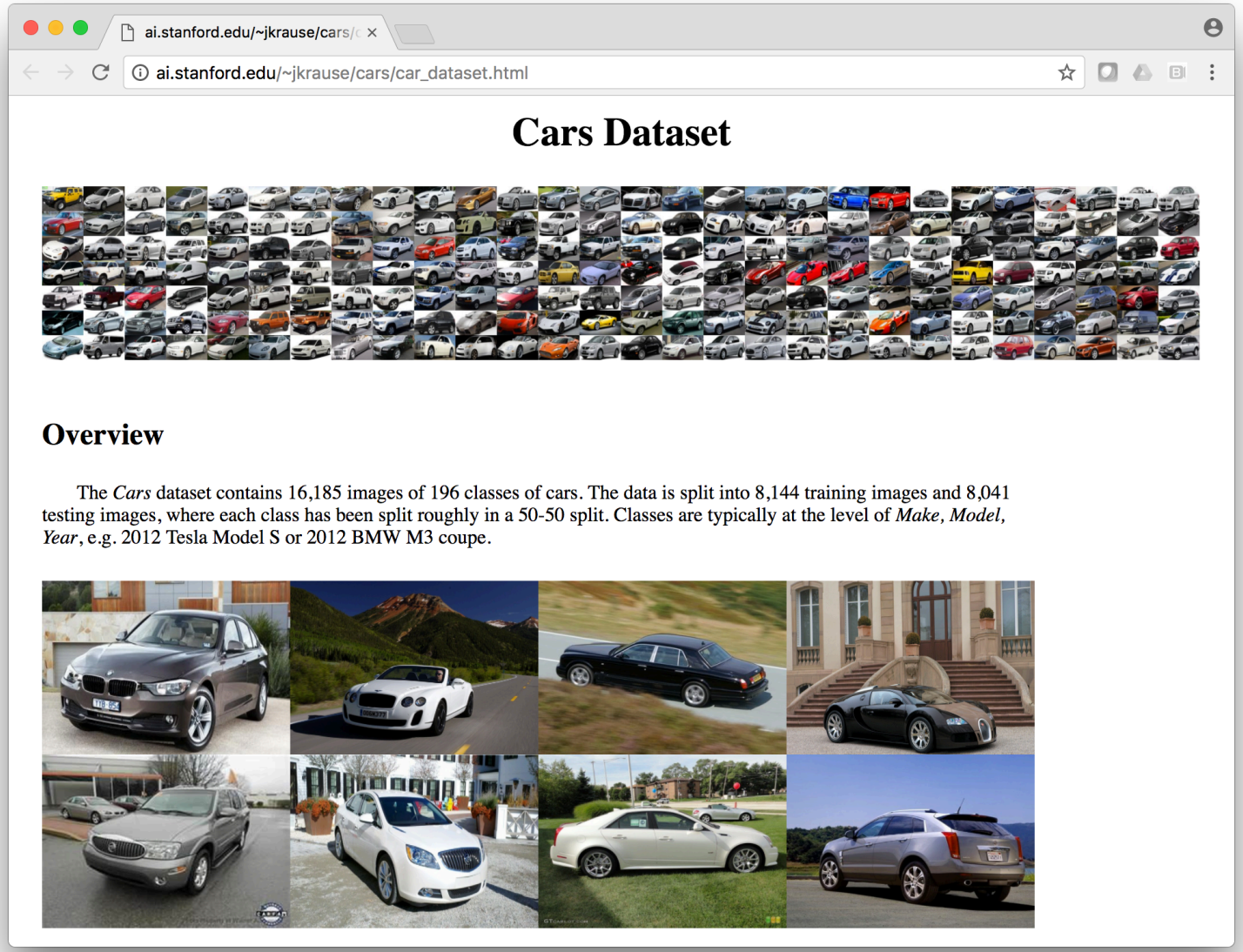
“Distance” Matrix

$$\mathbf{distance}\left(\mathbf{A},\mathbf{B}\right)=\mathbf{Z}:z_{i,j}=\mathbf{similarity}\left(\mathbf{a}_i,\mathbf{b}_j\right)=\mathbf{sim}\left(\mathbf{a}_i,\mathbf{b}_j\right)$$

$$\text{, where } \mathbf{A}\in\mathbb{R}^{M\times P},\mathbf{B}\in\mathbb{R}^{O\times P},\mathbf{Z}\in\mathbb{R}^{M\times O}$$

$$\mathbf{Z}=\begin{bmatrix}\mathbf{sim}\left(\mathbf{a}_1,\mathbf{b}_1\right) & \dots & \mathbf{sim}\left(\mathbf{a}_1,\mathbf{b}_O\right)\\ \vdots & \ddots & \vdots\\ \mathbf{sim}\left(\mathbf{a}_M,\mathbf{b}_1\right) & \dots & \mathbf{sim}\left(\mathbf{a}_M,\mathbf{b}_O\right)\end{bmatrix}$$

