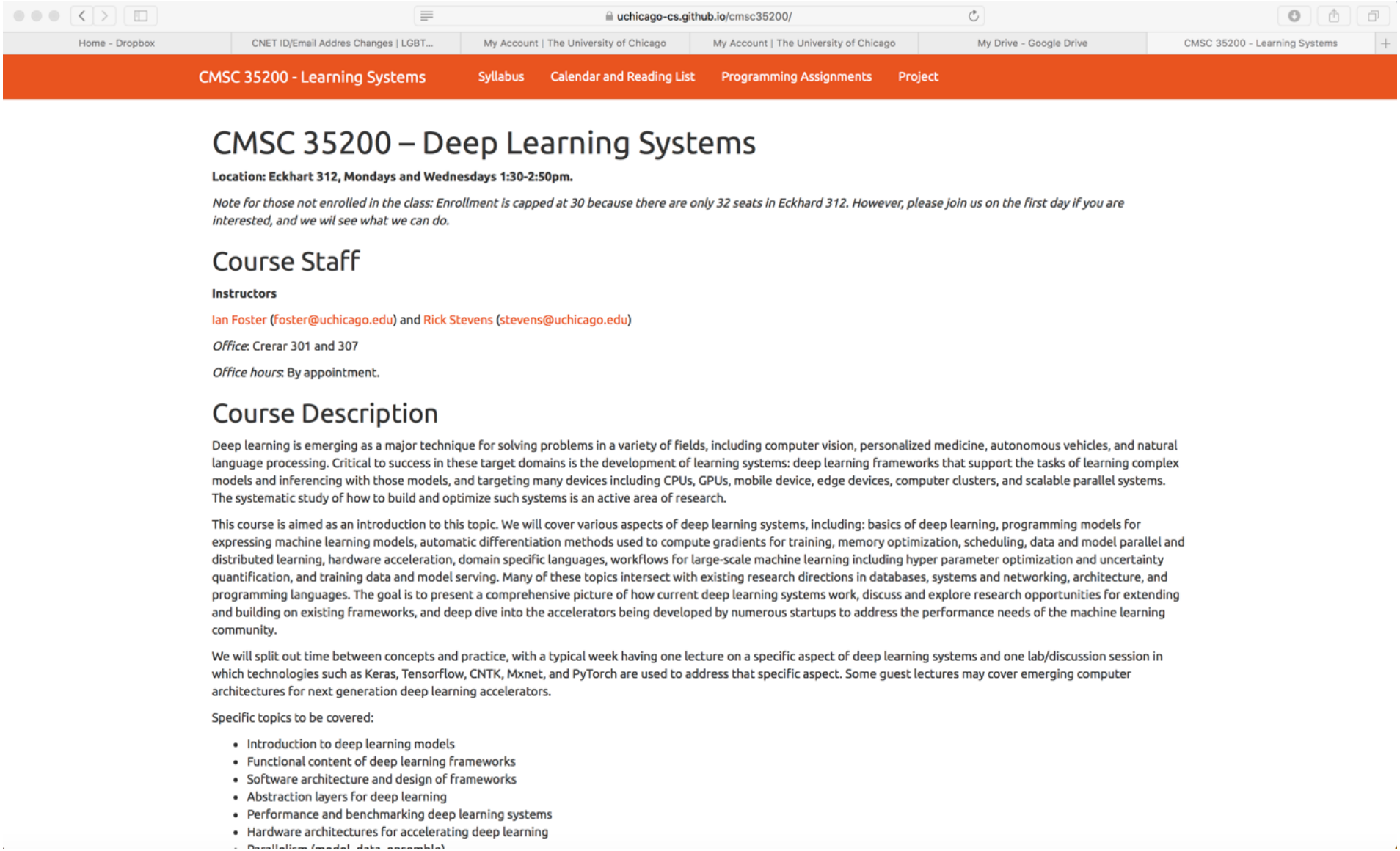
Learning Systems 2018:
Lecture 1 – Introduction to Deep Learning

Crescat scientia; vita excolatur

Ian Foster and Rick Stevens
Argonne National Laboratory
The University of Chicago



CMSC 35200 – Deep Learning Systems

Location: Eckhart 312, Mondays and Wednesdays 1:30-2:50pm.

Note for those not enrolled in the class: Enrollment is capped at 30 because there are only 32 seats in Eckhard 312. However, please join us on the first day if you are interested, and we wil see what we can do.

Course Staff

Instructors

Ian Foster (foster@uchicago.edu) and Rick Stevens (stevens@uchicago.edu)

Office: Crerar 301 and 307

Office hours: By appointment.

Course Description

Deep learning is emerging as a major technique for solving problems in a variety of fields, including computer vision, personalized medicine, autonomous vehicles, and natural language processing. Critical to success in these target domains is the development of learning systems: deep learning frameworks that support the tasks of learning complex models and inferencing with those models, and targeting many devices including CPUs, GPUs, mobile device, edge devices, computer clusters, and scalable parallel systems. The systematic study of how to build and optimize such systems is an active area of research.

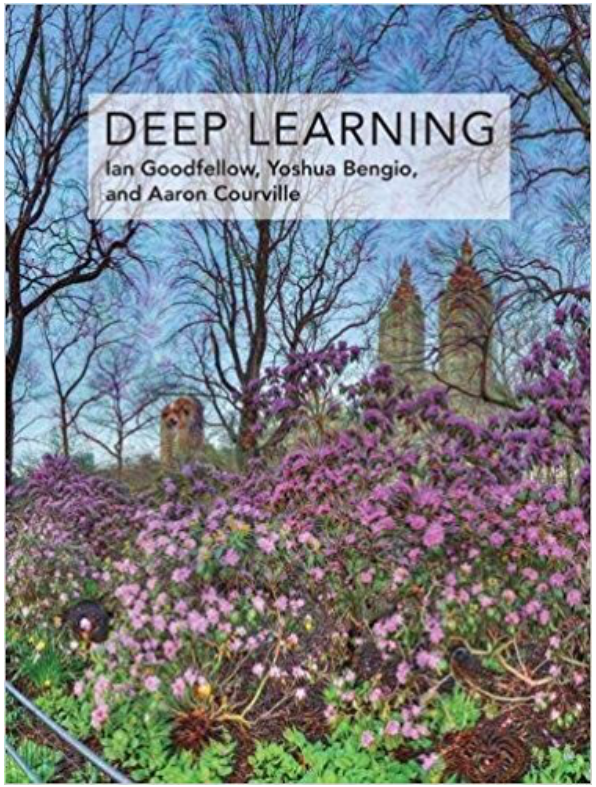
This course is aimed as an introduction to this topic. We will cover various aspects of deep learning systems, including: basics of deep learning, programming models for expressing machine learning models, automatic differentiation methods used to compute gradients for training, memory optimization, scheduling, data and model parallel and distributed learning, hardware acceleration, domain specific languages, workflows for large-scale machine learning including hyper parameter optimization and uncertainty quantification, and training data and model serving. Many of these topics intersect with existing research directions in databases, systems and networking, architecture, and programming languages. The goal is to present a comprehensive picture of how current deep learning systems work, discuss and explore research opportunities for extending and building on existing frameworks, and deep dive into the accelerators being developed by numerous startups to address the performance needs of the machine learning community.

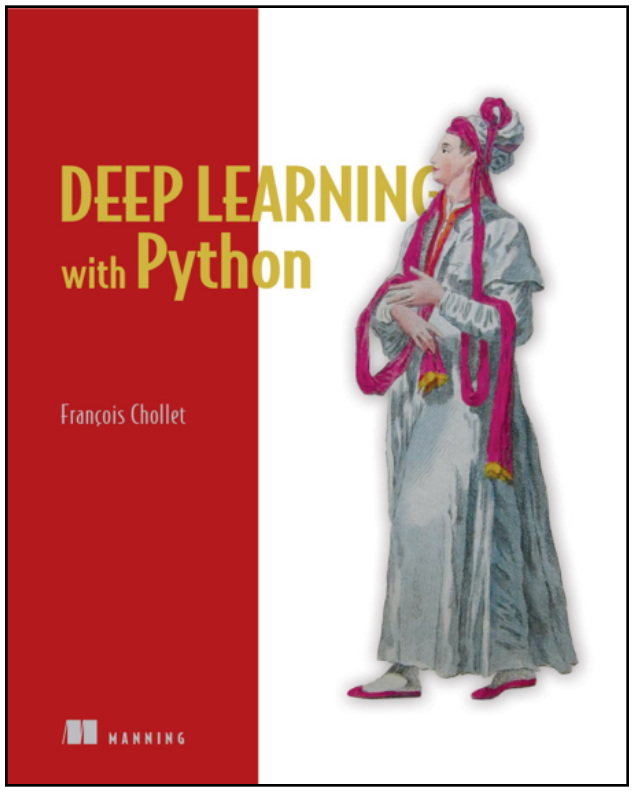
We will split out time between concepts and practice, with a typical week having one lecture on a specific aspect of deep learning systems and one lab/discussion session in which technologies such as Keras, Tensorflow, CNTK, Mxnet, and PyTorch are used to address that specific aspect. Some guest lectures may cover emerging computer architectures for next generation deep learning accelerators.

Specific topics to be covered:

- Introduction to deep learning models
- Functional content of deep learning frameworks
- Software architecture and design of frameworks
- Abstraction layers for deep learning
- Performance and benchmarking deep learning systems
- Hardware architectures for accelerating deep learning
- Parallelism (model, data, ensemble)

Deep Learning Books Worth Getting





Useful URLs

Deep Learning with Python (Francois Chollet)

- http://www.deeplearningitalia.com/wp-content/uploads/2017/12/Dropbox_Chollet.pdf
- https://www.manning.com/books/deep-learning-with-python
- https://github.com/fchollet/deep-learning-with-python-notebooks

Deep Learning Book (Goodfellow, Bengio, Courville)

- https://www.deeplearningbook.org
- https://github.com/janishar/mit-deep-learning-book-pdf

www.arxiv-sanity.com

Arxiv Sanity Preserver
Built in spare time by @karpathy to accelerate research.
Serving last 55551 papers from cs.[CV|CL|LG|AI|NE]/stat.ML

stevens | log out

Fork me on GitHub

User stevens logged in.

most recent | top recent | top hype | friends | discussions | recommended | library

Only show v1

Showing most recent Arxiv papers:

Towards Understanding Regularization in Batch Normalization
1809.00846v2 pdf
Ping Luo, Xinjiang Wang, Wenqi Shao, Zhanglin Peng
show similar | discuss
9/27/2018 (v1: 9/4/2018) cs.LG | cs.CV | cs.SY | stat.ML
Preprint. Work in progress. 17 pages

Batch Normalization (BN) improves both convergence and generalization in training neural networks. This work understands these phenomena theoretically. We analyze BN by using a basic block of neural networks, consisting of a kernel layer, a BN layer, and a nonlinear activation function. This basic network helps us understand the impacts of BN in three aspects. First, by viewing BN as an implicit regularizer, BN can be decomposed into population normalization (PN) and gamma decay as an explicit regularization. Second, learning dynamics of BN and the regularization show that training converged with large maximum and effective learning rate. Third, generalization of BN is explored by using statistical mechanics. Experiments demonstrate that BN in convolutional neural networks share the same traits of regularization as the above analyses.

Learning to Coordinate Multiple Reinforcement Learning Agents for Diverse Query Reformulation
1809.10658v1 pdf
Rodrigo Nogueira, Jannis Bulian, Massimiliano Ciaramita
show similar | discuss
9/27/2018 cs.LG | stat.ML

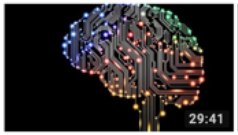
Alena Kruchkova

8,066 subscribers

SUBSCRIBED 8K

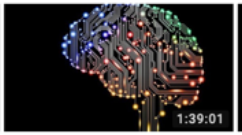
HOME | VIDEOS | PLAYLISTS | CHANNELS | DISCUSSION | ABOUT

Uploads | PLAY ALL | SORT BY



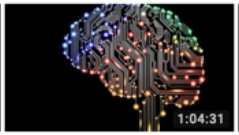
Live Stream Chapter 19: Approximate Inference with...

2K views • 11 months ago



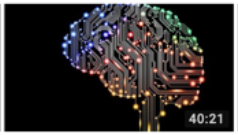
Live Stream Chapter 16: Structured Probabilistic...

2.1K views • 11 months ago



Live Stream Chapter 20: Deep Generative Models with...

1.6K views • 11 months ago

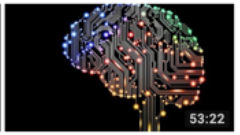


Live Stream Chapter 13: Linear Factor Models with...

1K views • 11 months ago

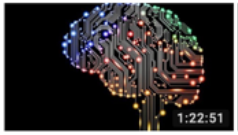
Live Stream Chapter18: Partition Function with Gavi...

833 views • 1 year ago



Live Stream Chapter 17: Monte Carlo Methods with...

982 views • 1 year ago



Live Stream Chapter 15: Representation Learning wit...

1K views • 1 year ago

Low Res Live Stream Capter 14: Autoencoders with...

1.2K views • 1 year ago

Low Res Live Stream Chapter 12: Applications with author...

1K views • 1 year ago

Deep Learning Chapter 10 Sequence Modeling:...

10K views • 1 year ago

Deep Learning Book Club CH11 Practical Methodology

1K views • 1 year ago

Deep Learning Book Club CH9 CNNs live stream

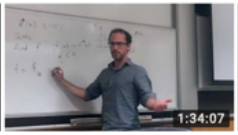
1.5K views • 1 year ago

Deep Learning Book Club CH8 live stream

1.2K views • 1 year ago

Deep Learning Book Club CH7 live stream

1.7K views • 1 year ago

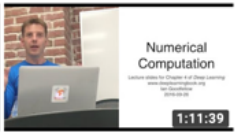


Deep Learning Chapter 6 Deep Feedforward Network...

7K views • 1 year ago

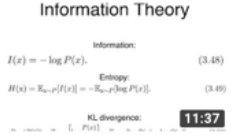
Deep Learning Book Club CH5 (Part 2) live stream

1.7K views • 1 year ago



Deep Learning Chapter 4 Numerical Computation...

6K views • 1 year ago



Deep Learning Chapter 3 Information Theory...

2.8K views • 1 year ago

Advance TensorFlow and Spark meetup

785 views • 1 year ago

Deep Learning Book Club CH5 live stream

4.8K views • 1 year ago

Deep Learing Chapter 3 Probability presented by...

8.1K views • 1 year ago

Deep Learning Q&A with Alexander Toshev

958 views • 1 year ago

Deep Learning Q&A with Yaroslav Bulatov

1.8K views • 1 year ago

Deep Learning Chapter 2 Linear Algebra presented by...

16K views • 1 year ago

What we will cover in this course

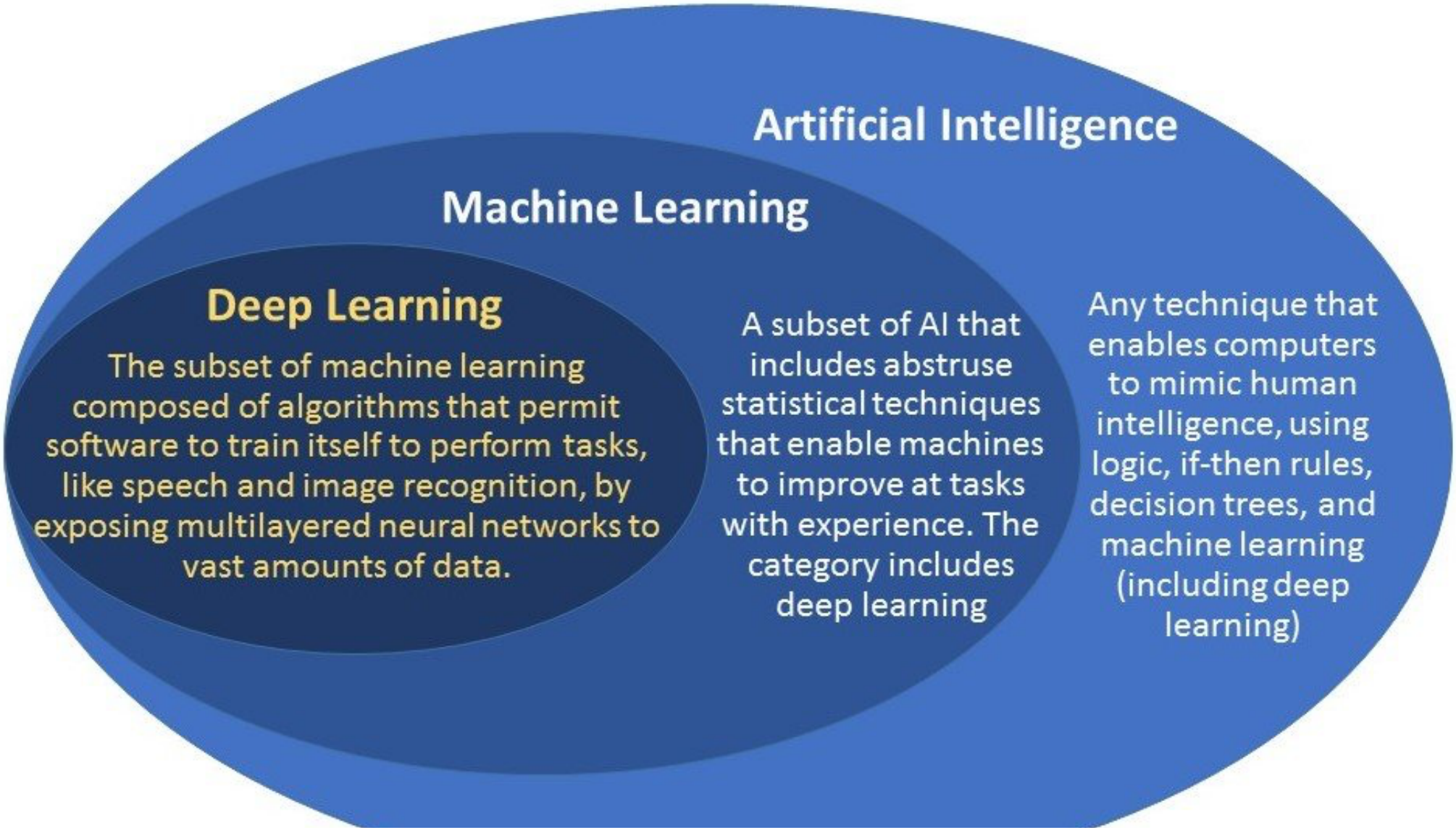
Course is aimed at the “systems” related topics associated with deep learning

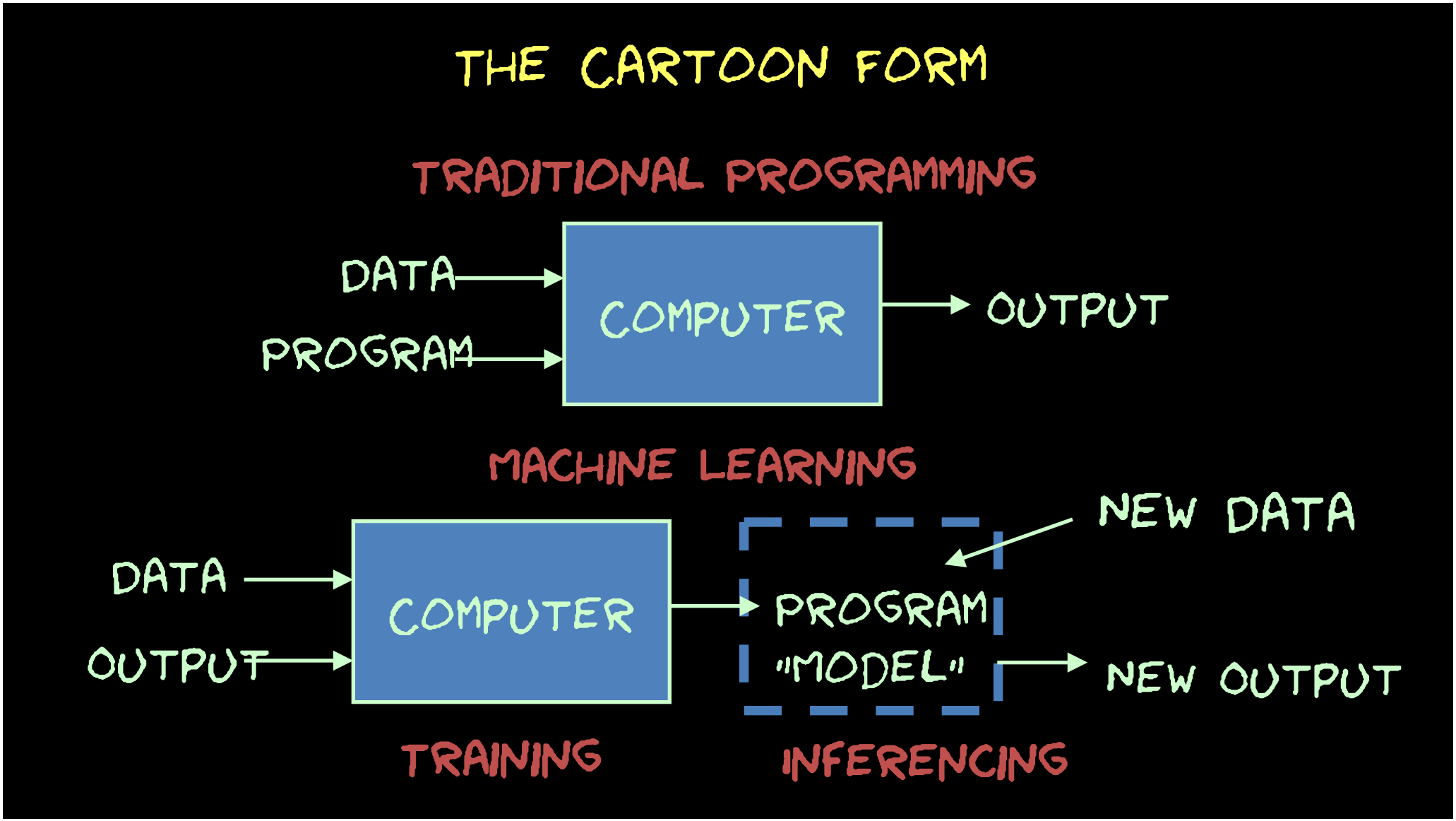
- Introduction to deep learning models
- Functional content of deep learning frameworks
- Software architecture and design of frameworks
- Abstraction layers for deep learning
- Performance and benchmarking deep learning systems
- Hardware architectures for accelerating deep learning
- Parallelism (model, data, ensemble)
- Portable representations and translations of models
- Optimization for training, inference
- Workflows for machine learning and workflow tools
- Hyper-parameter optimization and ensembles
- Uncertainty quantification

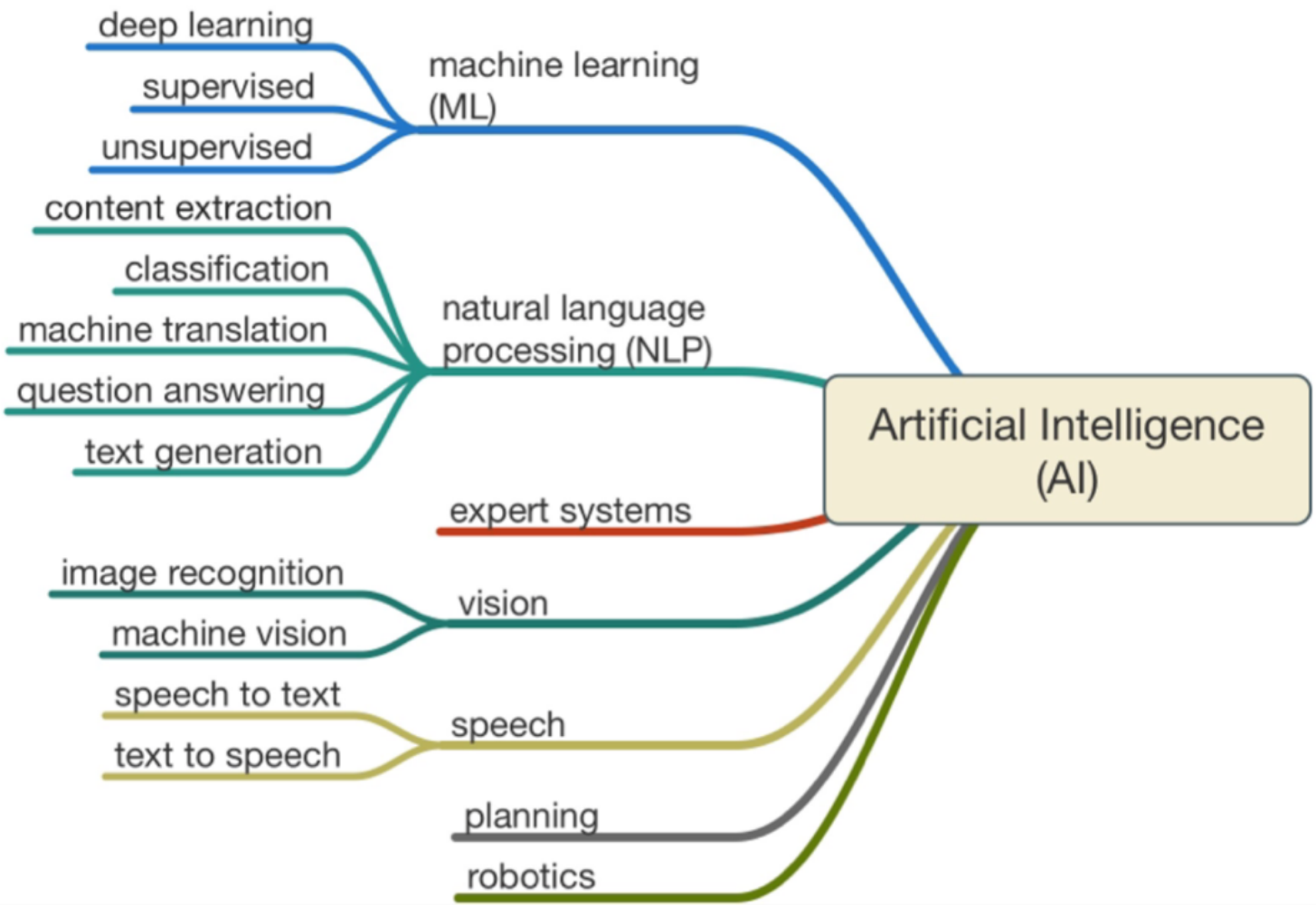
What we will not cover in this course

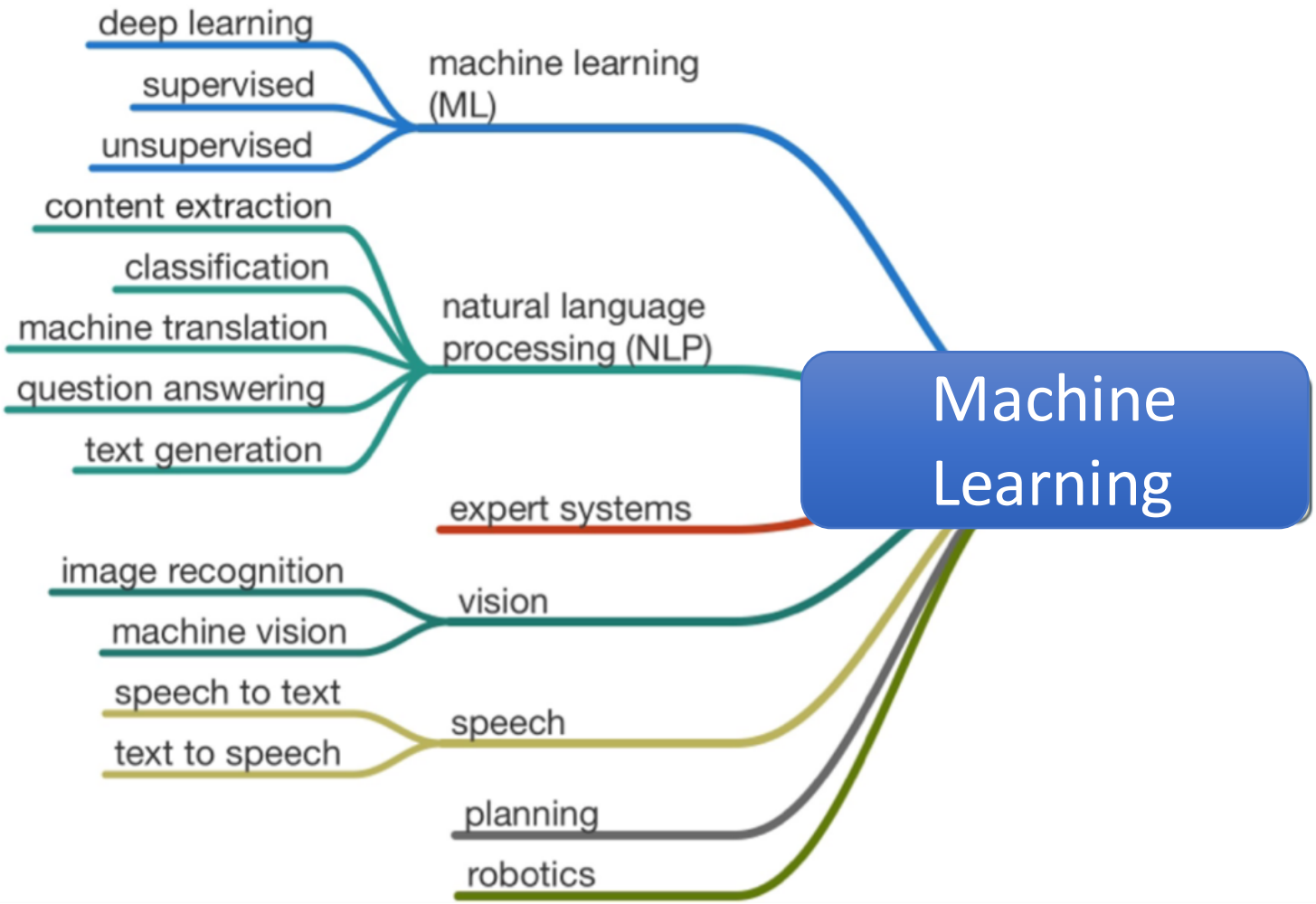
- Machine learning foundations
- Details of why deep learning methods work for various problems
- Research topics in deep learning methods development that are not related to systems issues

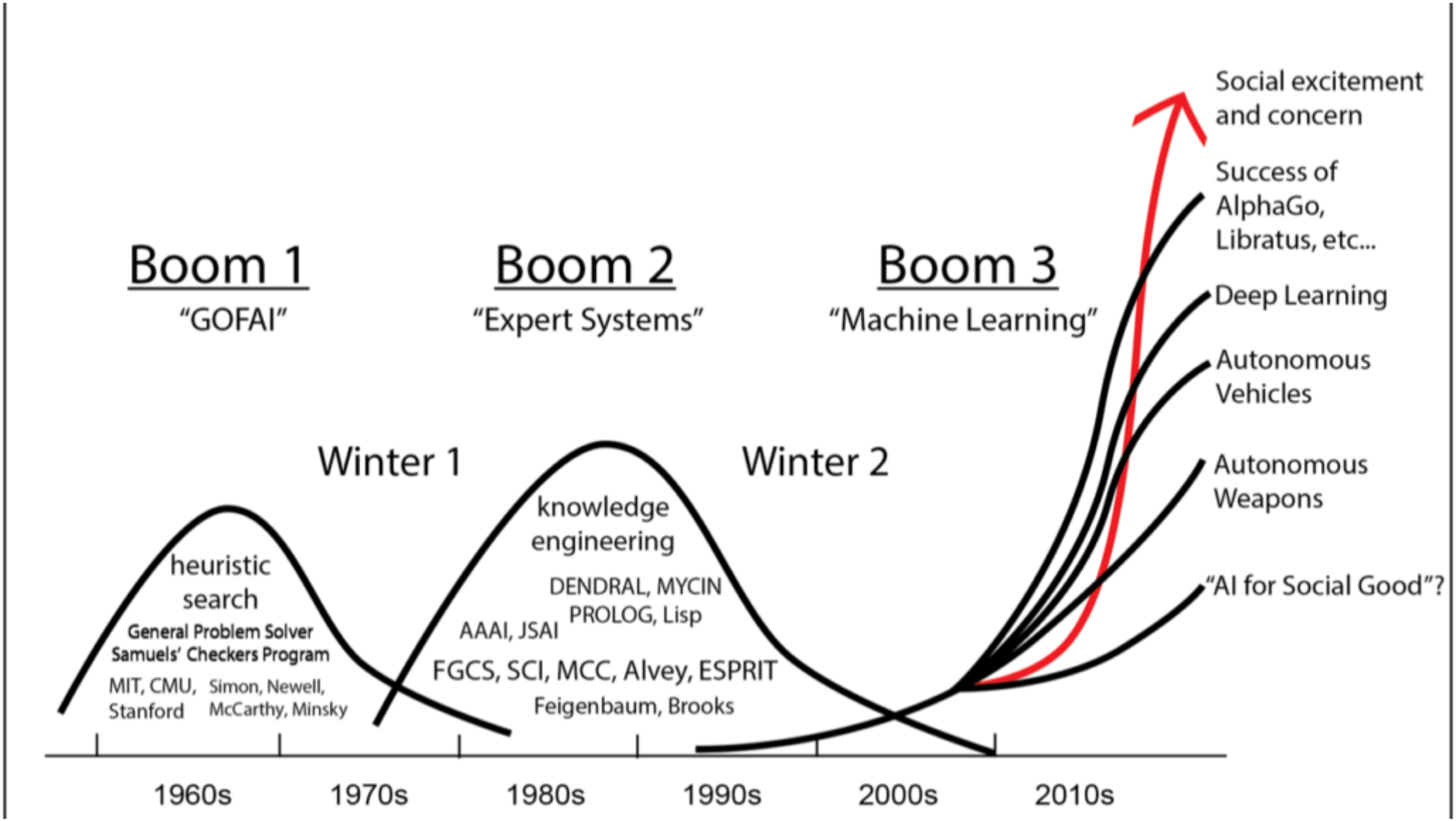
- These are all very important but are covered in other courses


Artificial Intelligence
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)
Machine Learning
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.


THE CARTOON FORM
TRADITIONAL PROGRAMMING
DATA
PROGRAM
COMPUTER
OUTPUT
MACHINE LEARNING
DATA
OUTPUT
COMPUTER
NEW DATA
PROGRAM
"MODEL"
NEW OUTPUT
TRAINING
INFERENCING

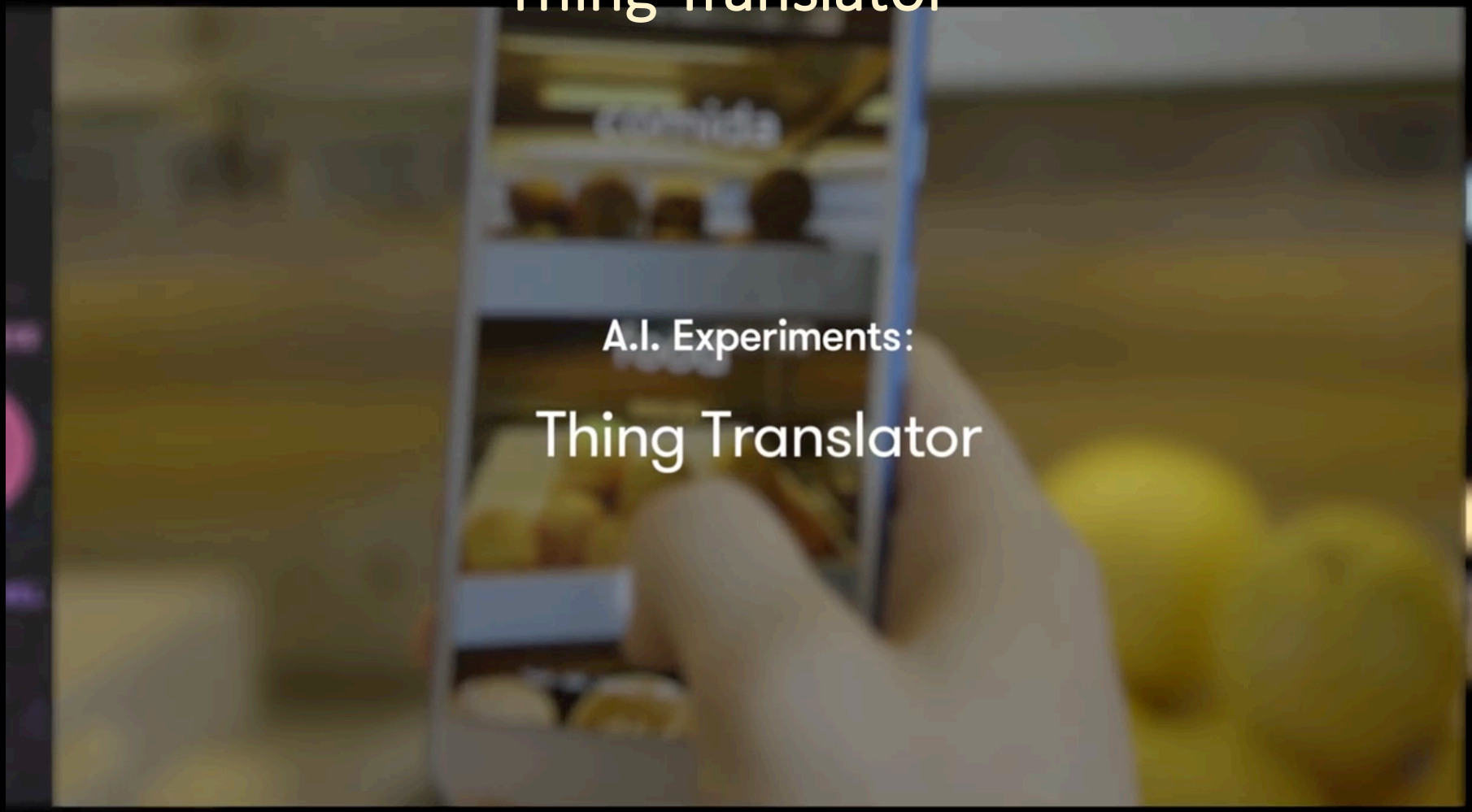

Artificial Intelligence (AI)
machine learning (ML)
deep learning
supervised
unsupervised
natural language processing (NLP)
content extraction
classification
machine translation
question answering
text generation
expert systems
vision
image recognition
machine vision
speech
speech to text
text to speech
planning
robotics

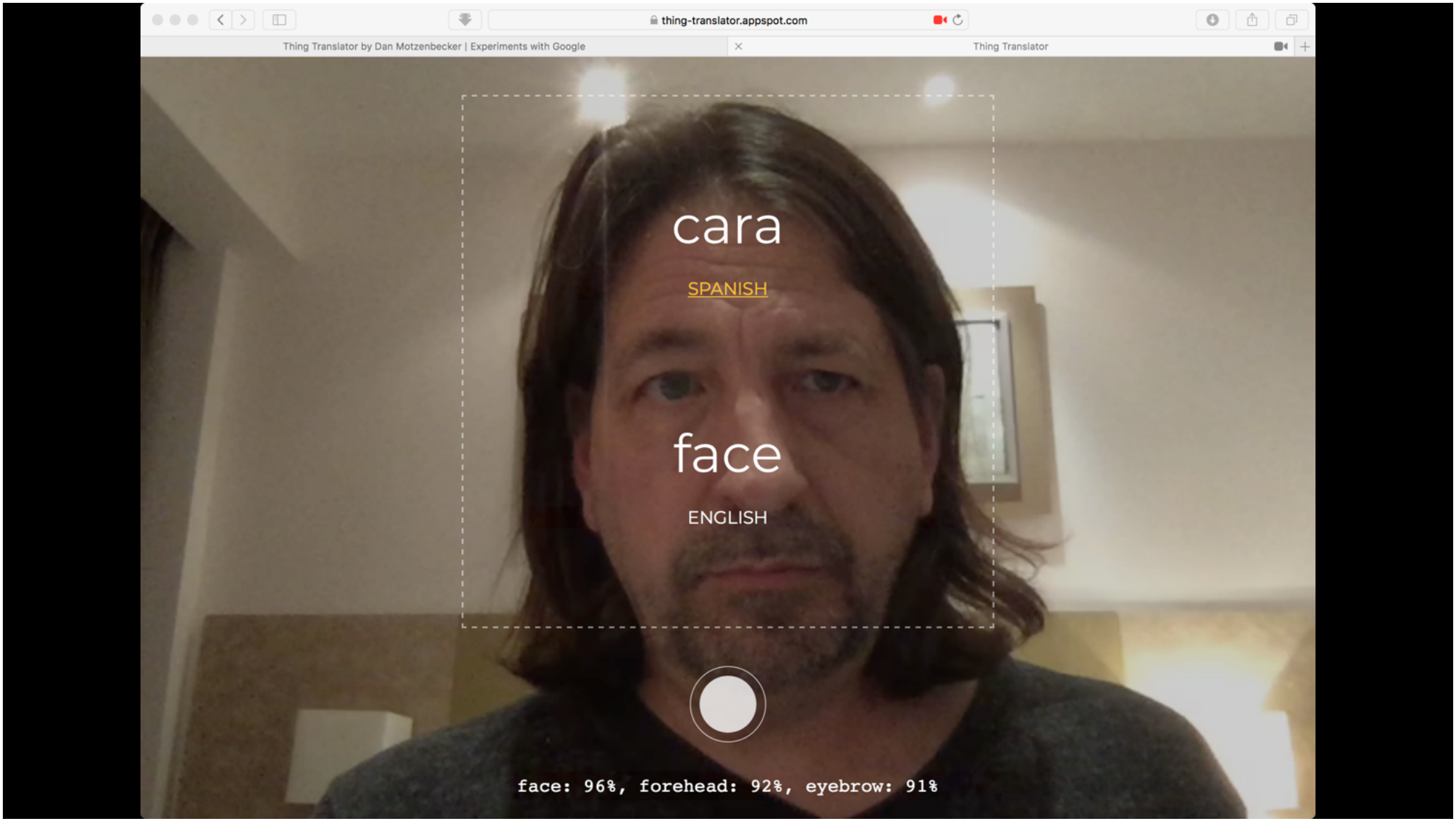

Machine Learning
machine learning (ML)
deep learning
supervised
unsupervised
natural language processing (NLP)
content extraction
classification
machine translation
question answering
text generation
expert systems
vision
image recognition
machine vision
speech
speech to text
text to speech
planning
robotics


Boom 1
"GOFAI"
Boom 2
"Expert Systems"
Boom 3
"Machine Learning"
Winter 1
Winter 2
heuristic search
General Problem Solver
Samuels' Checkers Program
MIT, CMU, Stanford
Simon, Newell, McCarthy, Minsky
knowledge engineering
DENDRAL, MYCIN
PROLOG, Lisp
AAAI, JSAI
FGCS, SCI, MCC, Alvey, ESPRIT
Feigenbaum, Brooks
Social excitement and concern
Success of AlphaGo, Libratus, etc...
Deep Learning
Autonomous Vehicles
Autonomous Weapons
"AI for Social Good"?
1960s
1970s
1980s
1990s
2000s
2010s

Spark

Thing Translator




thing-translator.appspot.com
Thing Translator by Dan Motzenbecker | Experiments with Google
Thing Translator
cara
SPANISH
face
ENGLISH
face: 96%, forehead: 92%, eyebrow: 91%

What Machine Learning Can Do

A simple way to think about supervised learning.

