CS7150 Deep Learning

Jiaji Huang

https://jiaji-huang.github.io

01/13/2024

About the lecturer

- PhD in electrical engineering from Duke University
- Thesis: statistical signal processing and machine learning
- Worked at Baidu Research on NLP and Speech
- Now at AWS, on Large Language Models

Agenda

- About this class
- Deep Learning nowadays
- Programming
- Math

What to learn from the class

- Basic building blocks, concepts
  - e.g., conv layer, attention layer, optimizers, overfitting
- Important applications
  - e.g., language modeling
- Recent topics
  - e.g., model compression

After the class, you should be

- able to implement and train typical deep nets
- aware of some emerging trends in the fields
- able to read recent deep learning papers

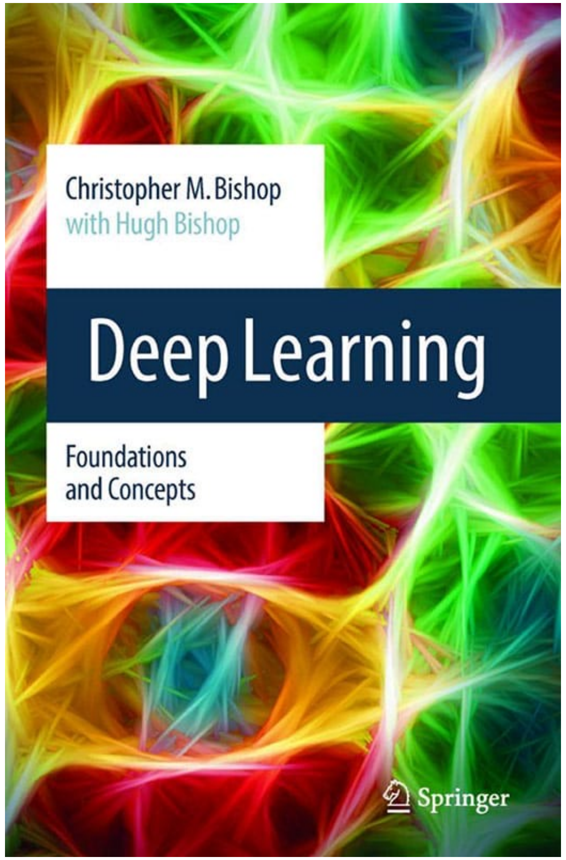
This class is not expected to …

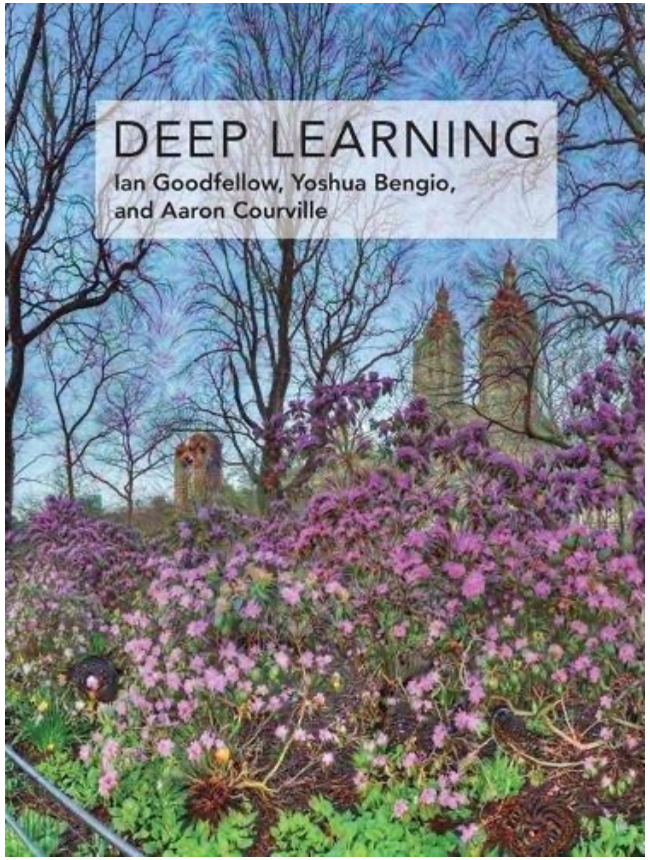
- teach coding with pytorch
- be a introductory course for machine learning