H2O.ai

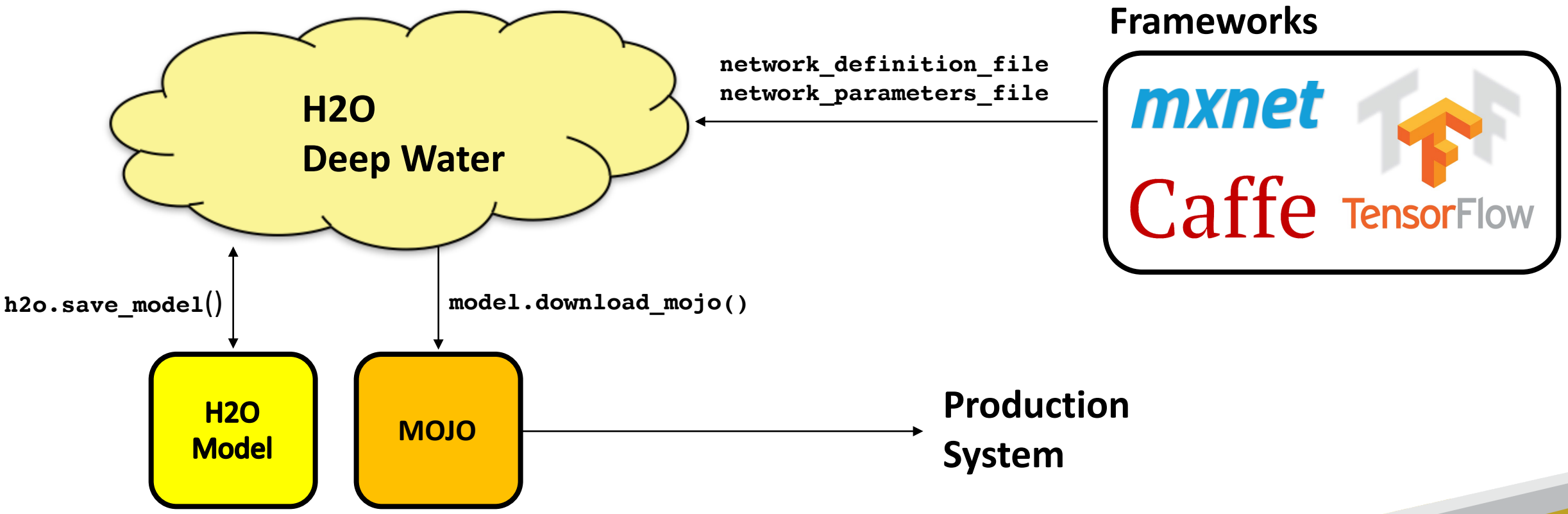


http://ai.stanford.edu/~jkrause/cars/car_dataset.html

H2O.ai

Deployment

Deployment

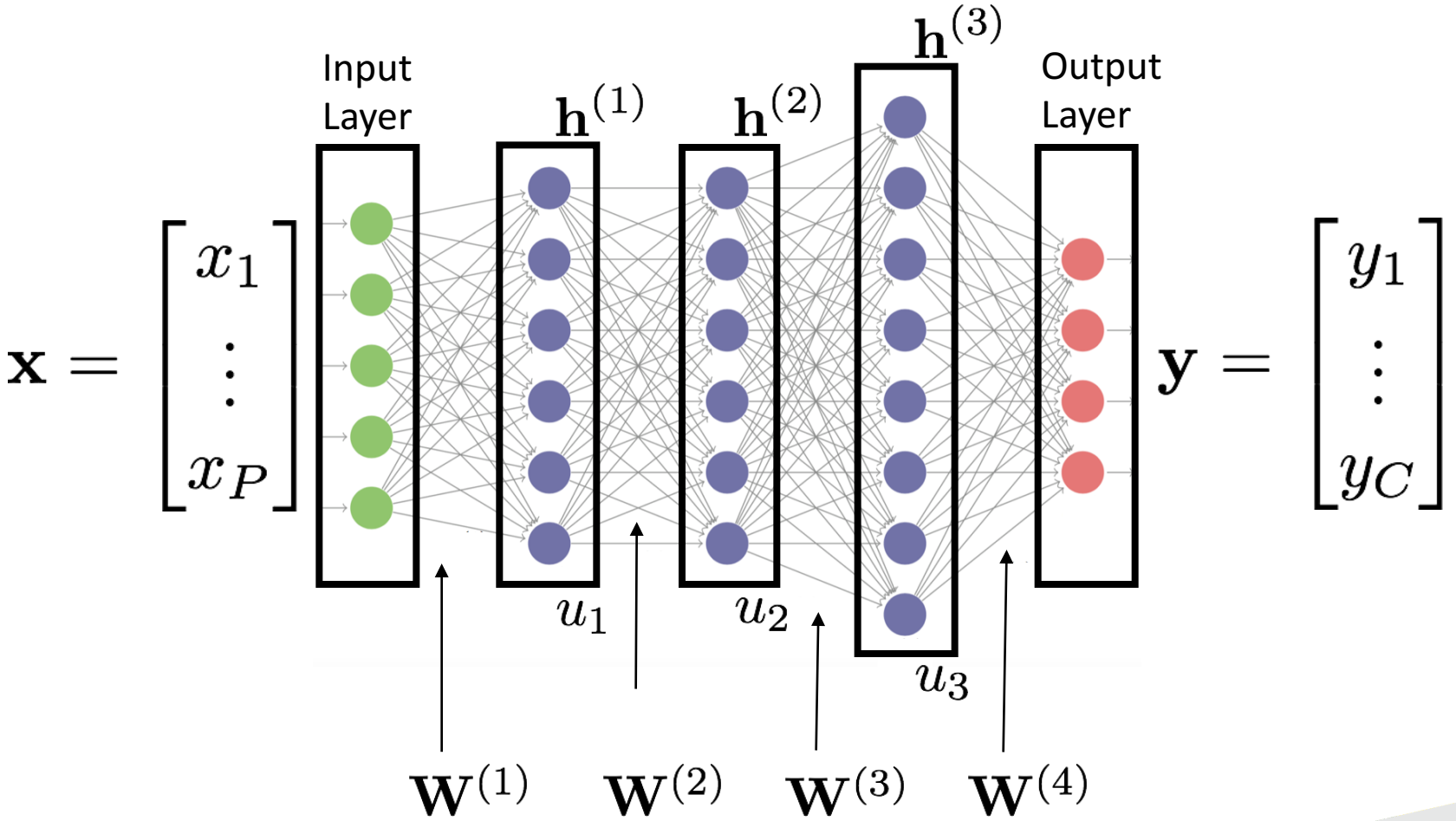

H2O
Deep Water
network_definition_file
network_parameters_file
Frameworks
mxnet
Caffe
TensorFlow
h2o.save_model()
model.download_mojo()
H2O
Model
MOJO
Production
System

Slides and Materials

https://github.com/h2oai/h2o-tutorials/gtc-2017-deep-water

Appendix: Deep Learning Review

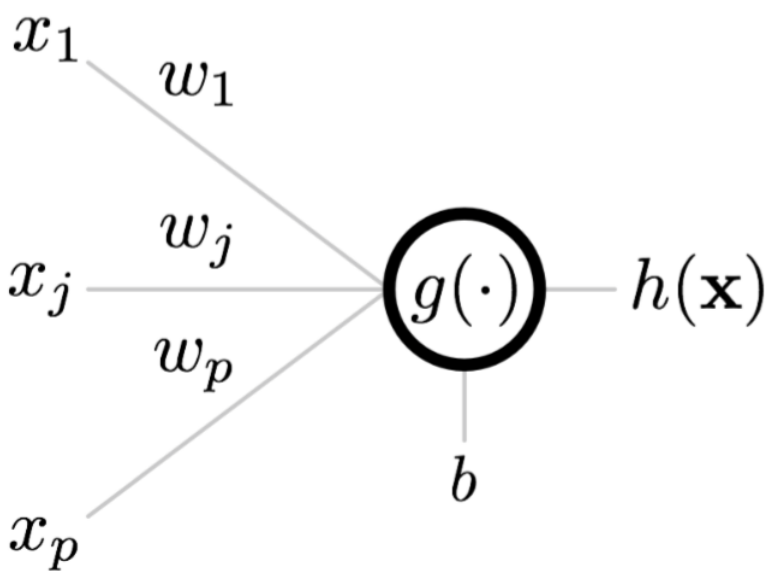
Neural Network



Efron and Hastie. Computer Age Statistical Inference.

Neuron

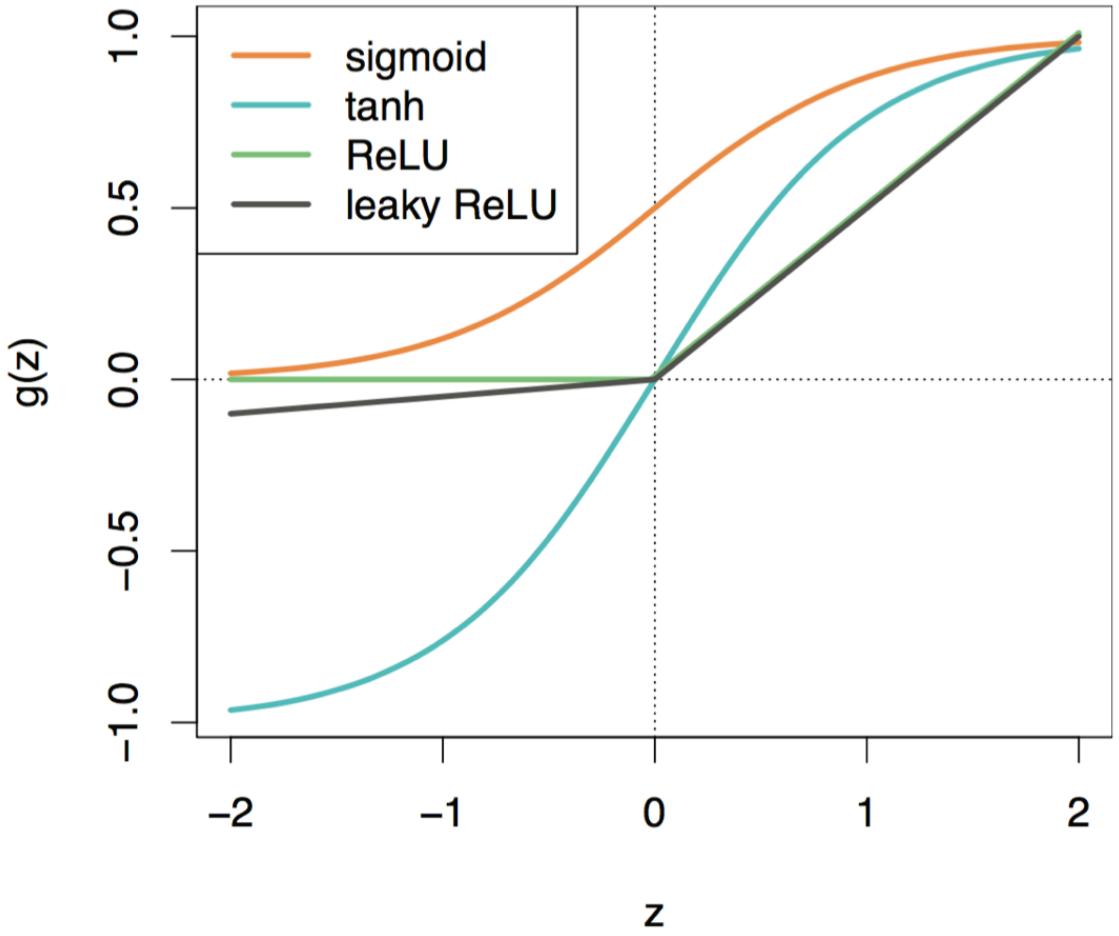
$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$$