INPUT A	RESPONSE B	APPLICATION
Picture	Are there human faces? (0 or 1)	Photo tagging
Loan application	Will they repay the loan? (0 or 1)	Loan approvals
Ad plus user information	Will user click on ad? (0 or 1)	Targeted online ads
Audio clip	Transcript of audio clip	Speech recognition
English sentence	French sentence	Language translation
Sensors from hard disk, plane engine, etc.	Is it about to fail?	Preventive maintenance
Car camera and other sensors	Position of other cars	Self-driving cars

SOURCE ANDREW NG

Can you do it in 1 second of thought?
If so an AI can probably do it soon and do it better than humans.

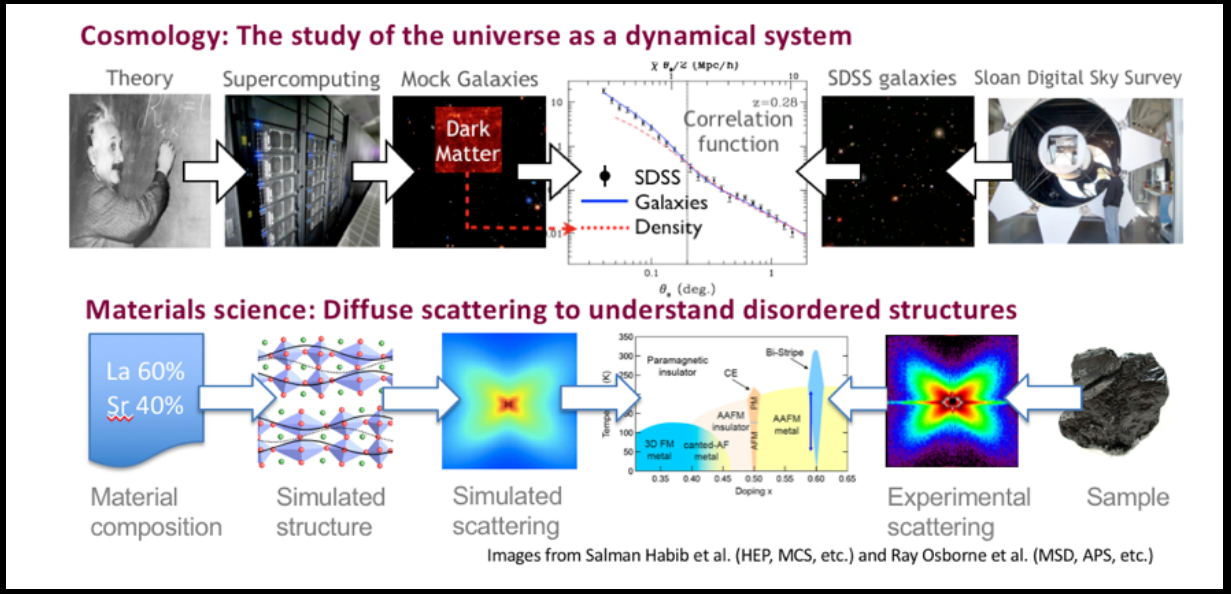
Current Accomplishments of Deep Learning

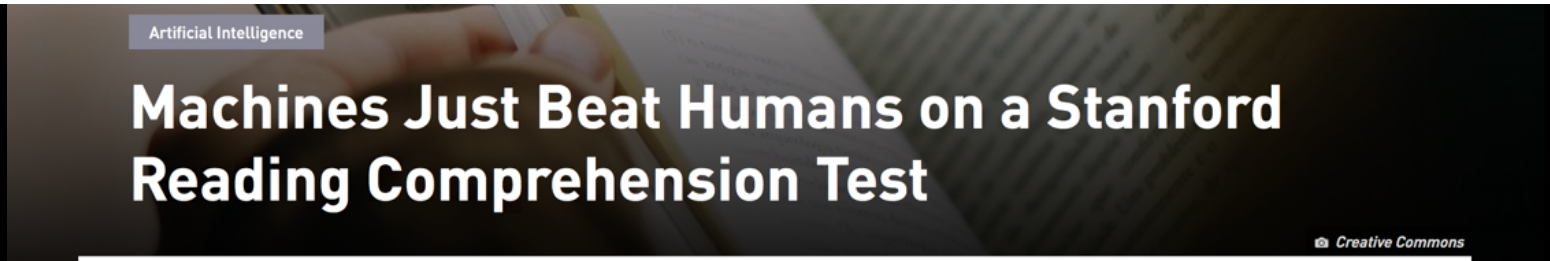
- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Improved machine translation
- Improved text-to-speech conversion
- Digital assistants such as Google Now and Amazon Alexa
- Near-human-level autonomous driving
- Improved ad targeting, as used by Google, Baidu, and Bing
- Improved search results on the web
- Ability to answer natural-language questions
- Superhuman Go playing

Machine Learning is becoming a major element of scientific computing applications

Across the DOE lab system hundreds of examples are emerging

- From fusion energy to precision medicine
- Materials design
- Cosmology
- High-Energy Physics
- Synthetic Biology
- Structural engineering
- Intelligent sensing
- Etc.





Artificial Intelligence

Machines Just Beat Humans on a Stanford Reading Comprehension Test

Creative Commons

IN BRIEF

The Stanford Question Answering Dataset is a well-respected means of testing machine reading. For the first time, an artificial intelligence has scored higher than a human participant.

READ ME

Chinese retail giant Alibaba has developed an artificial intelligence model that's managed to outdo human participants in a reading and comprehension test designed by Stanford University. The model scored 82.44, whereas humans recorded a score of 82.304.

The Stanford Question Answering Dataset is a set of 10,000 questions pertaining to some 500 Wikipedia articles. The answer to each question is a particular span of text from the corresponding piece of writing.

Alibaba claims that its accomplishment is the first time that humans have been outmatched on this particular test, according to a report from Bloomberg. Microsoft also managed a similar feat, scoring 82.650 — though, those results were finalized shortly after Alibaba's.

SHARE

WRITTEN BY

Brad Jones

Published: 2 hours ago

#Alibaba #machine reading #microsoft

Artificial Intelligence

Google's New AI Is Better at Creating AI Than the Company's Engineers

Shutterstock / Denis Linine

IN BRIEF

At its I/O '17 conference this week, Google shared details of its AutoML project, an artificial intelligence that can assist in the creation of other AIs. By automating some of the complicated process, AutoML could make machine learning more accessible to non-experts.

GOOGLE'S AUTOML

One of the more noteworthy remarks to come out of Google I/O '17 conference this week was CEO Sundar Pichai recalling how his team had joked that they have achieved "AI inception" with AutoML. Instead of crafting layers of dreams like in the Christopher Nolan flick, however, the AutoML system layers artificial intelligence (AI), with AI systems creating better AI systems.

SHARE

WRITTEN BY

AUTHOR
Tom Ward

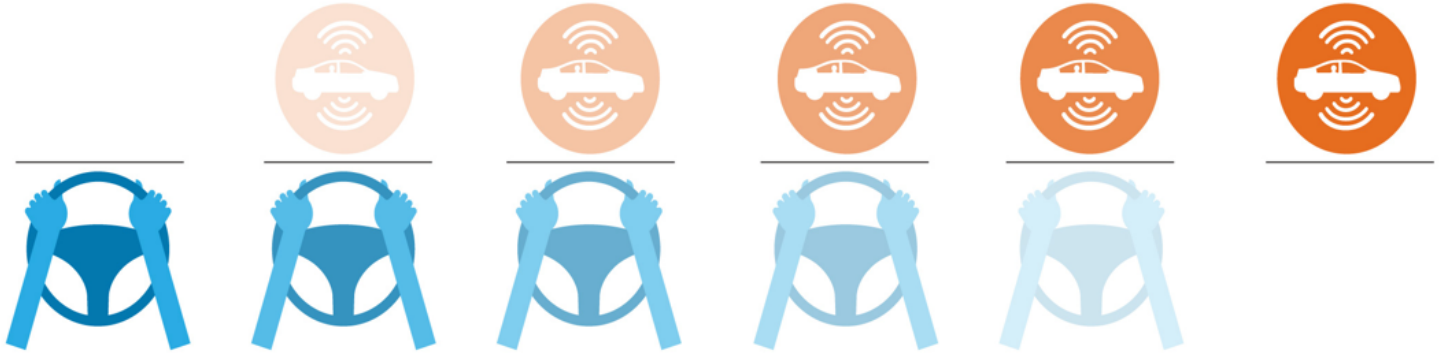
EDITOR
Kristin Houser
Website

Published: May 19, 2017

Last updated: May 19, 2017 at 12:29 pm

#artificial intelligence #deep learning
#Google #machine learning

Five Levels of Vehicle Autonomy



Level 0

No automation: the driver is in complete control of the vehicle at all times.

Level 1

Driver assistance: the vehicle can assist the driver or take control of either the vehicle's speed, through cruise control, or its lane position, through lane guidance.

Level 2

Occasional self-driving: the vehicle can take control of both the vehicle's speed and lane position in some situations, for example on limited-access freeways.

Level 3

Limited self-driving: the vehicle is in full control in some situations, monitors the road and traffic, and will inform the driver when he or she must take control.

Level 4

Full self-driving under certain conditions: the vehicle is in full control for the entire trip in these conditions, such as urban ride-sharing.

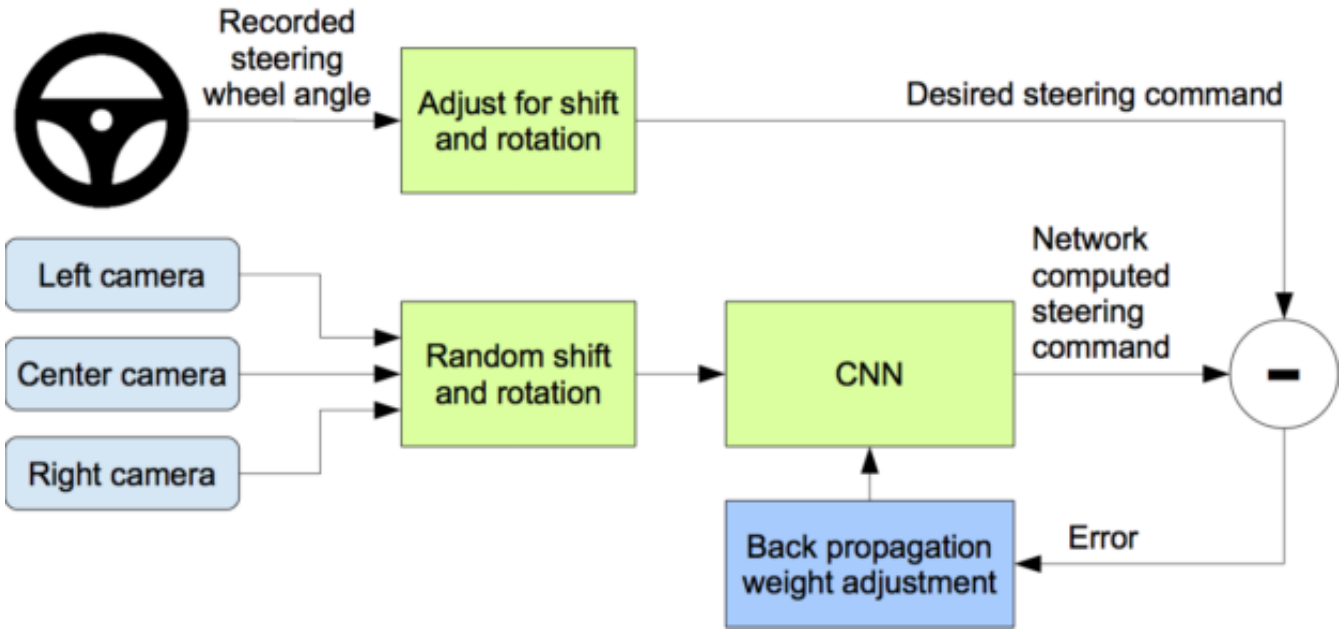
Level 5

Full self-driving under all conditions: the vehicle can operate without a human driver or occupants.

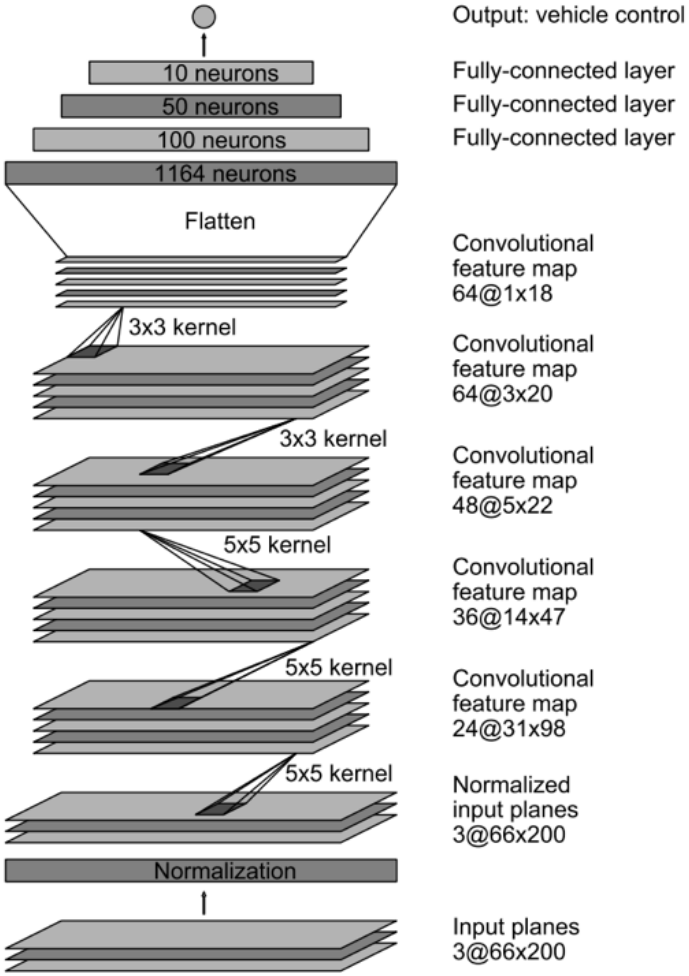
Source: SAE & NHTSA

THE PERSON IN THE DRIVER'S SEAT
IS ONLY THERE FOR LEGAL REASONS.

HE IS NOT DOING ANYTHING.
THE CAR IS DRIVING ITSELF.

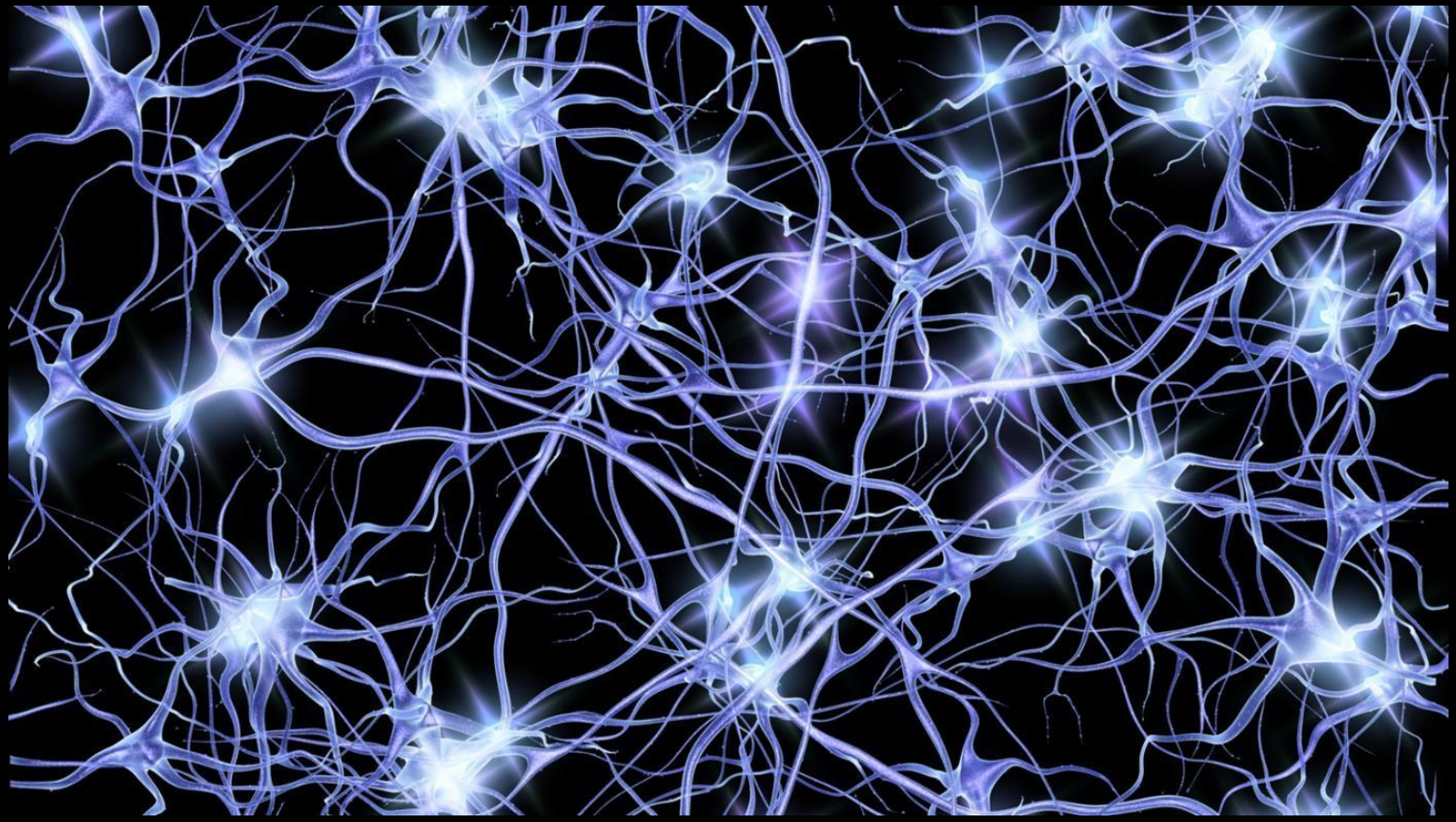


https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/

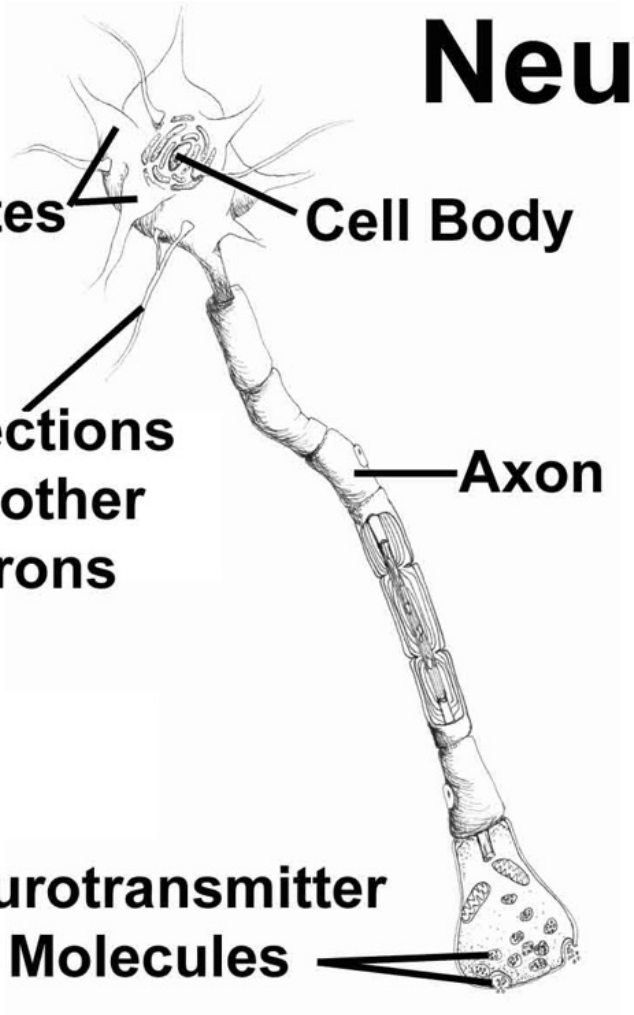


https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/

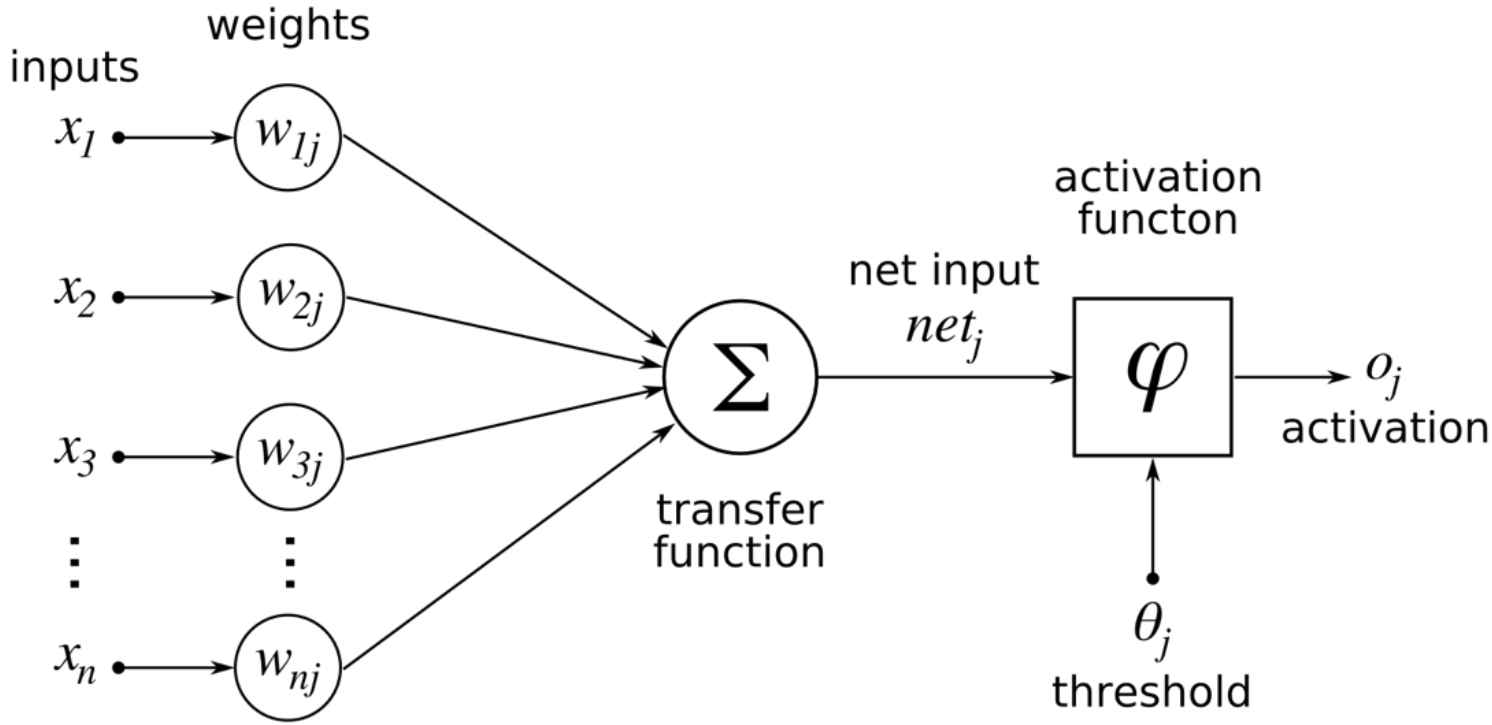

062M486



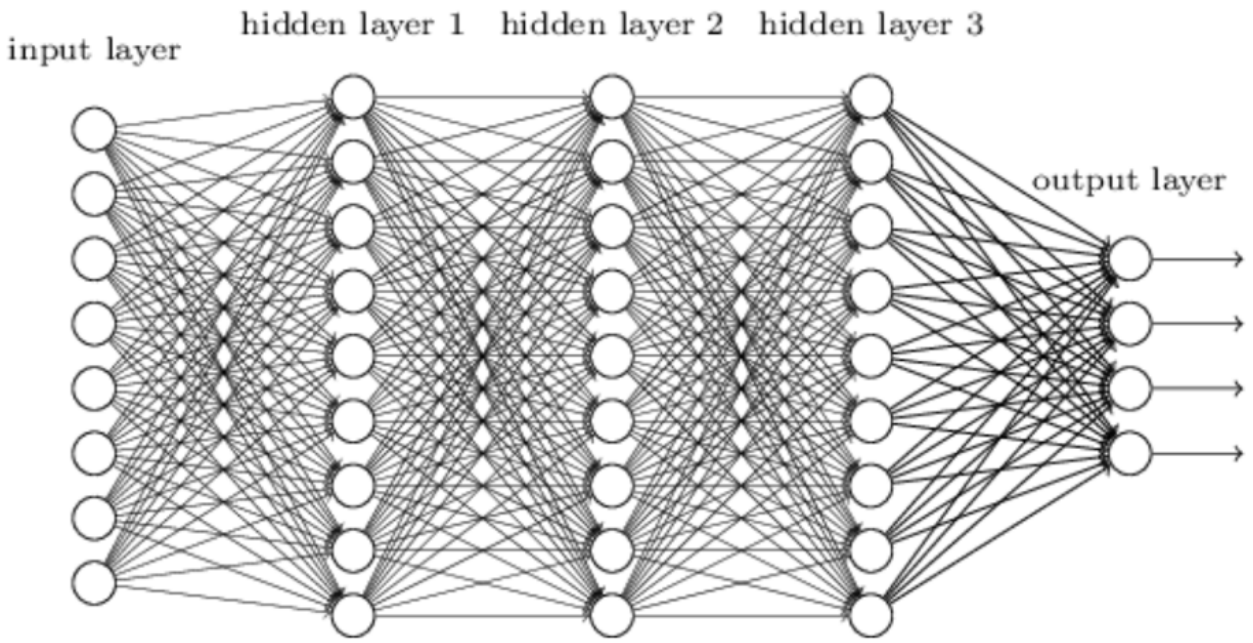
Neuron

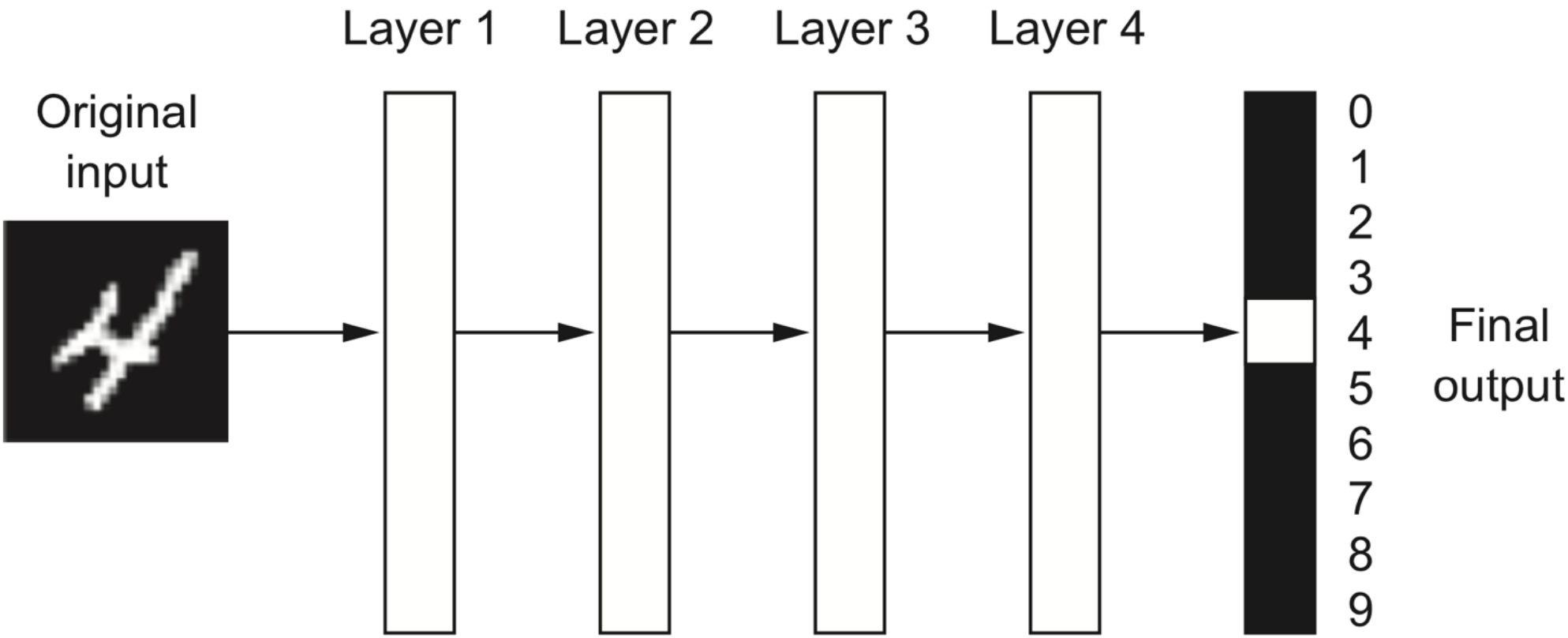

Dendrites
Cell Body
Connections from other Neurons
Axon
Neurotransmitter Molecules

Mathematical Model of a Neuron



Deep Neural Network




Layer 1
Layer 2
Layer 3
Layer 4
Original input
0
1
2
3
4
5
6
7
8
9
Final output

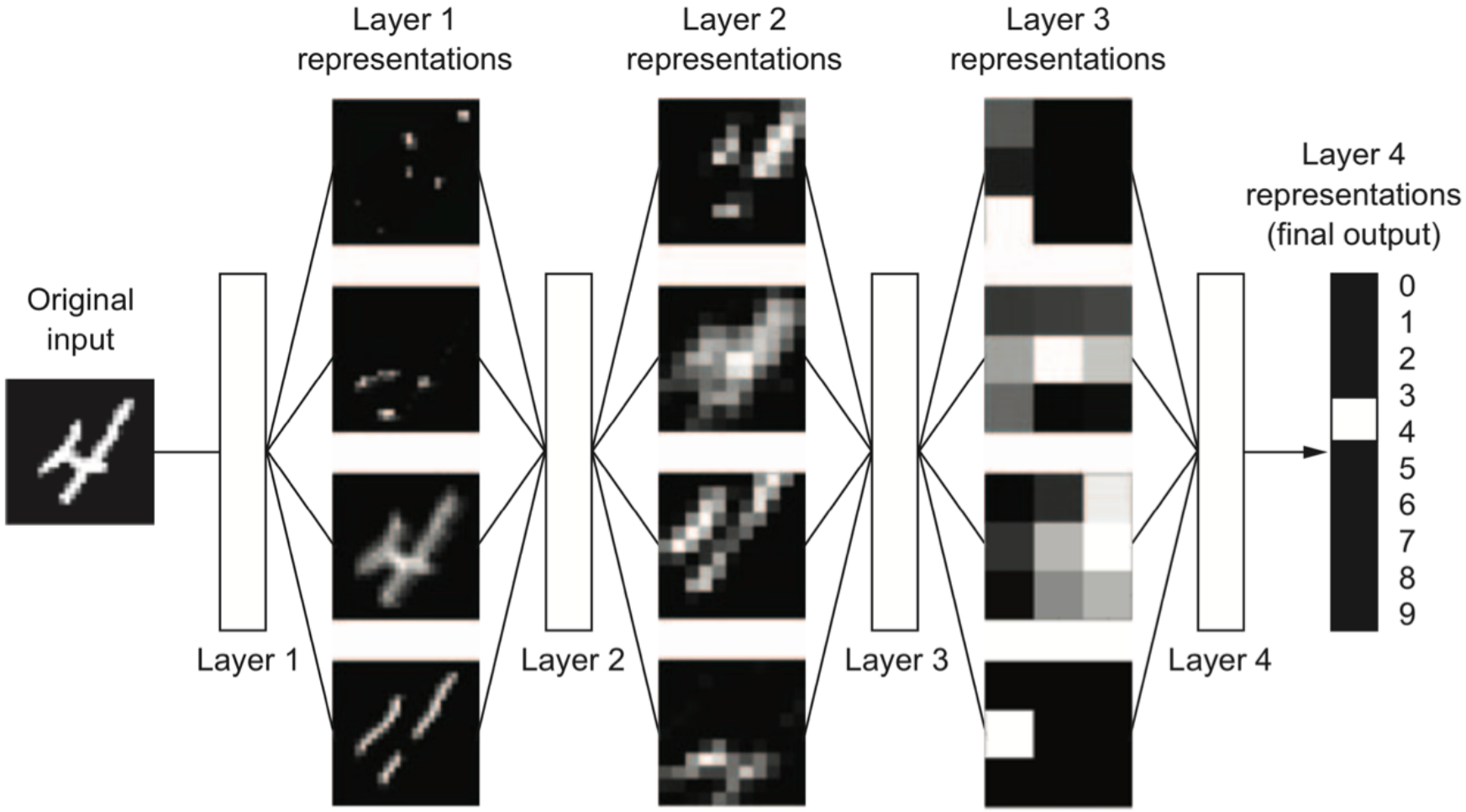


Figure 1.6 Deep representations learned by a digit-classification model

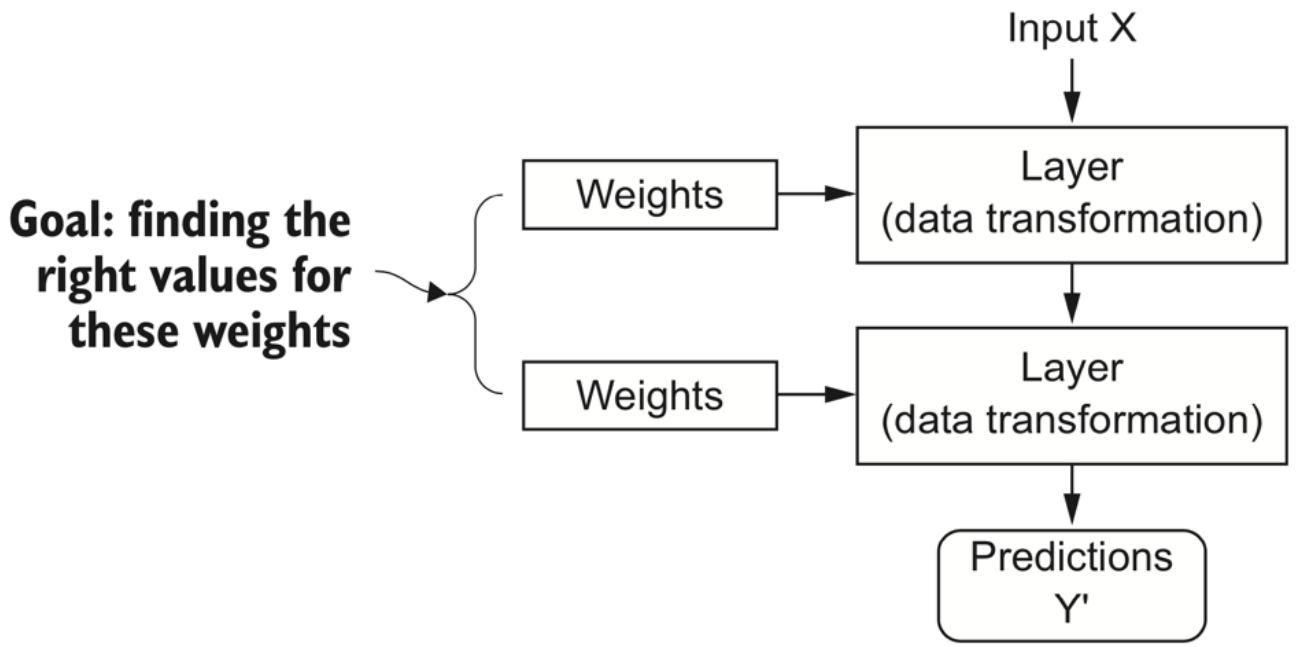


Figure 1.7 A neural network is parameterized by its weights.

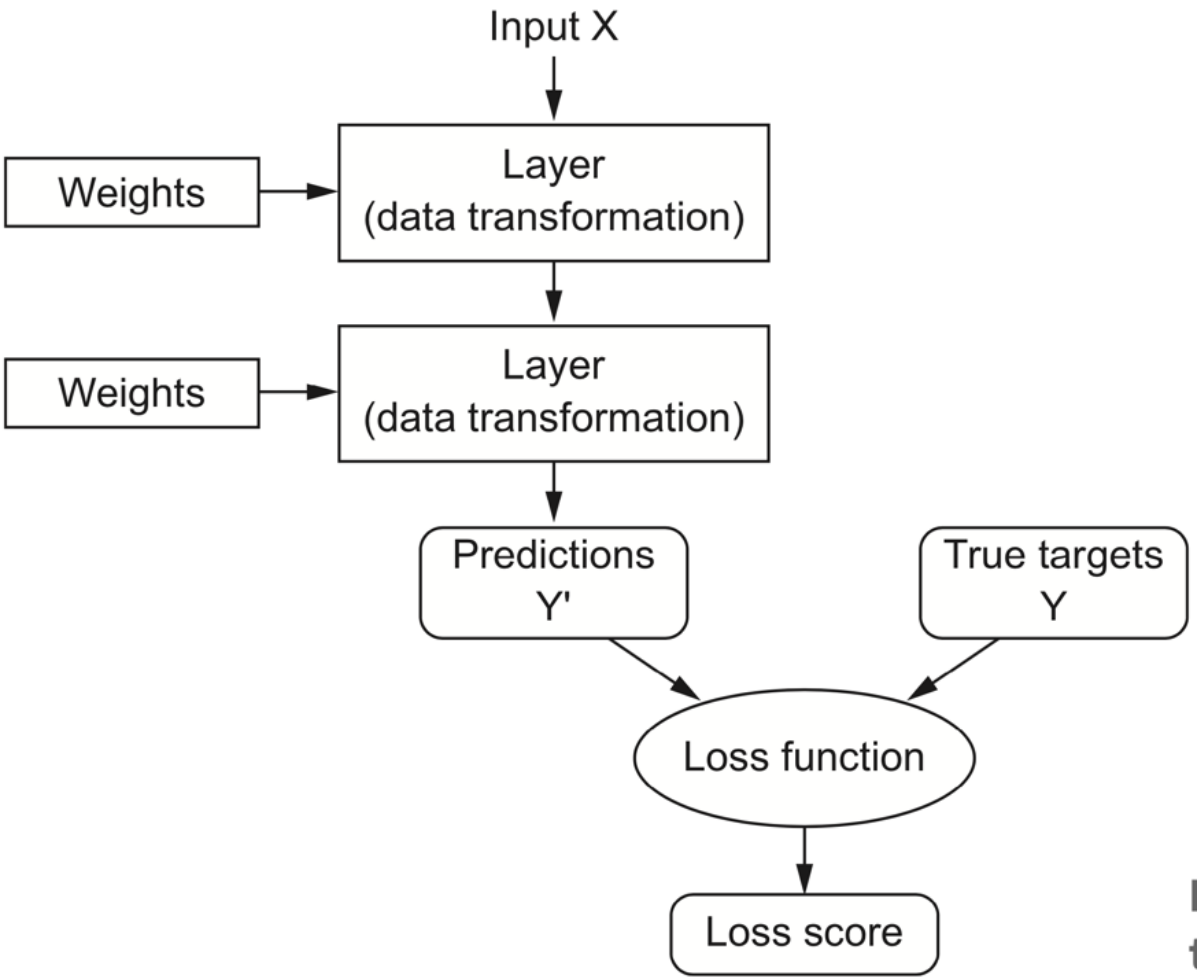


Figure 1.8 A loss function measures the quality of the network's output.