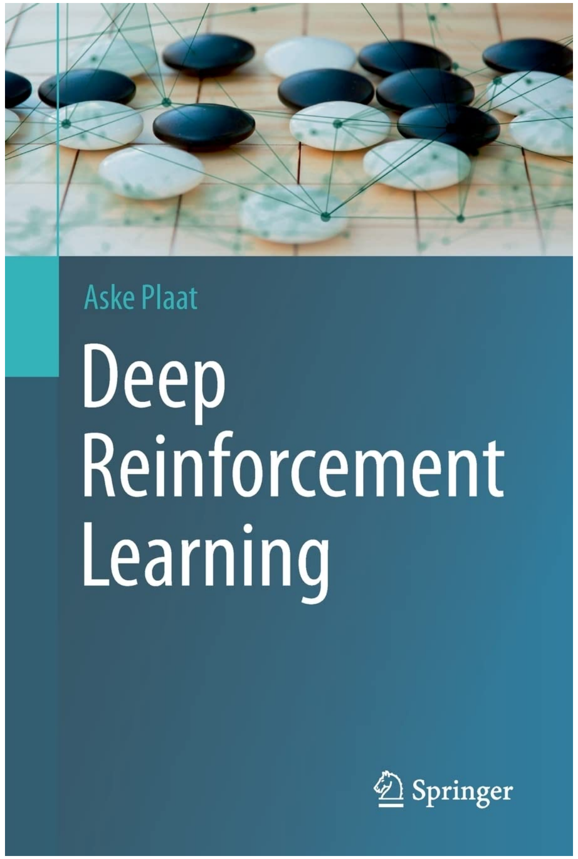
Logistics

- Classes
  - Each Saturday 9:00am-12:20pm
  - Before mid-term: focus on basics
  - After mid-term: advanced topics, 2-3 guest lectures (tentative)
  - Homework submission deadline: Friday 11:59pm
- Syllabus: https://jiaji-huang.github.io/CS7150.html
- TA: Pratyaksh Bhalla (bhalla.pr@northeastern.edu)
- Office hours
  - Lecturer: Saturday 1:00pm-3:00pm
  - TA: Friday Morning

Textbooks


Christopher M. Bishop
with Hugh Bishop
Deep Learning
Foundations and Concepts
Springer


DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville


Aske Plaat
Deep Reinforcement Learning
Springer

Logistics

- Grades
  - 20% homework + 20% paper presentation + 20% mid-term exam + 40% project
  - Policy for ☹
    - absence
    - late submission of homework
    - Plagiarism ([Integrity policy])
- Computational resources:
  - You'll need a laptop/desktop with python3
  - Khoury cloud (See instructions on syllabus page)

Paper Presentation

- Since class of 01/27, after lecture, 30min includes Q&A
- TA will generate a randomized list of presenters
- Paper selection:
  - Suggested reading materials (but only papers) in previous classes
  - paper identified by presenter
- Credits:
  - For presenter: clear presentation, good answers, driving discussion
  - For audience: raise question, involve in discussion

How to identify interesting papers

- Subscribe to arxiv cs.AI: send an email like this

To: cs@arxiv.org

Subject: subscribe yourFirstName yourLastName

add Artificial Intelligence

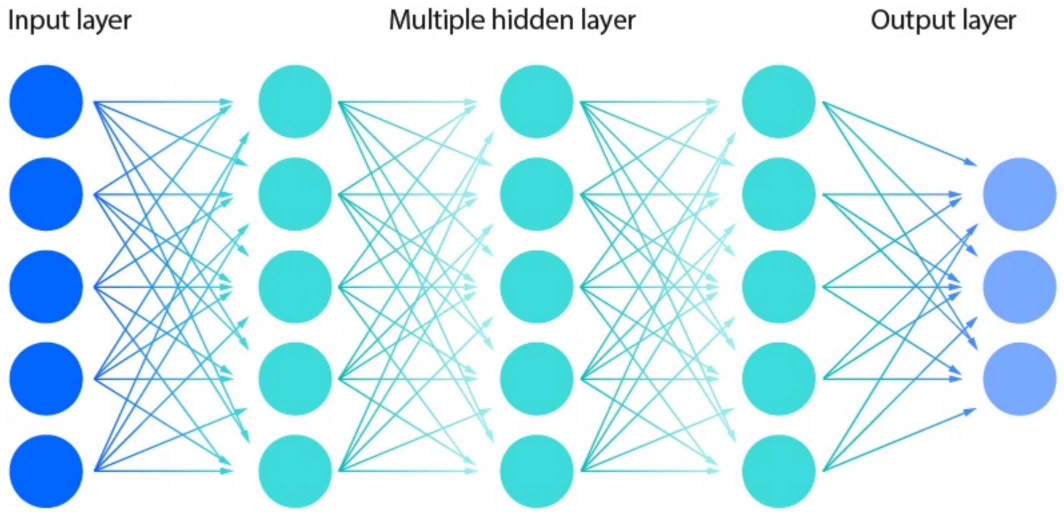
- Check recent accepted papers in conferences
  - Learning: Neurips, ICML, ICLR, AAAI, …
  - NLP: ACL, EMLNP, NAACL, …
  - Computer vision: ICCV, CVPR, ECCV, ...
- Media: twitter etc.

Agenda

- About this class
- Deep Learning nowadays
- Programming
- Math

What is Deep Learning

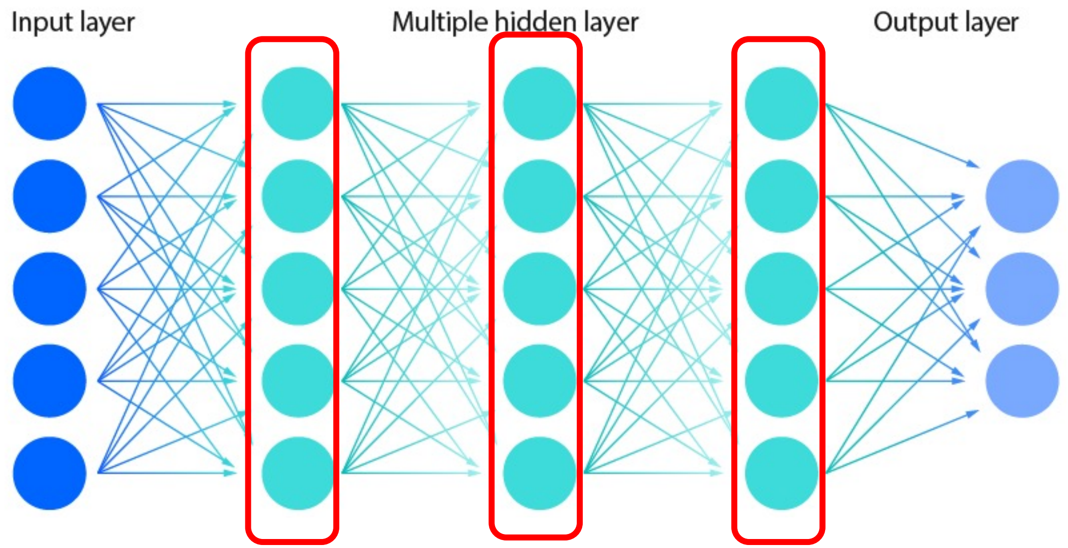
- Neural network: aka artificial neural networks (ANN)