$$h(\mathbf{x}) = g\left(\sum_{i=1}^{p} w_i x_i + b\right)$$
$$= g(\mathbf{w}^T \mathbf{x} + b)$$

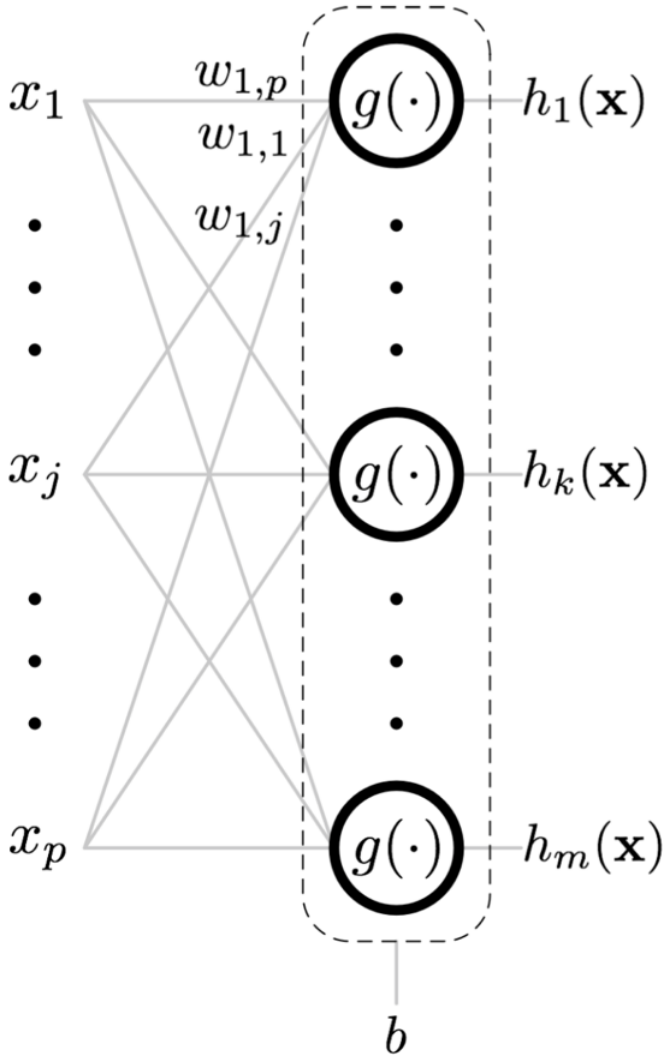
H$_2$O.ai

Activation Functions



Efron and Hastie. Computer Age Statistical Inference.

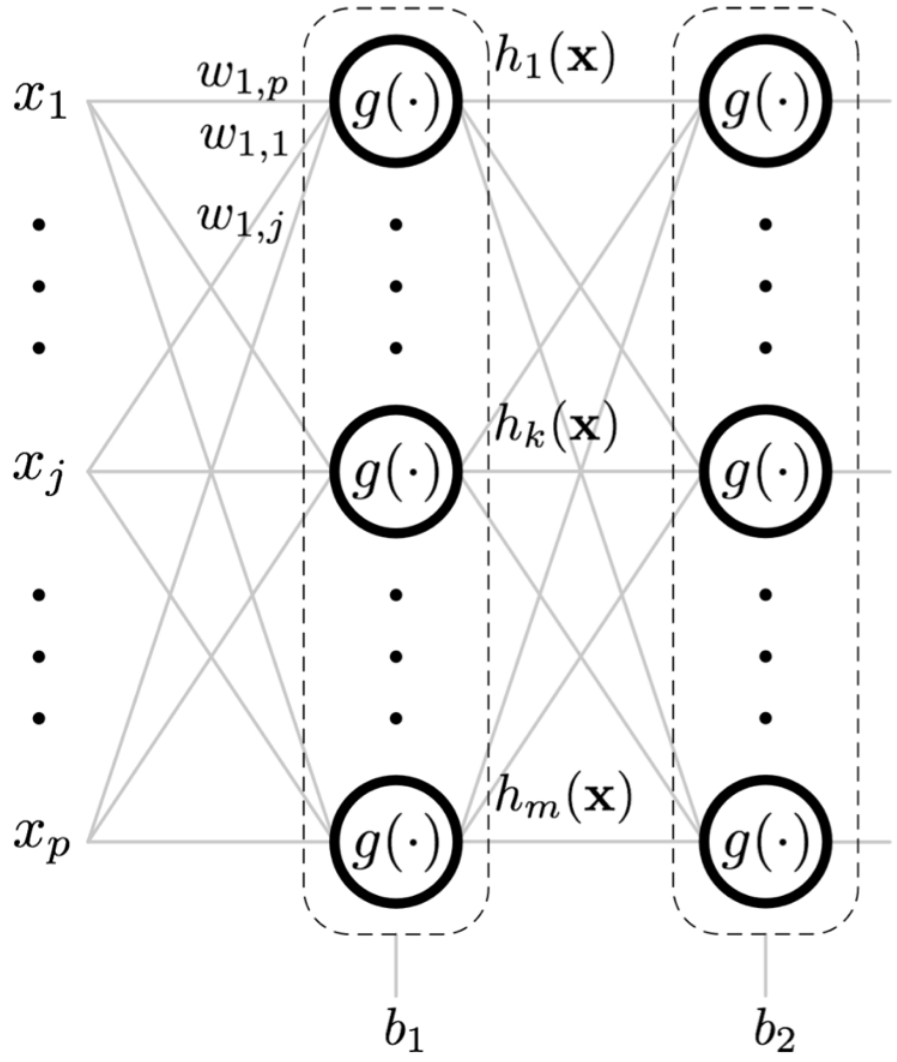
Layer of Neurons



$$h_m(\mathbf{x}) = g\left(\sum_{i=1}^{p} w_{m,i}x_i + b\right)$$

H$_2$O.ai

Hidden Layers



- Output from one layer is input to next layer
- L hidden layers
- Layer pre-activation, $k > 0$
  $a^{(k)}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}$
  $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$
- Hidden layer activation,
  $k \in [1, L]$
  $\mathbf{h}^{(k)}(\mathbf{x}) = g\left((a^{(k)}(\mathbf{x})\right)$