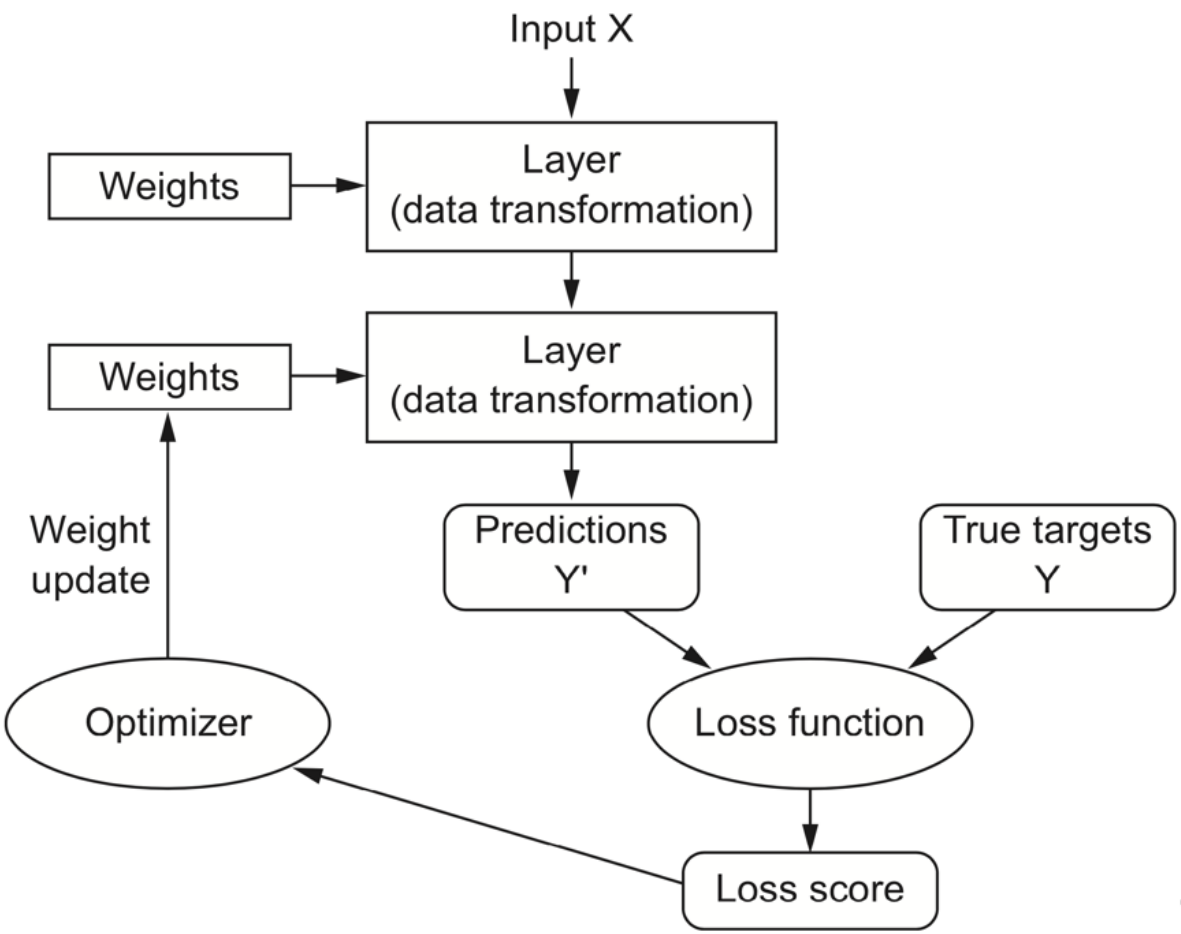
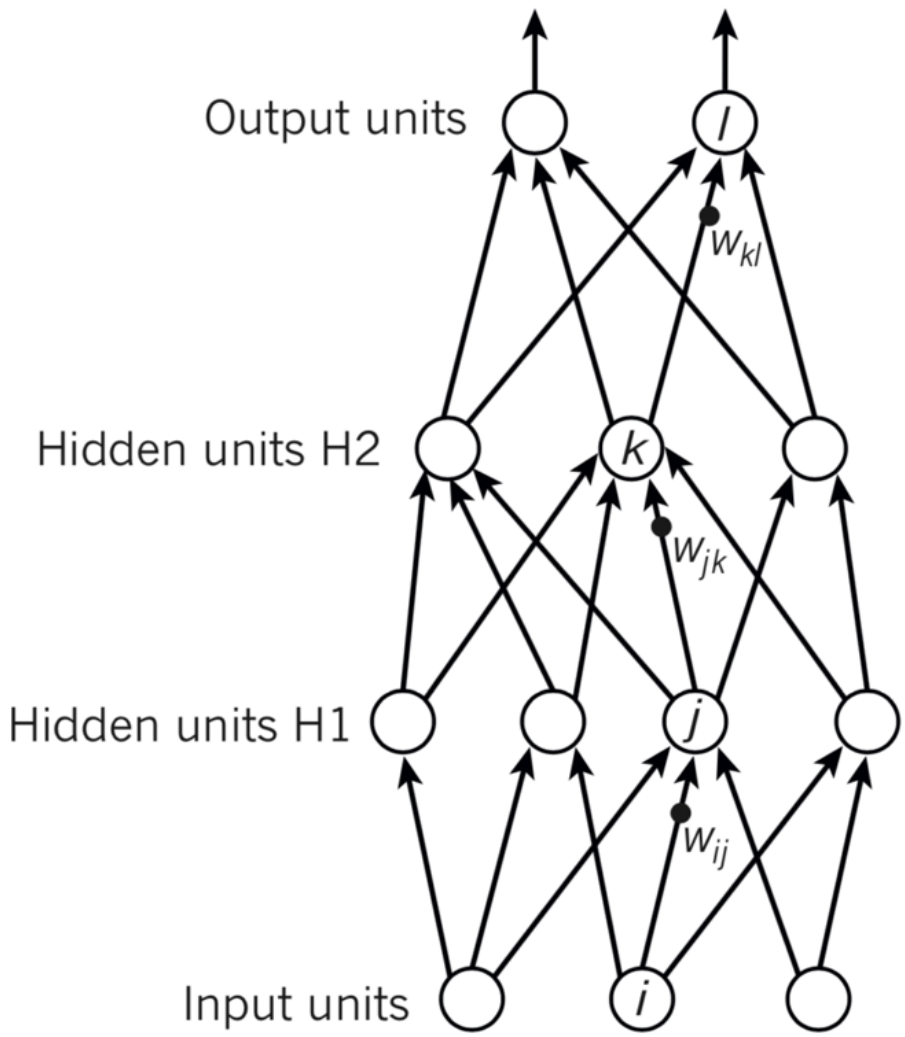


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

c



$$y_l = f(z_l)$$

$$z_l = \sum_{k \,\varepsilon\, \mathrm{H2}} w_{kl}\, y_k$$

$$y_k = f(z_k)$$

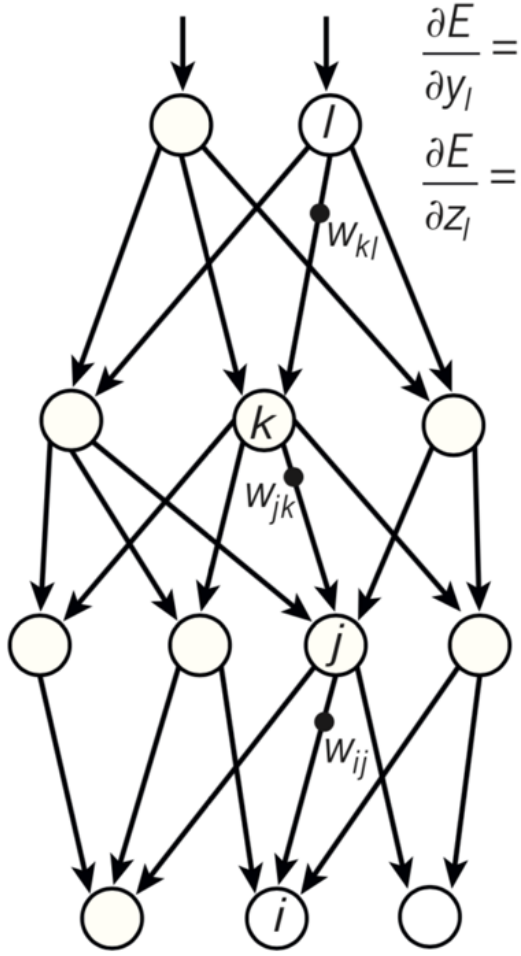
$$z_k = \sum_{j \,\varepsilon\, \mathrm{H1}} w_{jk}\, y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \,\varepsilon\, \mathrm{Input}} w_{ij}\, x_i$$

d

Compare outputs with correct answer to get error derivatives



$$\frac{\partial E}{\partial y_l} = y_l - t_l$$

$$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$$

$$\frac{\partial E}{\partial y_k} = \sum_{l \,\varepsilon\, \text{out}} w_{kl}\frac{\partial E}{\partial z_l}$$

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$$

$$\frac{\partial E}{\partial y_j} = \sum_{k \,\varepsilon\, \text{H2}} w_{jk}\frac{\partial E}{\partial z_k}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$$

jupyter Untitled3 Last Checkpoint: 4 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 2

Code CellToolbar

In [1]:
```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
Using TensorFlow backend.
```

In [2]:
```
from keras import models
from keras import layers
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```

In [3]:
```
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

In [4]:
```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
```

In [5]:
```
from keras.utils import to_categorical
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

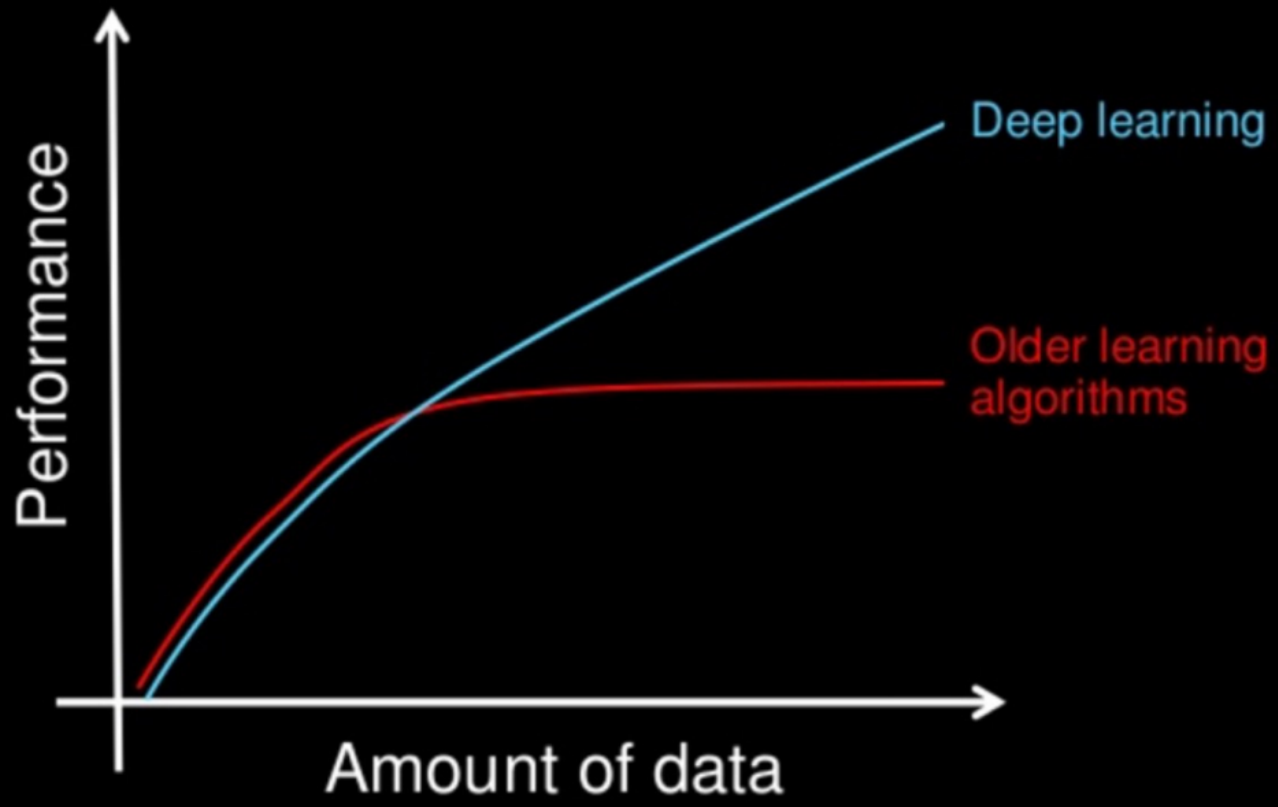
In [6]:
```
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 47us/step - loss: 0.2585 - acc: 0.9256
Epoch 2/5
60000/60000 [==============================] - 2s 36us/step - loss: 0.1035 - acc: 0.9697
Epoch 3/5
60000/60000 [==============================] - 2s 39us/step - loss: 0.0682 - acc: 0.9798
Epoch 4/5
60000/60000 [==============================] - 2s 36us/step - loss: 0.0487 - acc: 0.9856
Epoch 5/5
60000/60000 [==============================] - 2s 36us/step - loss: 0.0382 - acc: 0.9888
```

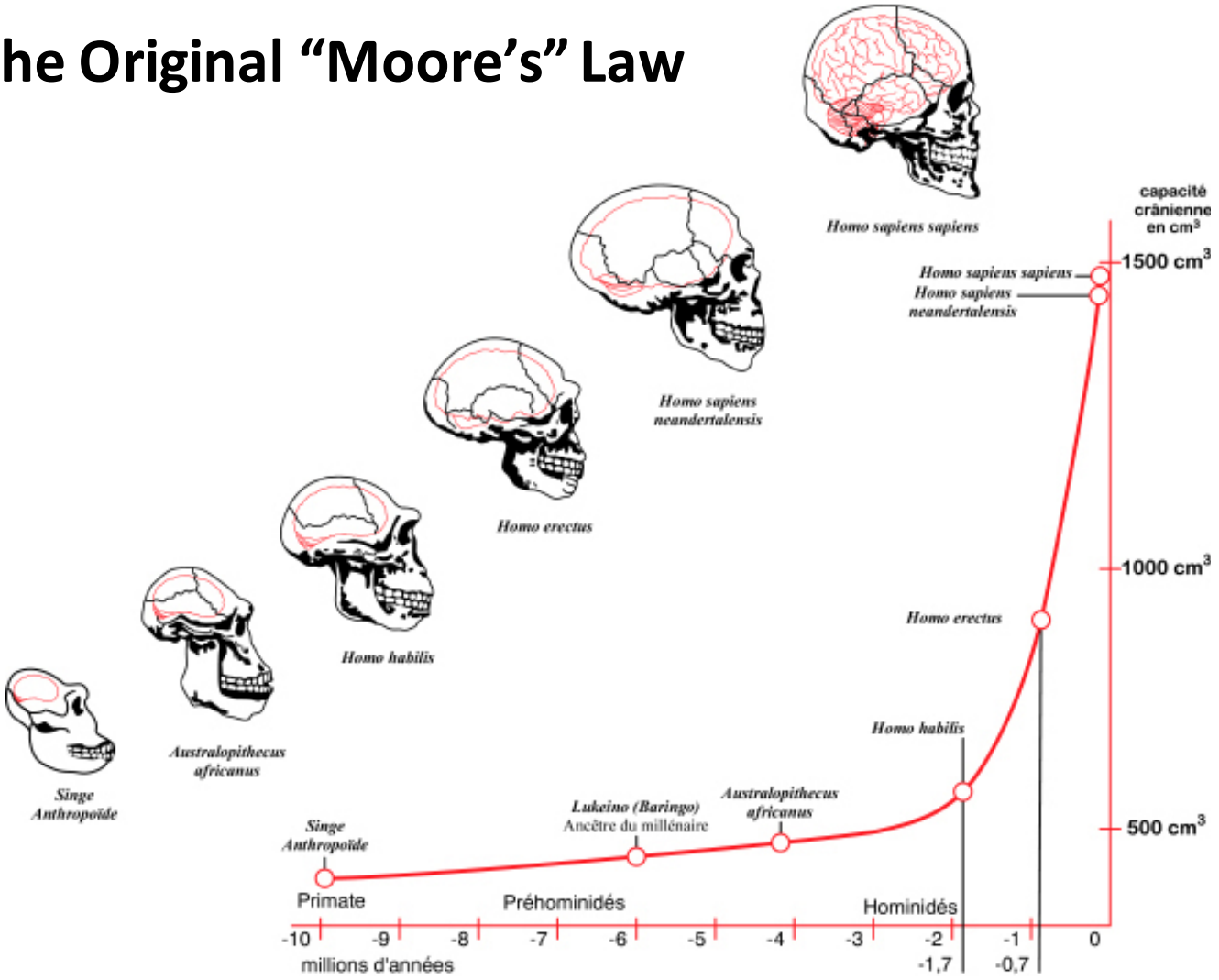
Out[6]:
```
<keras.callbacks.History at 0x12b194e10>
```

In [ ]:

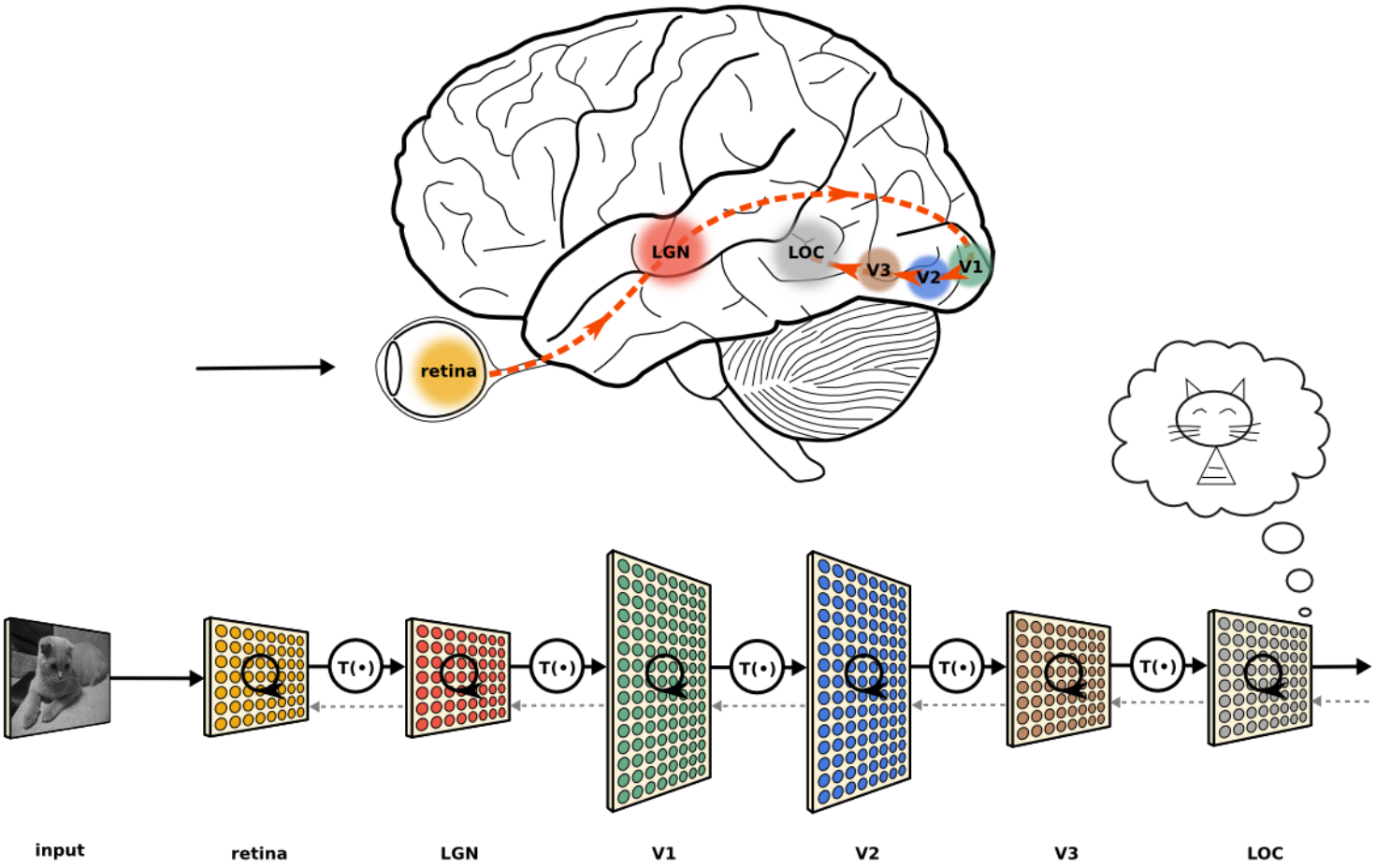
Why deep learning



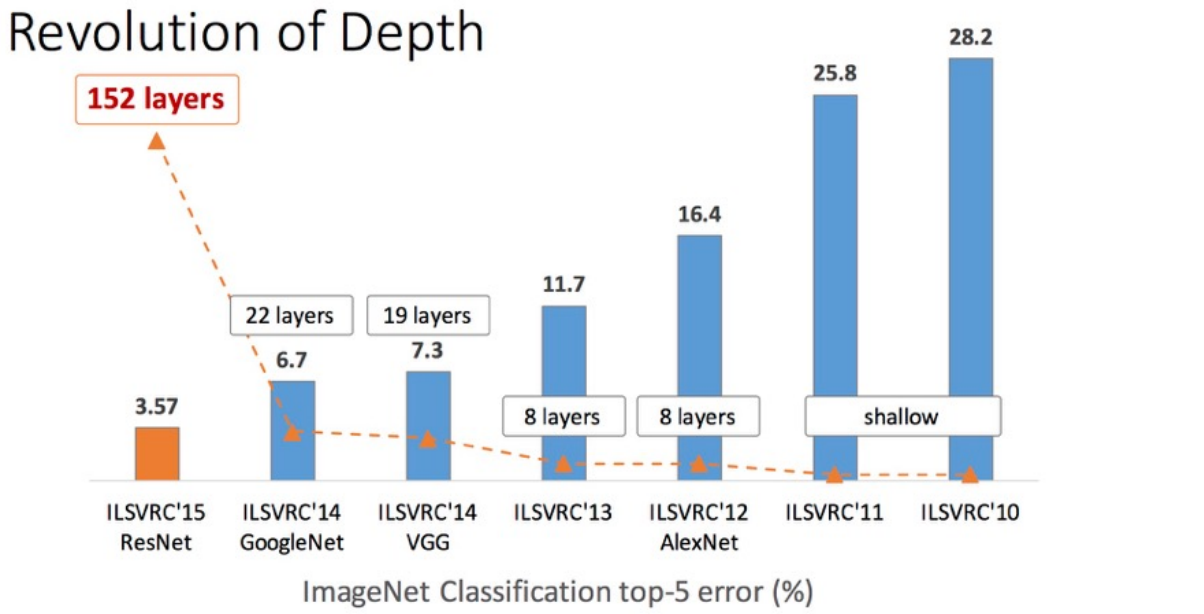
The Original “Moore’s” Law



Human Vision System



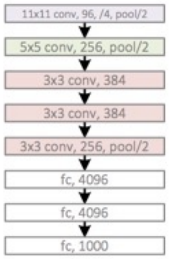
Increasing Depth Works!



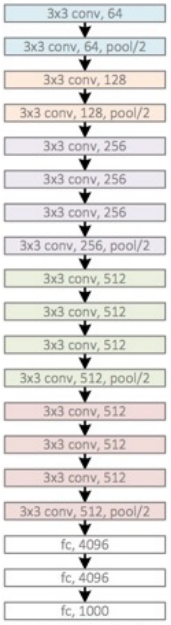
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Example CNNs Structures from ILSVRC Winners

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

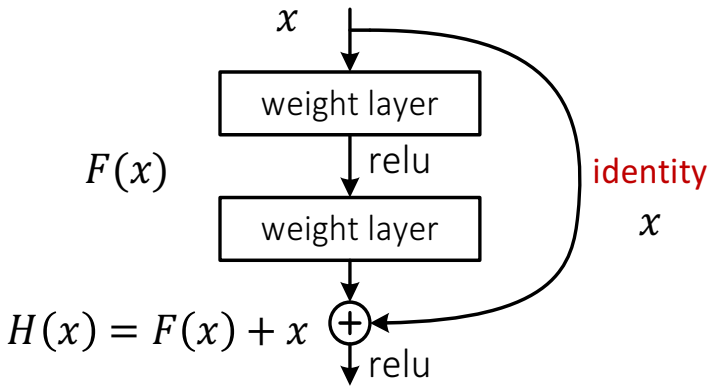


GoogleNet, 22 layers
(ILSVRC 2014)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity



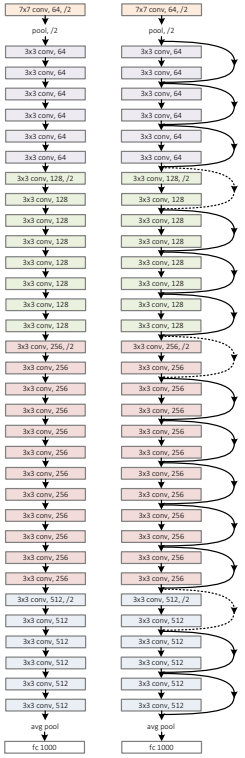
- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Deep Residual Learning

Network “Design”

- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design; just deep!
- Other remarks:
  - no hidden fc
  - no dropout

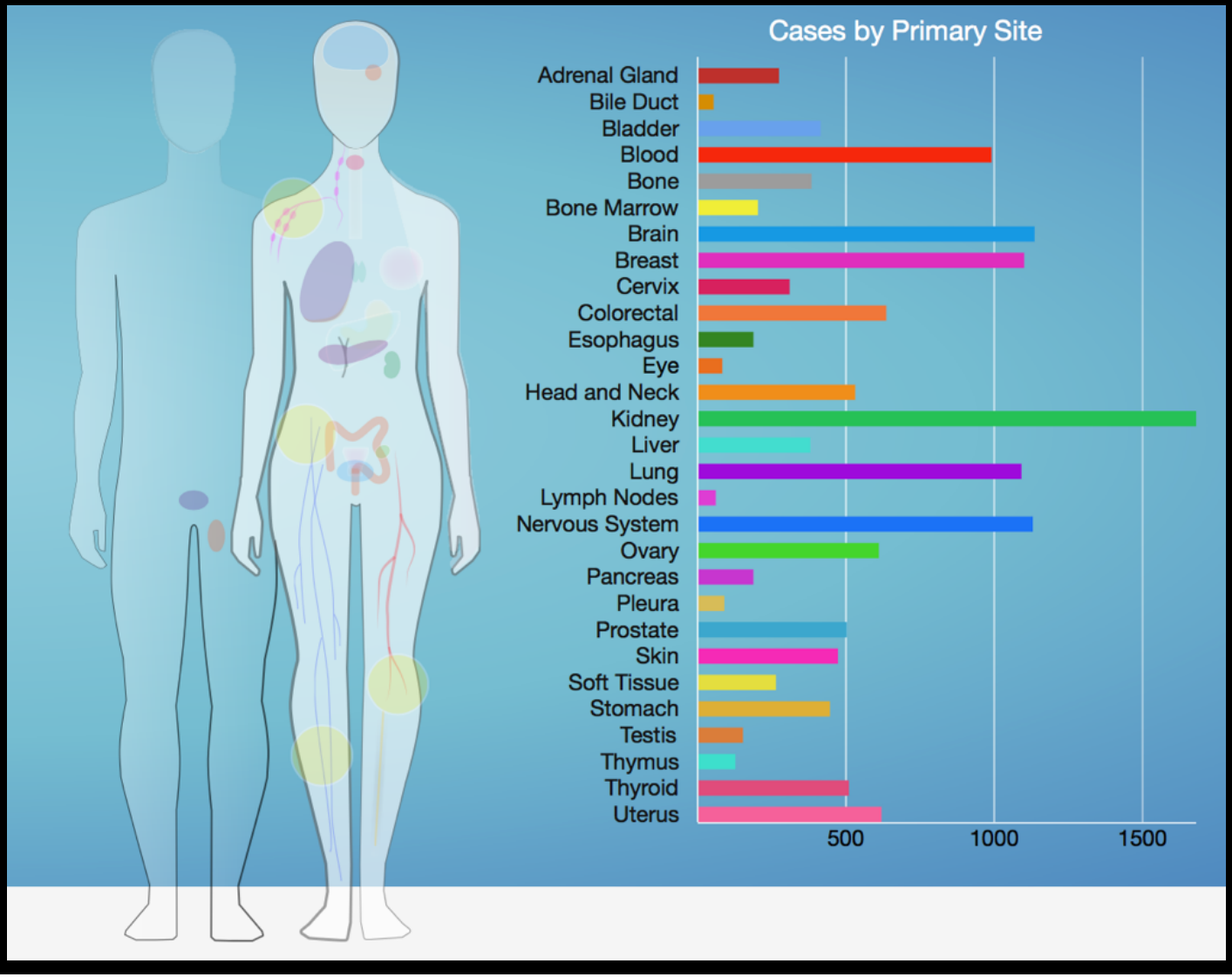
plain net

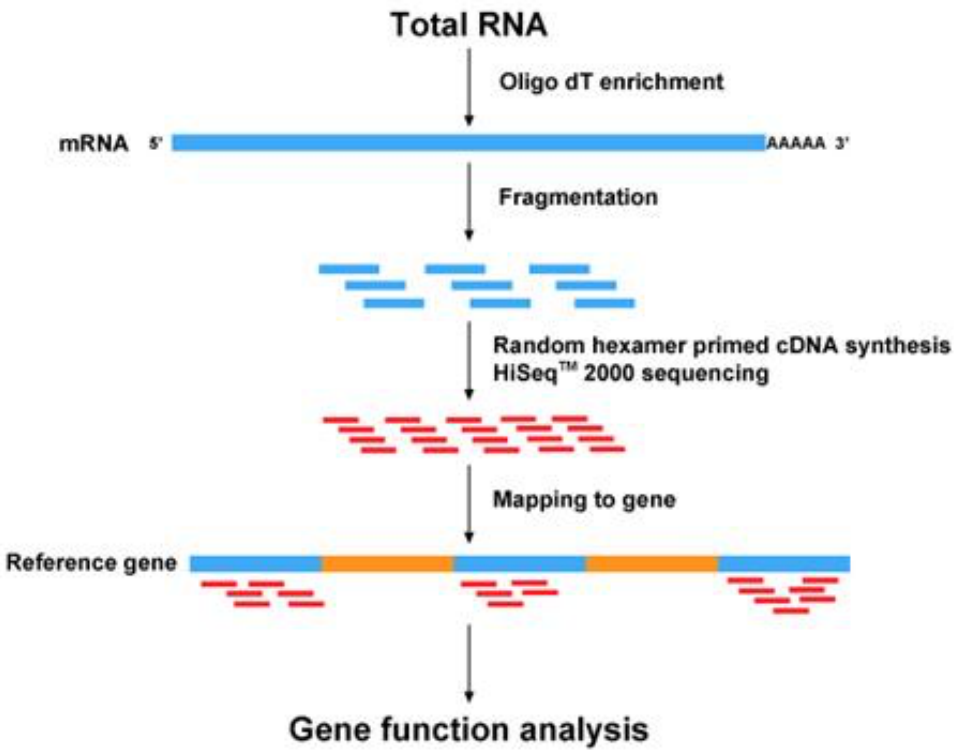


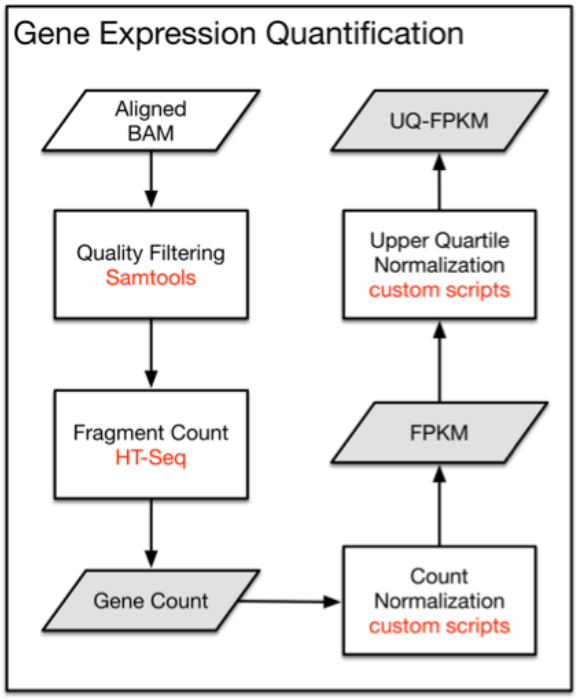
ResNet

Music Composition

Example from Cancer: Type Classification


Cases by Primary Site
Adrenal Gland
Bile Duct
Bladder
Blood
Bone
Bone Marrow
Brain
Breast
Cervix
Colorectal
Esophagus
Eye
Head and Neck
Kidney
Liver
Lung
Lymph Nodes
Nervous System
Ovary
Pancreas
Pleura
Prostate
Skin
Soft Tissue
Stomach
Testis
Thymus
Thyroid
Uterus
500
1000
1500


Total RNA
Oligo dT enrichment
mRNA 5'
AAAAA 3'
Fragmentation
Random hexamer primed cDNA synthesis
HiSeq™ 2000 sequencing
Mapping to gene
Reference gene
Gene function analysis



RPKM (reads per kilobase per million mapped reads)
Upper Quantile (UQ)

FPKM

The Fragments per Kilobase of transcript per Million mapped reads (FPKM) calculation normalizes read count by dividing it by the gene length and the total number of reads mapped to protein-coding genes.

Upper Quartile FPKM

The upper quartile FPKM (FPKM-UQ) is a modified FPKM calculation in which the total protein-coding read count is replaced by the 75th percentile read count value for the sample.

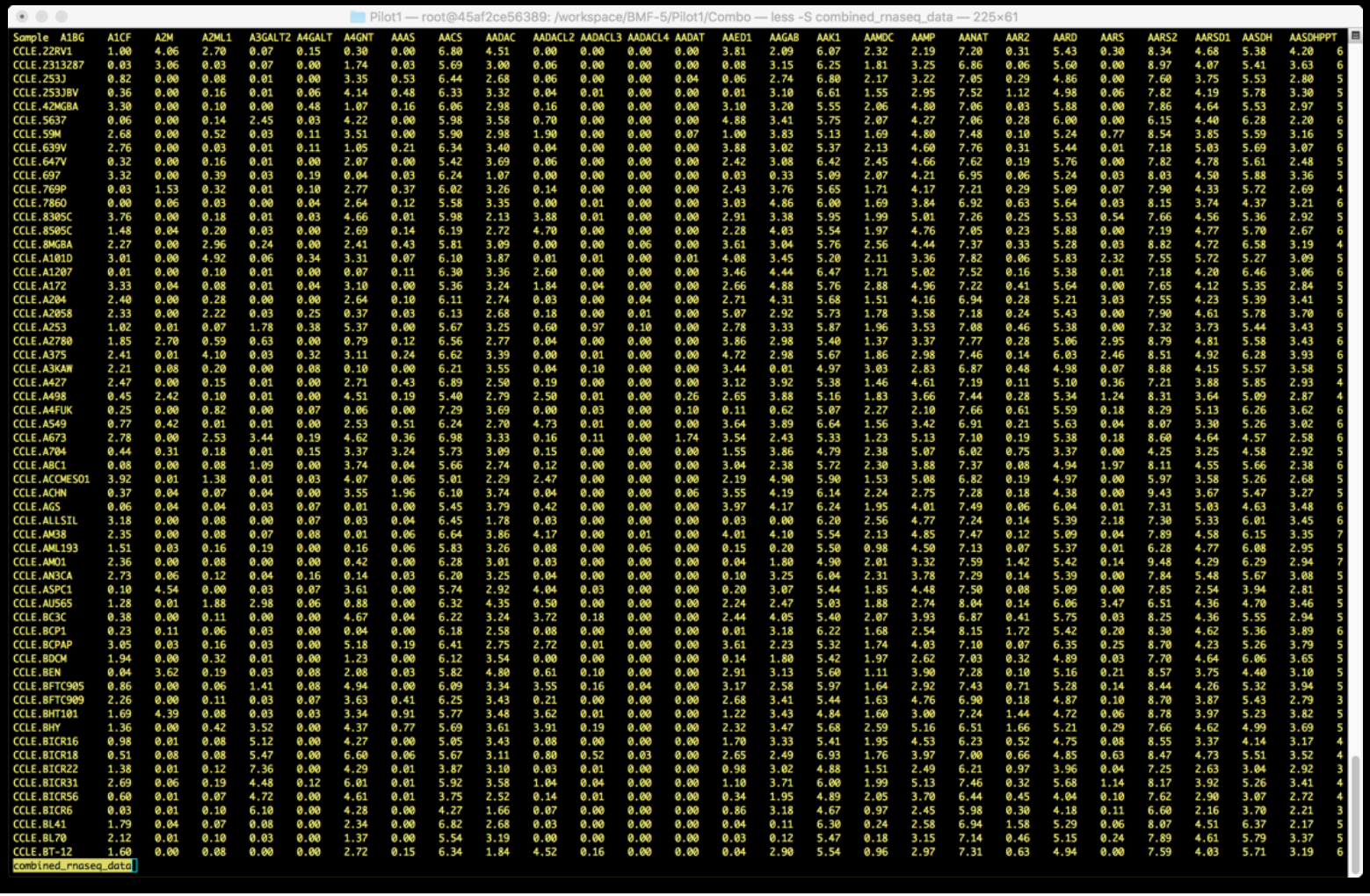
Calculations

$$FPKM = \frac{RC_g * 10^9}{RC_{pc} * L} \qquad FPKM-UQ = \frac{RC_g * 10^9}{RC_{g75} * L}$$

- RC_g: Number of reads mapped to the gene
- RC_{pc}: Number of reads mapped to all protein-coding genes
- RC_{g75}: The 75th percentile read count value for genes in the sample
- **L:** Length of the gene in base pairs

Note: The read count is multiplied by a scalar (10^9) during normalization to account for the kilobase and 'million mapped reads' units.

Each Sample has ~ 60,000 columns



Sample	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	AACS	AADAC	AADACL2	AADACL3	AADACL4	AADAT	AAED1	AAGAB	AAK1	AAMDC	AAMP	AANAT	AAR2	AARD	AARS	AARS2	AARSD1	AASDH	AASDHPPT	
CCLE.22RV1	1.00	4.06	2.70	0.07	0.15	0.30	0.00	6.80	4.51	0.00	0.00	0.00	0.00	3.81	2.09	6.07	2.32	2.19	7.20	0.31	5.43	0.30	8.34	4.68	5.38	4.20	6
CCLE.2313287	0.03	3.06	0.03	0.07	0.00	1.74	0.03	5.69	3.00	0.06	0.00	0.00	0.00	0.08	3.15	6.25	1.81	3.25	6.86	0.06	5.60	0.00	8.97	4.07	5.41	3.63	6
CCLE.253J	0.82	0.00	0.08	0.01	0.00	3.35	0.53	6.44	2.68	0.06	0.00	0.00	0.04	0.06	2.74	6.80	2.17	3.22	7.05	0.29	4.86	0.00	7.60	3.75	5.53	2.80	5
CCLE.253JBV	0.36	0.00	0.16	0.01	0.06	4.14	0.48	6.33	3.32	0.04	0.01	0.00	0.00	0.01	3.10	6.61	1.55	2.95	7.52	1.12	4.98	0.06	7.82	4.19	5.78	3.30	5
CCLE.42MGBA	3.30	0.00	0.10	0.00	0.48	1.07	0.16	6.06	2.98	0.16	0.00	0.00	0.00	3.10	3.20	5.55	2.06	4.80	7.06	0.03	5.88	0.00	7.86	4.64	5.53	2.97	5
CCLE.5637	0.06	0.00	0.14	2.45	0.03	4.22	0.00	5.98	3.58	0.70	0.00	0.00	0.00	4.88	3.41	5.75	2.07	4.27	7.06	0.28	6.00	0.00	6.15	4.40	6.28	2.20	6
CCLE.59M	2.68	0.00	0.52	0.03	0.11	3.51	0.00	5.90	2.98	1.90	0.00	0.00	0.07	1.00	3.83	5.13	1.69	4.80	7.48	0.10	5.24	0.77	8.54	3.85	5.59	3.16	5
CCLE.639V	2.76	0.00	0.03	0.01	0.11	1.05	0.21	6.34	3.40	0.04	0.00	0.00	0.00	3.88	3.02	5.37	2.13	4.60	7.76	0.31	5.44	0.01	7.18	5.03	5.69	3.07	6
CCLE.647V	0.32	0.00	0.16	0.01	0.00	2.07	0.00	5.42	3.69	0.06	0.00	0.00	0.00	2.42	3.08	6.42	2.45	4.66	7.62	0.19	5.76	0.00	7.82	4.78	5.61	2.48	5
CCLE.697	3.32	0.00	0.39	0.03	0.19	0.04	0.03	6.24	1.07	0.00	0.00	0.00	0.00	0.03	0.33	5.09	2.07	4.21	6.95	0.06	5.24	0.03	8.03	4.50	5.88	3.36	5
CCLE.769P	0.03	1.53	0.32	0.01	0.10	2.77	0.37	6.02	3.26	0.14	0.00	0.00	0.00	2.43	3.76	5.65	1.71	4.17	7.21	0.29	5.09	0.07	7.90	4.33	5.72	2.69	4
CCLE.786O	0.00	0.06	0.03	0.00	0.04	2.64	0.12	5.58	3.35	0.00	0.01	0.00	0.00	3.03	4.86	6.00	1.69	3.84	6.92	0.63	5.64	0.03	8.15	3.74	4.37	3.21	6
CCLE.8305C	3.76	0.00	0.18	0.01	0.03	4.66	0.01	5.98	2.13	3.88	0.01	0.00	0.00	2.91	3.38	5.95	1.99	5.01	7.26	0.25	5.53	0.54	7.66	4.56	5.36	2.92	5
CCLE.8505C	1.48	0.04	0.20	0.03	0.00	2.69	0.14	6.19	2.72	4.70	0.00	0.00	0.00	2.28	4.03	5.54	1.97	4.76	7.05	0.23	5.88	0.00	7.19	4.77	5.70	2.67	6
CCLE.8MGBA	2.27	0.00	2.96	0.24	0.00	2.41	0.43	5.81	3.09	0.00	0.00	0.06	0.00	3.61	3.04	5.76	2.56	4.44	7.37	0.33	5.28	0.03	8.82	4.72	6.58	3.19	4
CCLE.A101D	3.01	0.00	4.92	0.06	0.34	3.31	0.07	6.10	3.87	0.01	0.01	0.00	0.01	4.08	3.45	5.20	2.11	3.36	7.82	0.06	5.83	2.32	7.55	5.72	5.27	3.09	5
CCLE.A1207	0.01	0.00	0.10	0.01	0.00	0.07	0.11	6.30	3.36	2.60	0.00	0.00	0.00	3.46	4.44	6.47	1.71	5.02	7.52	0.16	5.38	0.01	7.18	4.20	6.46	3.06	6
CCLE.A172	3.33	0.04	0.08	0.01	0.04	3.10	0.00	5.36	3.24	1.84	0.04	0.00	0.00	2.66	4.88	5.76	2.88	4.96	7.22	0.41	5.64	0.00	7.65	4.12	5.35	2.84	5
CCLE.A204	2.40	0.00	0.28	0.00	0.00	2.64	0.10	6.11	2.74	0.03	0.00	0.04	0.00	2.71	4.31	5.68	1.51	4.16	6.94	0.28	5.21	3.03	7.55	4.23	5.39	3.41	5
CCLE.A2058	2.33	0.00	2.22	0.03	0.25	0.37	0.03	6.13	2.68	0.18	0.00	0.01	0.00	5.07	2.92	5.73	1.78	3.58	7.18	0.24	5.43	0.00	7.90	4.61	5.78	3.70	6
CCLE.A253	1.02	0.01	0.07	1.78	0.38	5.37	0.00	5.67	3.25	0.60	0.97	0.10	0.00	2.78	3.33	5.87	1.96	3.53	7.08	0.46	5.38	0.00	7.32	3.73	5.44	3.43	5
CCLE.A2780	1.85	2.70	0.59	0.63	0.00	0.79	0.12	6.56	2.77	0.04	0.00	0.00	0.00	3.86	2.98	5.40	1.37	3.37	7.77	0.28	5.06	2.95	8.79	4.81	5.58	3.43	6
CCLE.A375	2.41	0.01	4.10	0.03	0.32	3.11	0.24	6.62	3.39	0.00	0.01	0.00	0.00	4.72	2.98	5.67	1.86	2.98	7.46	0.14	6.03	2.46	8.51	4.92	6.28	3.93	6
CCLE.A3KAW	2.21	0.08	0.20	0.00	0.08	0.10	0.00	6.21	3.55	0.04	0.10	0.00	0.00	3.44	0.01	4.97	3.03	2.83	6.87	0.48	4.98	0.07	8.88	4.15	5.57	3.58	5
CCLE.A427	2.47	0.00	0.15	0.01	0.00	2.71	0.43	6.89	2.50	0.19	0.00	0.00	0.00	3.12	3.92	5.38	1.46	4.61	7.19	0.11	5.10	0.36	7.21	3.88	5.85	2.93	4
CCLE.A498	0.45	2.42	0.10	0.01	0.00	4.51	0.19	5.40	2.79	2.50	0.01	0.00	0.26	2.65	3.88	5.16	1.83	3.66	7.44	0.28	5.34	1.24	8.31	3.64	5.09	2.87	4
CCLE.A4FUK	0.25	0.00	0.82	0.00	0.07	0.06	0.00	7.29	3.69	0.00	0.03	0.00	0.10	0.11	0.62	5.07	2.27	2.10	7.66	0.61	5.59	0.18	8.29	5.13	6.26	3.62	6
CCLE.A549	0.77	0.42	0.01	0.01	0.00	2.53	0.51	6.24	2.70	4.73	0.01	0.00	0.00	3.64	3.89	6.64	1.56	3.42	6.91	0.21	5.63	0.04	8.07	3.30	5.26	3.02	6
CCLE.A673	2.78	0.00	2.53	3.44	0.19	4.62	0.36	6.98	3.33	0.16	0.11	0.00	1.74	3.54	2.43	5.33	1.23	5.13	7.10	0.19	5.38	0.18	8.60	4.64	4.57	2.58	6
CCLE.A704	0.44	0.31	0.18	0.01	0.15	3.37	3.24	5.73	3.09	0.15	0.00	0.00	0.00	1.55	3.86	4.79	2.38	5.07	6.02	0.75	3.37	0.00	4.25	3.25	4.58	2.92	5
CCLE.ABC1	0.08	0.00	0.08	1.09	0.00	3.74	0.04	5.66	2.74	0.12	0.00	0.00	0.00	3.04	2.38	5.72	2.30	3.88	7.37	0.08	4.94	1.97	8.11	4.55	5.66	2.38	6
CCLE.ACCMESO1	3.92	0.01	1.38	0.01	0.03	4.07	0.06	5.01	2.29	2.47	0.00	0.00	0.00	2.19	4.90	5.90	1.53	5.08	6.82	0.19	4.97	0.00	5.97	3.58	5.26	2.68	5
CCLE.ACHN	0.37	0.04	0.07	0.04	0.00	3.55	1.96	6.10	3.74	0.04	0.00	0.00	0.06	3.55	4.19	6.14	2.24	2.75	7.28	0.18	4.38	0.00	9.43	3.67	5.47	3.27	5
CCLE.AGS	0.06	0.04	0.04	0.03	0.07	0.01	0.00	5.45	3.79	0.42	0.00	0.00	0.00	3.97	4.17	6.24	1.95	4.01	7.49	0.06	6.04	0.01	7.31	5.03	4.63	3.48	6
CCLE.ALLSIL	3.18	0.00	0.08	0.00	0.07	0.03	0.04	6.45	1.78	0.03	0.00	0.00	0.00	0.03	0.00	6.20	2.56	4.77	7.24	0.14	5.39	2.18	7.30	5.33	6.01	3.45	6
CCLE.AM38	2.35	0.00	0.08	0.07	0.08	0.01	0.06	6.64	3.86	4.17	0.00	0.01	0.00	4.01	4.10	5.54	2.13	4.85	7.47	0.12	5.09	0.04	7.89	4.58	6.15	3.35	7
CCLE.AML193	1.51	0.03	0.16	0.19	0.00	0.16	0.06	5.83	3.26	0.08	0.00	0.06	0.00	0.15	0.20	5.50	0.98	4.50	7.13	0.07	5.37	0.01	6.28	4.77	6.08	2.95	5
CCLE.AMO1	2.36	0.00	0.08	0.00	0.00	0.42	0.00	6.28	3.01	0.03	0.00	0.00	0.00	0.04	1.80	4.90	2.01	3.32	7.59	1.42	5.42	0.14	9.48	4.29	6.29	2.94	7
CCLE.AN3CA	2.73	0.06	0.12	0.04	0.16	0.14	0.03	6.20	3.25	0.04	0.00	0.00	0.00	0.10	3.25	6.04	2.31	3.78	7.29	0.14	5.39	0.00	7.84	5.48	5.67	3.08	5
CCLE.ASPC1	0.10	4.54	0.00	0.03	0.07	3.61	0.00	5.74	2.92	4.04	0.03	0.00	0.00	0.20	3.07	5.44	1.85	4.48	7.50	0.08	5.09	0.00	7.85	2.54	3.94	2.81	5
CCLE.AU565	1.28	0.01	1.88	2.98	0.06	0.88	0.00	6.32	4.35	0.50	0.00	0.00	0.00	2.24	2.47	5.03	1.88	2.74	8.04	0.14	6.06	3.47	6.51	4.36	4.70	3.46	5
CCLE.BC3C	0.38	0.00	0.11	0.00	0.00	4.67	0.04	6.22	3.24	3.72	0.18	0.00	0.00	2.44	4.05	5.40	2.07	3.93	6.87	0.41	5.75	0.03	8.25	4.36	5.55	2.94	5
CCLE.BCP1	0.23	0.11	0.06	0.03	0.00	0.04	0.00	6.18	2.58	0.08	0.00	0.00	0.00	0.01	3.18	6.22	1.68	2.54	8.15	1.72	5.42	0.20	8.30	4.62	5.36	3.89	6
CCLE.BCPAP	3.05	0.03	0.16	0.03	0.00	5.18	0.19	6.41	2.75	2.72	0.01	0.00	0.00	3.61	2.23	5.32	1.74	4.03	7.10	0.07	6.35	0.25	8.70	4.23	5.26	3.79	5
CCLE.BDCM	1.94	0.00	0.32	0.01	0.00	1.23	0.00	6.12	3.54	0.00	0.00	0.00	0.00	0.14	1.80	5.42	1.97	2.62	7.03	0.32	4.89	0.03	7.70	4.64	6.06	3.65	5
CCLE.BEN	0.04	3.62	0.19	0.03	0.08	2.08	0.03	5.82	4.80	0.61	0.10	0.00	0.00	2.91	3.13	5.60	1.11	3.90	7.28	0.10	5.16	0.21	8.57	3.75	4.40	3.10	5
CCLE.BFTC905	0.86	0.00	0.06	1.41	0.08	4.94	0.00	6.09	3.34	3.55	0.16	0.04	0.00	3.17	2.58	5.97	1.64	2.92	7.43	0.71	5.28	0.14	8.44	4.26	5.32	3.94	5
CCLE.BFTC909	2.26	0.00	0.11	0.03	0.07	3.63	0.41	6.25	3.43	0.21	0.00	0.00	0.00	2.68	3.41	5.44	1.63	4.76	6.90	0.18	4.87	0.10	8.70	3.87	5.43	2.79	3
CCLE.BHT101	1.69	4.39	0.08	0.03	0.03	3.34	0.91	5.77	3.48	3.62	0.01	0.00	0.00	1.22	3.43	4.84	1.60	3.00	7.24	1.44	4.72	0.06	8.78	3.97	5.23	3.82	5
CCLE.BHY	1.36	0.00	0.42	3.52	0.00	4.37	0.77	5.69	3.61	3.91	0.19	0.00	0.00	2.32	3.47	5.68	2.59	5.16	6.51	1.66	5.21	0.29	7.66	4.62	4.99	3.69	5
CCLE.BICR16	0.98	0.01	0.08	5.12	0.00	4.27	0.00	5.05	3.43	0.08	0.00	0.00	0.00	1.70	3.33	5.41	1.95	4.53	6.23	0.52	4.75	0.08	8.55	3.37	4.14	3.17	4
CCLE.BICR18	0.51	0.08	0.08	5.47	0.00	6.60	0.06	5.67	3.11	0.80	0.52	0.03	0.00	2.65	2.49	6.93	1.76	3.97	7.00	0.66	4.85	0.63	8.47	4.73	5.51	3.52	4
CCLE.BICR22	1.38	0.01	0.12	7.36	0.00	4.29	0.01	3.87	3.10	0.03	0.01	0.00	0.00	0.98	3.02	4.88	1.51	2.49	6.21	0.97	3.96	0.04	7.25	2.63	3.04	2.92	3
CCLE.BICR31	2.69	0.06	0.19	4.48	0.12	6.01	0.01	5.92	3.58	1.04	0.04	0.00	0.00	1.10	3.71	6.00	1.99	5.13	7.46	0.32	5.68	1.14	8.17	3.92	5.26	3.41	4
CCLE.BICR56	0.60	0.01	0.07	4.72	0.00	4.61	0.01	3.75	2.52	0.14	0.01	0.00	0.00	0.34	1.95	4.89	2.05	3.70	6.44	0.45	4.04	0.10	7.62	2.90	3.07	2.72	4
CCLE.BICR6	0.03	0.01	0.10	6.10	0.00	4.28	0.00	4.27	1.66	0.07	0.00	0.00	0.00	0.86	3.18	4.67	0.97	2.45	5.98	0.30	4.18	0.11	6.60	2.16	3.70	2.21	3
CCLE.BL41	1.79	0.04	0.07	0.08	0.00	2.34	0.00	6.82	2.68	0.03	0.00	0.00	0.00	0.04	0.11	6.30	0.24	2.58	6.94	1.58	5.29	0.06	8.07	4.51	6.37	2.17	5
CCLE.BL70	2.12	0.01	0.10	0.03	0.00	1.37	0.00	5.54	3.19	0.00	0.00	0.00	0.00	0.03	0.12	5.47	0.18	3.15	7.14	0.46	5.15	0.24	7.89	4.61	5.79	3.37	5
CCLE.BT-12	1.60	0.00	0.08	0.00	0.00	2.72	0.15	6.34	1.84	4.52	0.16	0.00	0.00	0.04	2.90	5.54	0.96	2.97	7.31	0.63	4.94	0.00	7.59	4.03	5.71	3.19	6

One Hot Encoding of Categories

State	Binary	One-Hot	Hamming 2	Hamming 3
S0	000	00000001	0000	000000
S1	001	00000010	0011	000111
S2	010	00000100	0101	011001
S3	011	00001000	0110	011110
S4	100	00010000	1001	101010
S5	101	00100000	1010	101101
S6	110	01000000	1100	110011
S7	111	10000000	1111	110100

Open Source Framework Comparison

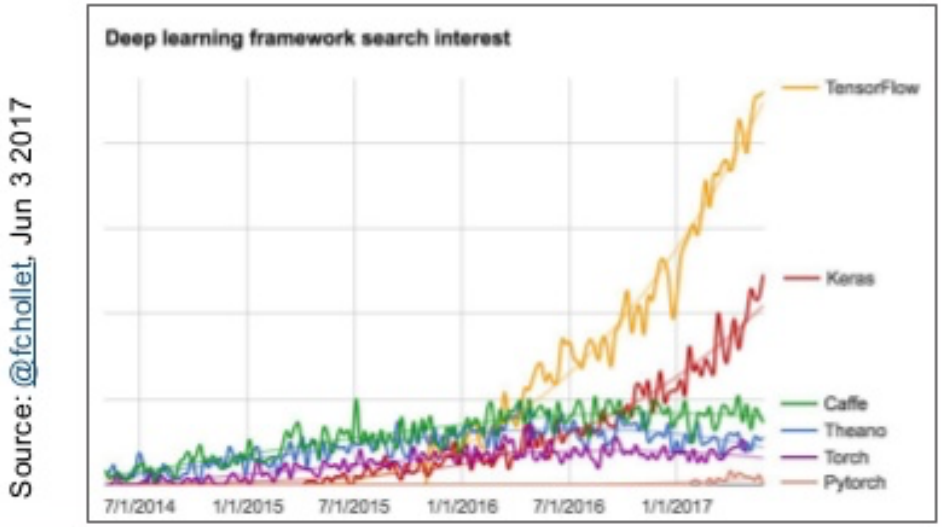
	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	+
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	+

Keras

- https://keras.io/
- Minimalist, highly modular neural networks library
- Written in Python
- Capable of running on top of either TensorFlow/Theano and CNTK
- Developed with a focus on enabling fast experimentation

Keras

- Keras is the *de facto* deep learning frontend



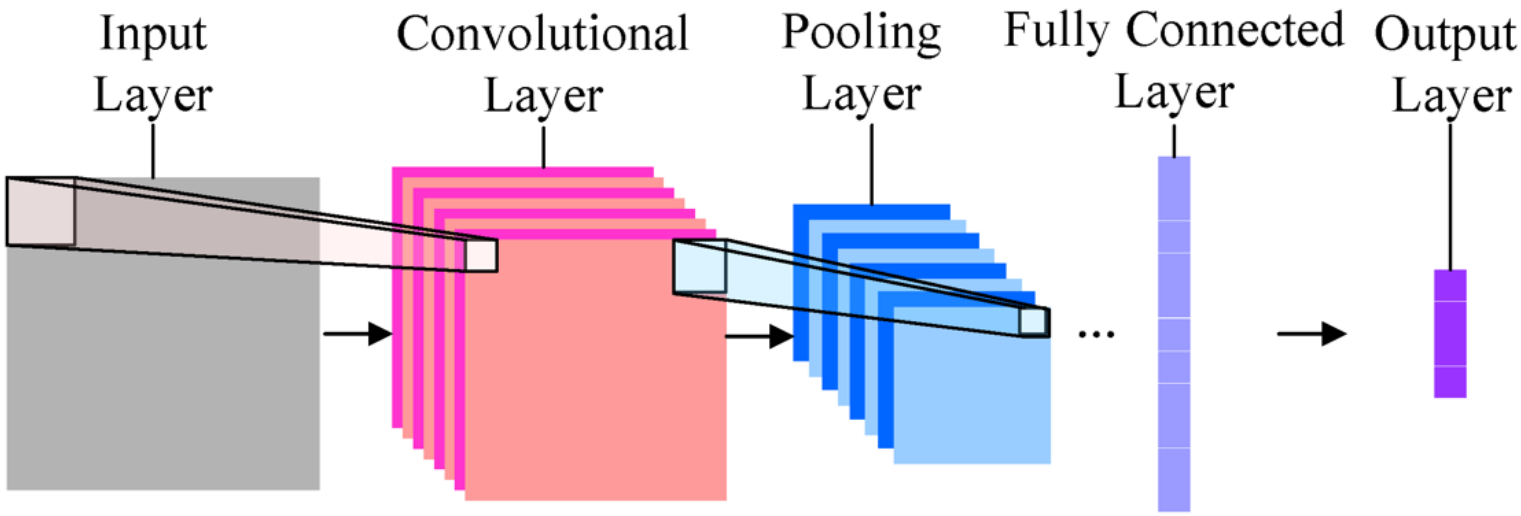
Source: @fchollet, Jun 3 2017

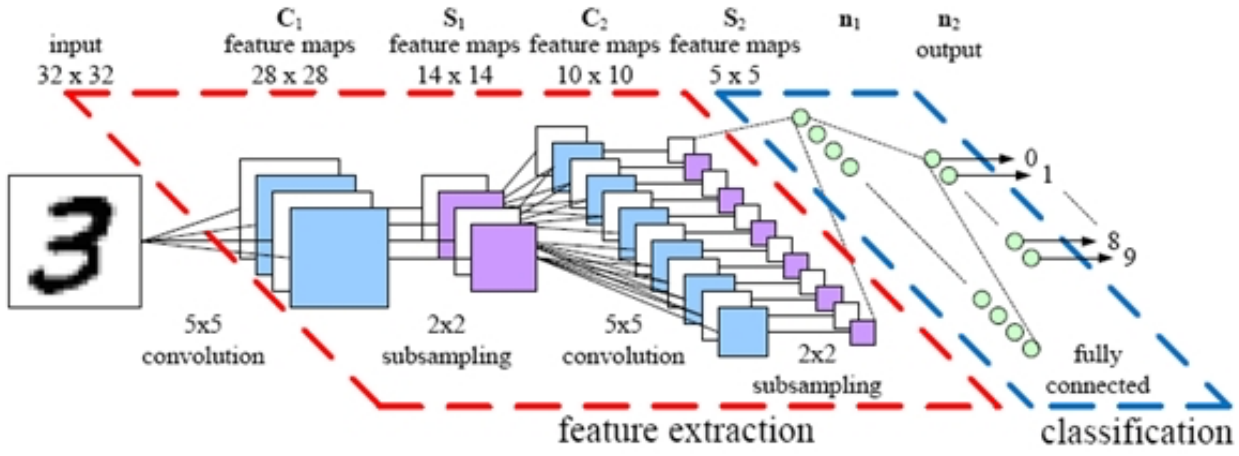
```
from keras.layers import Input, Dense
from keras.models import Model