- **Deep** neural network: many layers
- **Deep Learning**: A sub-area of machine learning that builds deep neural network to model the world

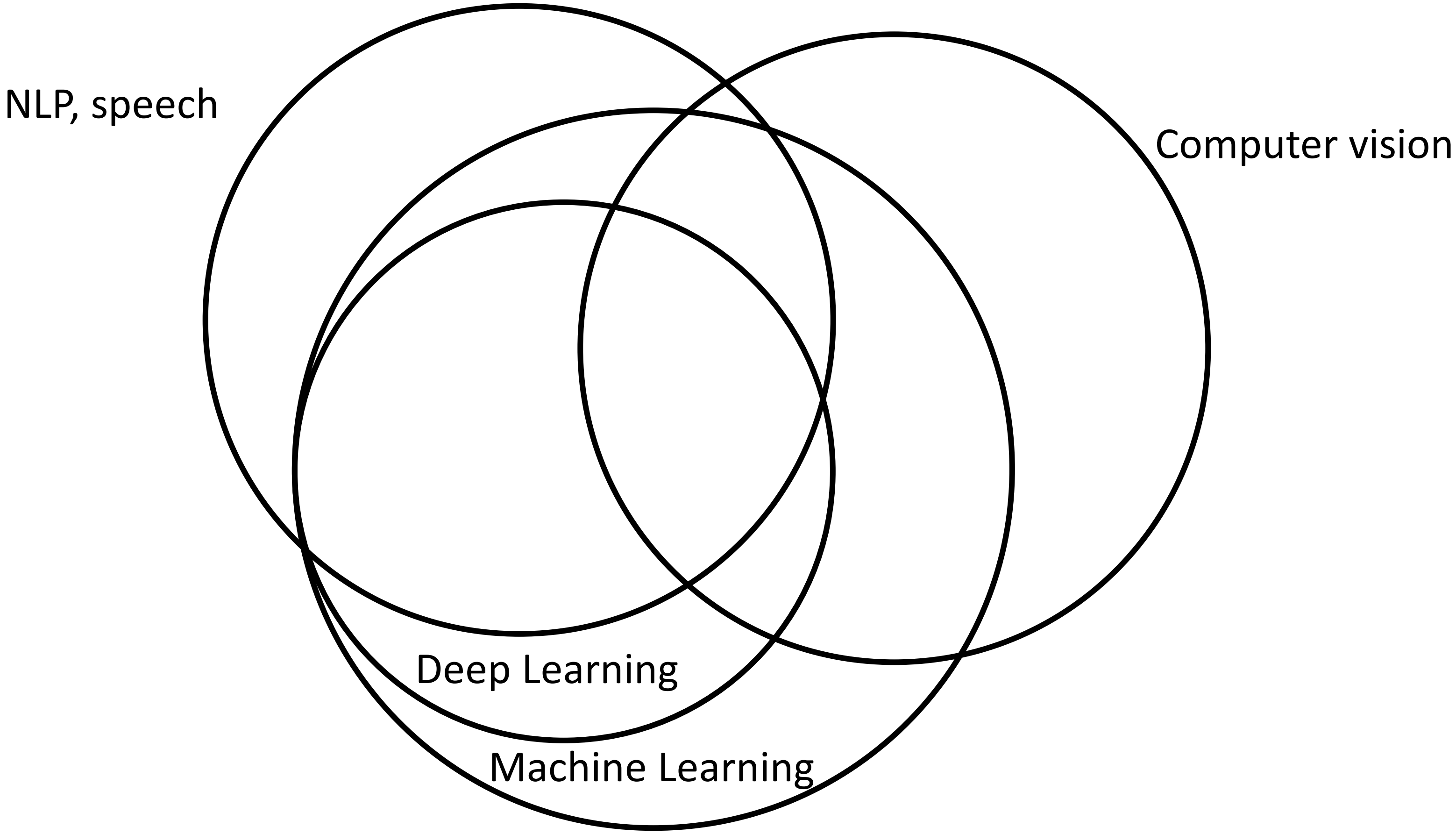
Illustration from https://www.ibm.com/topics/neural-networks