H$_2$O.ai

Output Layer

$$0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, 1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \cdots, 9 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{Pr}\left(y = k|\mathbf{x}\right) = \begin{bmatrix} \mathbf{Pr}\left(y = 0|\mathbf{x}\right) \\ \mathbf{Pr}\left(y = 1|\mathbf{x}\right) \\ \vdots \\ \mathbf{Pr}\left(y = 9|\mathbf{x}\right) \end{bmatrix}$$

$, k \in K = \{0, 1, \ldots, 9\}$

Output Layer Requirements

- All probabilities must sum up to one.
  $$\sum_{k \in K} \mathbf{Pr}\left(y = k | \mathbf{x}\right) = 1$$
- Probabilities must be between 0 and 1.
  $$0 \leq \mathbf{Pr}\left(y = k | \mathbf{x}\right) \leq 1, k \in K$$

Softmax

$$\mathbf{softmax}_k(a_1, a_2, \ldots, a_{|K|}) = \frac{\exp(a_k)}{\sum_{j \in K} \exp(a_j)}$$

$$= \frac{\text{evidence for class } k}{\text{total evidence for all classes}}$$

Cross Entropy

$$H(p, q) = -\sum_i p_i \log q_i$$

$$= -\sum_i p_i \log_e q_i$$

$$= -\sum_i p_i \ln q_i$$

Cross-Entropy (cont.)

- Binary Classification: $p \in \{y, 1-y\},\ q \in \{\hat{y}, 1-\hat{y}\}$

$$
\begin{aligned}
H(p,q) &= -\sum_{i=1}^{2} p_i \ln q_i \\
&= -p_1 \ln q_1 - p_2 \ln q_2 \\
H(y,\hat{y}) &= -y \ln \hat{y} - (1-y)\ln(1-\hat{y})
\end{aligned}
$$

  - $y = 1$: $H(1,\hat{y}) = \ln \hat{y}$
  - $y = 0$: $H(0,\hat{y}) = \ln(1-\hat{y})$

H$_2$O.ai

Cross-Entropy Loss Function

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} H(p_j, q_j)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[\sum_{j=1}^{K} y_j^{(i)} \log \hat{y}_j^{(i)} \right]$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[\sum_{j=1}^{10} y_j^{(i)} \log \hat{y}_j^{(i)} \right]$$

Training and Optimization

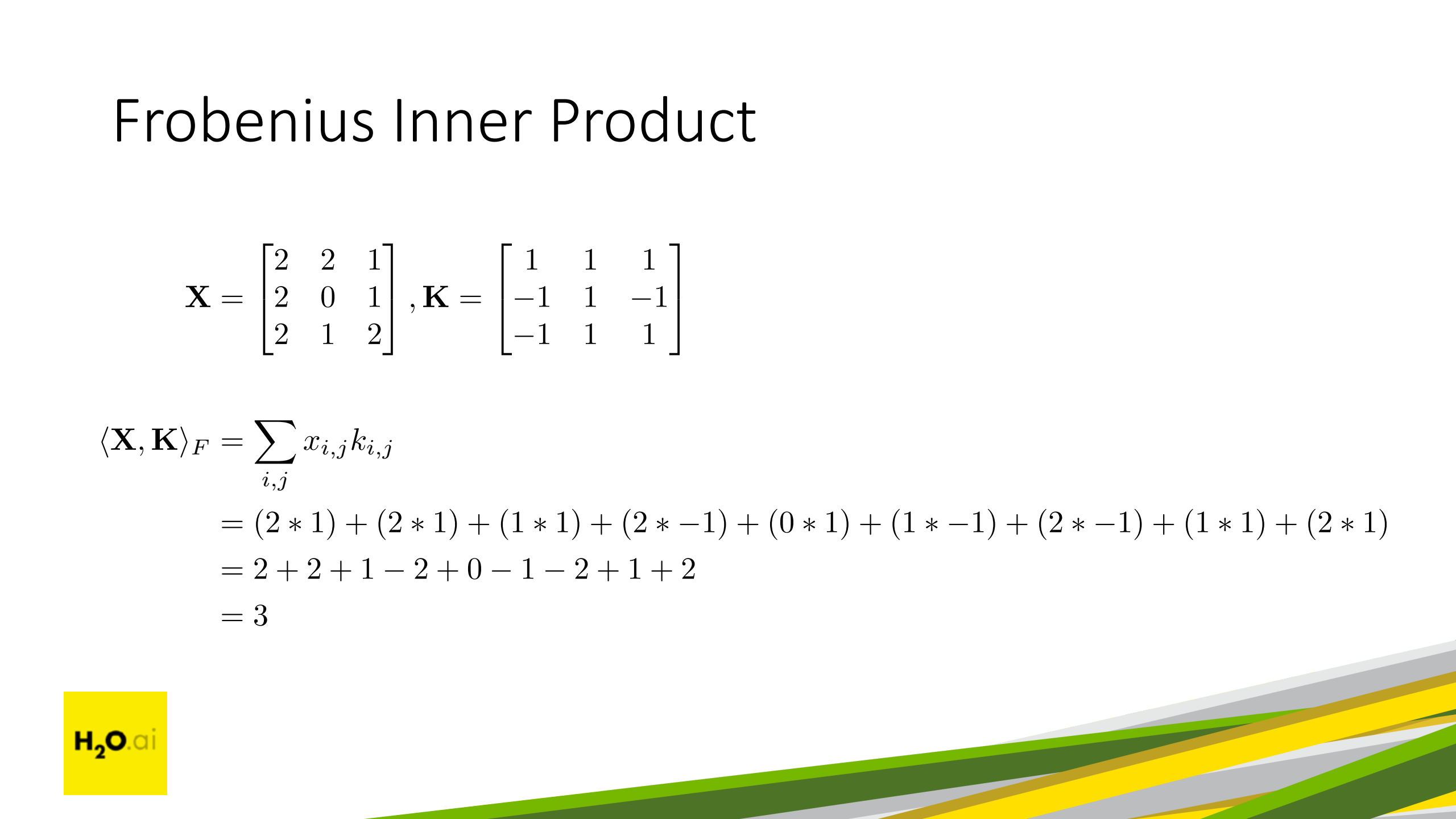
$$\underset{\mathbf{W},\mathbf{b}}{\text{minimize}} \; -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{j=1}^{K} y_j \log \hat{y}_j\right]$$

Appendix: Convolutional Neural Networks

Frobenius Inner Product

$$\mathbf{X} = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 0 & 1 \\ 2 & 1 & 2 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$$