input_layer = Input(shape=(1000,))
fc_1 = Dense(512, activation='relu')(input_layer)
fc_2 = Dense(256, activation='relu')(fc_1)
output_layer = Dense(10, activation='softmax')(fc_2)

model = Model(input=input_layer, output=output_layer)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(bow, newsgroups.target)
predictions = model.predict(features).argmax(axis=1)
```

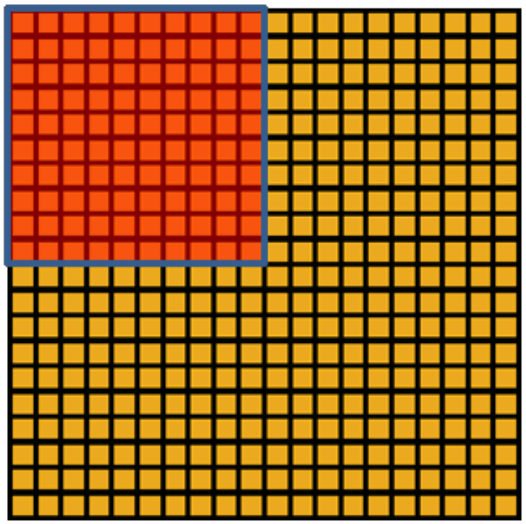
Neural Network for Classification

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 60464, 128)	2688
activation_1 (Activation)	(None, 60464, 128)	0
max_pooling1d_1 (MaxPooling1	(None, 60464, 128)	0
conv1d_2 (Conv1D)	(None, 60455, 128)	163968
activation_2 (Activation)	(None, 60455, 128)	0
max_pooling1d_2 (MaxPooling1	(None, 6045, 128)	0
flatten_1 (Flatten)	(None, 773760)	0
dense_1 (Dense)	(None, 200)	154752200
activation_3 (Activation)	(None, 200)	0
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 20)	4020
activation_4 (Activation)	(None, 20)	0
dropout_2 (Dropout)	(None, 20)	0
dense_3 (Dense)	(None, 36)	756
activation_5 (Activation)	(None, 36)	0

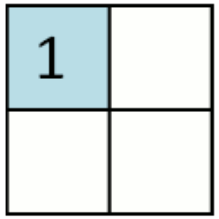
Total params: 154,923,632
Trainable params: 154,923,632
Non-trainable params: 0


Input Layer
Convolutional Layer
Pooling Layer
Fully Connected Layer
Output Layer

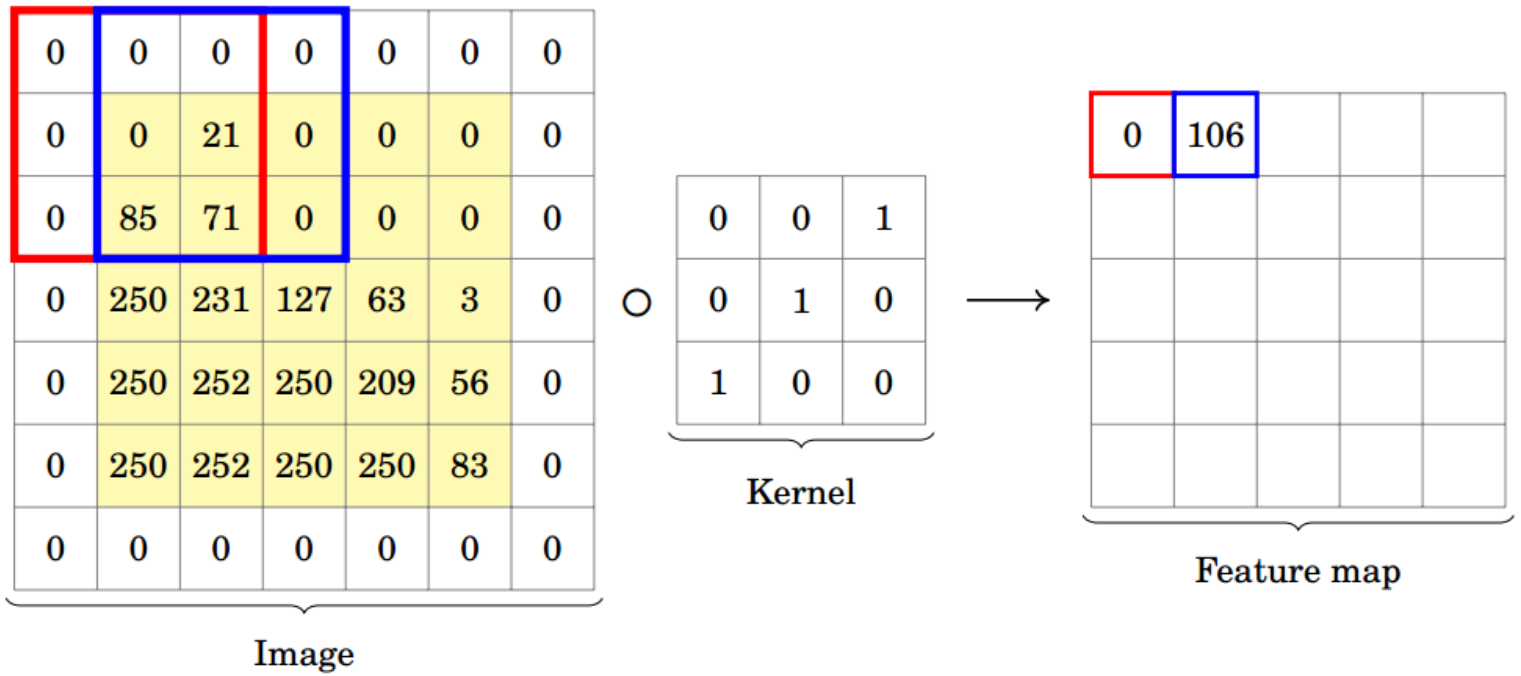

input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification



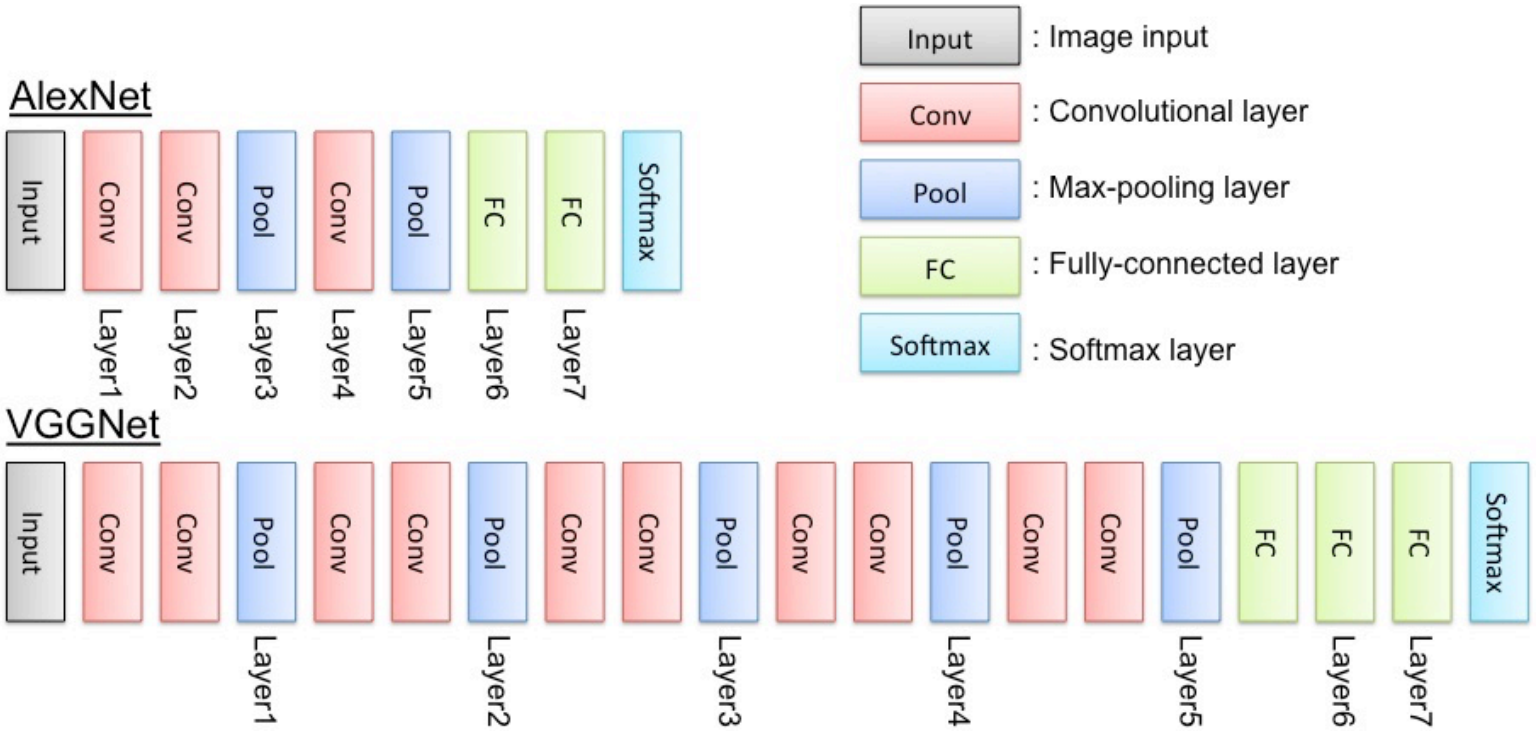
Convolved feature



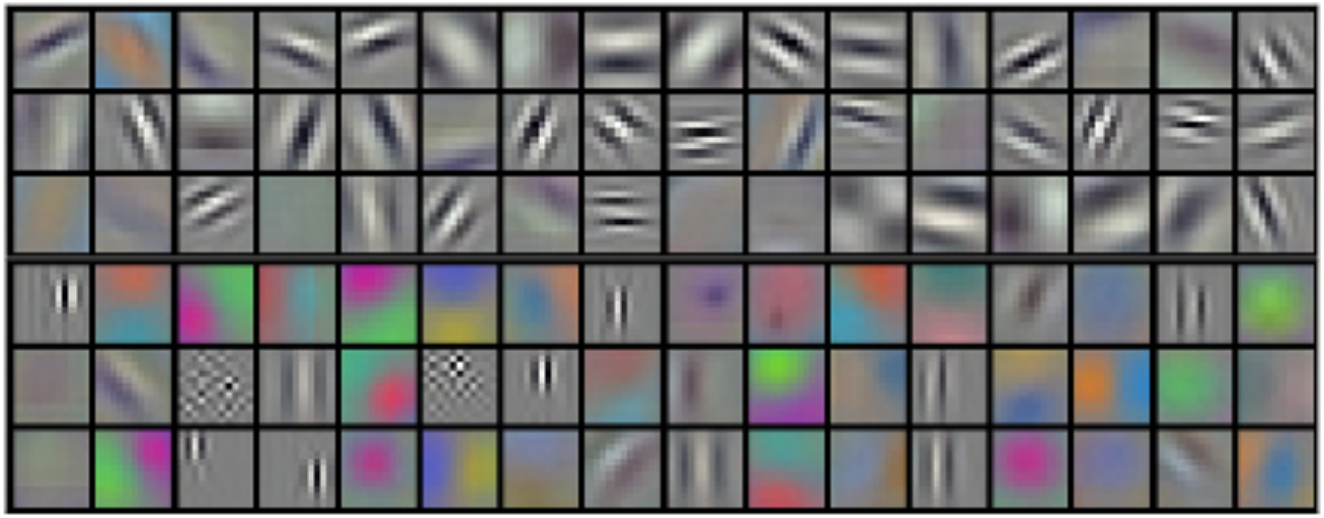
Pooled feature


0 0 0 0 0 0 0
0 0 21 0 0 0 0
0 85 71 0 0 0 0
0 250 231 127 63 3 0
0 250 252 250 209 56 0
0 250 252 250 250 83 0
0 0 0 0 0 0 0
Image
0 0 1
0 1 0
1 0 0
Kernel
0 106
Feature map

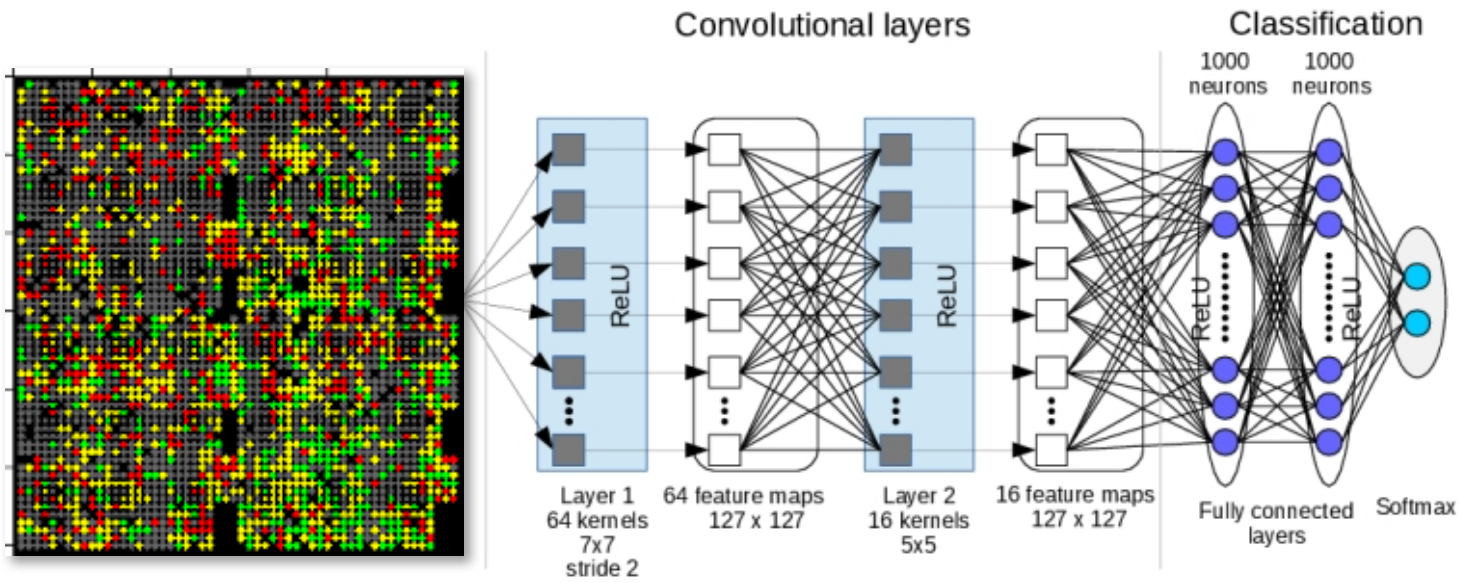
Networks Used to Classify Images



Example Features Learned in 2D Convolution Lower Layers

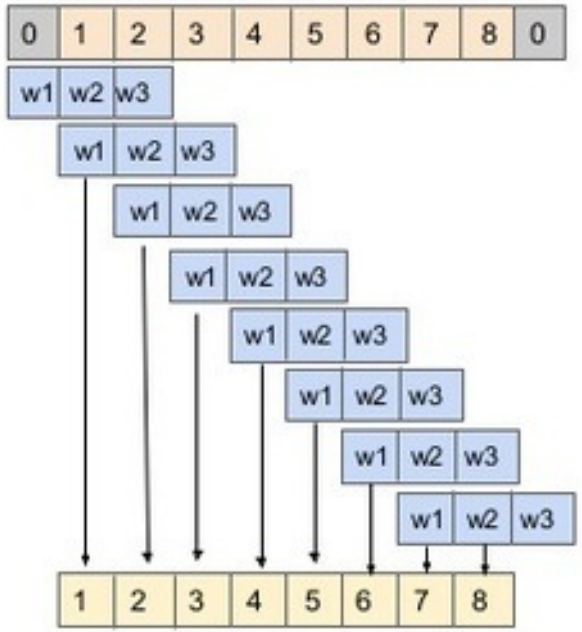


Convolution vs
Fully (Dense) Connected Layers
for



1D Convolutions

When we add zero padding, we normally do so on both sides of the sequence (as in image padding)



Neural Network for Classification

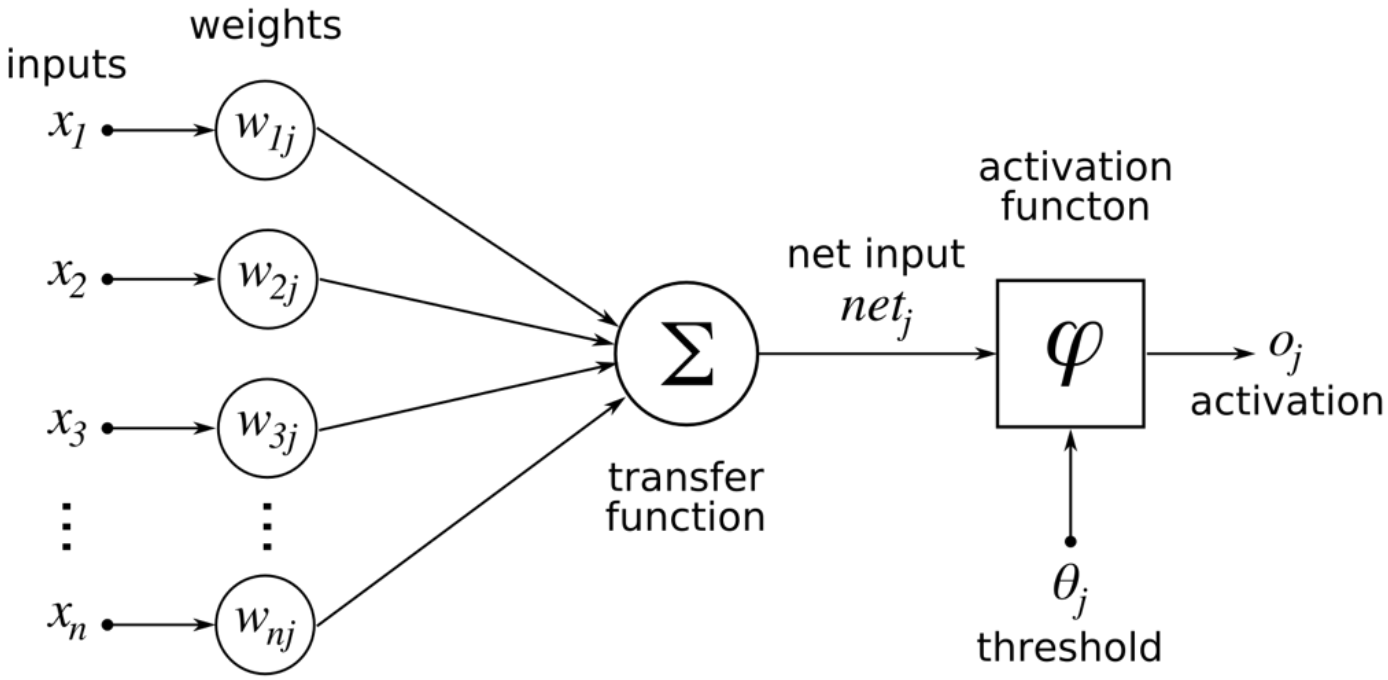
Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 60464, 128)	2688
activation_1 (Activation)	(None, 60464, 128)	0
max_pooling1d_1 (MaxPooling1	(None, 60464, 128)	0
conv1d_2 (Conv1D)	(None, 60455, 128)	163968
activation_2 (Activation)	(None, 60455, 128)	0
max_pooling1d_2 (MaxPooling1	(None, 6045, 128)	0
flatten_1 (Flatten)	(None, 773760)	0
dense_1 (Dense)	(None, 200)	154752200
activation_3 (Activation)	(None, 200)	0
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 20)	4020
activation_4 (Activation)	(None, 20)	0
dropout_2 (Dropout)	(None, 20)	0
dense_3 (Dense)	(None, 36)	756
activation_5 (Activation)	(None, 36)	0

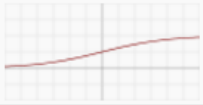
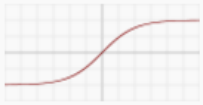
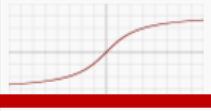
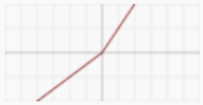
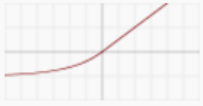
Total params: 154,923,632
Trainable params: 154,923,632
Non-trainable params: 0

Setting up the Graph Structure

```
model = Sequential()
model.add(Conv1D(filters=128, kernel_size=20, strides=1, padding='valid', input_shape=(P, 1)))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Conv1D(filters=128, kernel_size=10, strides=1, padding='valid'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=10))
model.add(Flatten())
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(20))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(CLASSES))
model.add(Activation('softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=SGD(),
              metrics=['accuracy'])
```

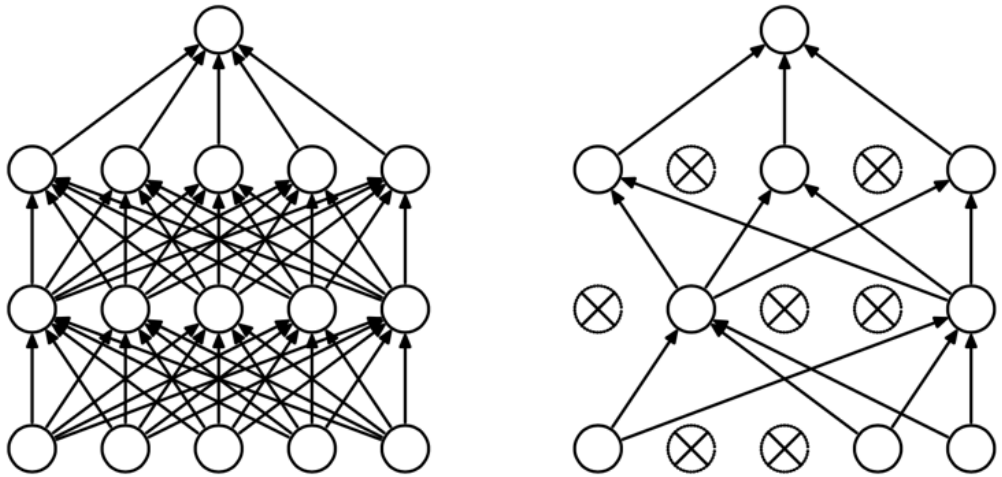
What Activation Function to Use?



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Dropout!

Srivastava, Hinton, Krizhevsky, Sutskever and Salakhutdinov



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

What Loss Function to use?
What Optimizer to use?