A few more terminologies

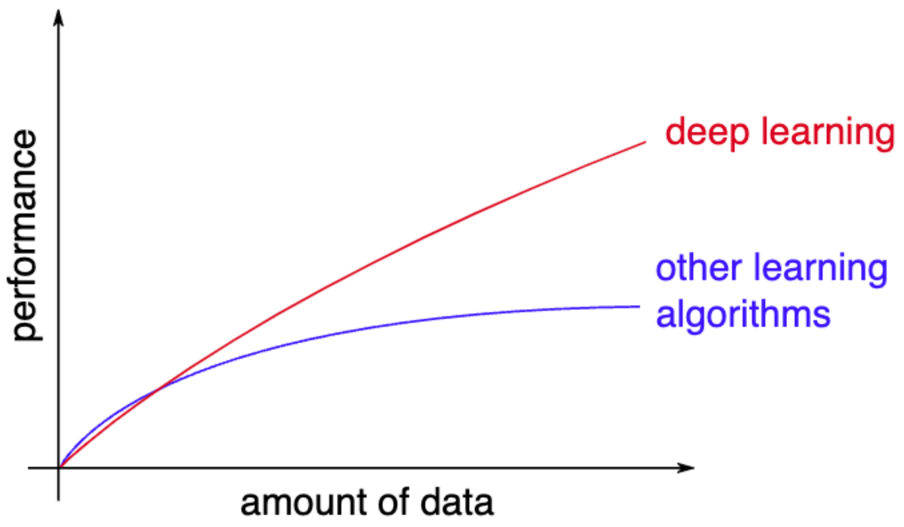


- Hidden representations/feature maps
- Learned Representation (as apposed to hand-crafted features)
- Distributed representations
- **End-to-end**: jointly learn the representation for the task

Related areas



Why is DL useful?



- Given enough computational power (though)

DL and vision

- Image Classification (e.g., ImageNet, 1000 classes)
  - [Alexnet (Krizhevsky *et. al, 2012*)]