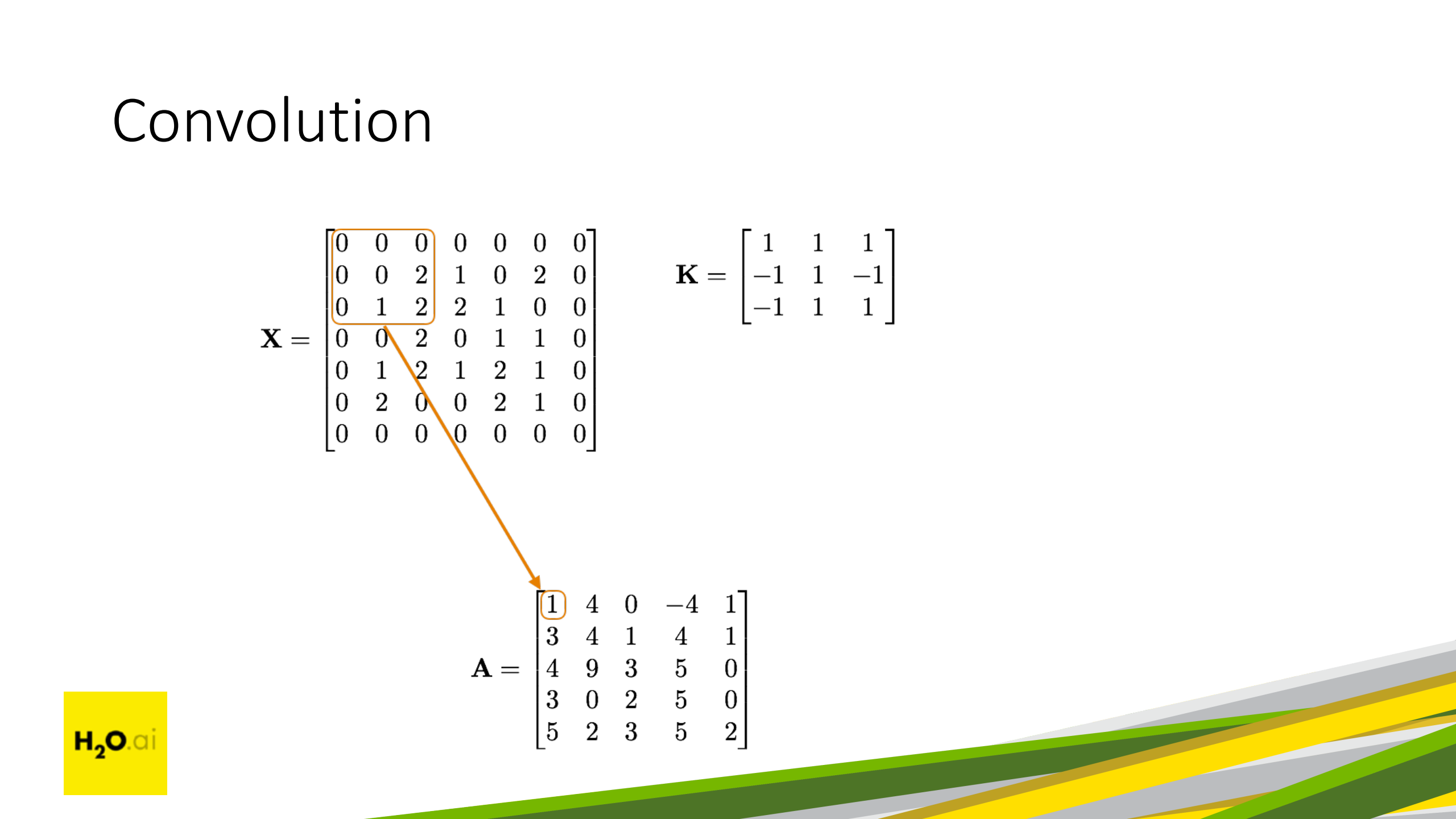
$$\begin{aligned} \langle \mathbf{X}, \mathbf{K} \rangle_F &= \sum_{i,j} x_{i,j} k_{i,j} \\ &= (2 * 1) + (2 * 1) + (1 * 1) + (2 * -1) + (0 * 1) + (1 * -1) + (2 * -1) + (1 * 1) + (2 * 1) \\ &= 2 + 2 + 1 - 2 + 0 - 1 - 2 + 1 + 2 \\ &= 3 \end{aligned}$$



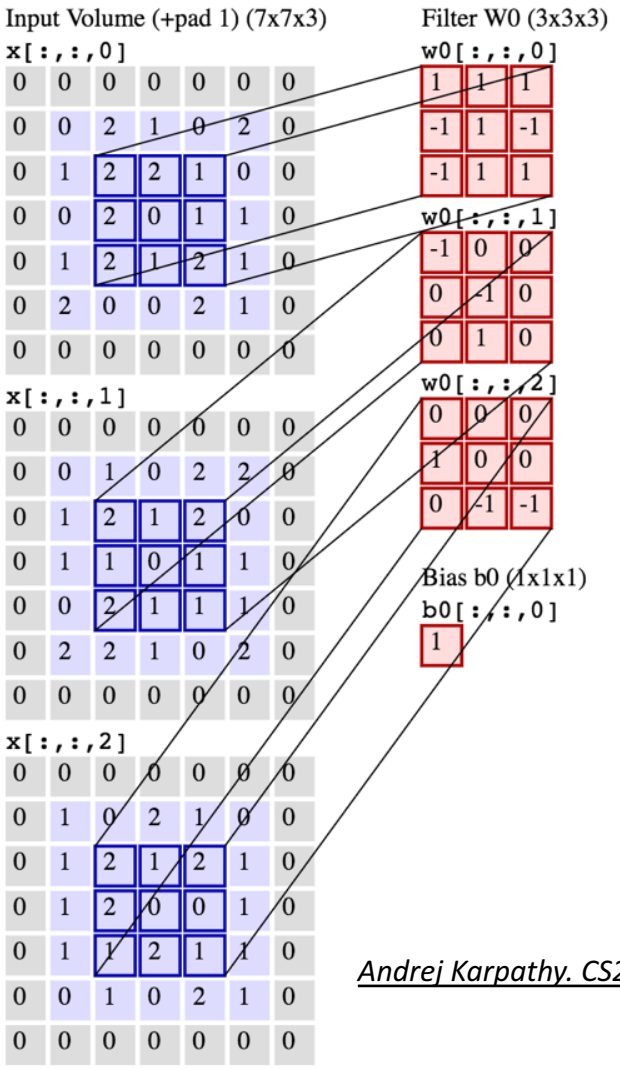
Convolution

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 2 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{K} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 0 & -4 & 1 \\ 3 & 4 & 1 & 4 & 1 \\ 4 & 9 & 3 & 5 & 0 \\ 3 & 0 & 2 & 5 & 0 \\ 5 & 2 & 3 & 5 & 2 \end{bmatrix}$$