```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(),
              metrics=['accuracy'])


# set up a bunch of callbacks to do work during model training..

checkpointer = ModelCheckpoint(filepath='nt3.autosave.model.h5', verbose=1, save_weights_only=False, save_best_only=True)
csv_logger = CSVLogger('training.log')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=10, verbose=1, mode='auto', epsilon=0.0001, cooldown=0, min_lr=0)

history = model.fit(X_train, Y_train,
                    batch_size=BATCH,
                    epochs=EPOCH,
                    verbose=1,
                    validation_data=(X_test, Y_test),
                    callbacks = [checkpointer, csv_logger, reduce_lr])

score = model.evaluate(X_test, Y_test, verbose=0)
```

Example Loss functions

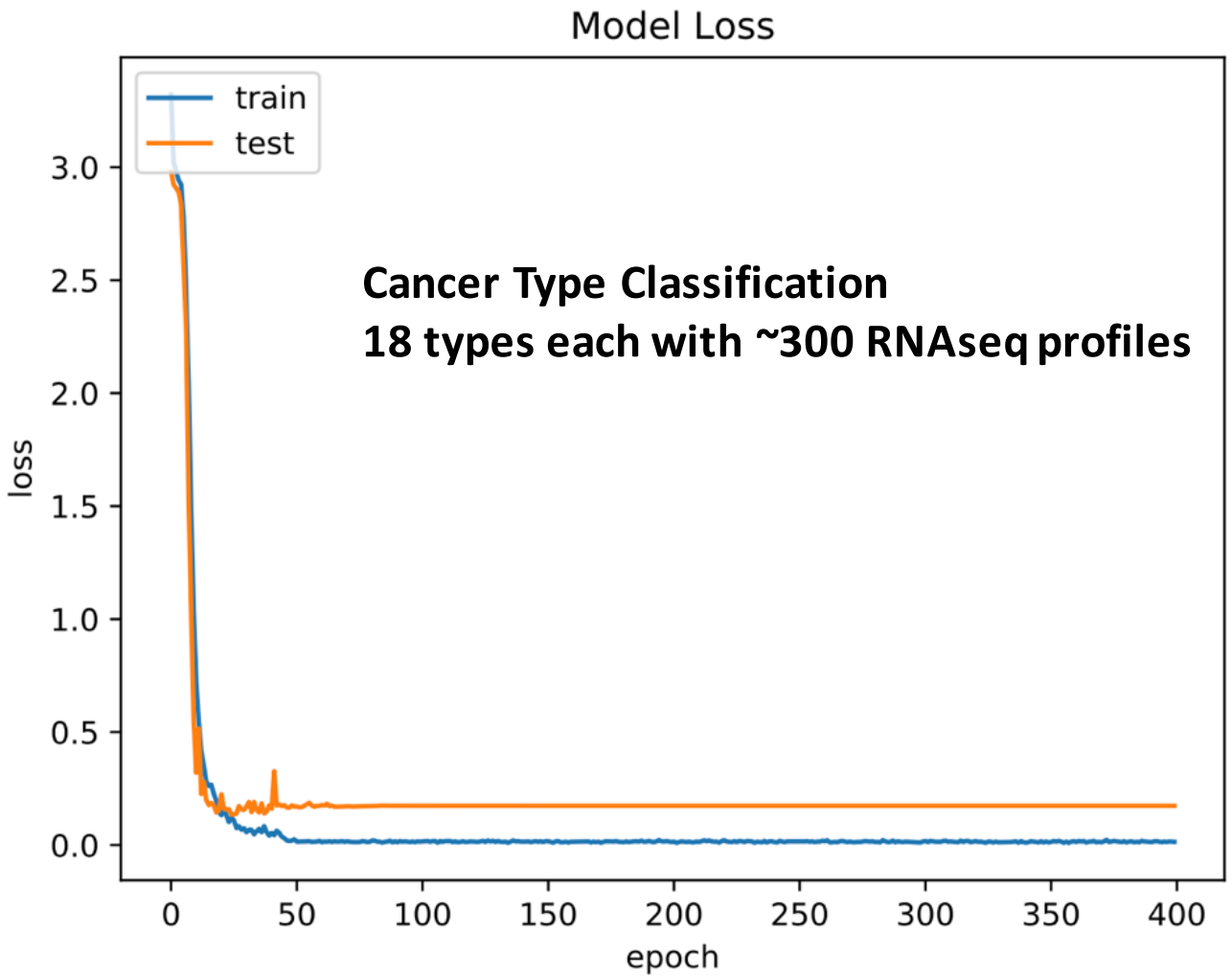
Regression: $$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$$

Classification: cross-entropy (deviance)

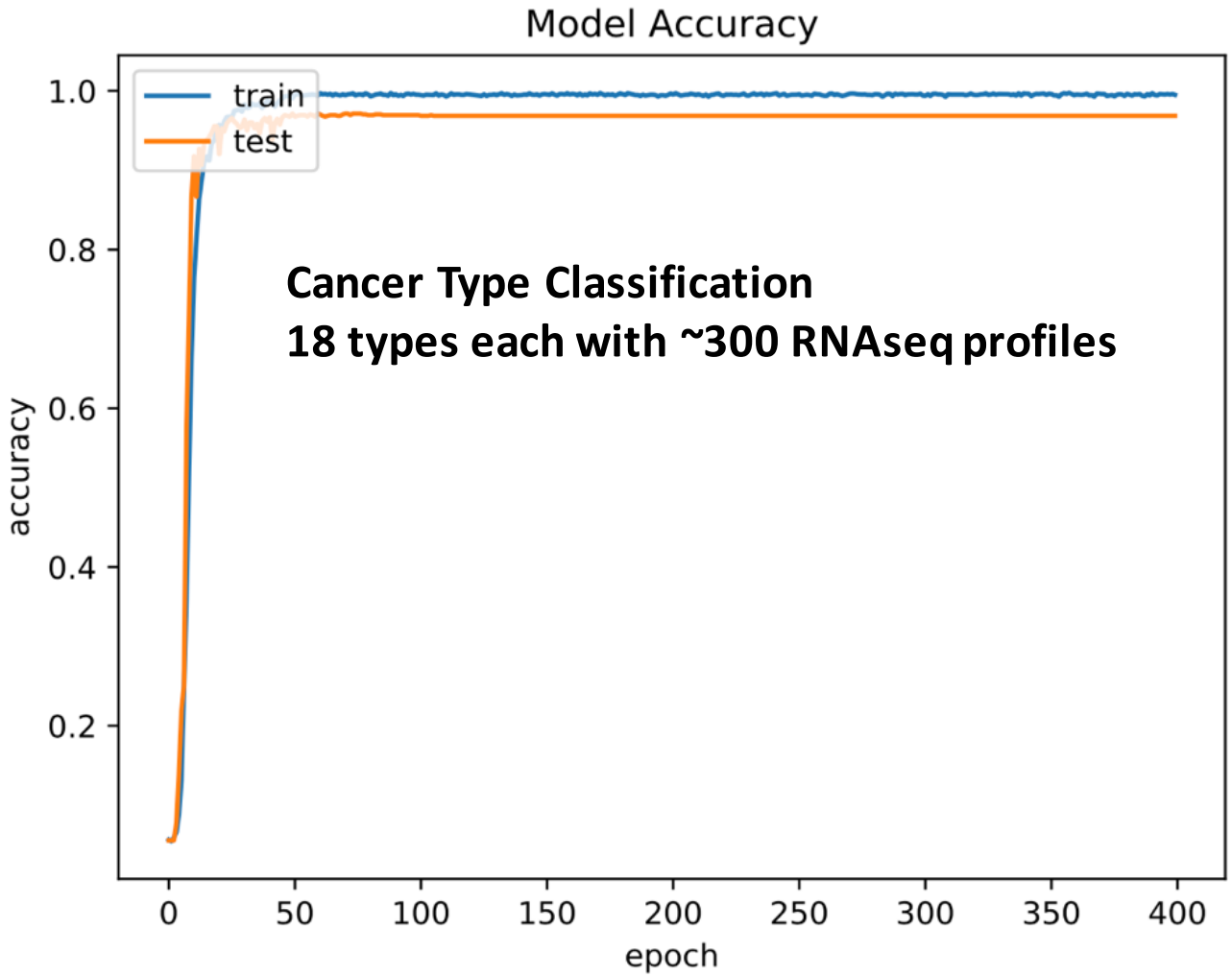
$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i).$$

Cancer Type Classification