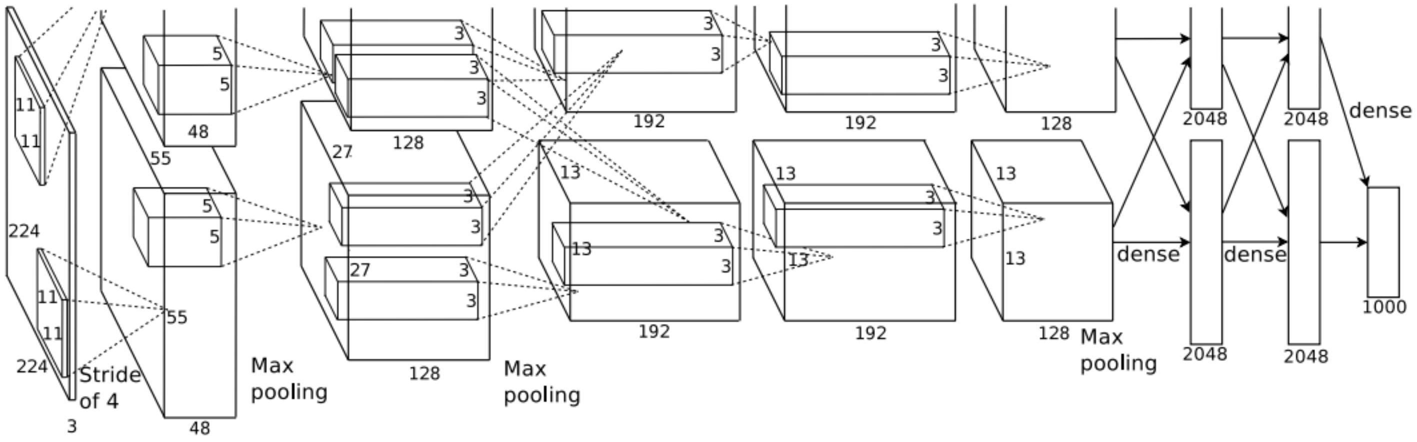


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

DL and vision

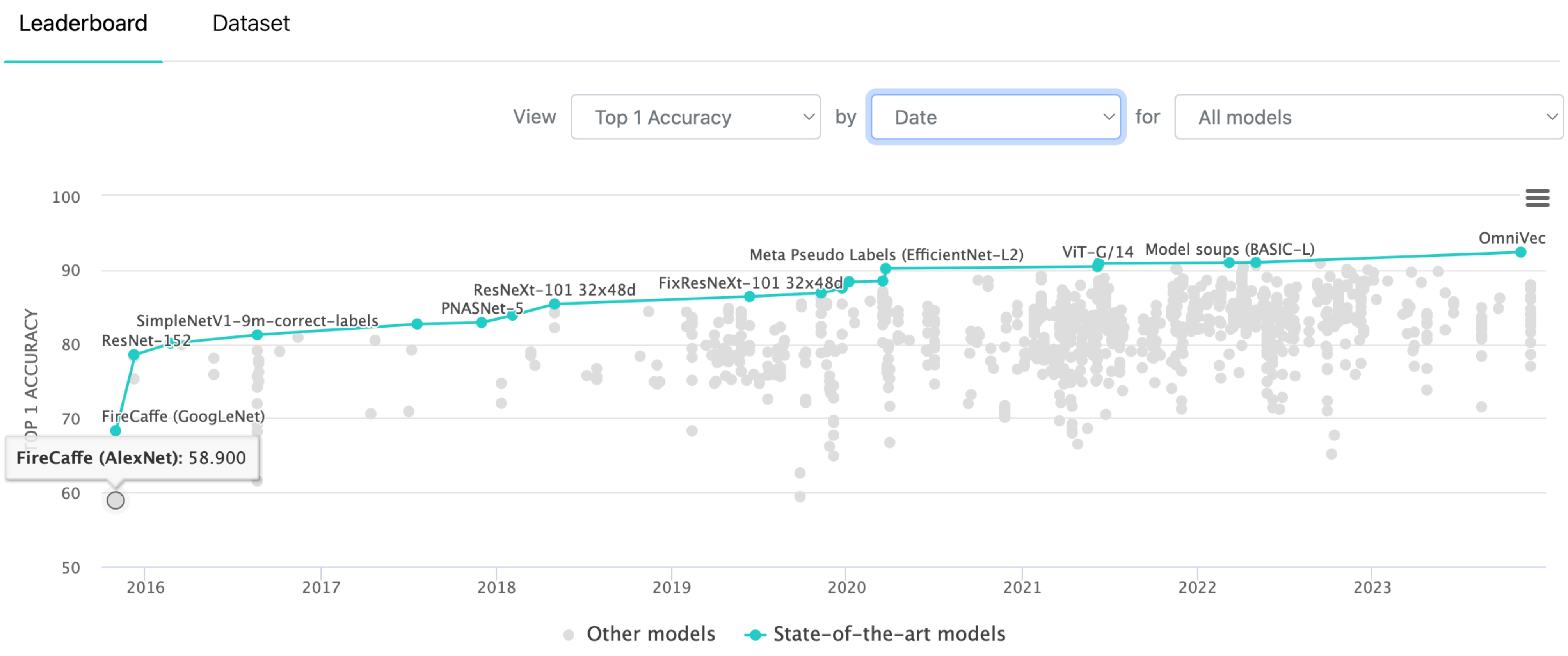
- Beat non-DL methods by a large margin

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

- Limitations
  - The GPU memory was tiny (3GB), and slow
  - So they have to <u>split the layer parameters</u> (we may talk about this later)
  - and limit training time

Since AlexNet …



Trend according to [paperwithcode]

DL and vision

- Object Detection
- [Fast R-CNN (Girshick 2015)]
- Track trend on [paperwithcode]

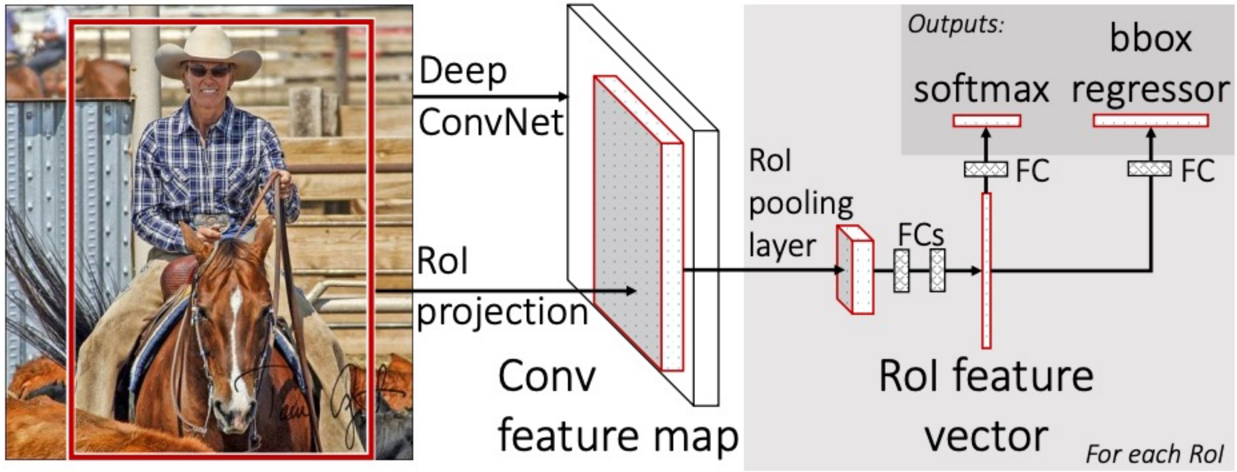
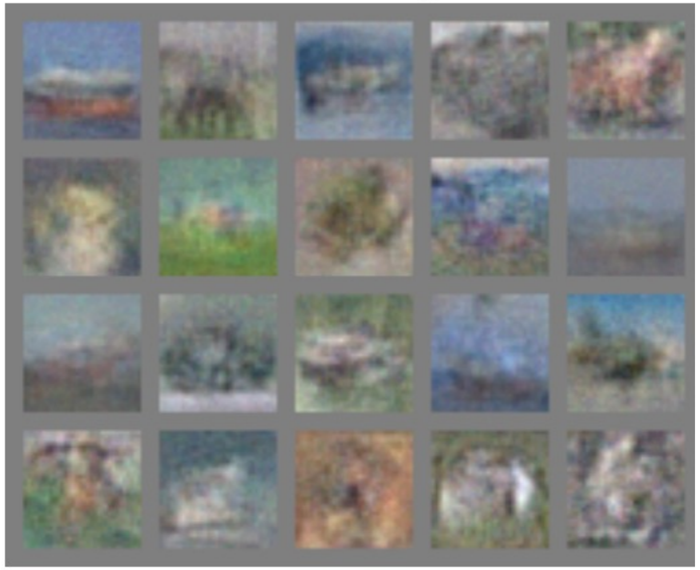