Convolution

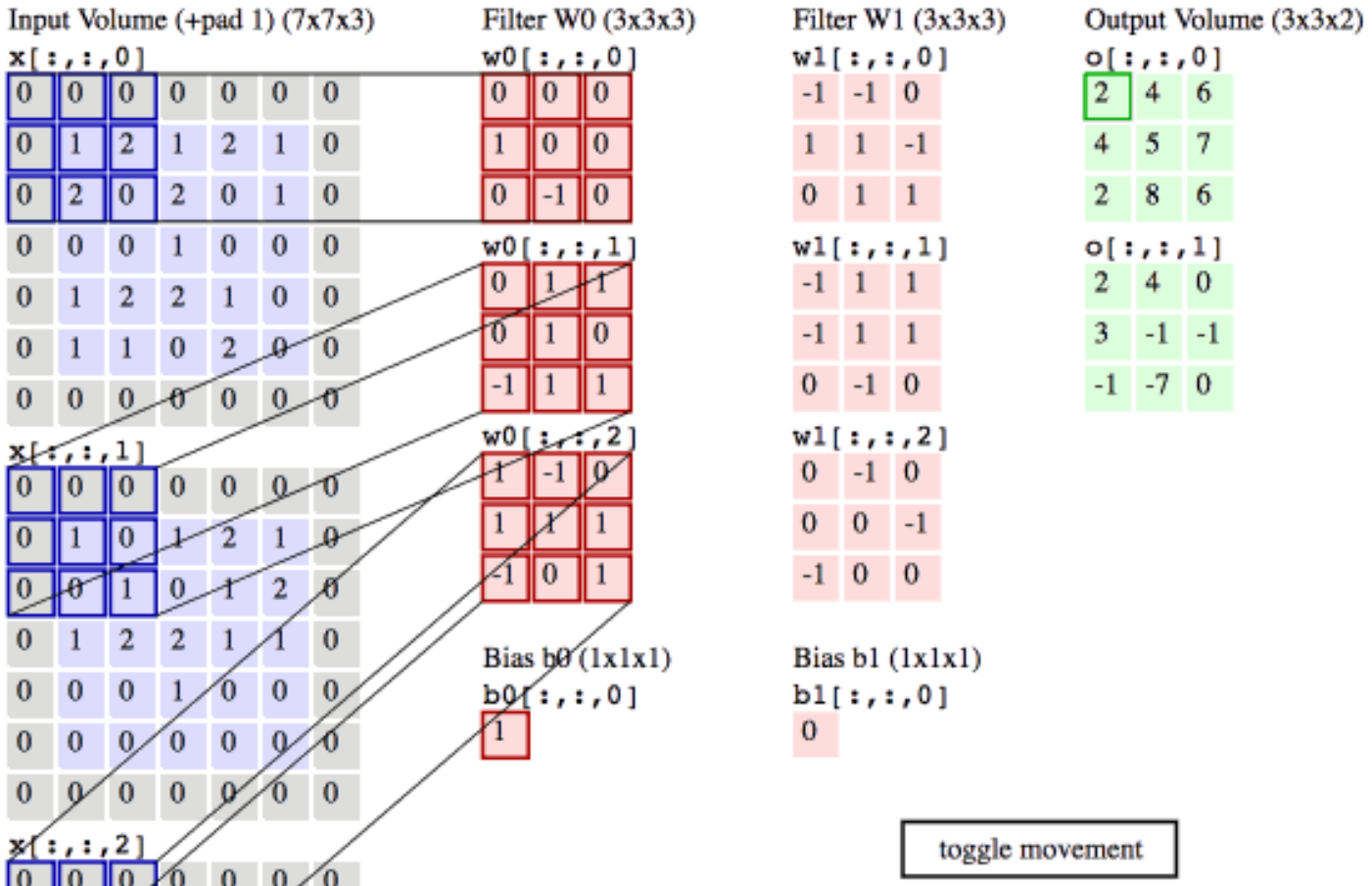


$$\sum_{d} \sum_{i,j} x_{i,j,d} k_{i,j,d}$$

Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition.

H2O.ai

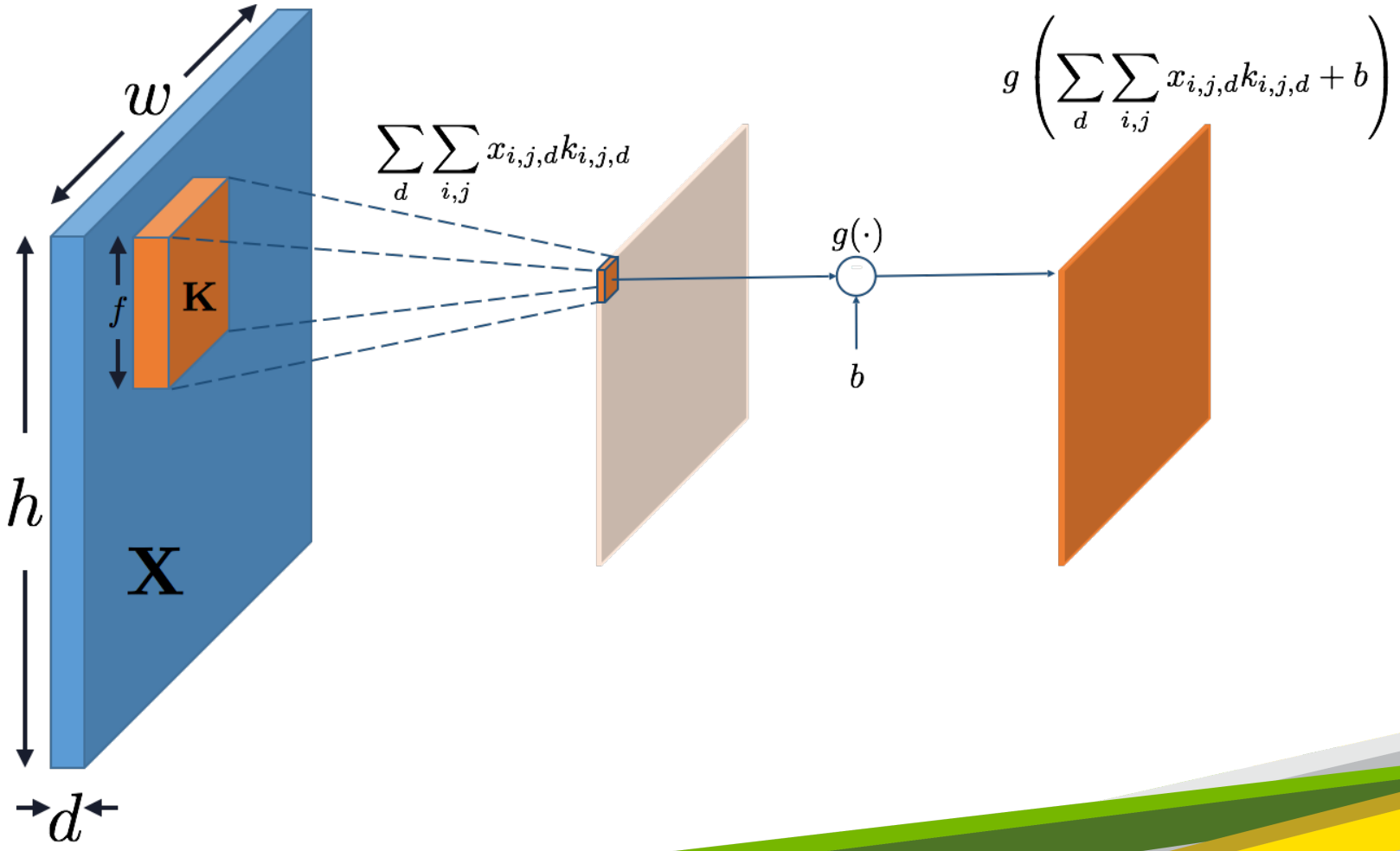
Convolution Layer



Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition.