```
4320/4320 [==============================] - 87s - loss: 3.2885 - acc: 0.0537 - val_loss: 2.9542 - val_acc: 0.0556
Epoch 2/400
4320/4320 [==============================] - 76s - loss: 2.9777 - acc: 0.0752 - val_loss: 2.8273 - val_acc: 0.1083
Epoch 3/400
4320/4320 [==============================] - 78s - loss: 2.8117 - acc: 0.1176 - val_loss: 2.5971 - val_acc: 0.2194
Epoch 4/400
4320/4320 [==============================] - 77s - loss: 2.5094 - acc: 0.2060 - val_loss: 2.1191 - val_acc: 0.3306
Epoch 5/400
4320/4320 [==============================] - 78s - loss: 2.0385 - acc: 0.3442 - val_loss: 1.6411 - val_acc: 0.4648
Epoch 6/400
4320/4320 [==============================] - 75s - loss: 1.4995 - acc: 0.5079 - val_loss: 0.9846 - val_acc: 0.7704
Epoch 7/400
4320/4320 [==============================] - 77s - loss: 1.0688 - acc: 0.6481 - val_loss: 0.5628 - val_acc: 0.8796
Epoch 8/400
4320/4320 [==============================] - 76s - loss: 0.7657 - acc: 0.7461 - val_loss: 0.4952 - val_acc: 0.8509
Epoch 9/400
4320/4320 [==============================] - 76s - loss: 0.5729 - acc: 0.8123 - val_loss: 0.2803 - val_acc: 0.9287
Epoch 10/400
4320/4320 [==============================] - 79s - loss: 0.4389 - acc: 0.8620 - val_loss: 0.1962 - val_acc: 0.9398
```

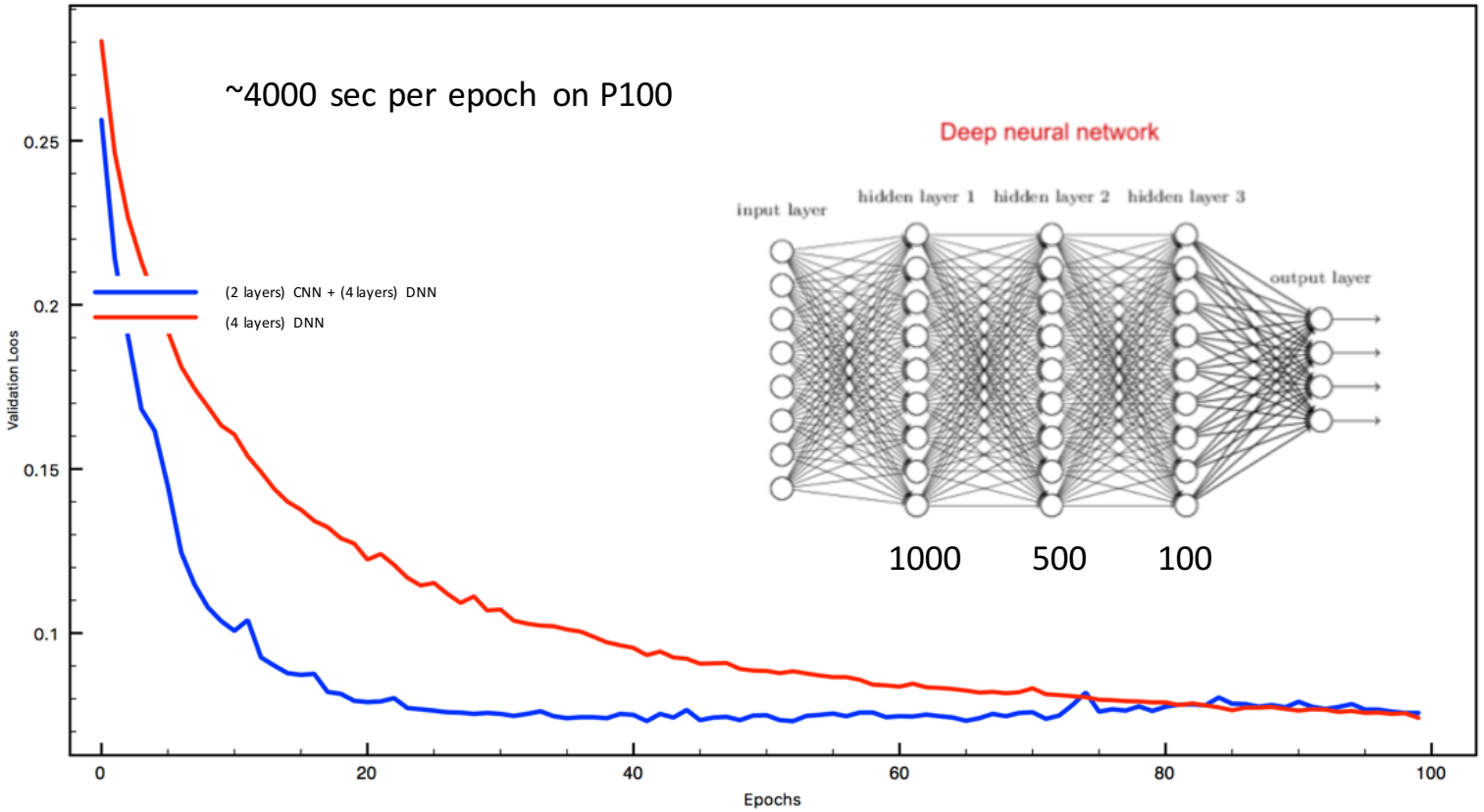


https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/TC1



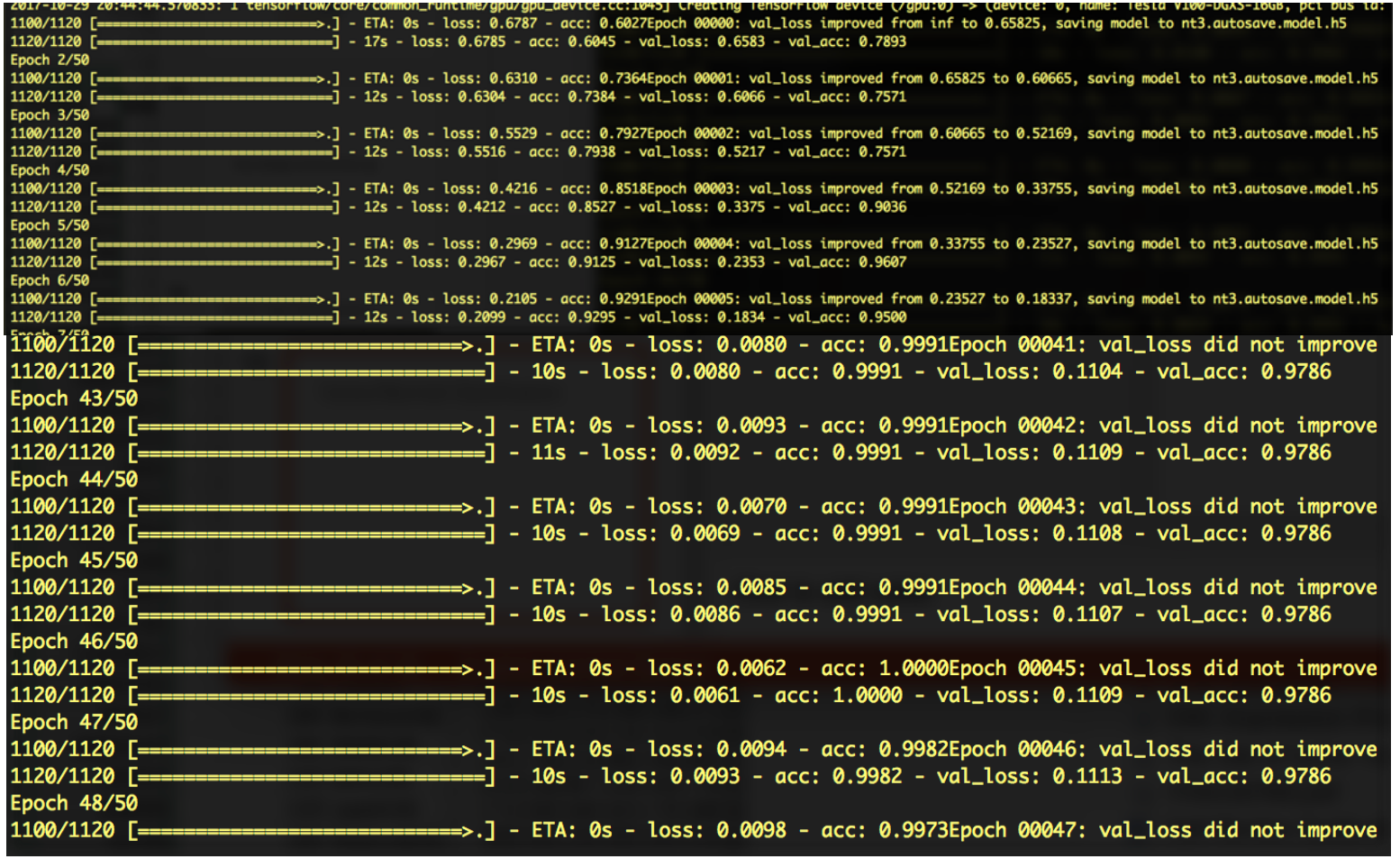
https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/TC1

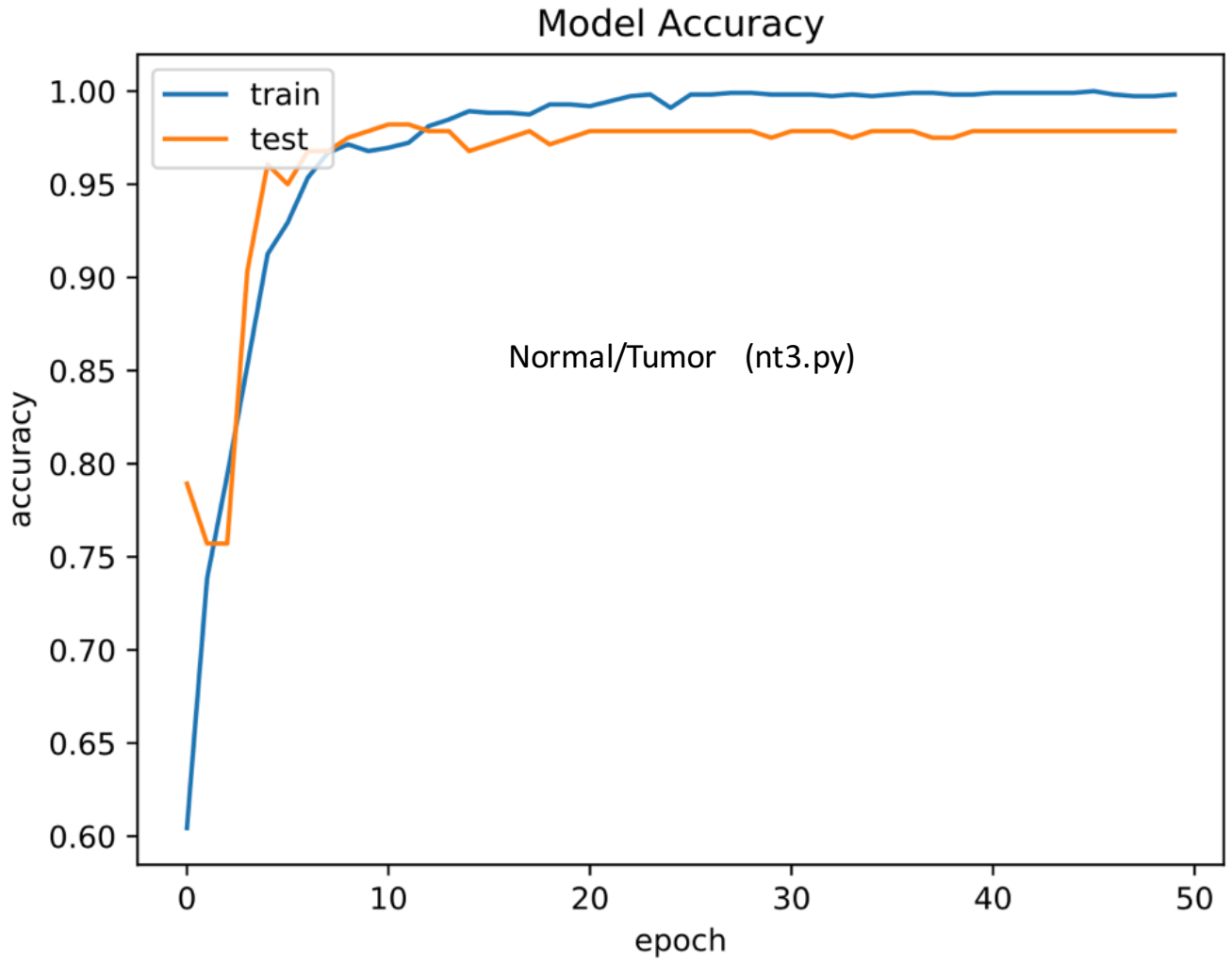
P1B3 Convergence ([C(100)xC(50)]x1000x500x100x50)



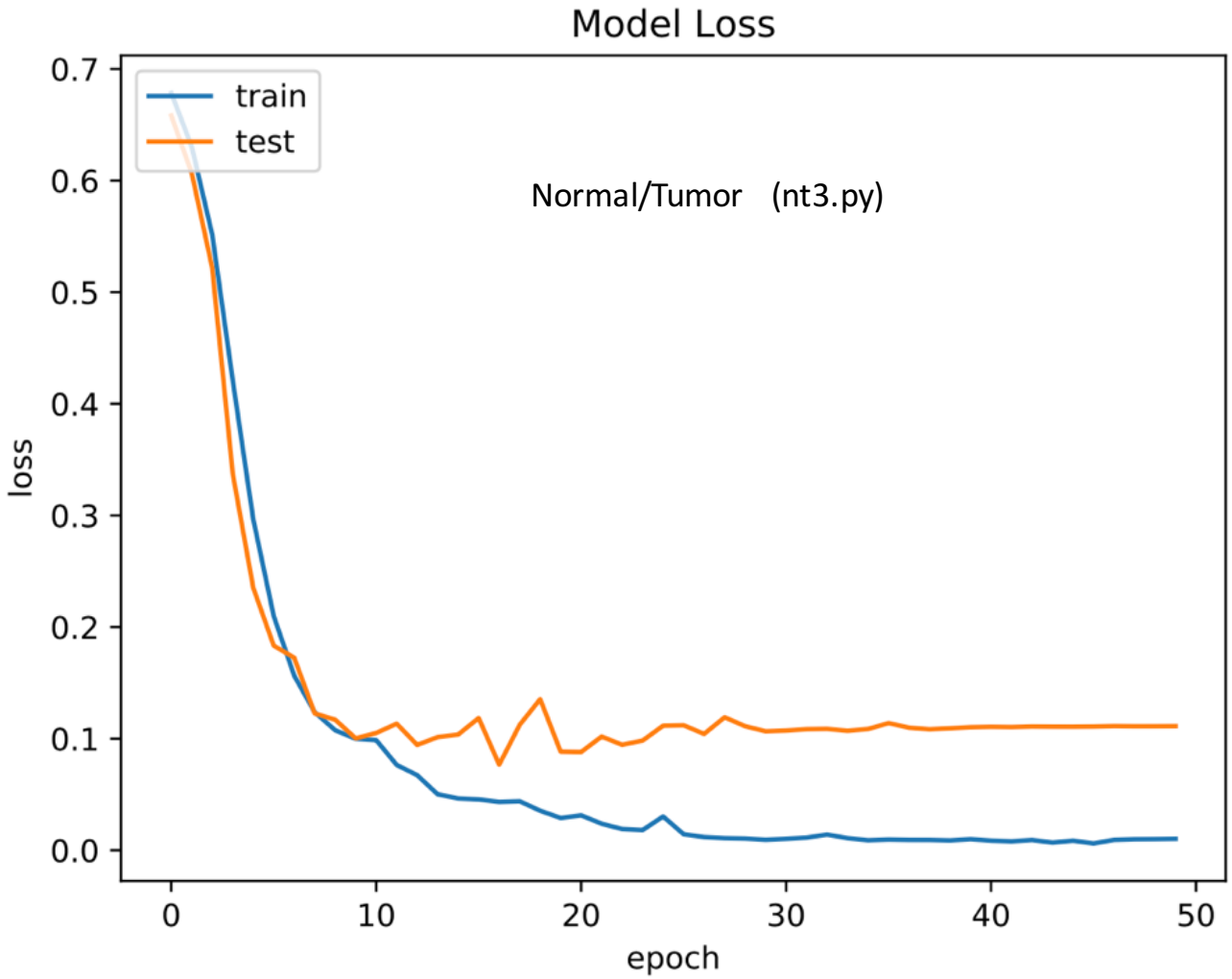
https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/P1B3

Tumor/Normal Classification





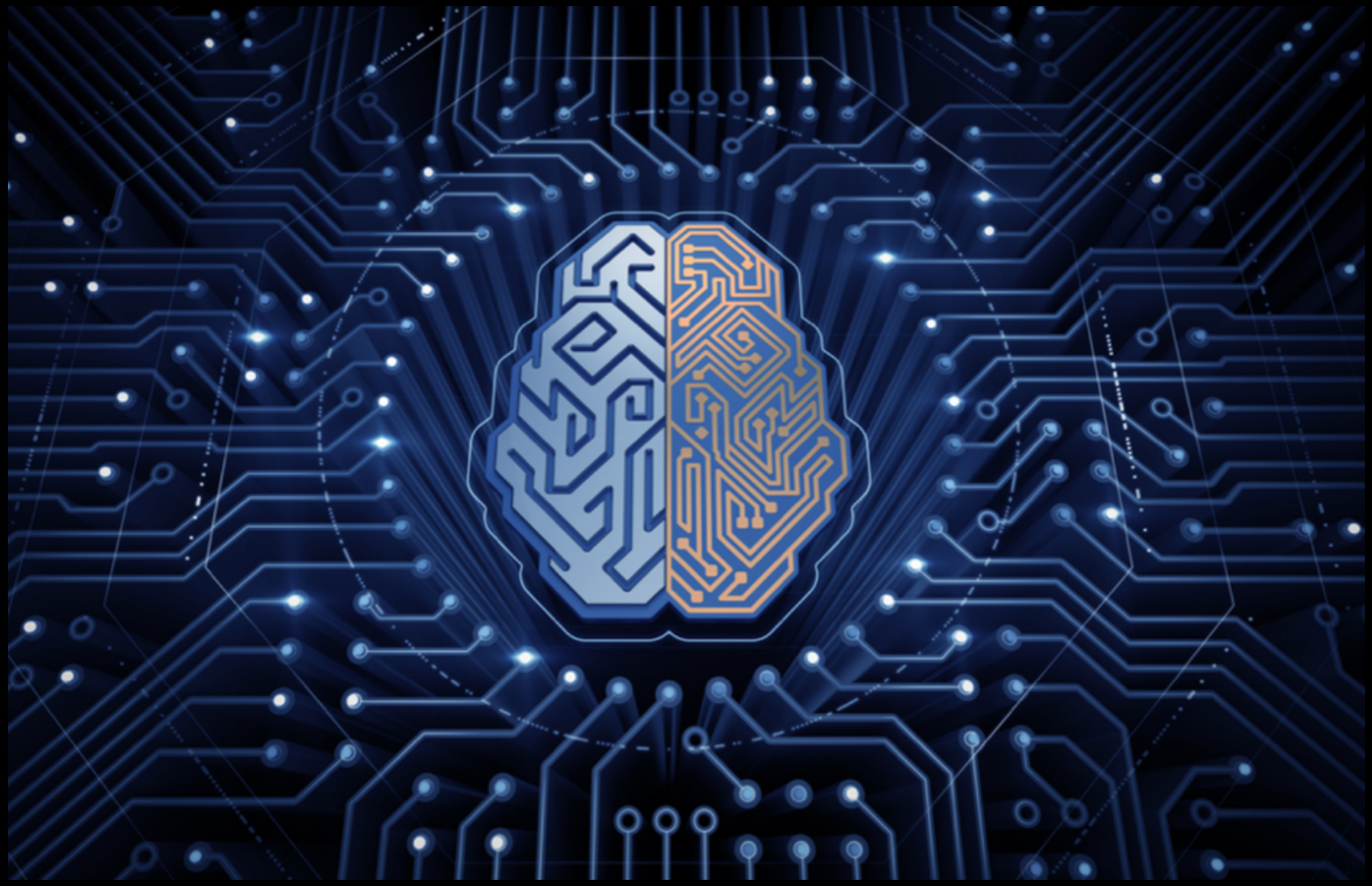
https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/NT3

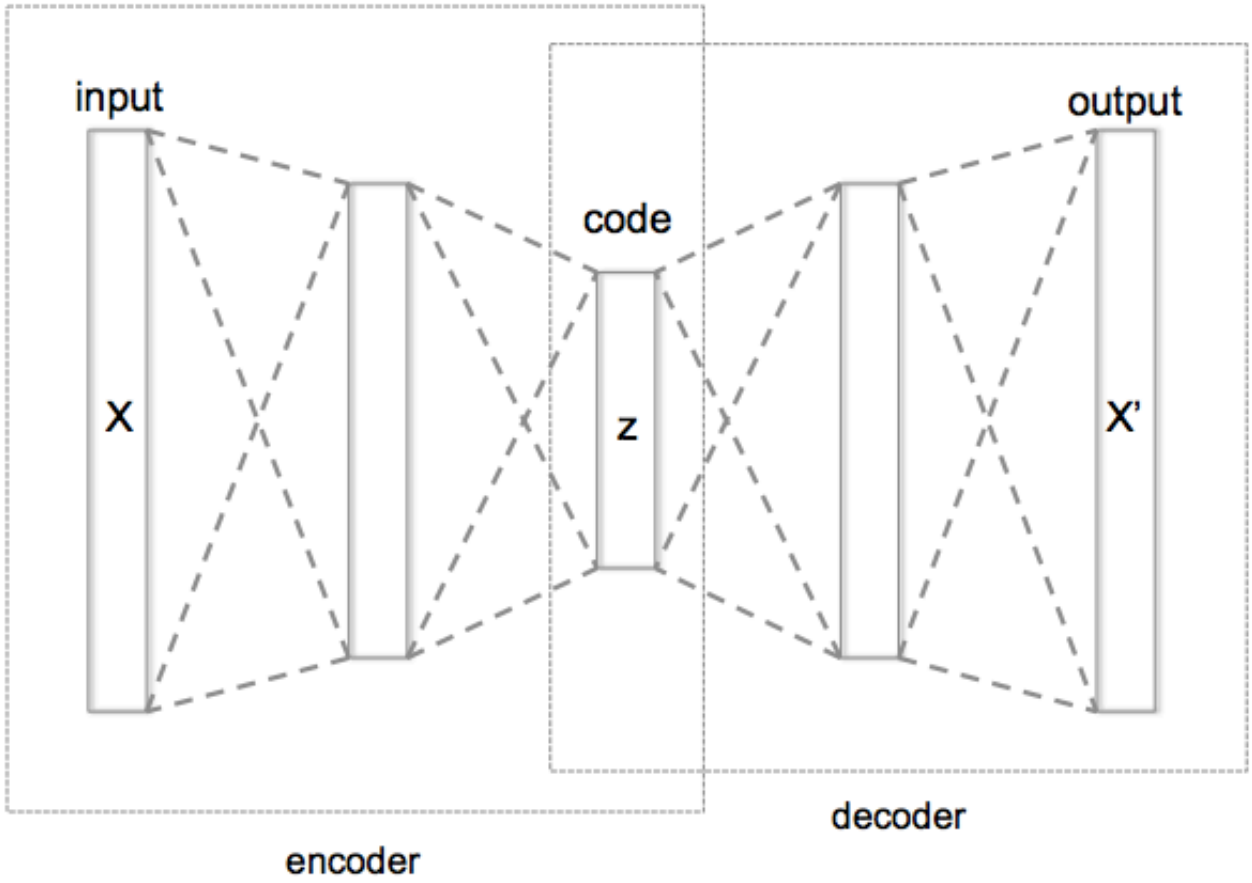


https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/NT3

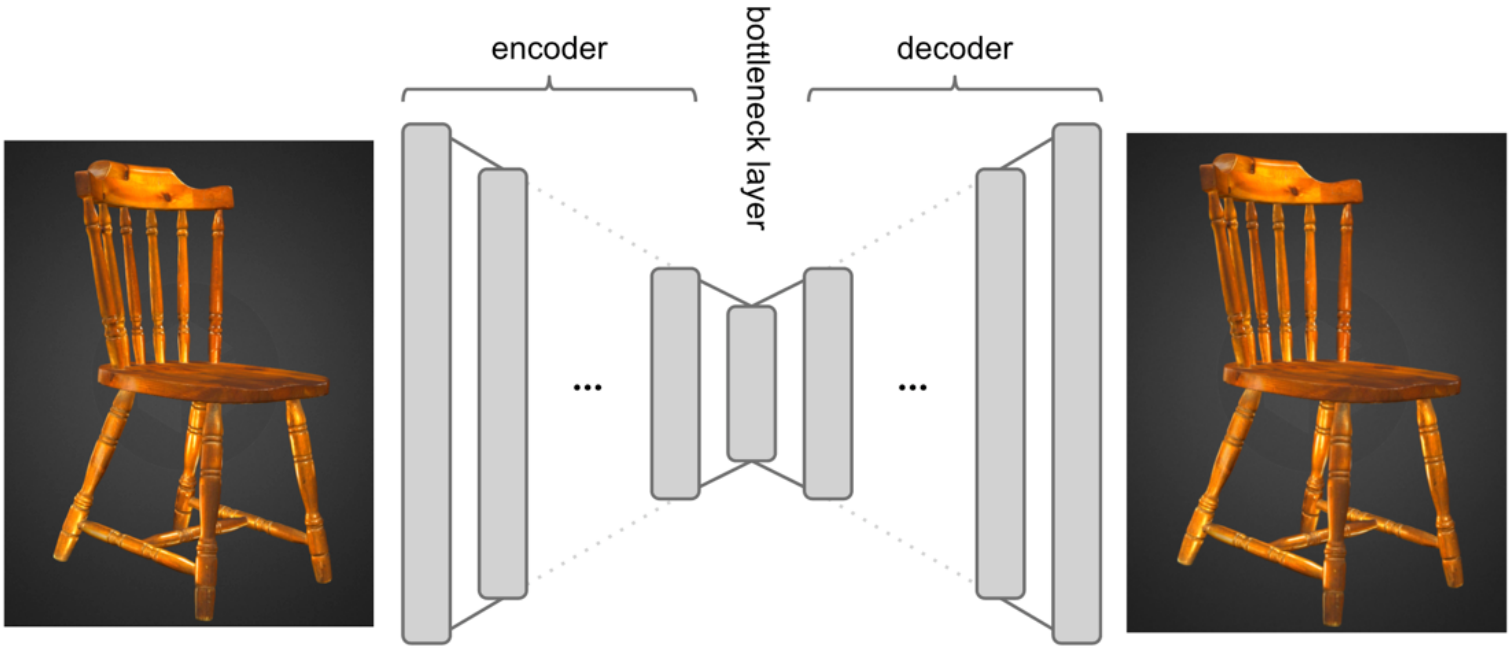
How did we know it might work?

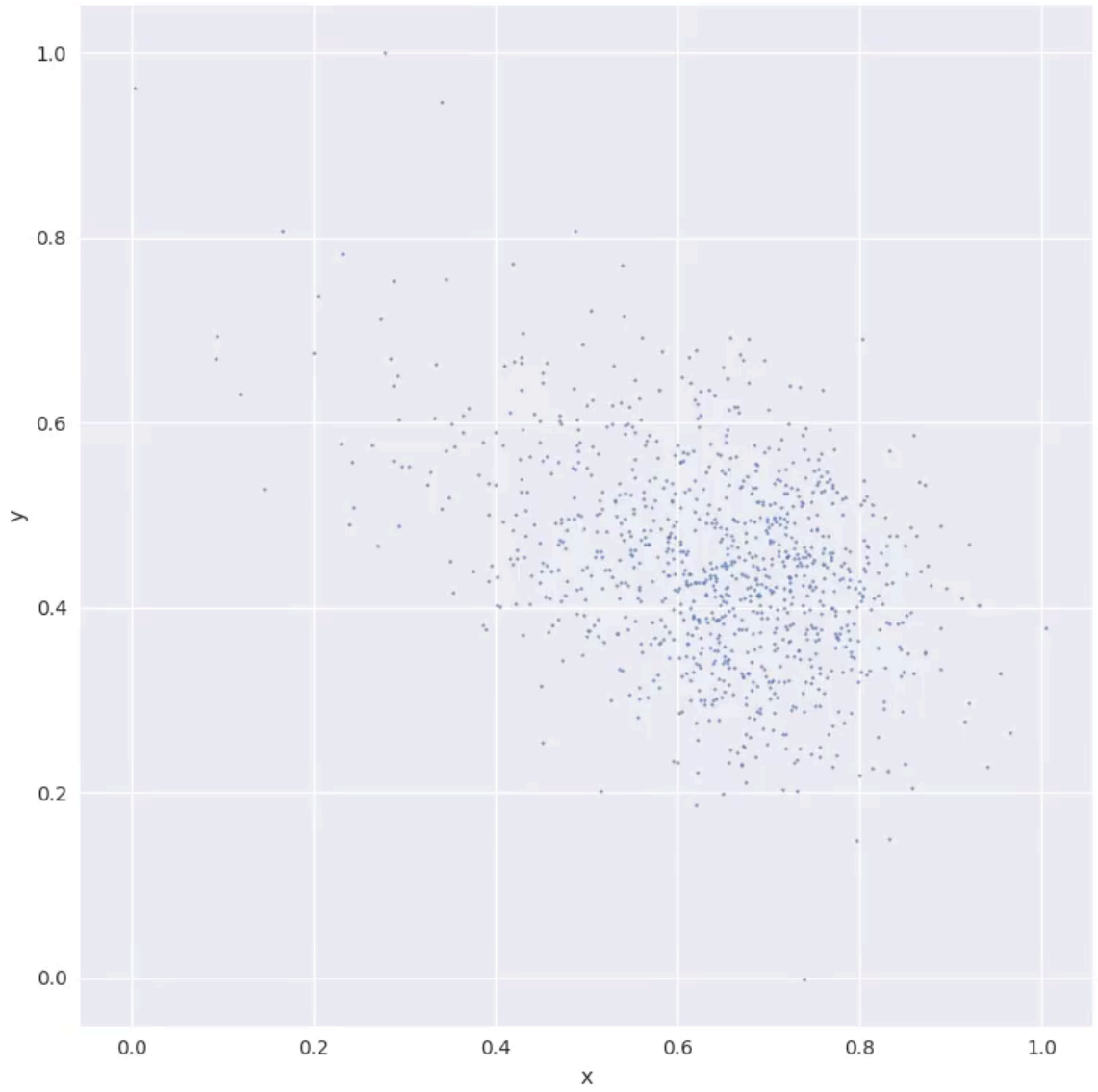
- Build autoencoders first with the features you are going to work with
- If you get reasonable accuracy then the model can learn a representation and that is a good sign
- Class balance seems to matter
- Number of training examples matters > 1000 is good > 10,000 better, > 100,000 much better
- Hyper parameter search is also important once you get something that basically works

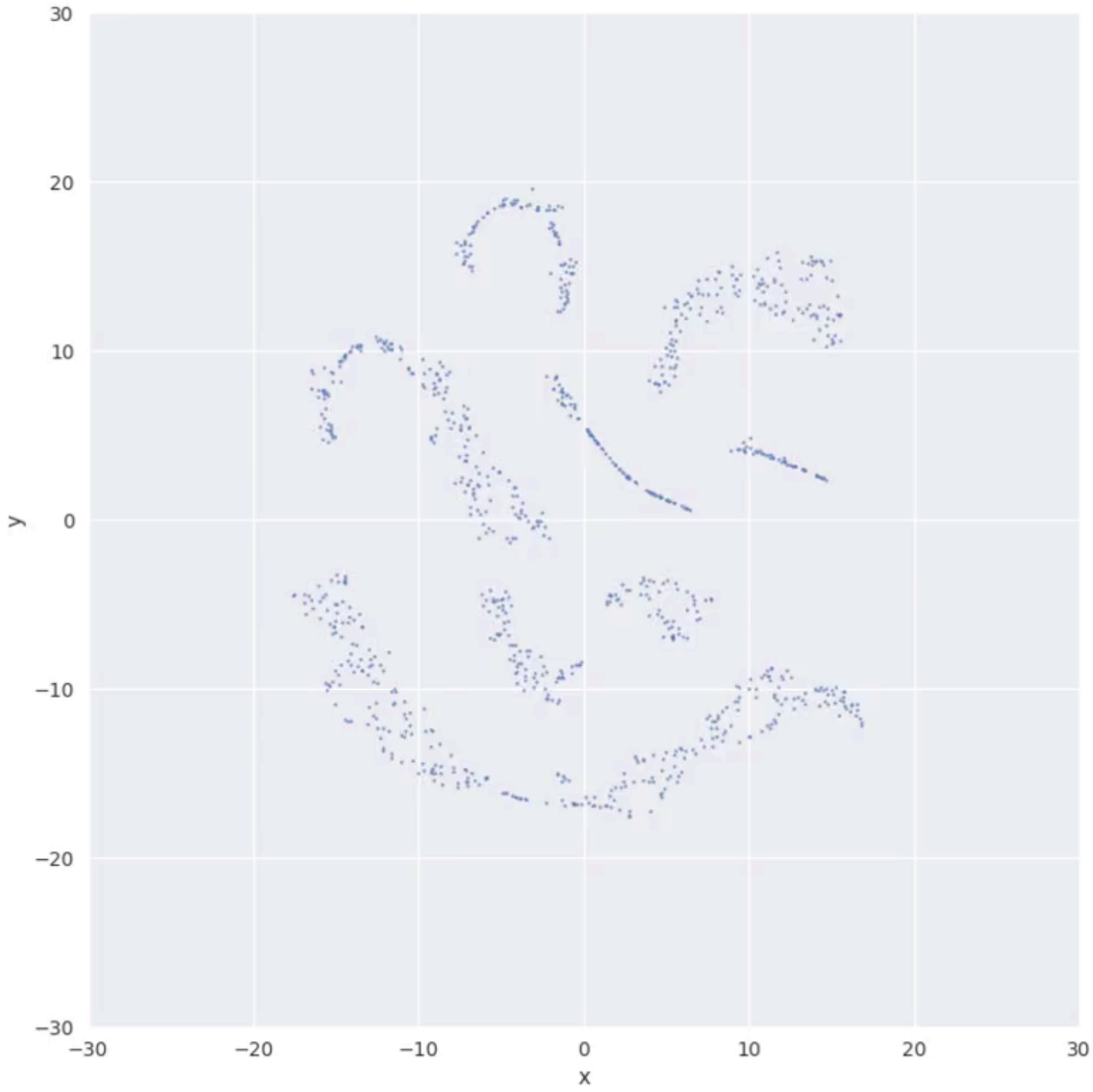



input
output
code
X
z
X'
encoder
decoder

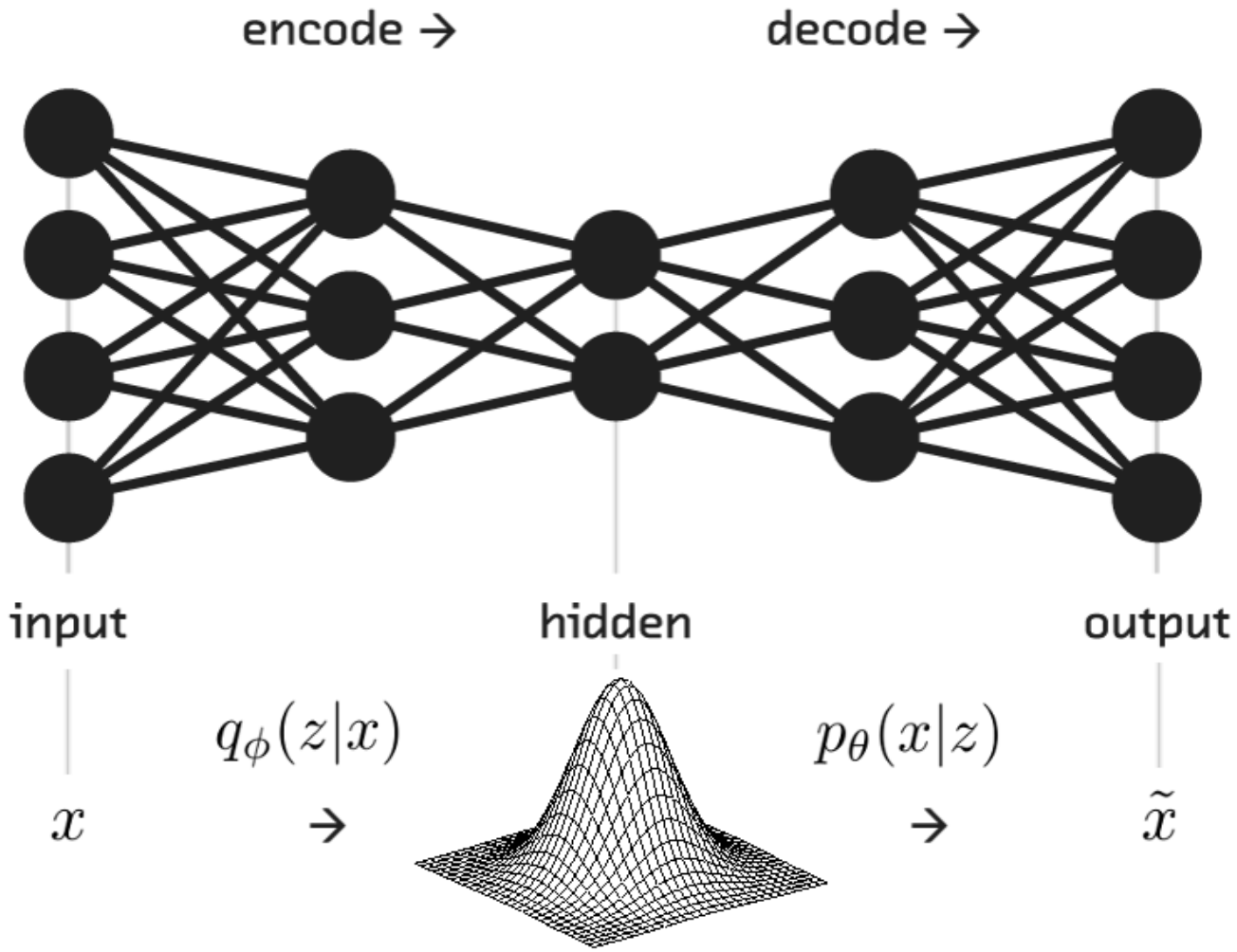
Autoencoder



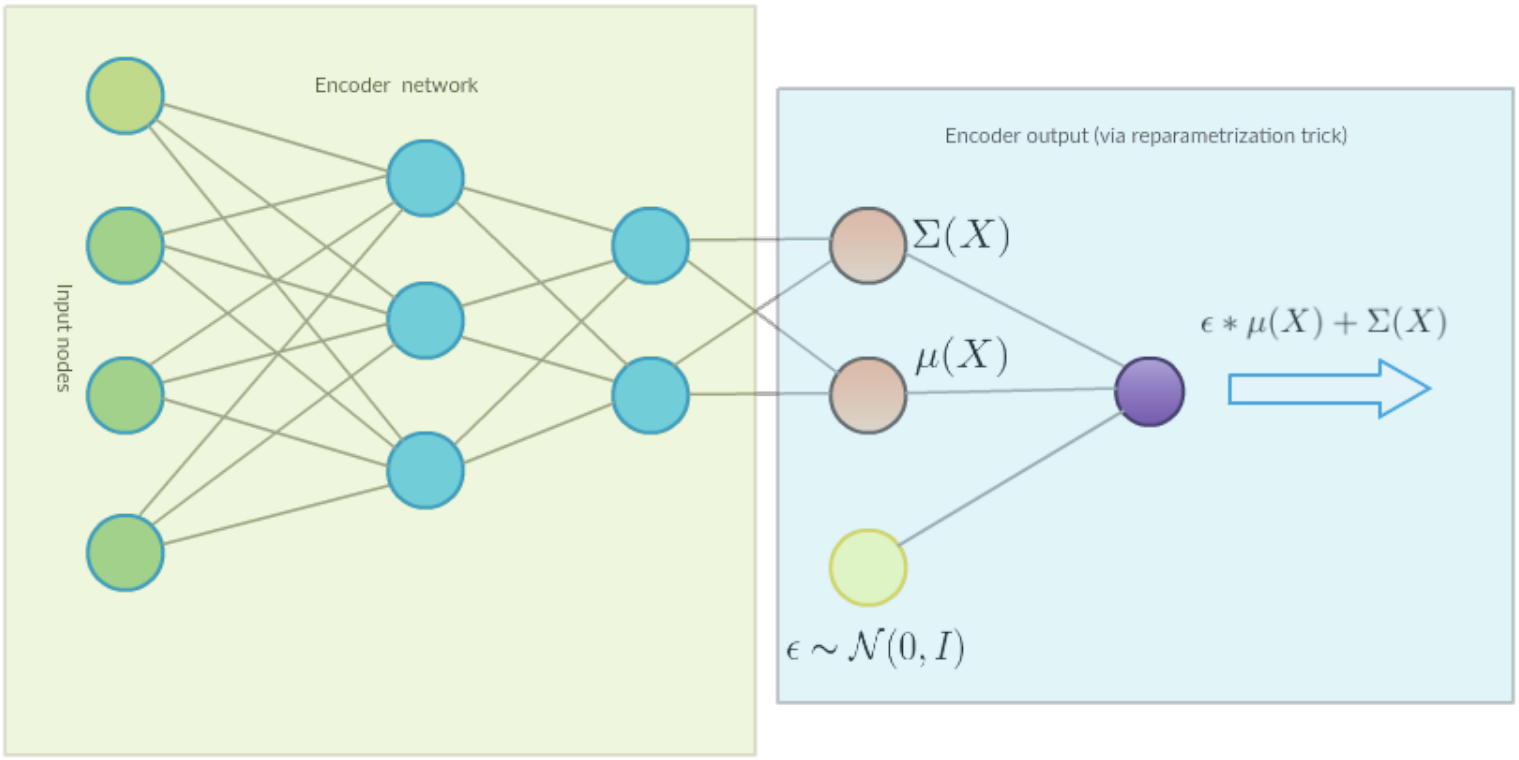

1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


30
20
10
0
−10
−20
−30
y
−30
−20
−10
0
10
20
30
x

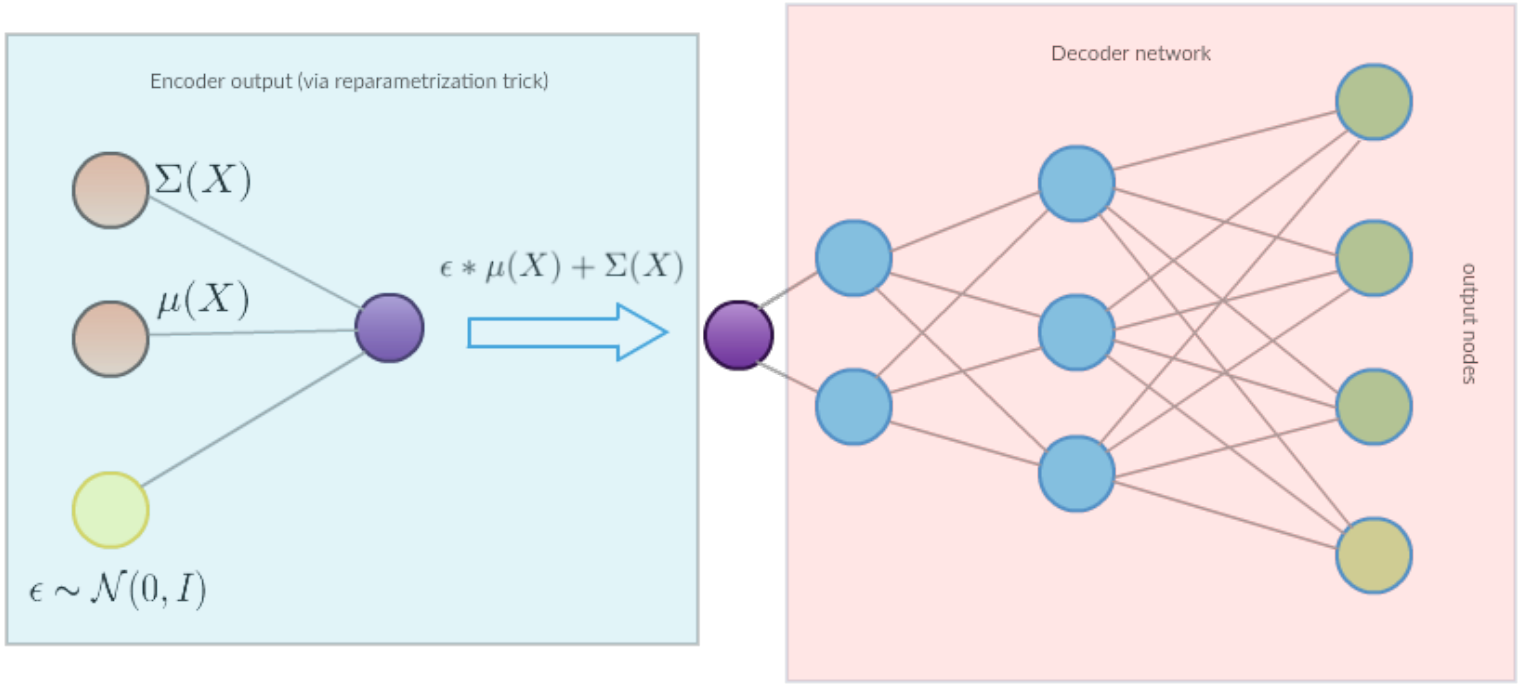
Variational Autoencoder


encode →
decode →
input
hidden
output
$q_\phi(z|x)$
$p_\theta(x|z)$
x
→
→
$\tilde{x}$

Encoder to Latent



Latent to Decoder

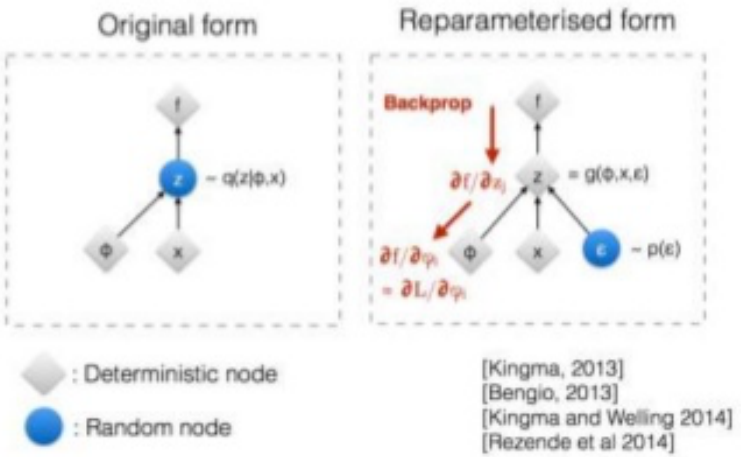


Re-Parameterization Trick

Backpropagation not possible through random sampling!

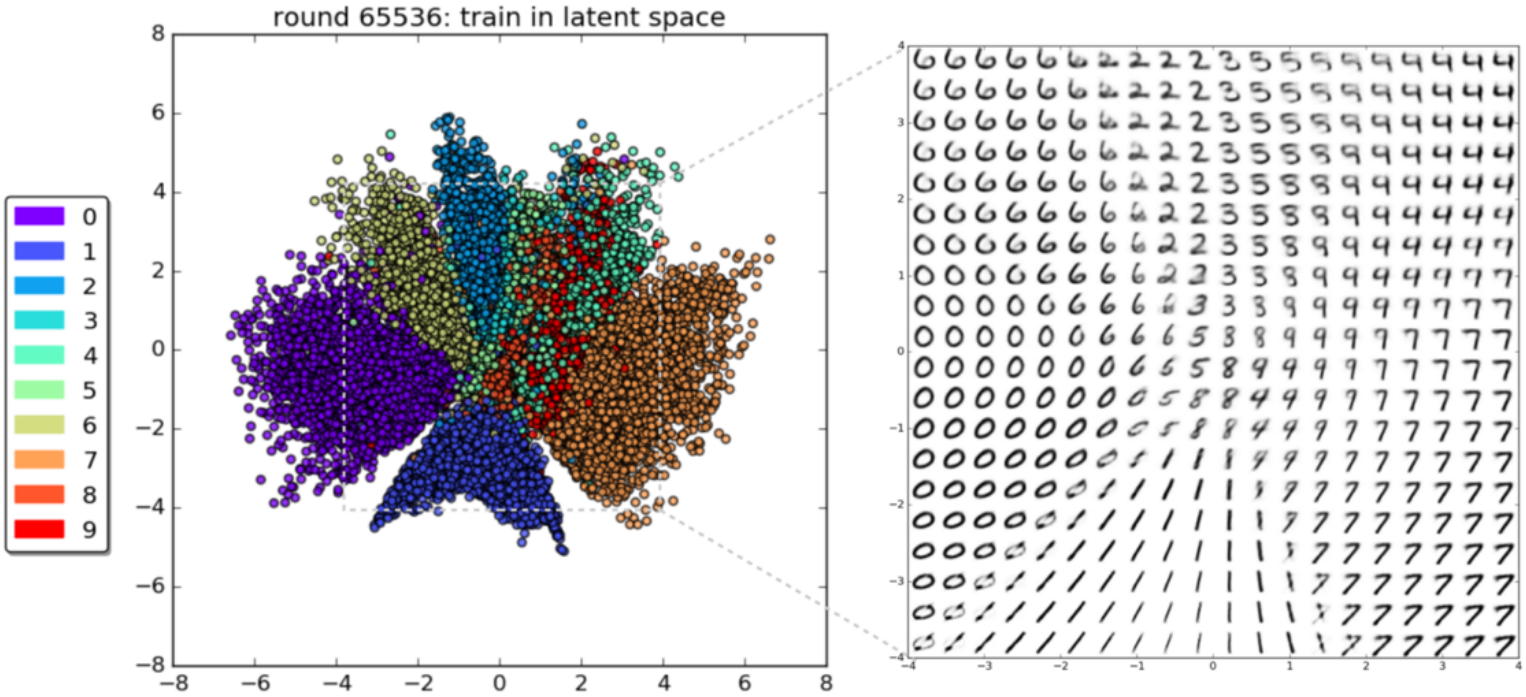
$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \varepsilon_i$$

$$\varepsilon_i \sim N(0,1)$$



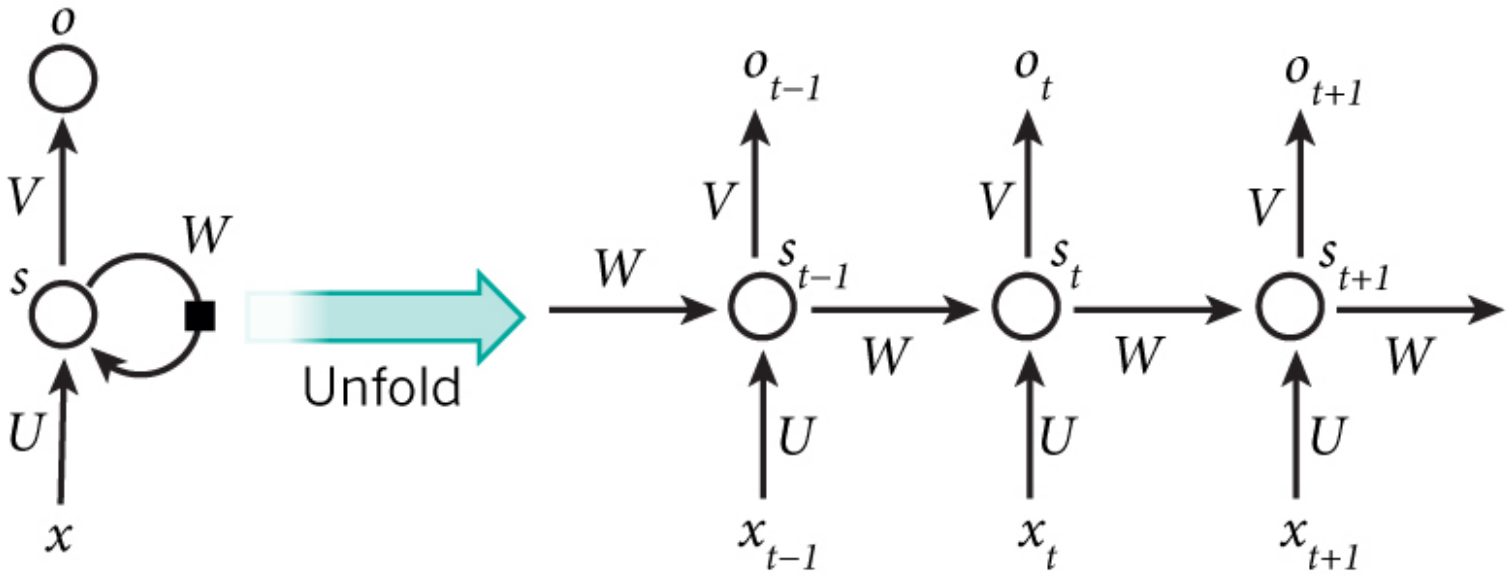
[https://arxiv.org/abs/1609.04468]

MNIST Latent Space Sampling

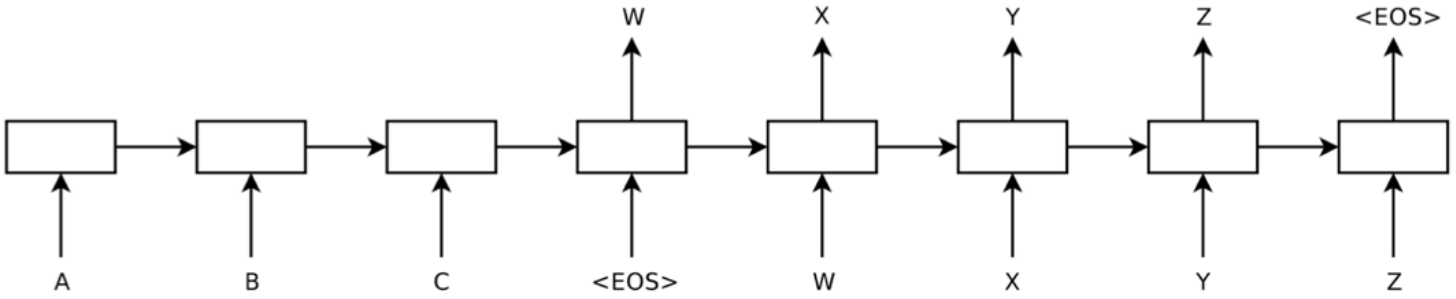


Recurrent Neural Network
Long-Short Term Memory

Recurrent Neural Network



Seq2seq Neural Networks



$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

where $(x_1, \ldots, x_T)$ is input sequence

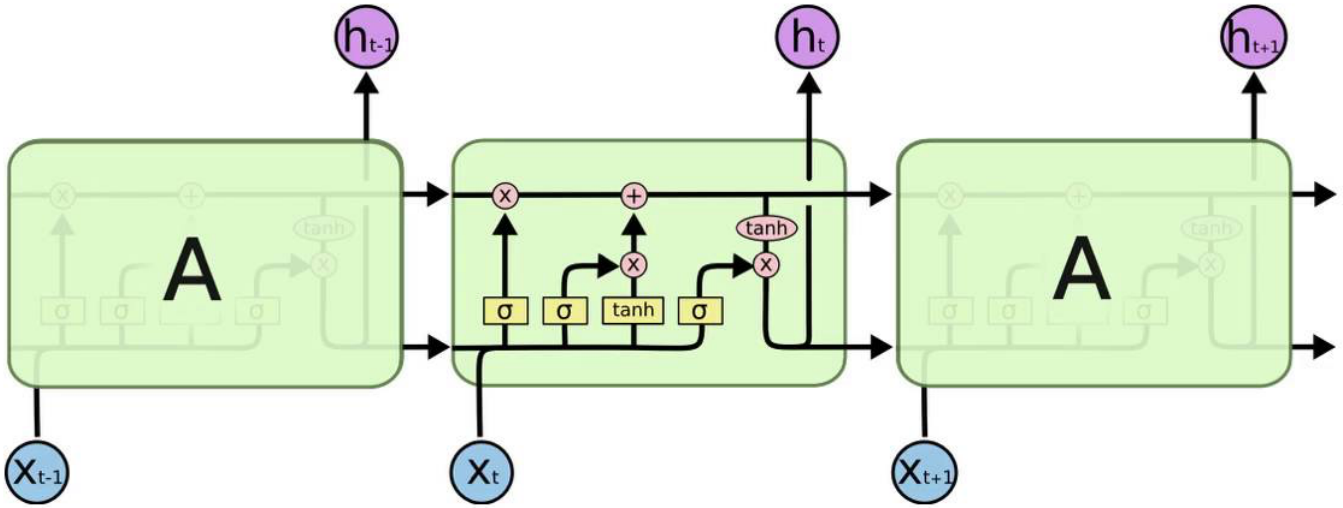
$y_1, \ldots, y_{T'}$ is corresponding output sequence

v the last hidden state of the LSTM

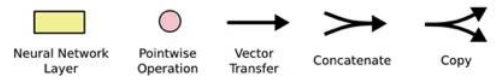
Ref: Sequence to Sequence Learning with Neural Networks; https://arxiv.org/pdf/1409.3215.pdf

The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM.

Long-Short Term Memory module: LSTM



long-short term memory modules used in an RNN



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Variational Autoencoder

+

Recurrent Neural Network

=

Generative Chemistry

Automatic Chemical Design using Variational Autoencoders

Alán Aspuru-Guzik
Harvard University

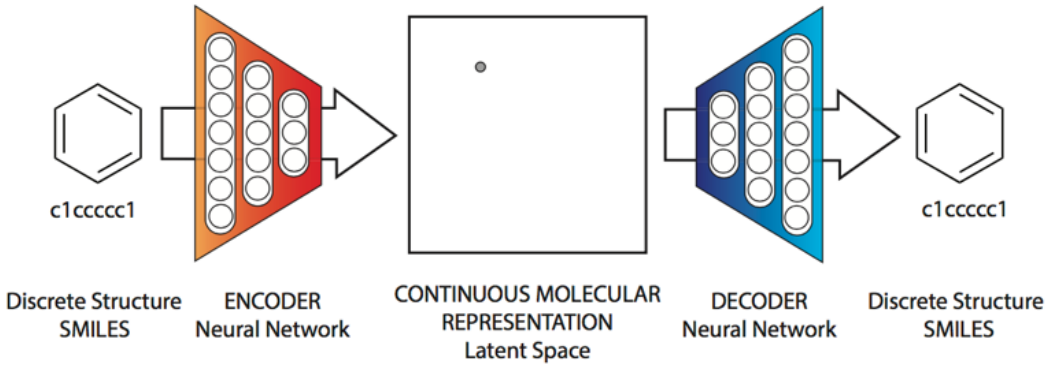
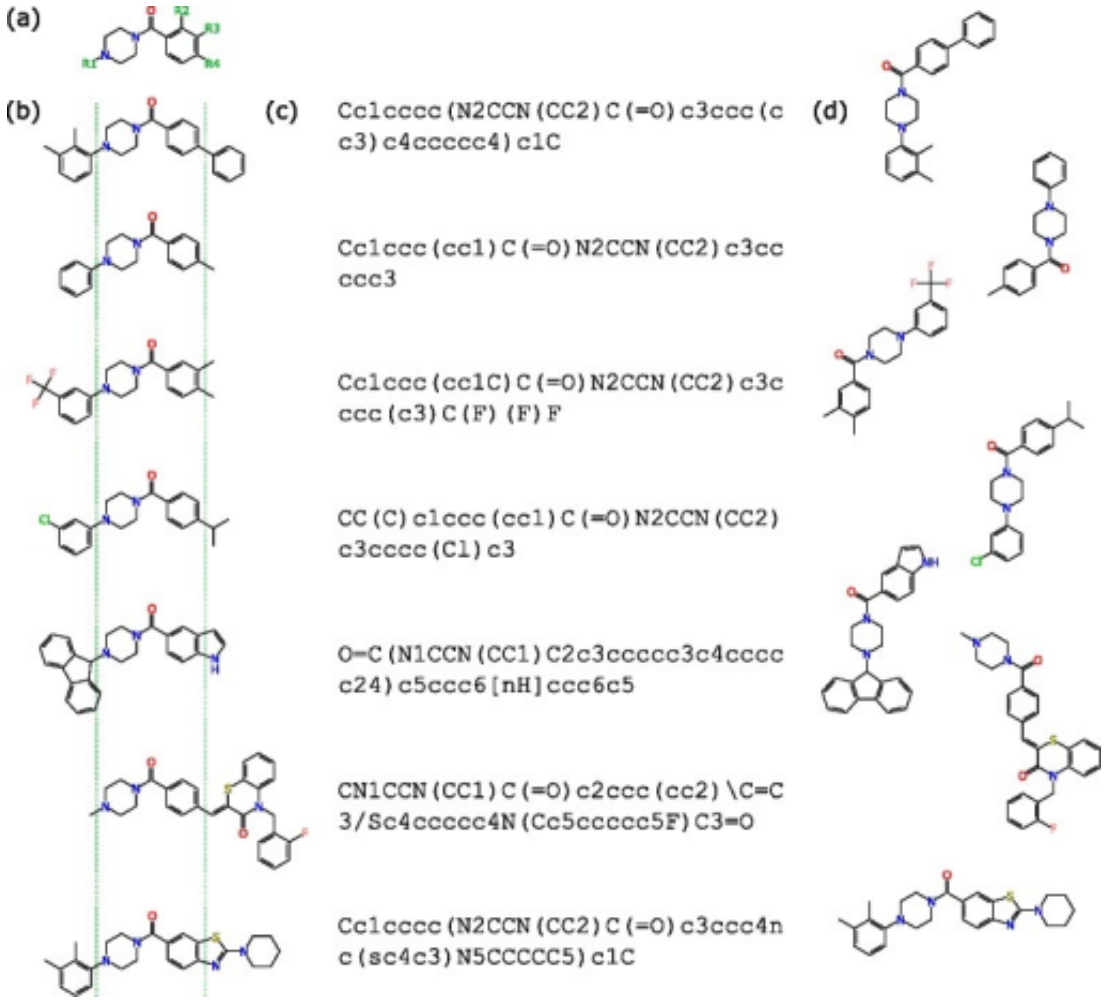


Figure 1: A diagram of the proposed autoencoder for molecular design. Starting from a discrete molecular representation, such as a SMILES string, the encoder network converts each molecule into a vector in the latent space, which is effectively a continuous molecular representation. Given a point in the latent space, the decoder network produces a corresponding SMILES string.


(a)
(b)
(c)
Cc1cccc(N2CCN(CC2)C(=O)c3ccc(cc3)c4ccccc4)c1C
Cc1ccc(cc1)C(=O)N2CCN(CC2)c3ccccc3
Cc1ccc(cc1C)C(=O)N2CCN(CC2)c3cccc(c3)C(F)(F)F
CC(C)c1ccc(cc1)C(=O)N2CCN(CC2)c3cccc(Cl)c3
O=C(N1CCN(CC1)C2c3ccccc3c4ccccc24)c5ccc6[nH]ccc6c5
CN1CCN(CC1)C(=O)c2ccc(cc2)\C=C3/Sc4ccccc4N(Cc5ccccc5F)C3=O
Cc1cccc(N2CCN(CC2)C(=O)c3ccc4nc(sc4c3)N5CCCCC5)c1C
(d)

CNN to RNN (LSTM)

- Three layers of CNN filters 9,9,11 and 9,9,10 convolutional kernels
- Two fully connected layers of size 435, 292

- Three layers of gated RNN with hidden dimension of 501

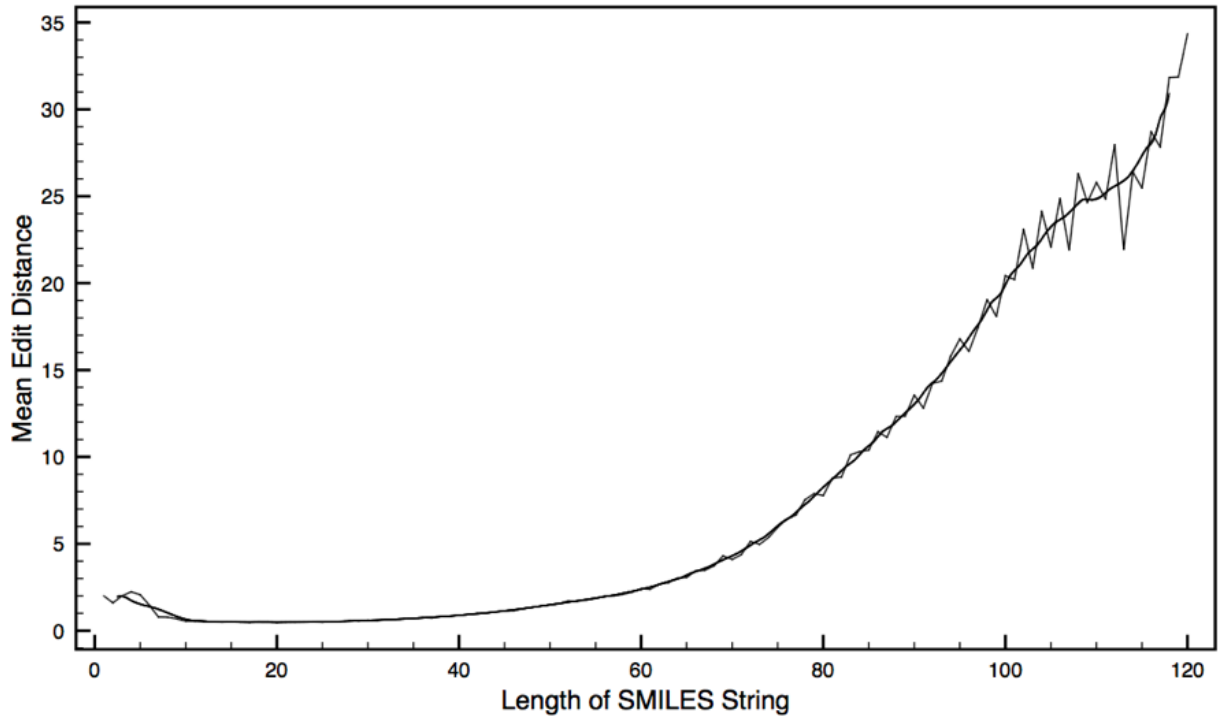
- Input SMILES upto 120 characters

Aspuru-Guzik et. al. Results

Molecular family	Autoencoder training loss	Latent dimension	Training set reconstruction %	Test set reconstruction %
drug-like	naïve	56	99.1	98.3
drug-like	variational	292	96.4	95.3
OLED	naïve	56	96.7	91.2
OLED	variational	292	91.4	79.4

Table 1: Reconstruction accuracy for the deep autoencoders used in this work. Accuracy is defined as the percentage of correct characters in decoded SMILES strings. An autoencoder with a large enough latent dimension could achieve perfect reconstruction, but exploration of the latent space tends to become more difficult as the latent dimension increases.

Error rate for the Molecular VAE



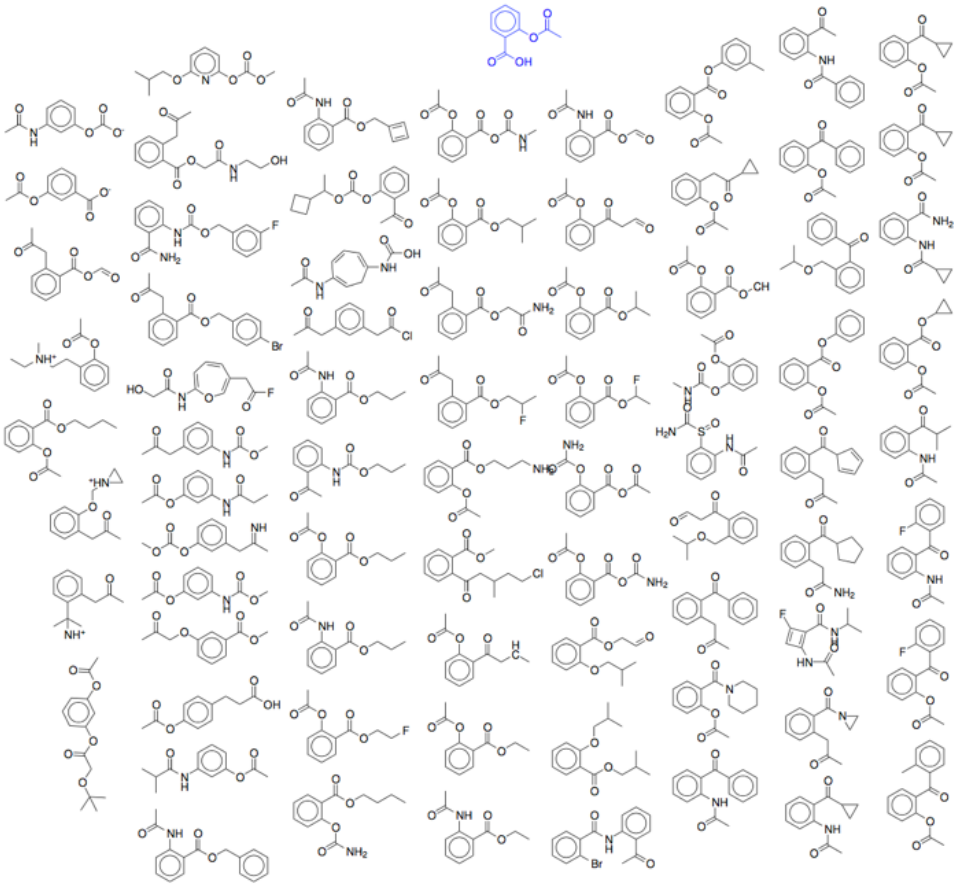