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

DL and vision

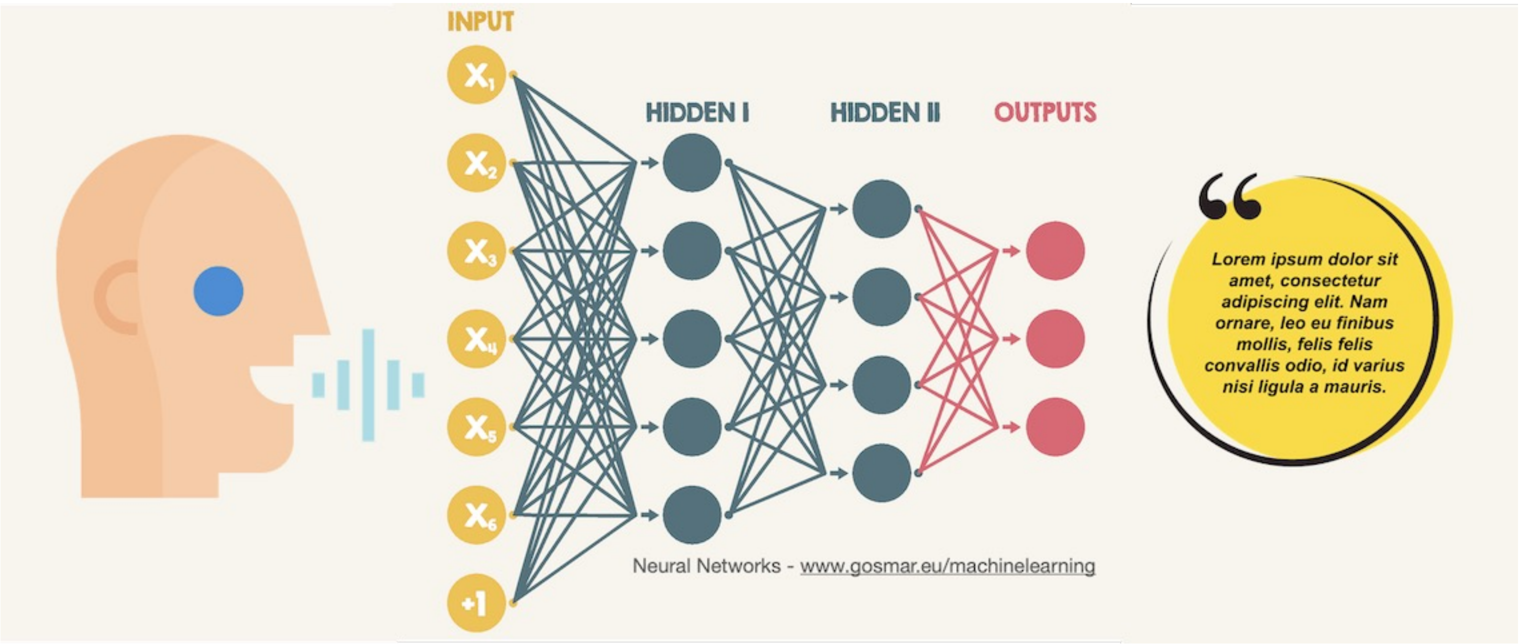
- Image generation
- GAN (Goodfellow *et. al, 2014*)