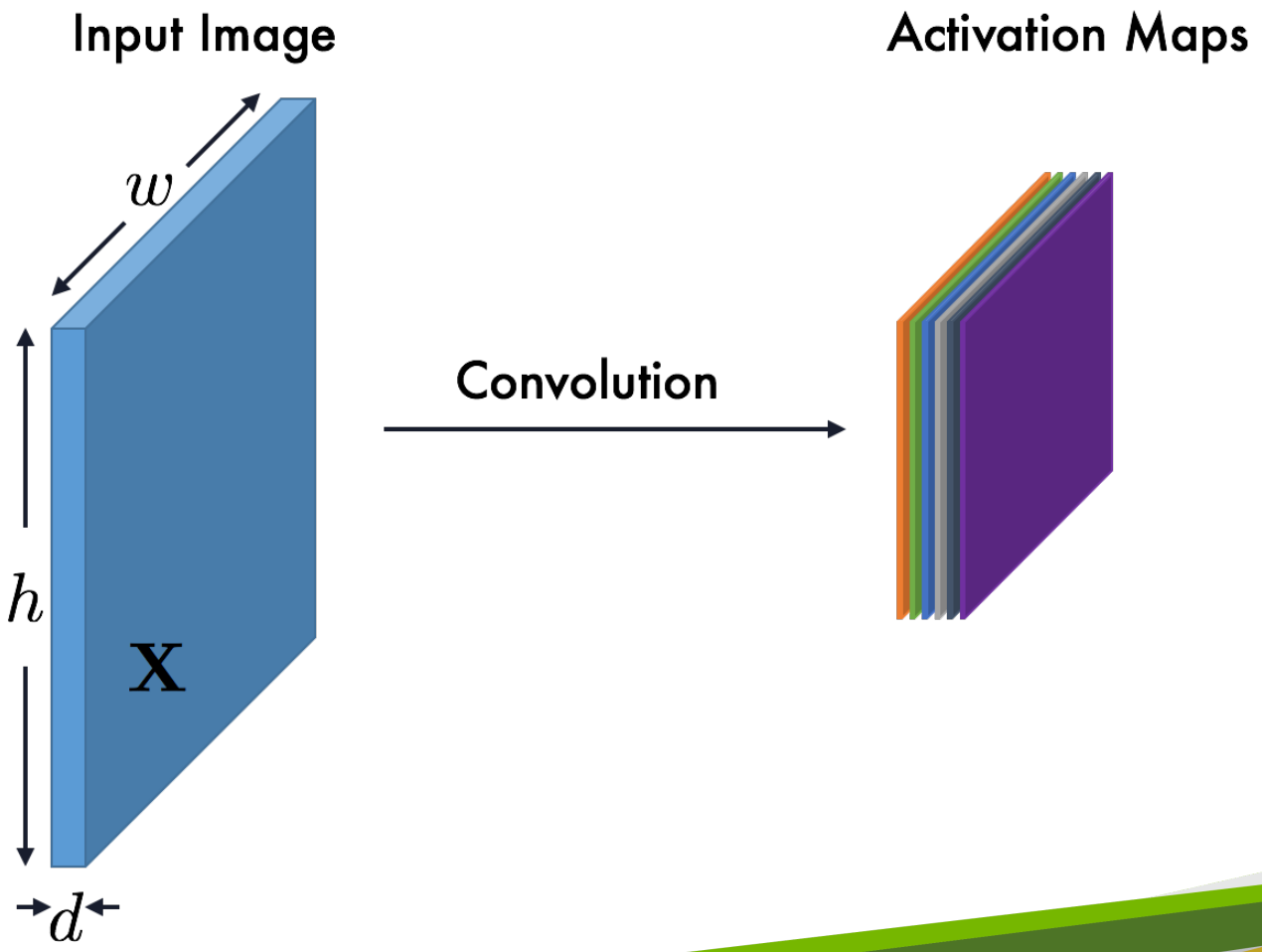
H2O.ai

Convolution Layer

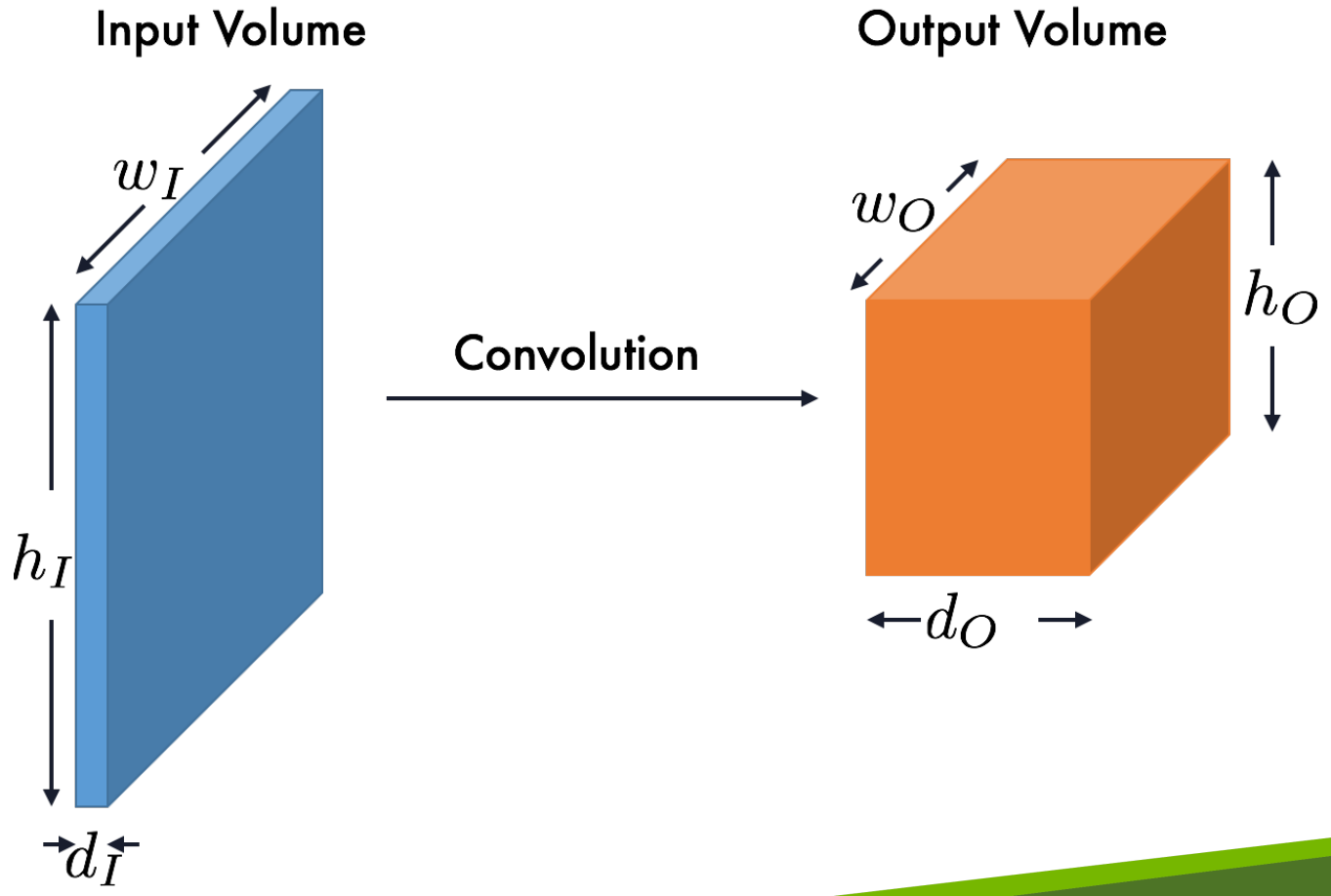


H2O.ai

Convolution Layer



Convolution Layer



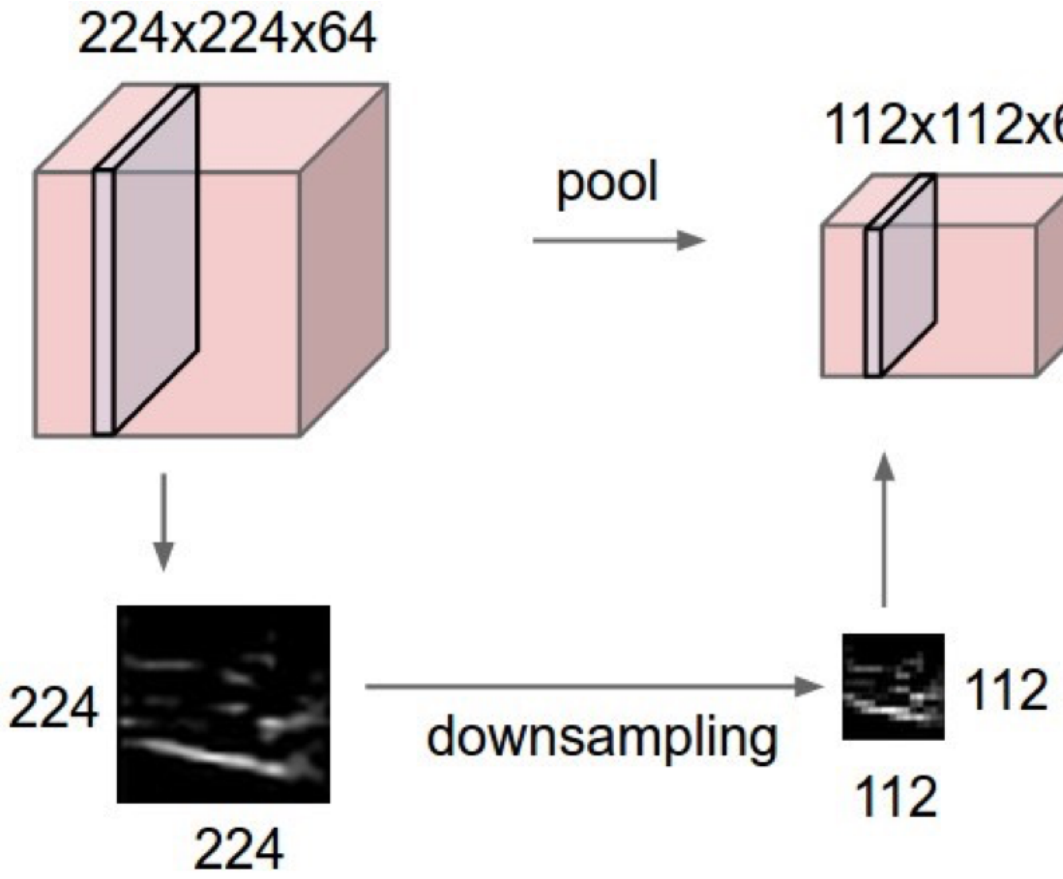
$$w_O = \frac{w_I - f + 2p}{s} + 1$$

$$h_O = \frac{w_I - f + 2p}{s} + 1$$

$$d_O = m$$

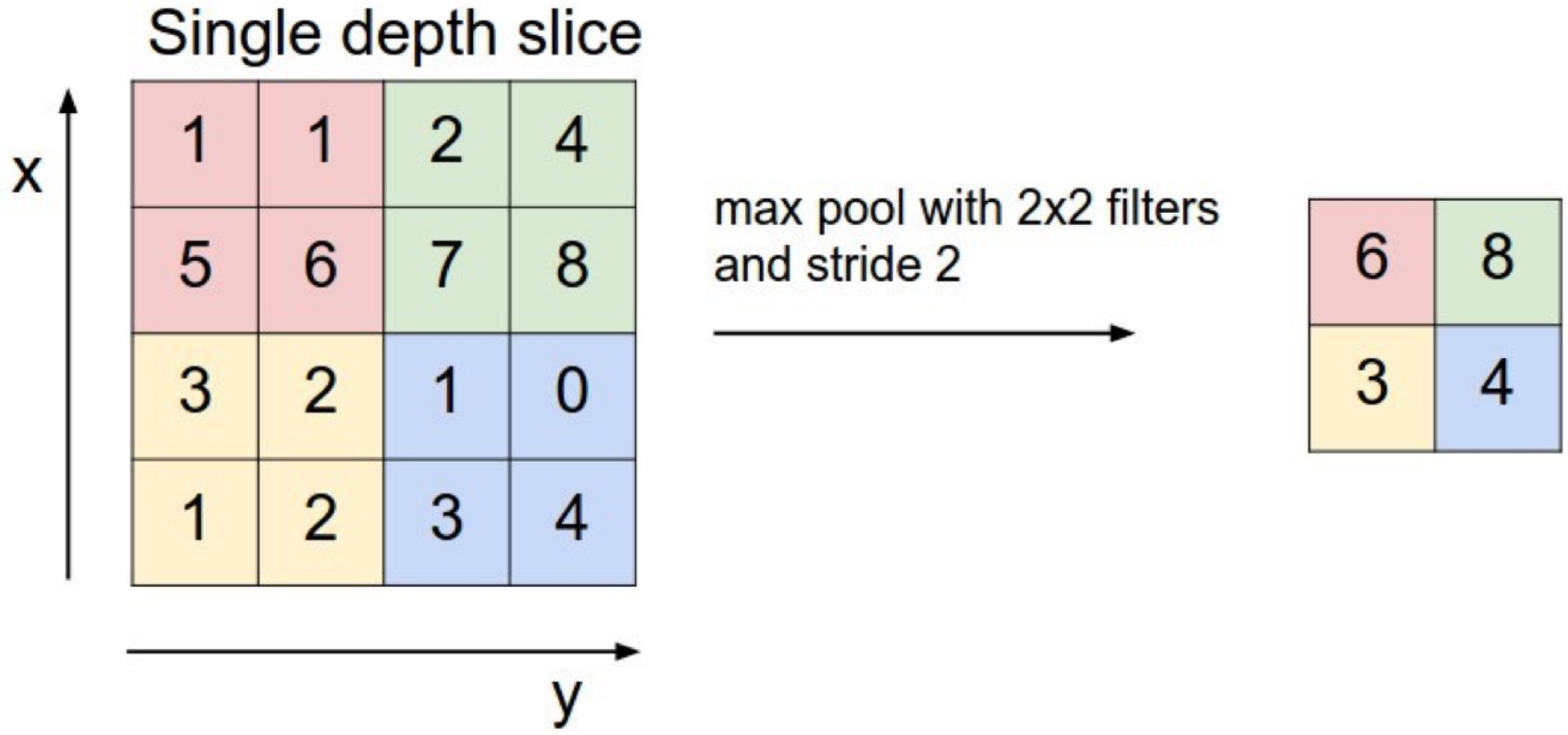
H$_2$O.ai

Pooling



Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition.