Figure 4: Molecules decoded from randomly-sampled points in the latent space of a variational autoencoder, near to a given molecule (aspirin [2-(acetyloxy)benzoic acid], highlighted in blue).

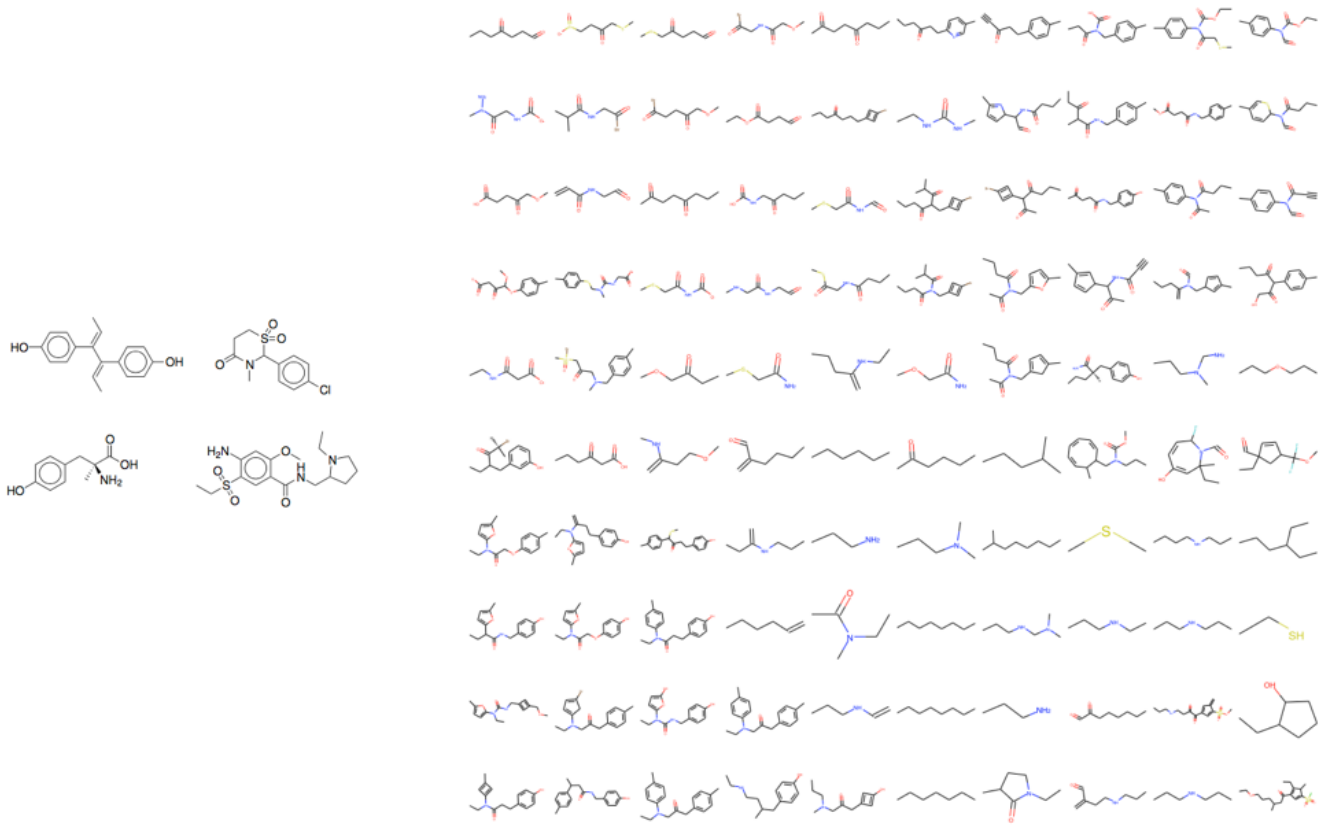
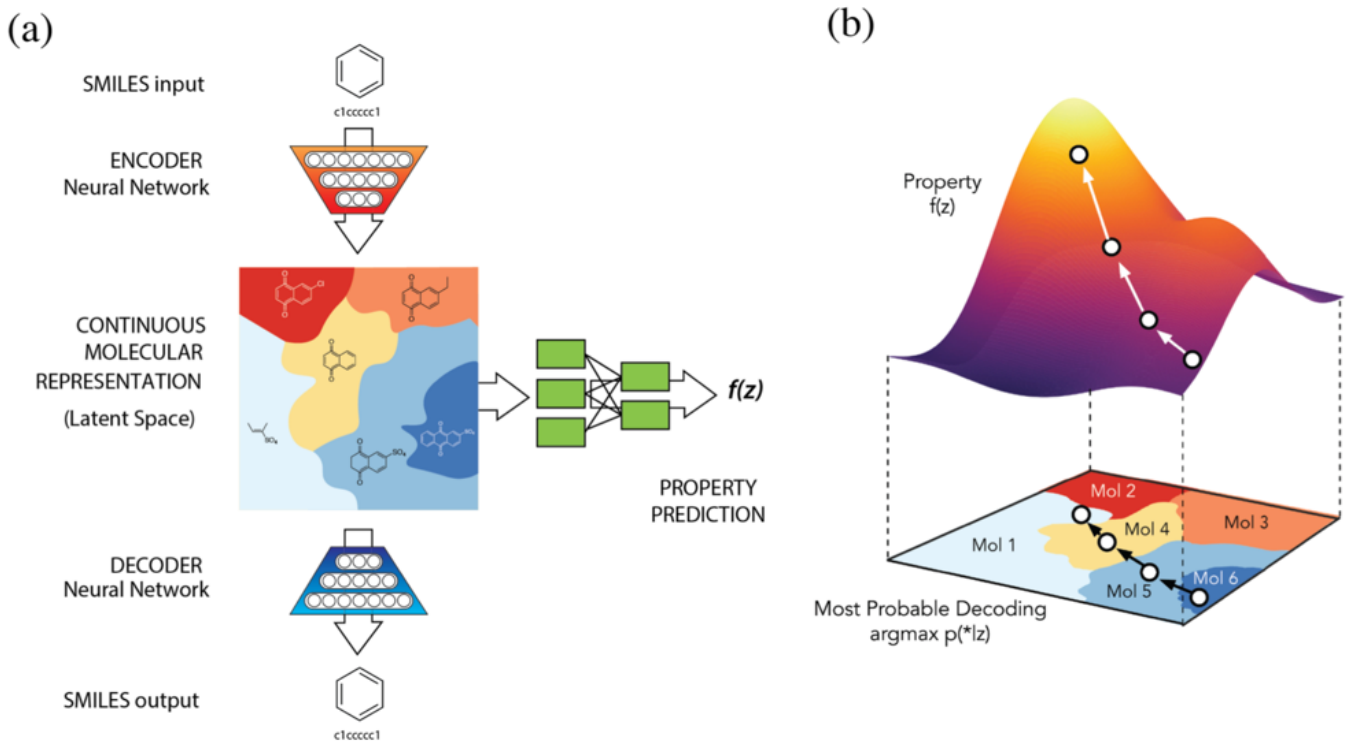


Figure 5: Interpolation. Two-dimensional interpolation between four random drugs. *Left* Starting molecules encoded, whose decodings correspond to the respective four corners of the figure to the right. *Right* Decodings of interpolating linearly between the latent representations of the four molecules to the right.

Associating Properties with Latent Space Representation



Visualization of Latent Space

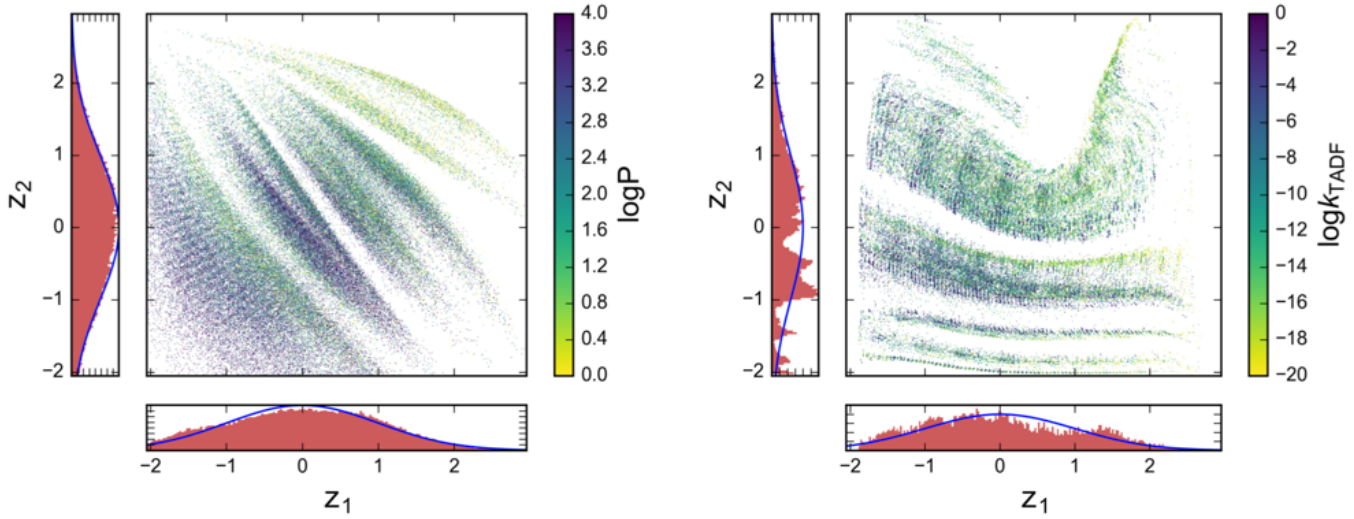


Figure 2: Projection of the molecular training sets onto learned two-dimensional latent spaces. The one-dimensional histograms show the distribution of the training data along each dimension, overlaid with the Gaussian prior imposed in the variational autoencoder. The points are colored along a chemical property that is relevant to their function, and will be the target of optimization experiments. *Left*: A natural library of drug-like molecules, colored by their predicted water-octanol partition coefficient. *Right*: A combinatorially-generated library of organic LED molecules, colored by their predicted delayed fluorescent emission rate (k_{TADF} in μs^{-1}).

Mining the Learned Representations

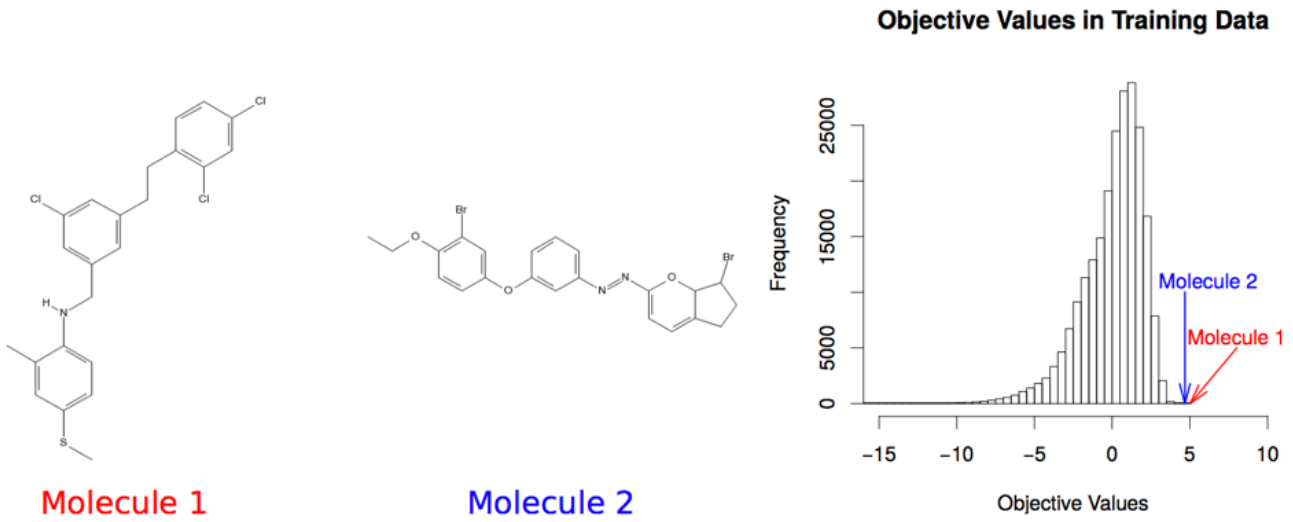
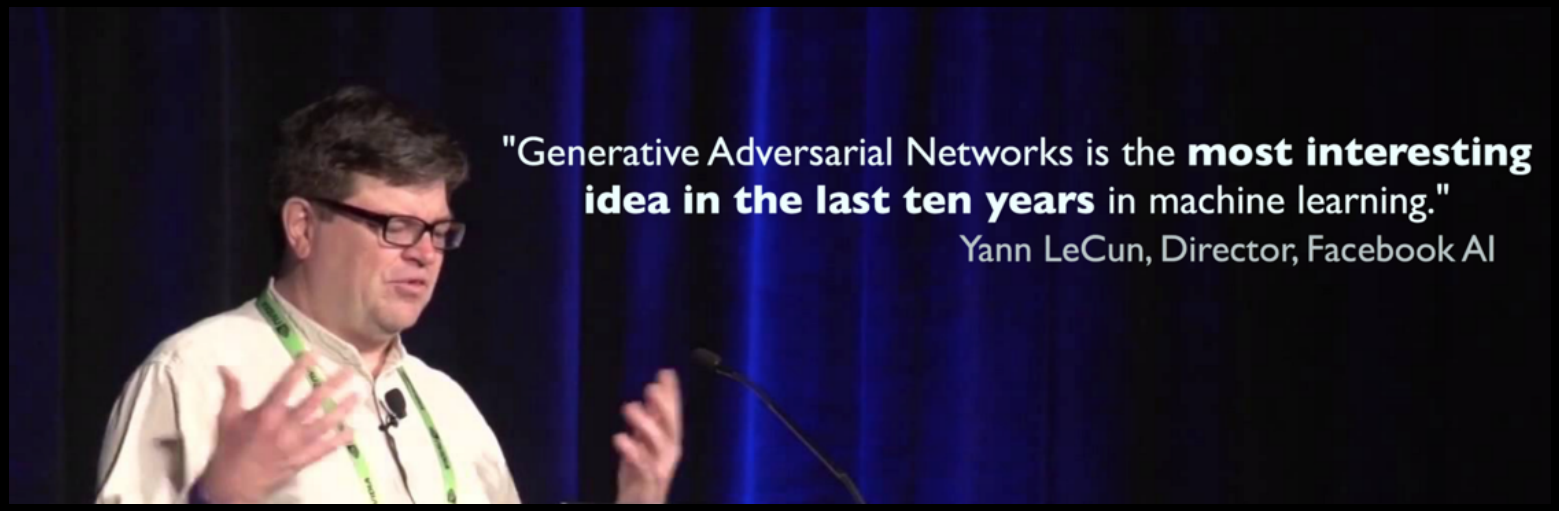


Figure 7: *Left*: Molecules generated by the optimization process with better score values than any other molecule in the training data. *Right*: Histogram of objective values in the training data.

Generative Adversarial Networks


"Generative Adversarial Networks is the **most interesting idea in the last ten years** in machine learning."
Yann LeCun, Director, Facebook AI

What are Generative Models?

Key Idea: our model cares about what distribution generated the input data points, and we want to mimic it with our probabilistic model. **Our learned model should be able to make up new samples from the distribution, not just copy and paste existing samples!**

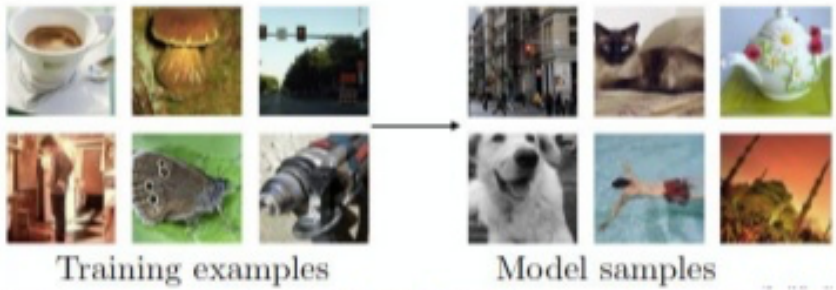
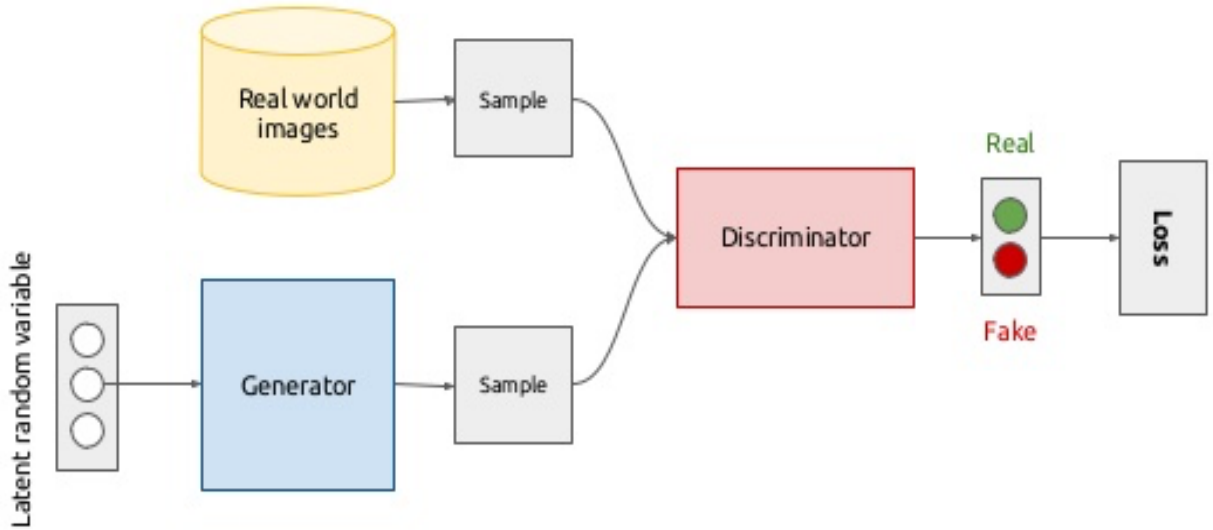
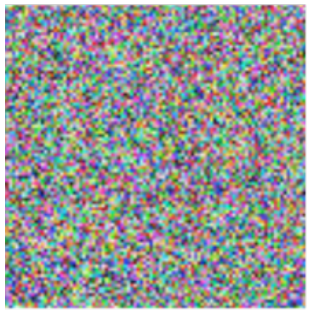


Figure from NIPS 2016 Tutorial: Generative Adversarial Networks (I. Goodfellow)

Generative adversarial networks (conceptual)

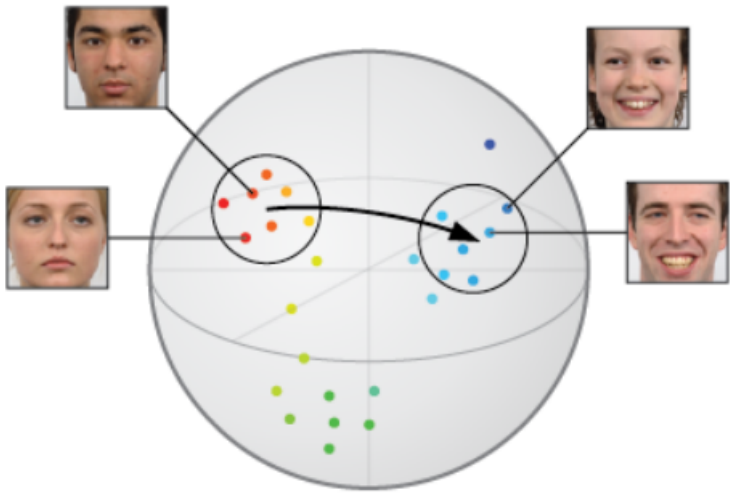


Noise ~ N(0,1)

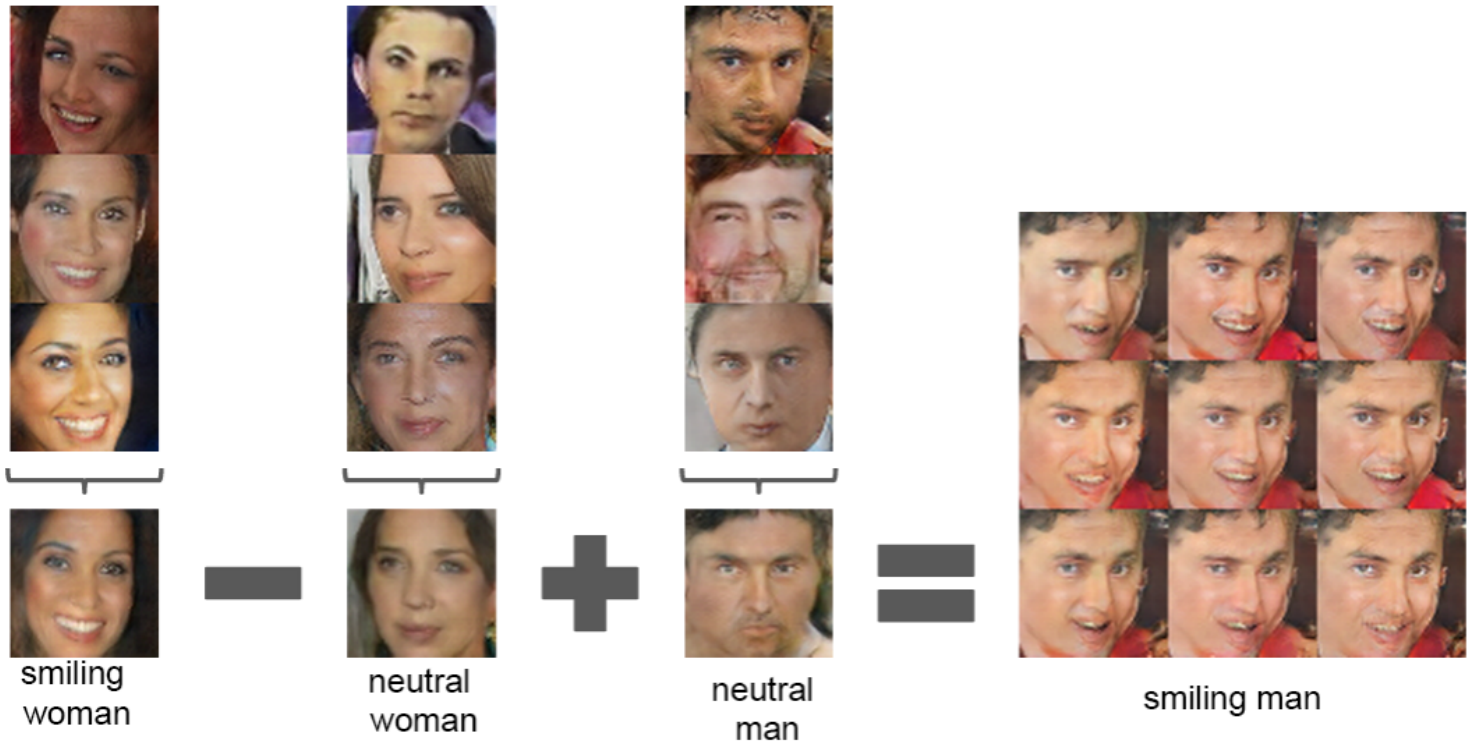


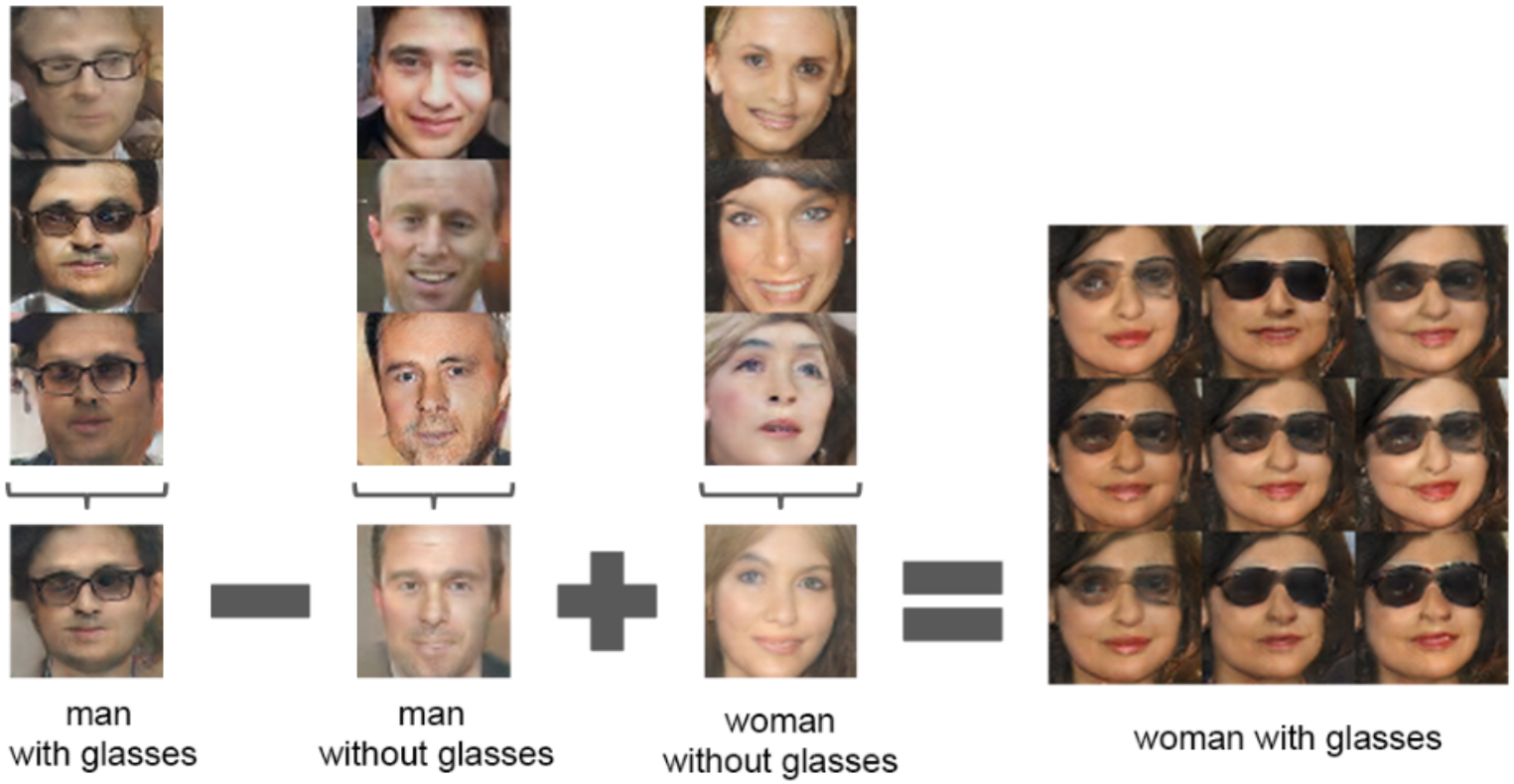
Generative Model

If you do it right!

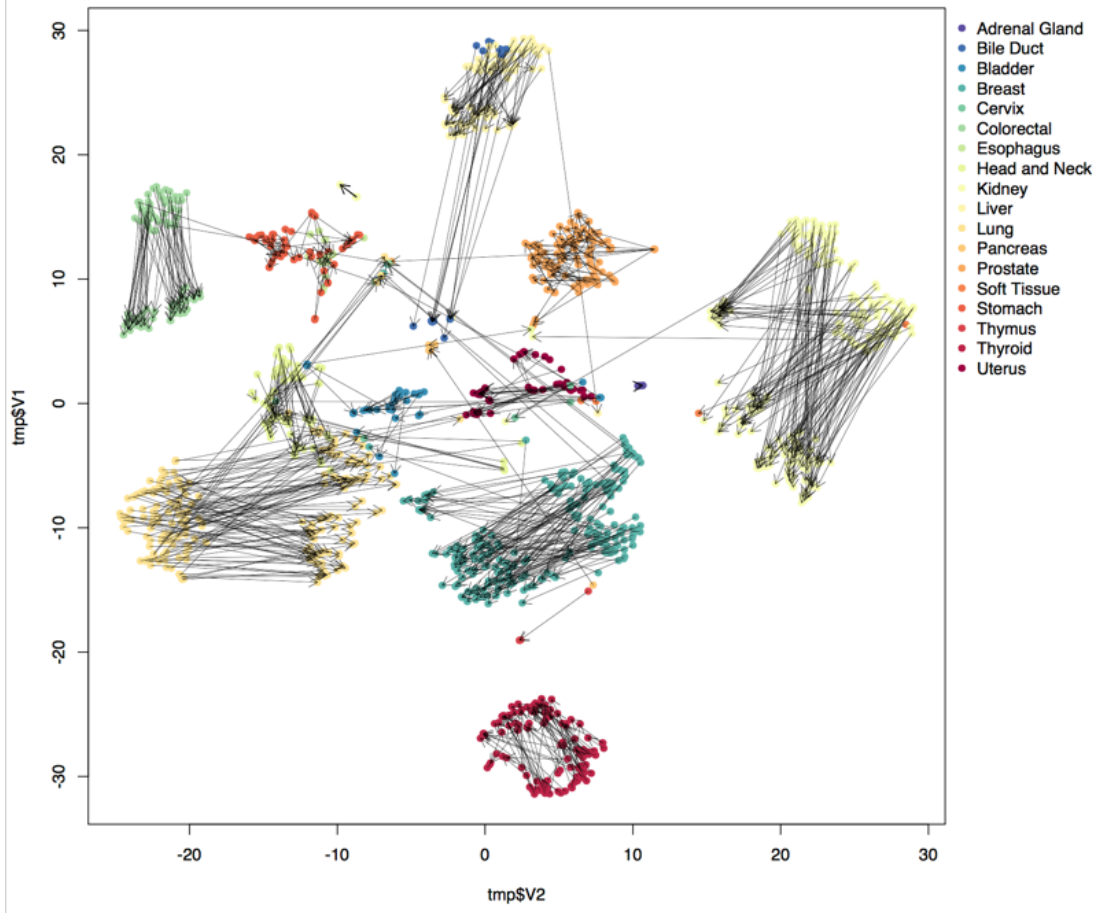


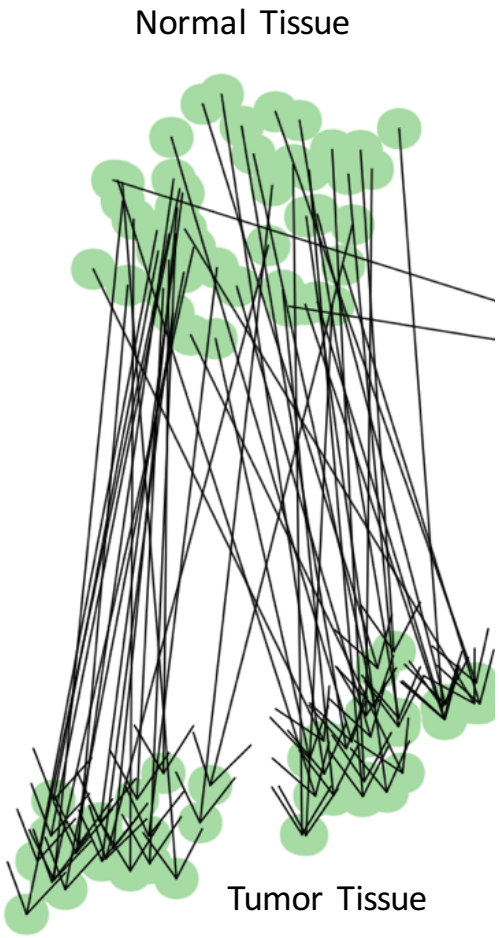
Arithmetic in the Latent Vector Space


smiling woman
neutral woman
neutral man
smiling man


man
with glasses
man
without glasses
woman
without glasses
woman with glasses

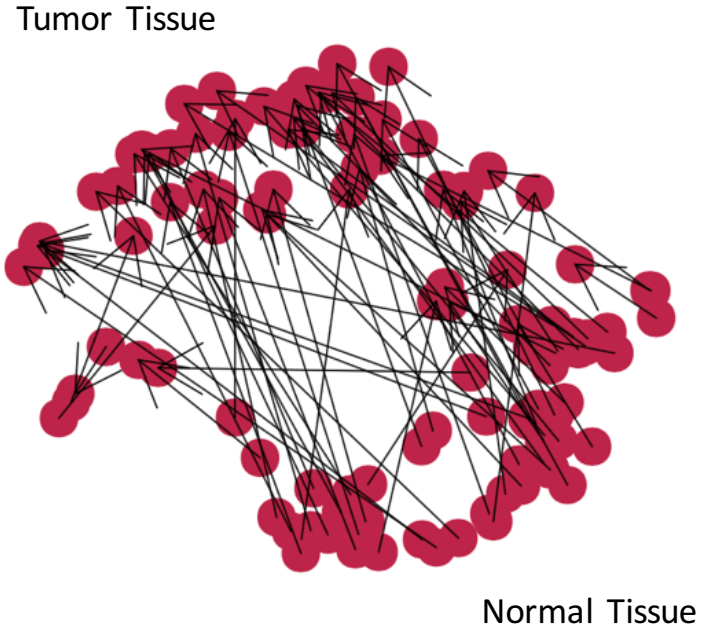
t-sne Plot of Matched Normal Pairs Showing Translation in Features Space





Colon-Rectal

Uterus



Cycle Consistent Generative Adversarial Networks

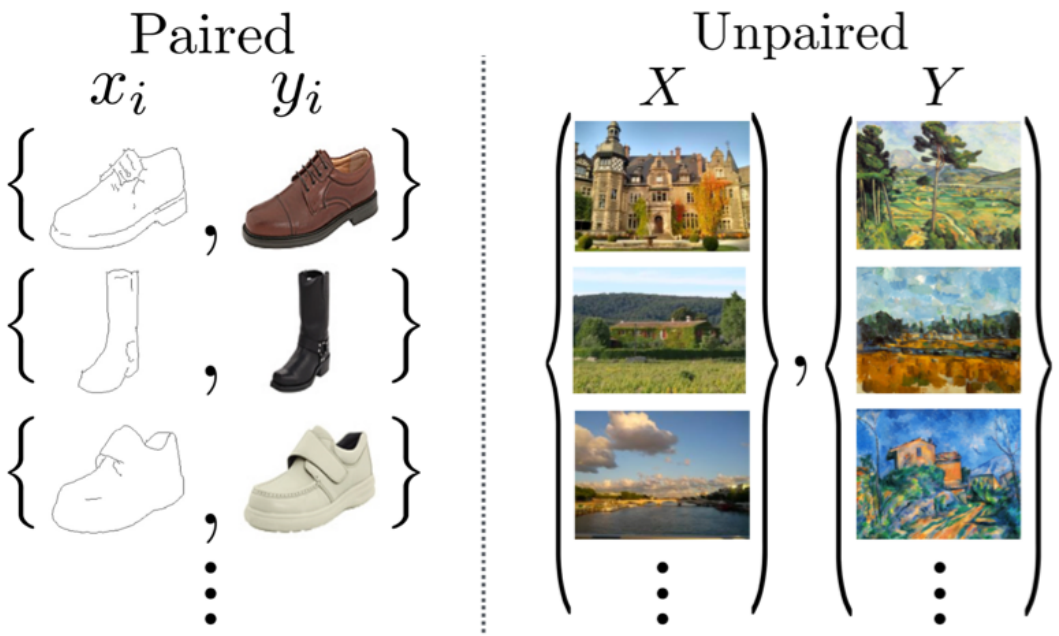


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [21]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

Two Mapping Functions $G: X \rightarrow Y$ and $F: Y \rightarrow X$ and discriminators D_Y and D_X

$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$

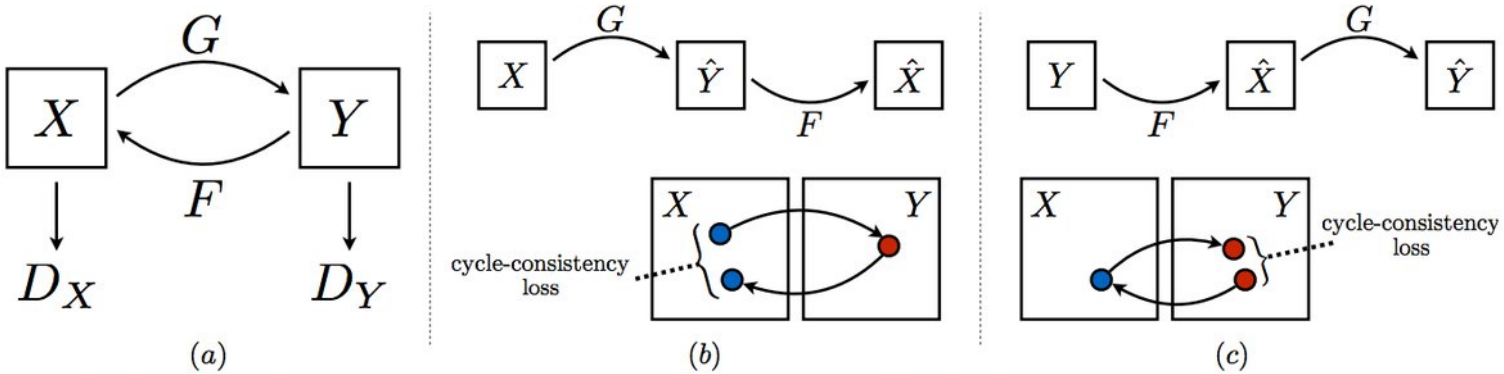
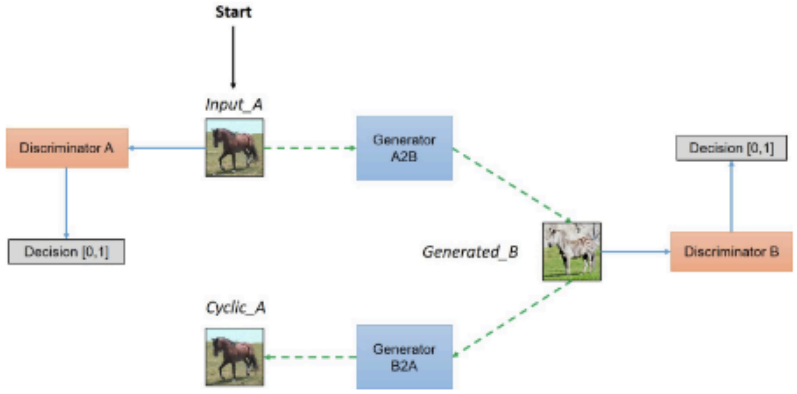
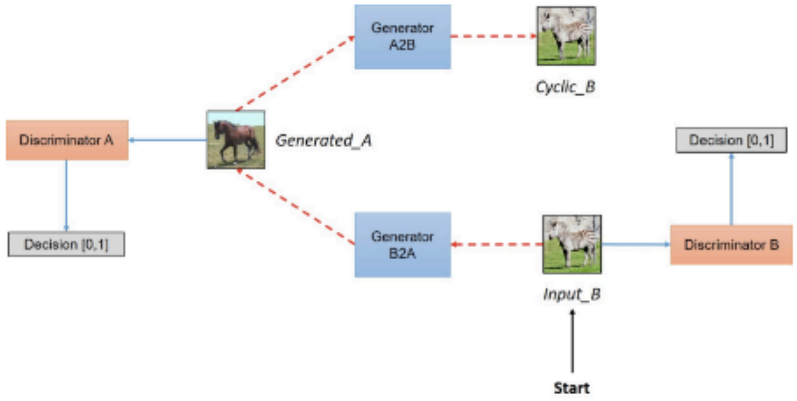


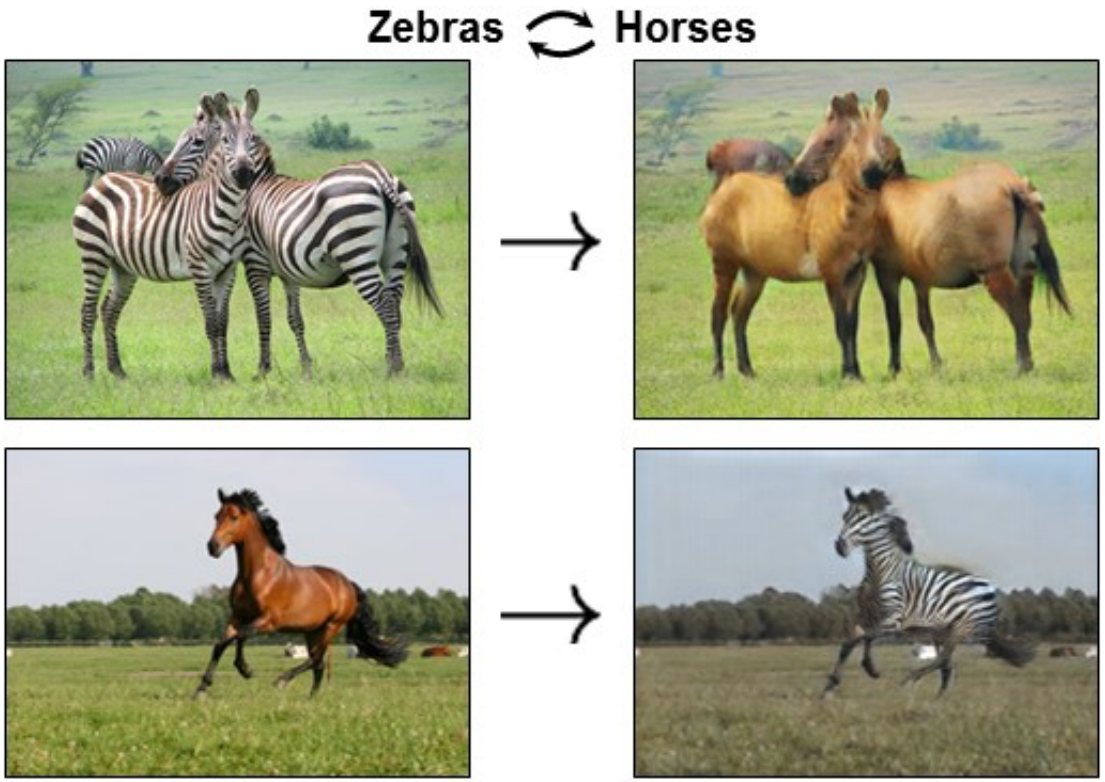
Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X, F, and X. To further regularize the mappings, we introduce two "cycle consistency losses" that capture the intuition that if we translate from one domain to the other and back again we should arrive where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Network Architecture





Simplified view of CycleGAN architecture


Zebras
Horses


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Combination Generative Adversarial Networks

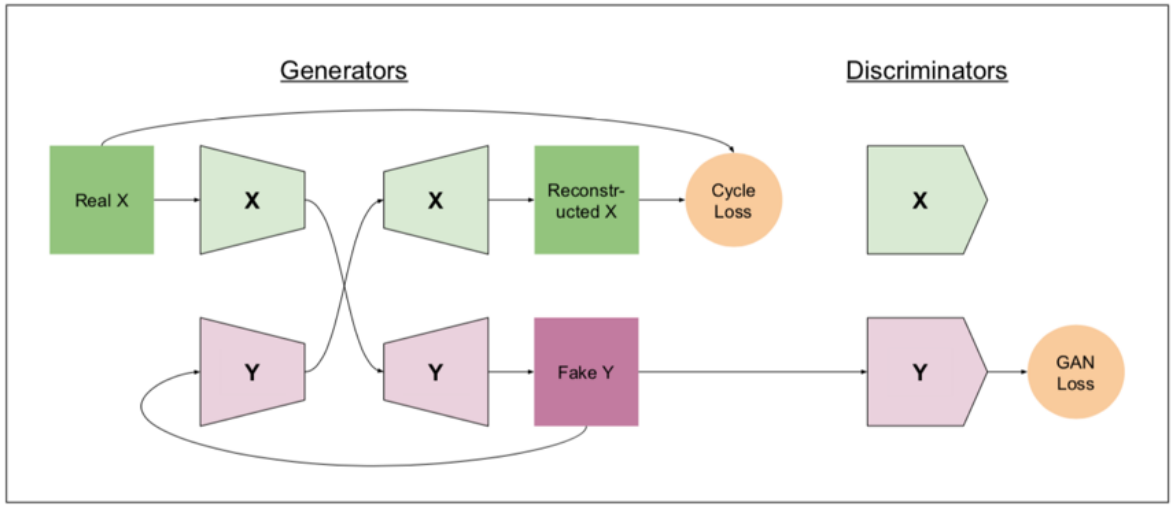


Figure 2. Generator training pass for direction $X \rightarrow Y$, where $X, Y \in \{1, .., n\} : X \neq Y$ are randomly chosen from our n domains at the start of every iteration. This pass is always repeated symmetrically for direction $Y \rightarrow X$ as well.

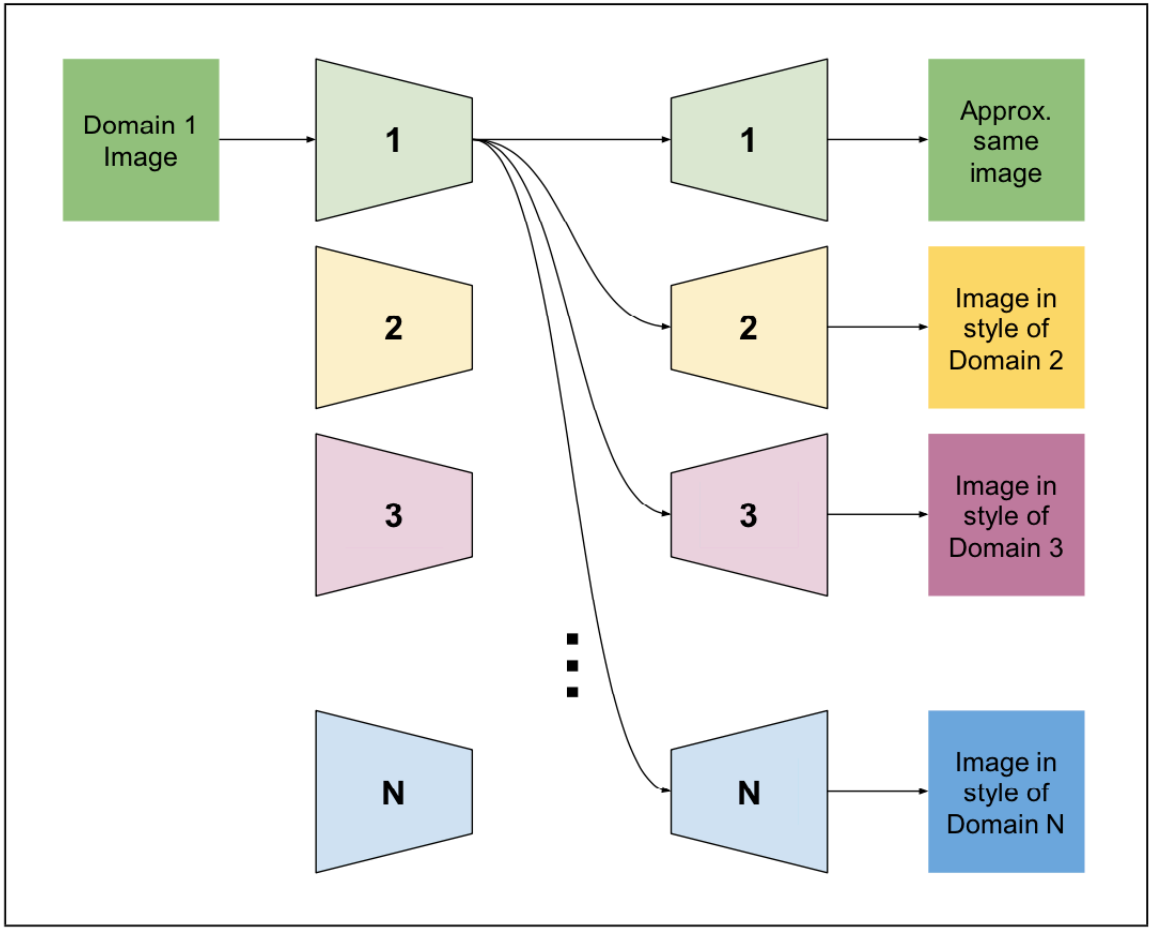


Figure 3. Example inference functionality of translation from one domain to all others.

Figure 4. Validation results for pictures of the Alps in all four seasons. Original images lie on the diagonal.

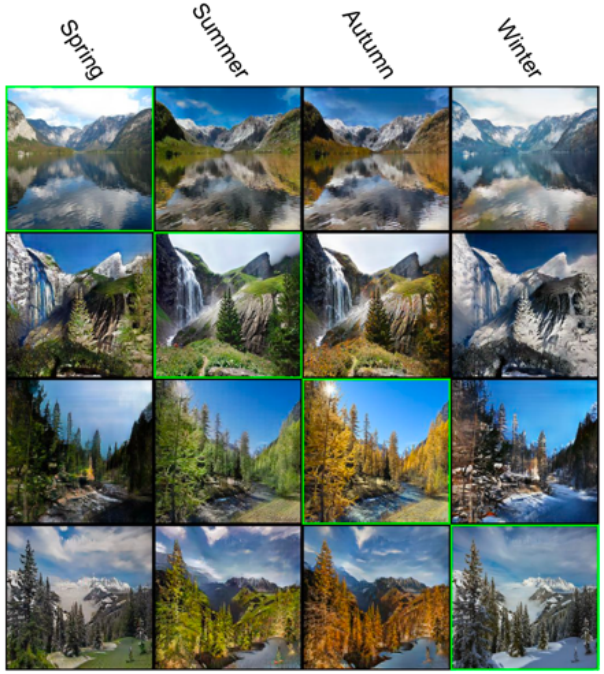
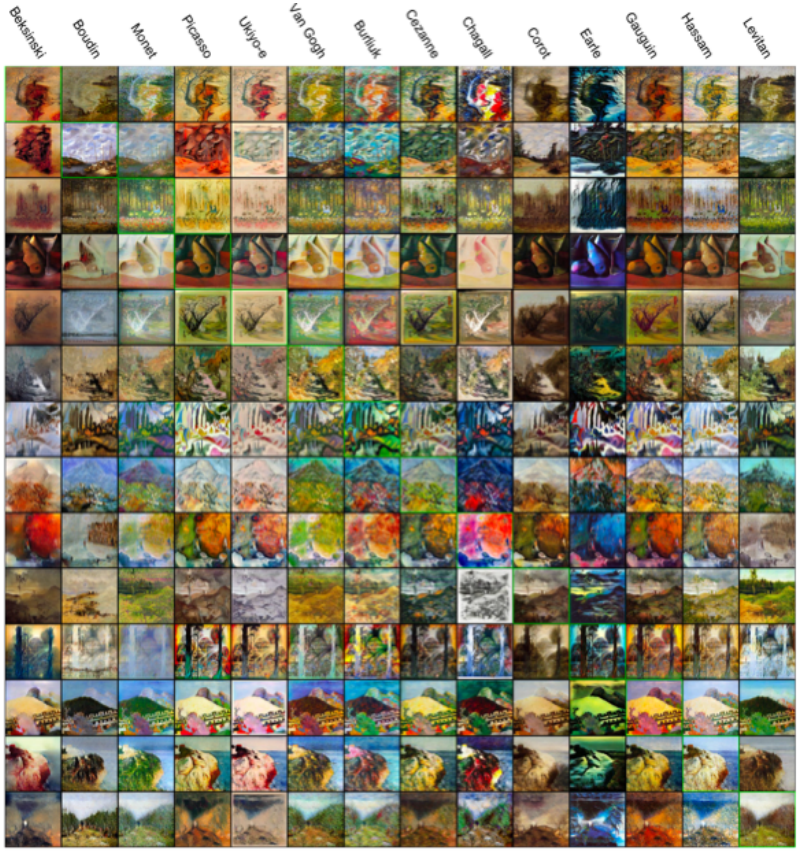
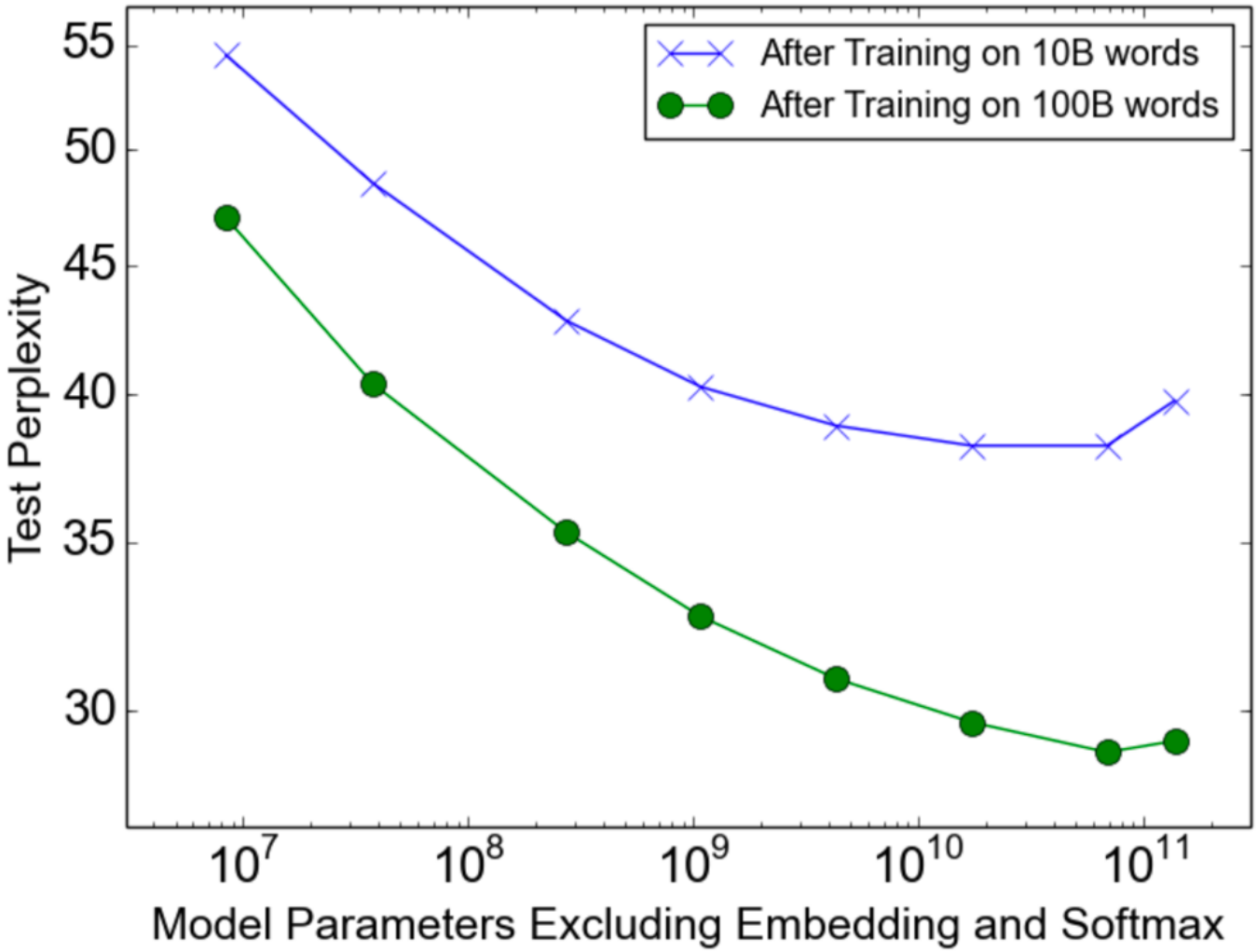


Figure 5. Same Alps images but from standard CycleGAN results instead. Original images lie on the diagonal.


Beksinski
Boudin
Monet
Picasso
Ukiyo-e
Van Gogh
Burliuk
Cezanne
Chagall
Corot
Earle
Gauguin
Hassam
Levitan

Really Large Networks
Multimodal Networks
Multitask Networks


After Training on 10B words
After Training on 100B words
Test Perplexity
55
50
45
40
35
30
10^7
10^8
10^9
10^{10}
10^{11}
Model Parameters Excluding Embedding and Softmax

1000x Model Capacity, 137 Billion Parameters

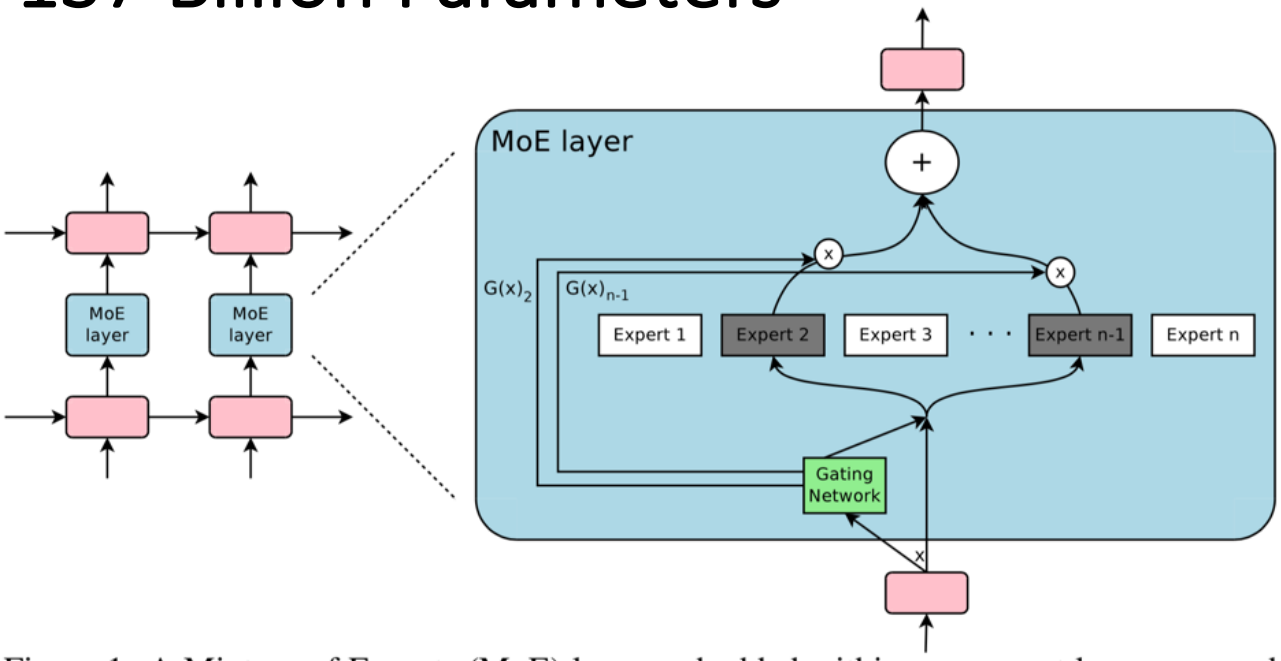


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

OUTRAGEOUSLY LARGE NEURAL NETWORKS:
THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

Can we create a unified deep learning model to solve tasks across multiple domains?

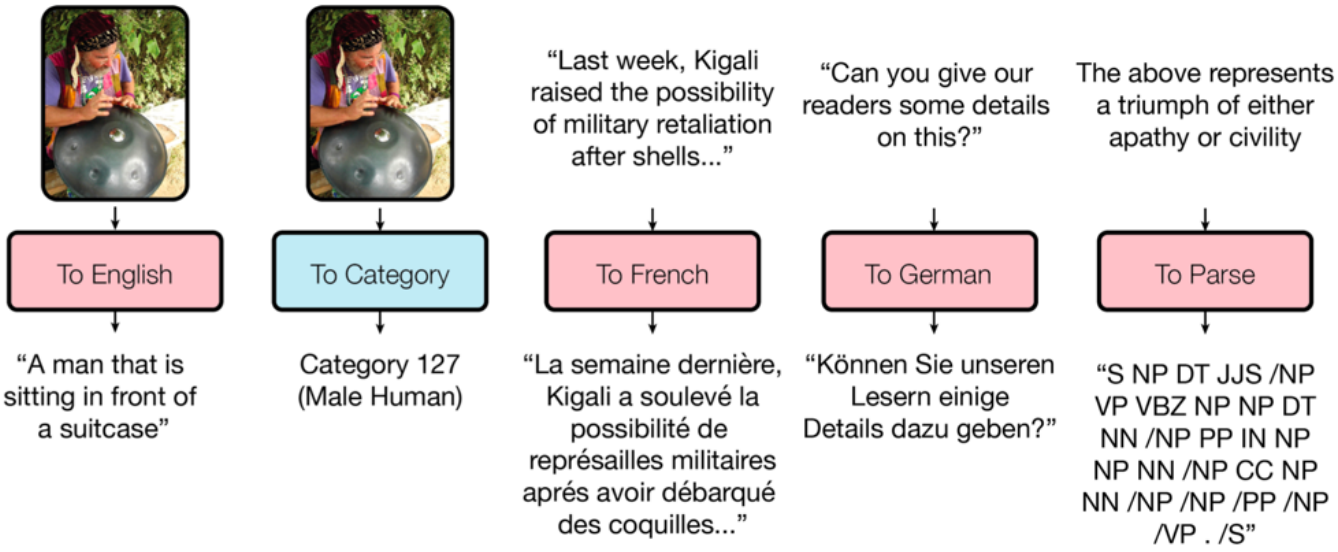


Figure 1: Examples decoded from a single MultiModel trained jointly on 8 tasks. Red depicts a language modality while blue depicts a categorical modality.

One Model To Learn Them All

Aggregating Blocks with Gates

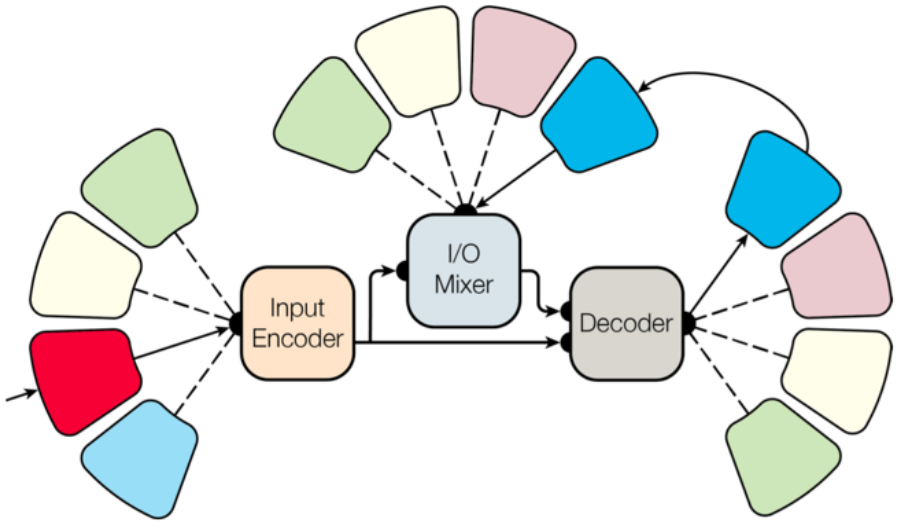


Figure 2: The MultiModel, with modality-nets, an encoder, and an autoregressive decoder.

“This leads us to conclude that mixing different computation blocks is in fact a good way to improve performance on many various tasks.”