- Stable Diffusion

DL and Speech

- Speech Recognition

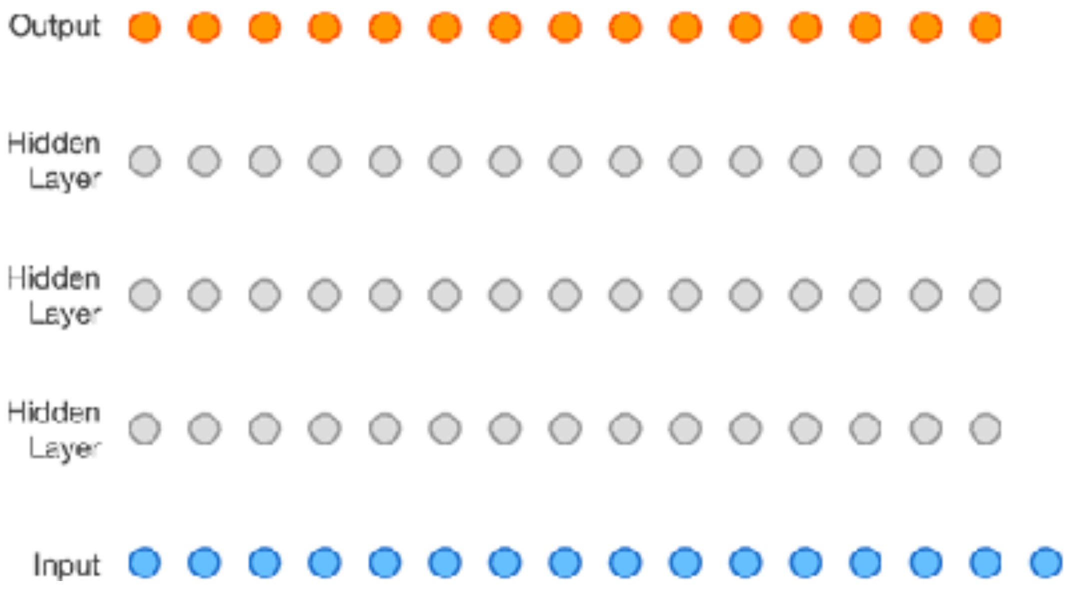


- Check trend of Word Error Rate (WER) on paperwithcode

Illustration from this blogpost

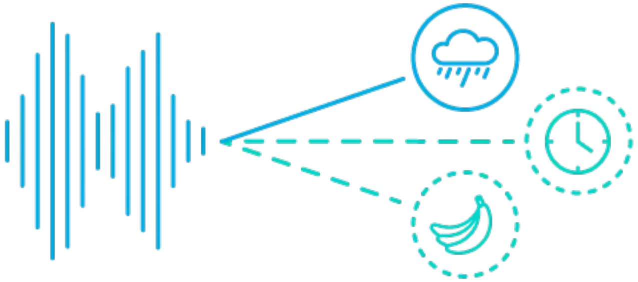
DL and Speech

- Speech synthesis
  - Wavenet



DL and NLP

- Language Understanding



Illustrations from here and here

DL and NLP

- Language generation
  - conversation

You
suggest 5 warm places to visit during Christmas, describe each within 20 words

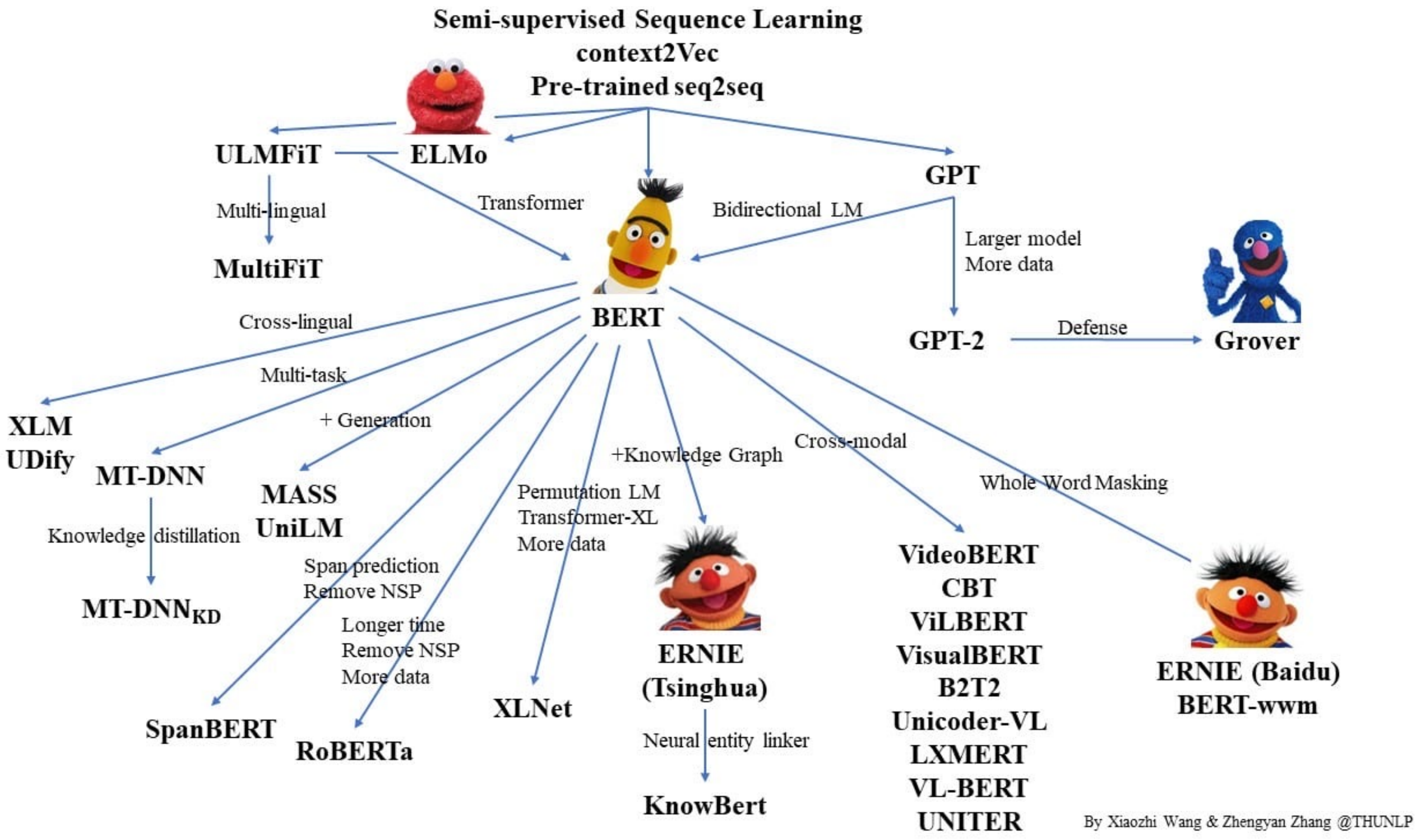
ChatGPT
1. **Maui, Hawaii:** Sunny beaches, festive luaus, and lush landscapes create a tropical Christmas paradise with a unique island charm.
2. **Sydney, Australia:** Celebrate down under with beach picnics, vibrant fireworks at the iconic Opera House, and summer festivities.
3. **Phuket, Thailand:** Escape to stunning beaches, enjoy Thai hospitality, and witness vibrant celebrations with a tropical twist.
4. **Cancun, Mexico:** White sandy beaches, lively fiestas, and cultural warmth make Christmas in Cancun a blissful escape.
5. **Cape Town, South Africa:** Experience a summer Christmas with breathtaking landscapes, outdoor adventures, and diverse cultural celebrations.