Max Pooling



Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition.

LeNet

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. **Gradient-Based Learning Applied to Document Recognition**. In *Proceedings of the IEEE*, Nov. 1998. URL `http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf`

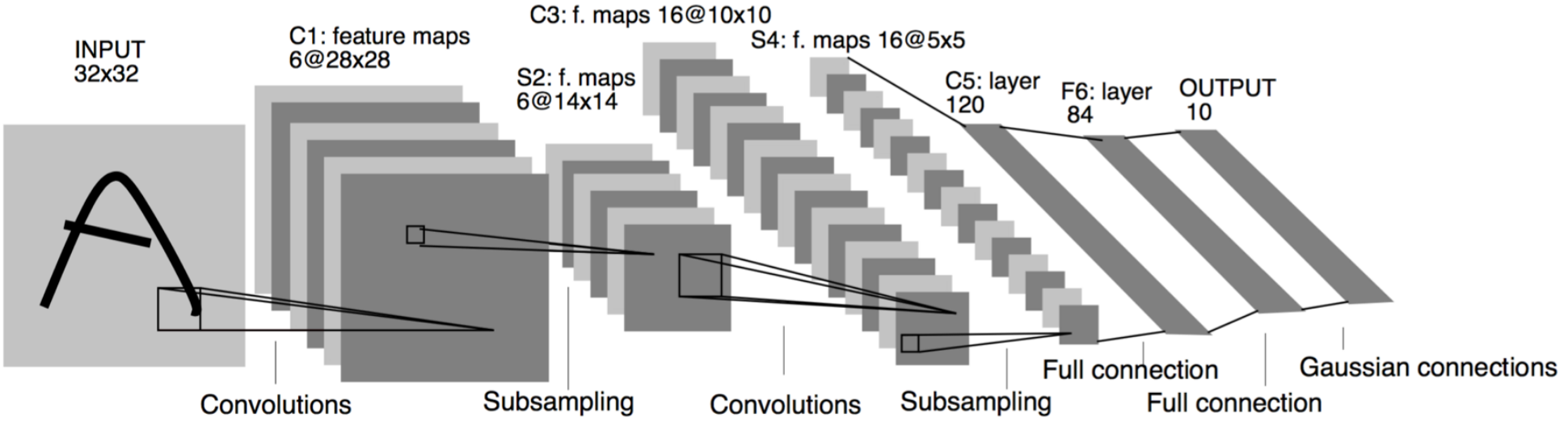


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

H2O.ai