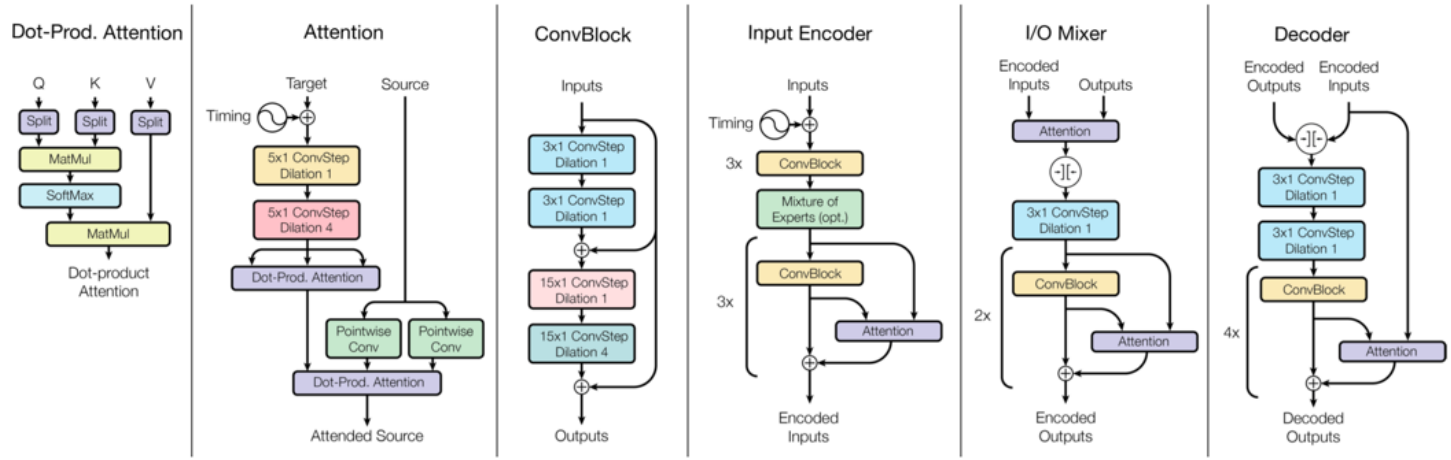


Figure 3: Architecture of the MultiModel; see text for details.

Problem	Alone			W/ ImageNet			W/ 8 Problems		
	log(ppl)	acc.	full	log(ppl)	acc.	full	log(ppl)	acc.	full
Parsing	0.20	97.1%	11.7%	0.16	97.5%	12.7%	0.15	97.9%	14.5%

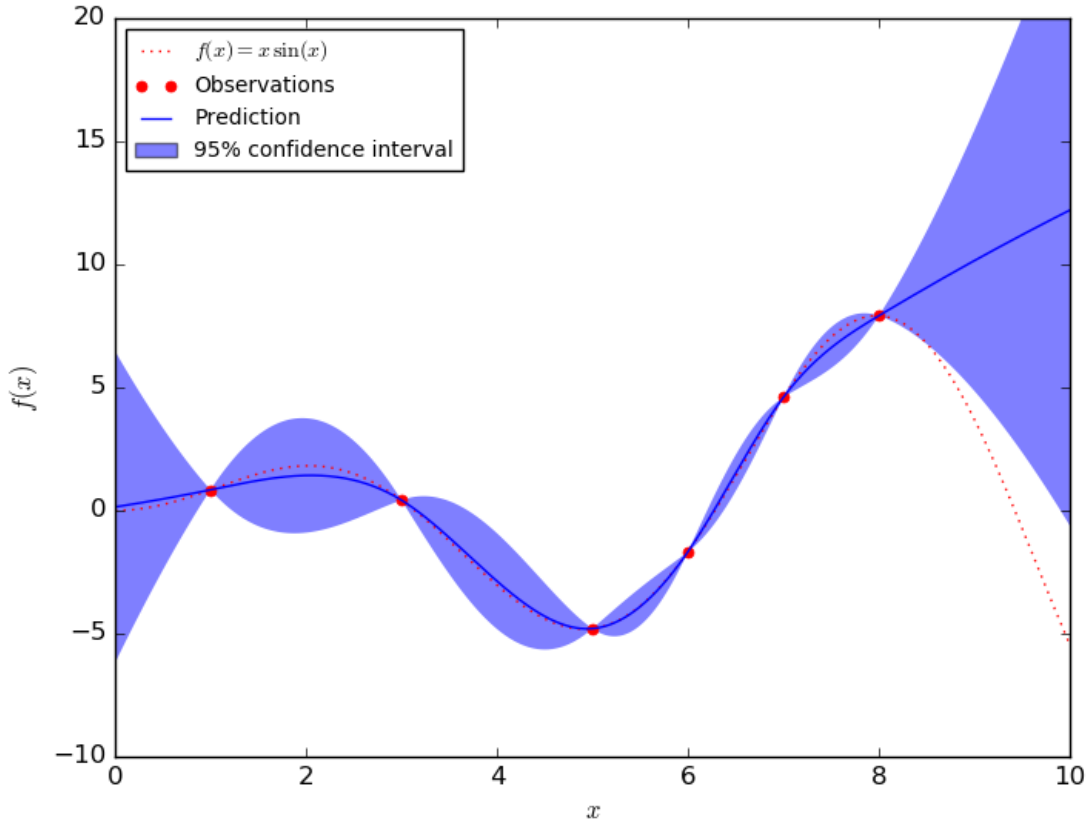
Table 3: Results on training parsing alone, with ImageNet, and with 8 other tasks. We report log-perplexity, per-token accuracy, and the percentage of fully correct parse trees.

Problem	All Blocks		Without MoE		Without Attention	
	log(perpexity)	accuracy	log(perplexity)	accuracy	log(perplexity)	accuracy
ImageNet	1.6	67%	1.6	66%	1.6	67%
WMT EN→FR	1.2	76%	1.3	74%	1.4	72%

Table 4: Ablating mixture-of-experts and attention from MultiModel training.

Deep Learning Uncertainty Quantification

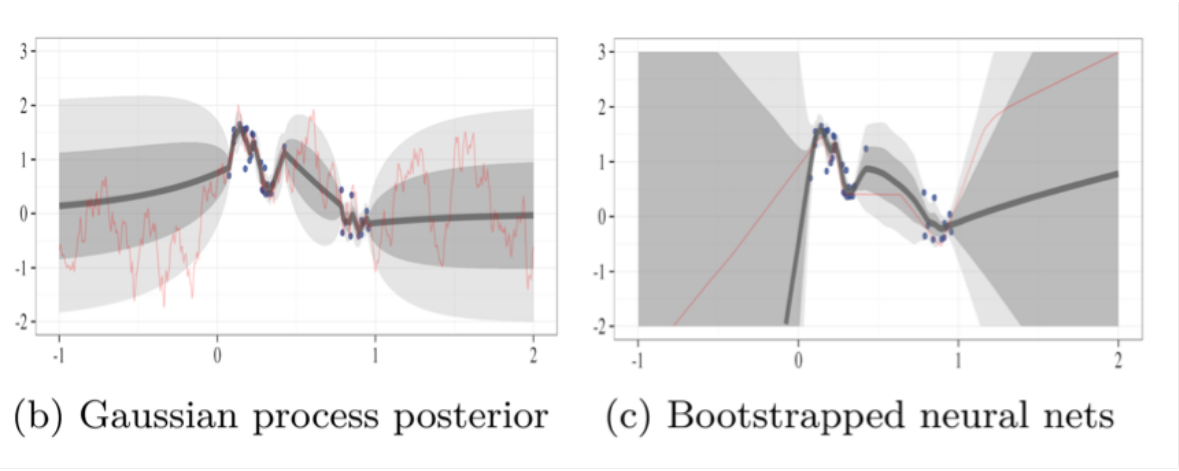
Intuition behind UQ



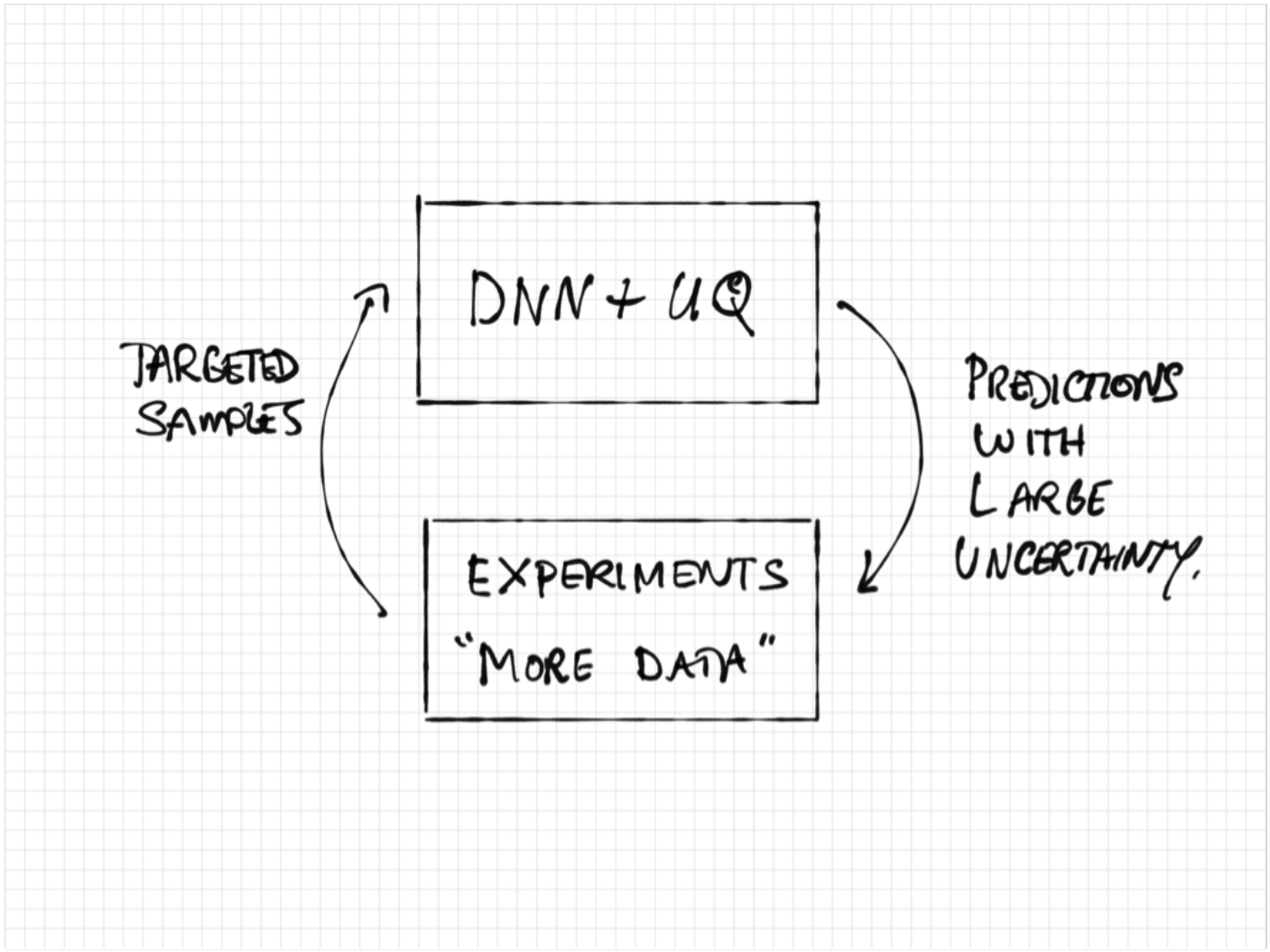
Three Approaches to DL UQ:

- **Train on distributions and predict distributions**
- **Bootstrap with ensembles with optional data augmentation**
- **Dropout as a Bayesian approximation**

Bootstrapping UQ in Deep Neural Networks



(b) Gaussian process posterior (c) Bootstrapped neural nets


DNN + UQ
TARGETED SAMPLES
PREDICTIONS WITH LARGE UNCERTAINTY.
EXPERIMENTS
"MORE DATA"

Artificial Intelligence

Google's New AI Is Better at Creating AI Than the Company's Engineers

Shutterstock / Denis Linine

IN BRIEF

At its I/O '17 conference this week, Google shared details of its AutoML project, an artificial intelligence that can assist in the creation of other AIs. By automating some of the complicated process, AutoML could make machine learning more accessible to non-experts.

GOOGLE'S AUTOML

One of the more noteworthy remarks to come out of Google I/O '17 conference this week was CEO Sundar Pichai recalling how his team had joked that they have achieved "AI inception" with AutoML. Instead of crafting layers of dreams like in the Christopher Nolan flick, however, the AutoML system layers artificial intelligence (AI), with AI systems creating better AI systems.

SHARE

WRITTEN BY

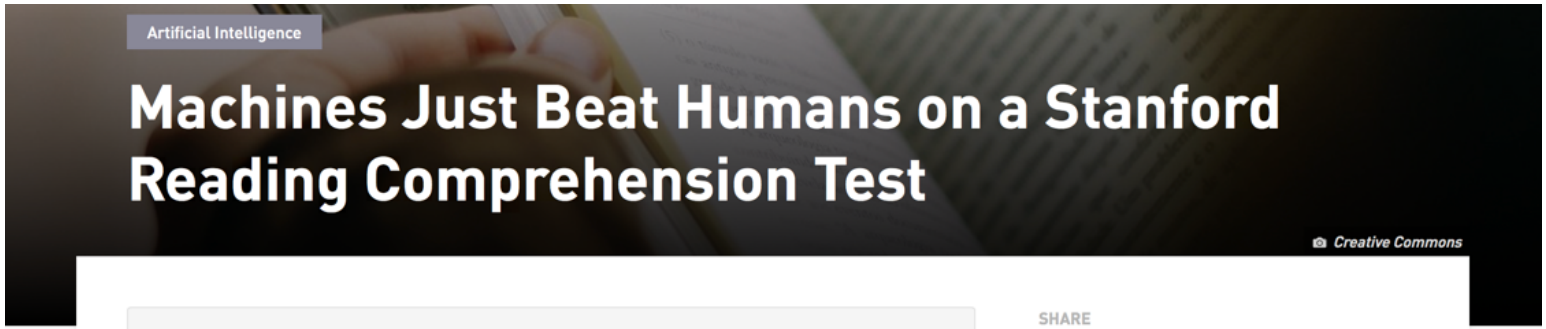
AUTHOR
Tom Ward

EDITOR
Kristin Houser
Website

Published: May 19, 2017

Last updated: May 19, 2017 at 12:29 pm

#artificial intelligence #deep learning #Google #machine learning



Artificial Intelligence

Machines Just Beat Humans on a Stanford Reading Comprehension Test

Creative Commons

IN BRIEF

The Stanford Question Answering Dataset is a well-respected means of testing machine reading. For the first time, an artificial intelligence has scored higher than a human participant.

READ ME

Chinese retail giant Alibaba has developed an artificial intelligence model that's managed to outdo human participants in a reading and comprehension test designed by Stanford University. The model scored 82.44, whereas humans recorded a score of 82.304.

The Stanford Question Answering Dataset is a set of 10,000 questions pertaining to some 500 Wikipedia articles. The answer to each question is a particular span of text from the corresponding piece of writing.

Alibaba claims that its accomplishment is the first time that humans have been outmatched on this particular test, according to a report from Bloomberg. Microsoft also managed a similar feat, scoring 82.650 — though, those results were finalized shortly after Alibaba's.

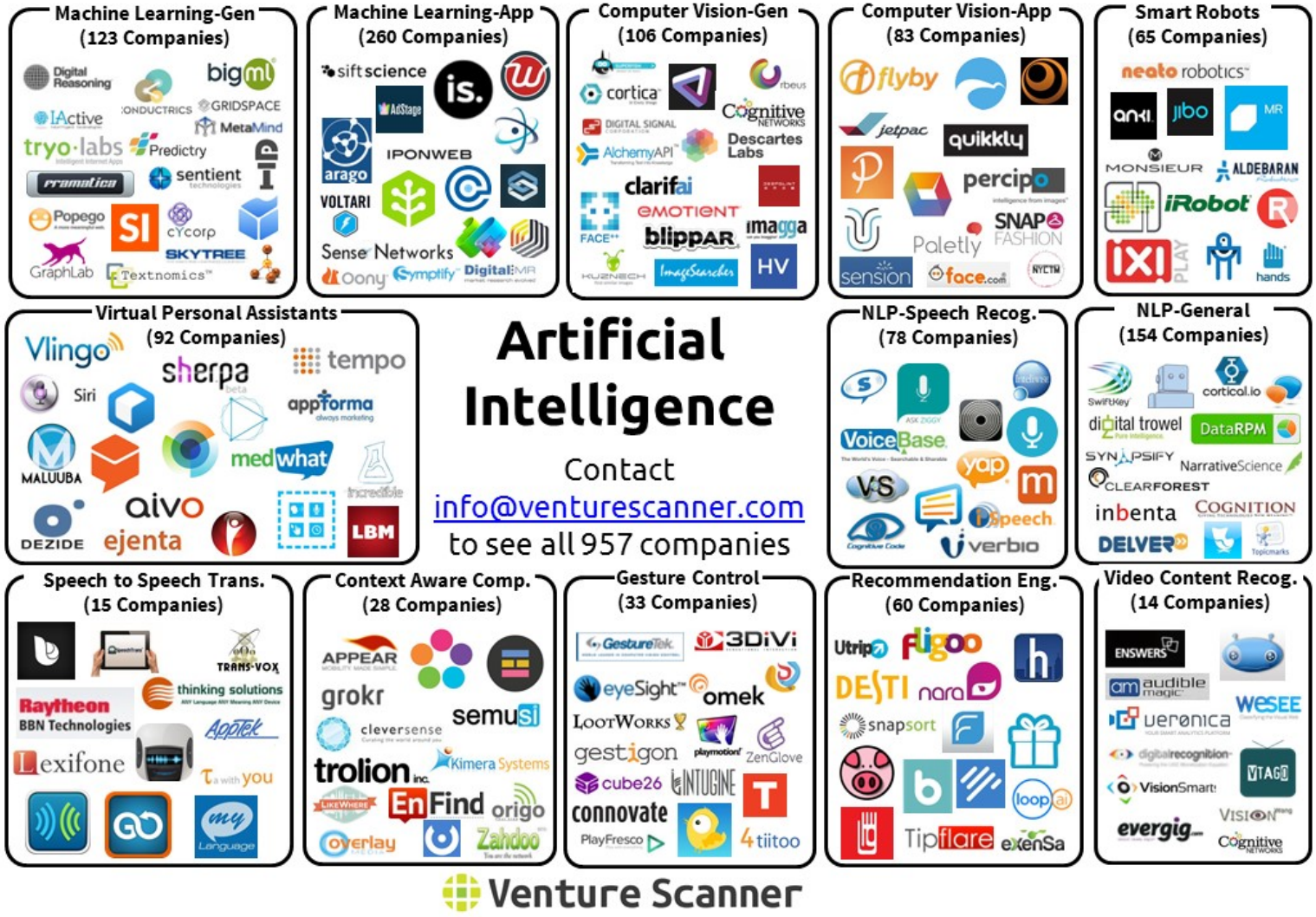
SHARE

WRITTEN BY

Brad Jones

Published: 2 hours ago

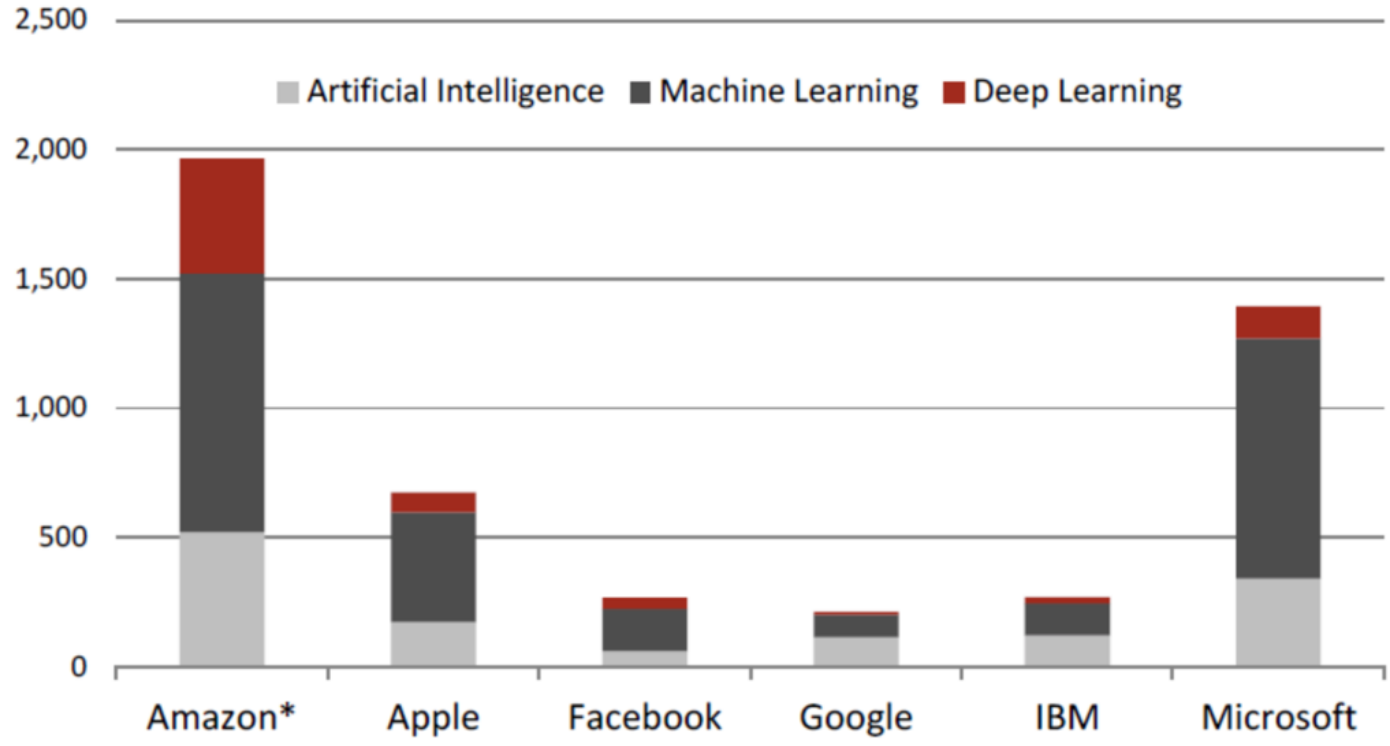
#Alibaba #machine reading #microsoft


Machine Learning-Gen
(123 Companies)
Machine Learning-App
(260 Companies)
Computer Vision-Gen
(106 Companies)
Computer Vision-App
(83 Companies)
Smart Robots
(65 Companies)
Virtual Personal Assistants
(92 Companies)
Artificial Intelligence
Contact
info@venturescanner.com
to see all 957 companies
NLP-Speech Recog.
(78 Companies)
NLP-General
(154 Companies)
Speech to Speech Trans.
(15 Companies)
Context Aware Comp.
(28 Companies)
Gesture Control
(33 Companies)
Recommendation Eng.
(60 Companies)
Video Content Recog.
(14 Companies)
Venture Scanner

By 2020, the market for machine learning will reach $40 billion, according to market research firm IDC.

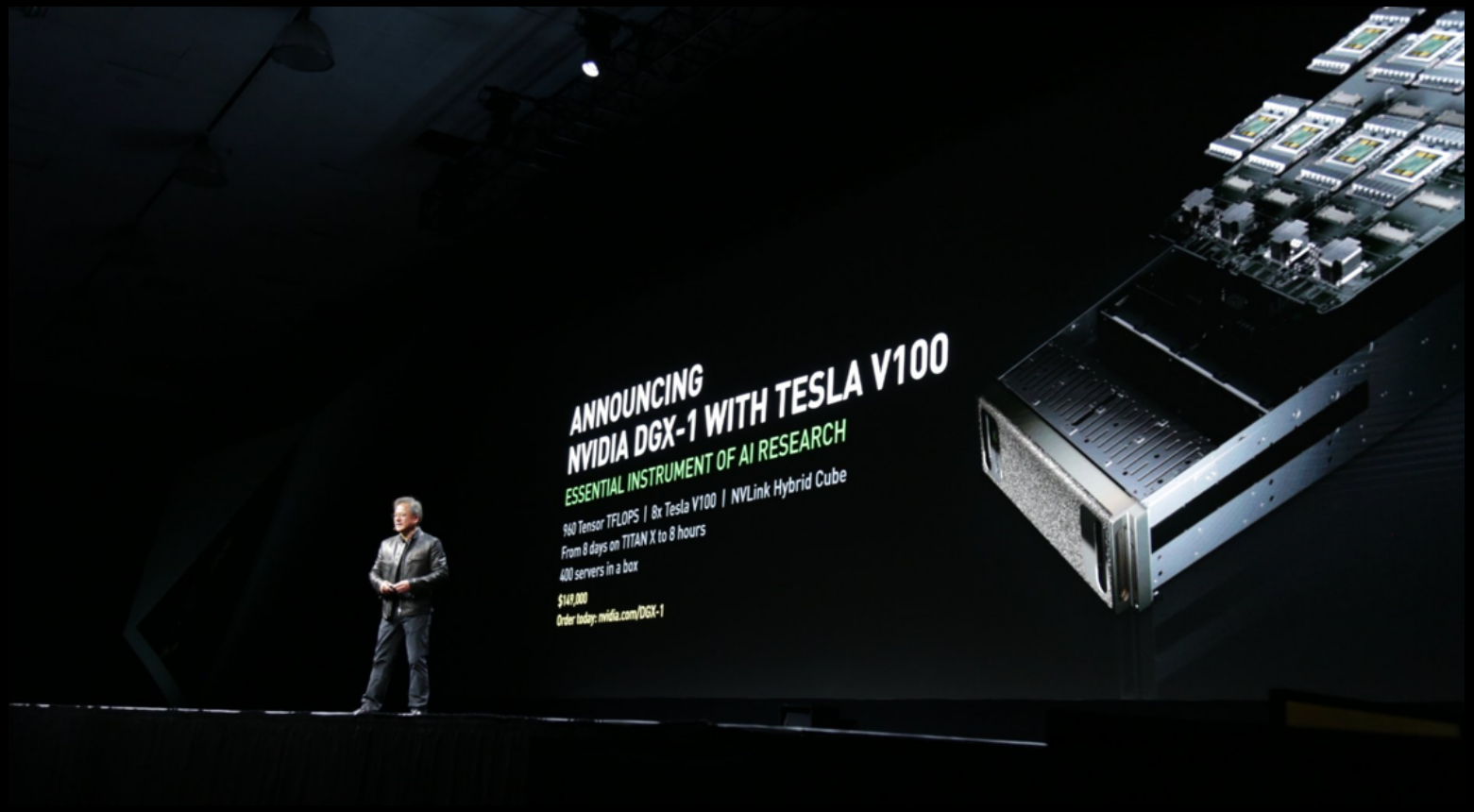
Deep Learning market is projected to be ~$5B by 2020

Exhibit 23: Monster.com Postings by Company, Search Terms: Artificial Intelligence, Machine Learning, and Deep Learning



*Machine Learning results listed as "1,000+"

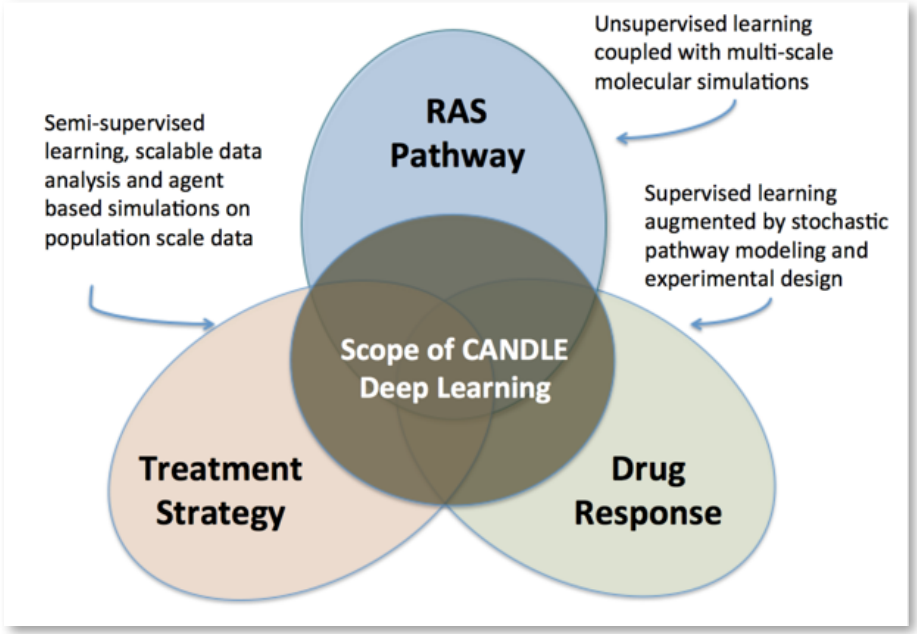
Source: monster.com


ANNOUNCING
NVIDIA DGX-1 WITH TESLA V100
ESSENTIAL INSTRUMENT OF AI RESEARCH
960 Tensor TFLOPS | 8x Tesla V100 | NVLink Hybrid Cube
From 8 days on TITAN X to 8 hours
400 servers in a box
$149,000
Order today: nvidia.com/DGX-1


DGX STATION
TESLA
TESLA
TESLA
TESLA
NVIDIA


intel
AI AND MORE ON IA
Dr. Rajeeb Hazra
Vice President, Data Center Group
General Manager, Enterprise and Government Group
ISC High Performance
ISC High Performance
ISC

ECP-CANDLE : CANcer Distributed Learning Environment



CANDLE Goals

Develop an exscale deep learning environment for cancer

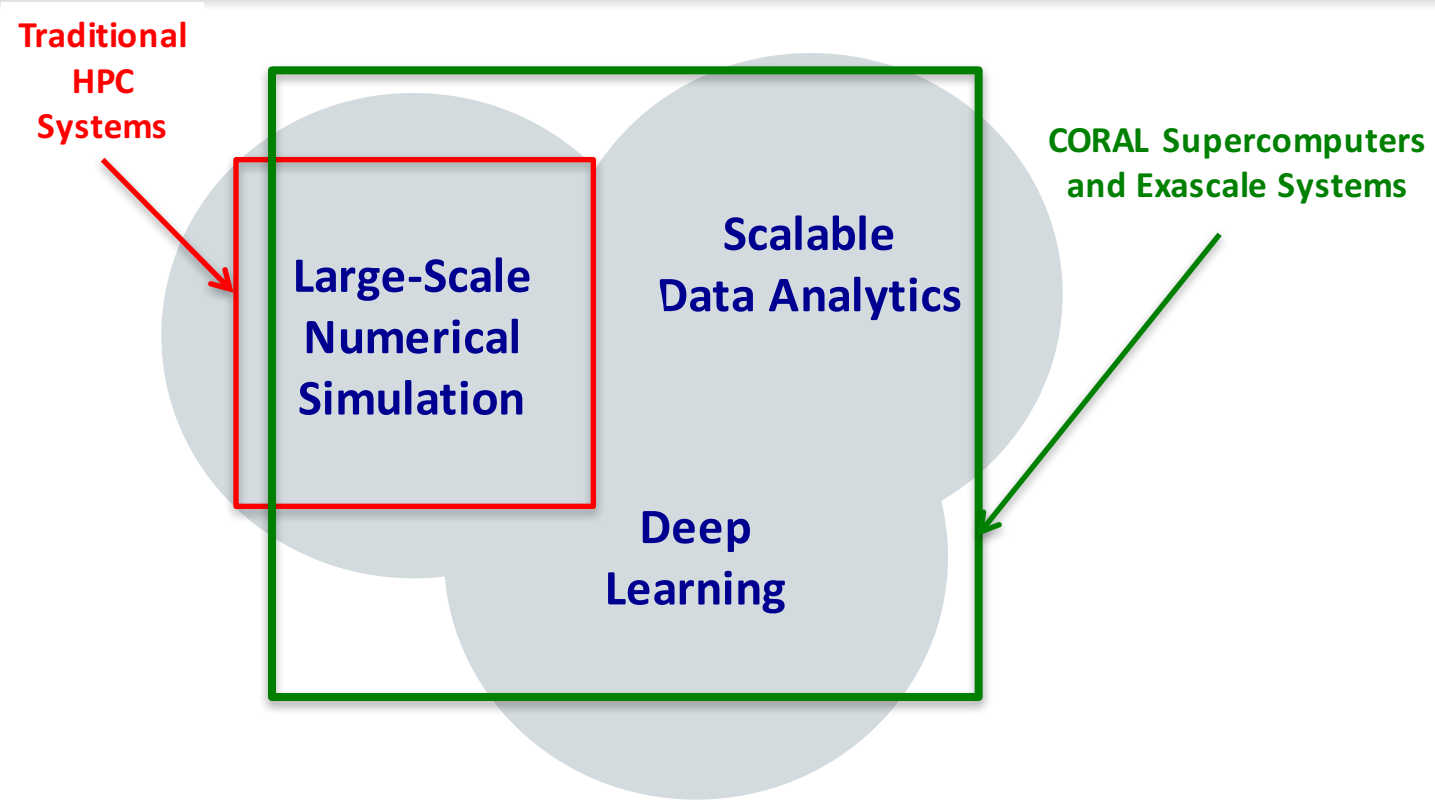
Building on open source Deep learning frameworks

Optimization for CORAL and exascale platforms

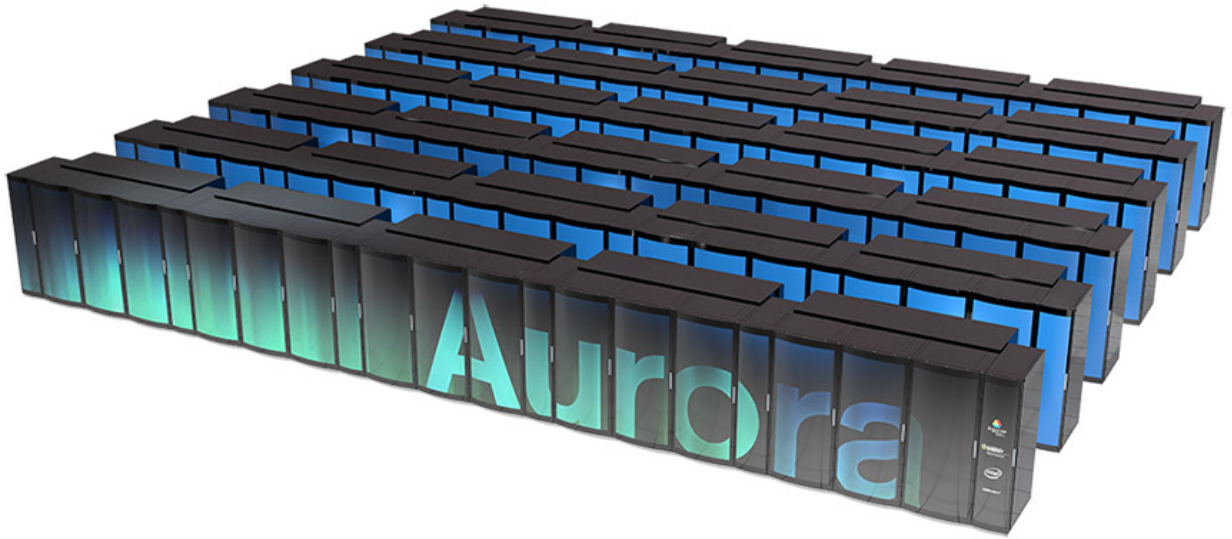
Support all three pilot project needs for deep dearning

Collaborate with DOE computing centers, HPC vendors and ECP co-design and software technology projects

DOE Objective: Dirve Integration of Simulation, Data Analytics and Machine Learning



Aurora 2021 (A21) Exascale System



Architectural support for three pillars

- **Large-scale Simulation (PDEs, traditional HPC)**
- **Data Intensive Applications (science pipelines)**
- **Deep Learning and Emerging Science AI**

Acknowledgements

Many thanks to DOE, NSF, NIH, DOD, ANL, UC, Moore Foundation, Sloan Foundation, Apple, Microsoft, Cray, Intel, NVIDIA and IBM for supporting our research group over the years

Deep Dream