You
pick the cheapest one for me

ChatGPT
Phuket, Thailand: Offers affordability with budget-friendly accommodations, delicious street food, and a variety of affordable activities amid stunning tropical scenery.

- Translation
- Summarization
- Essay writing
- ....

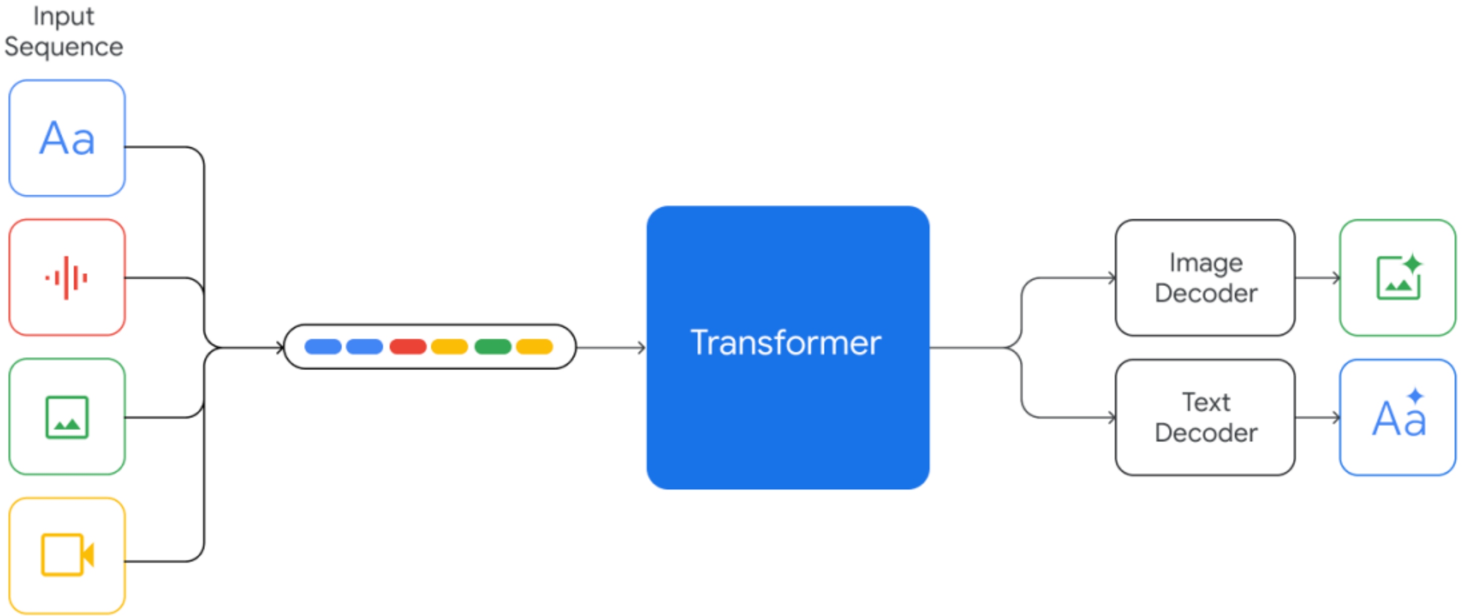
DL and NLP



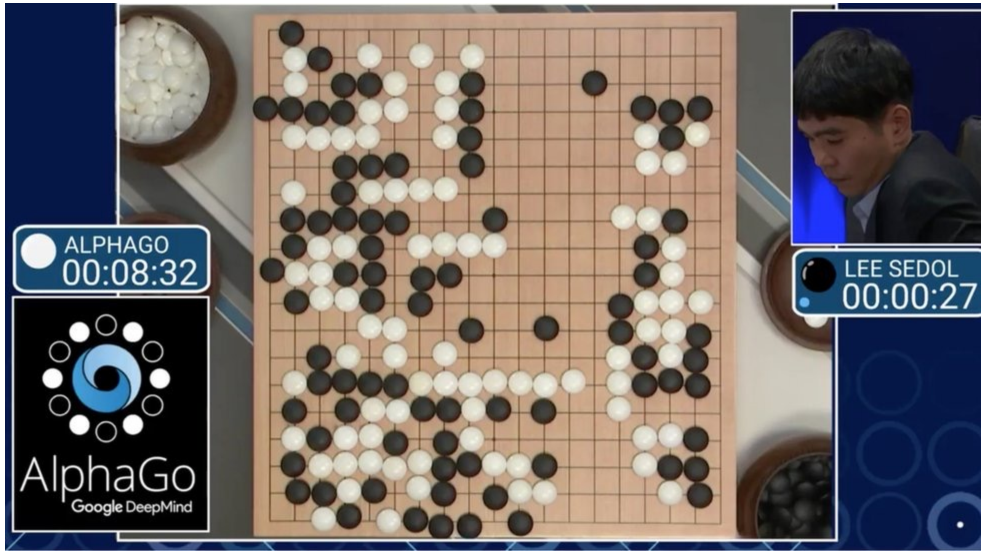
Taxonomy of language models from this page (outdated of course)

DL and multi-modality

- [Gemini]()



DL and Decision Making



Images from here and here

DL and Science

- AlphaFold

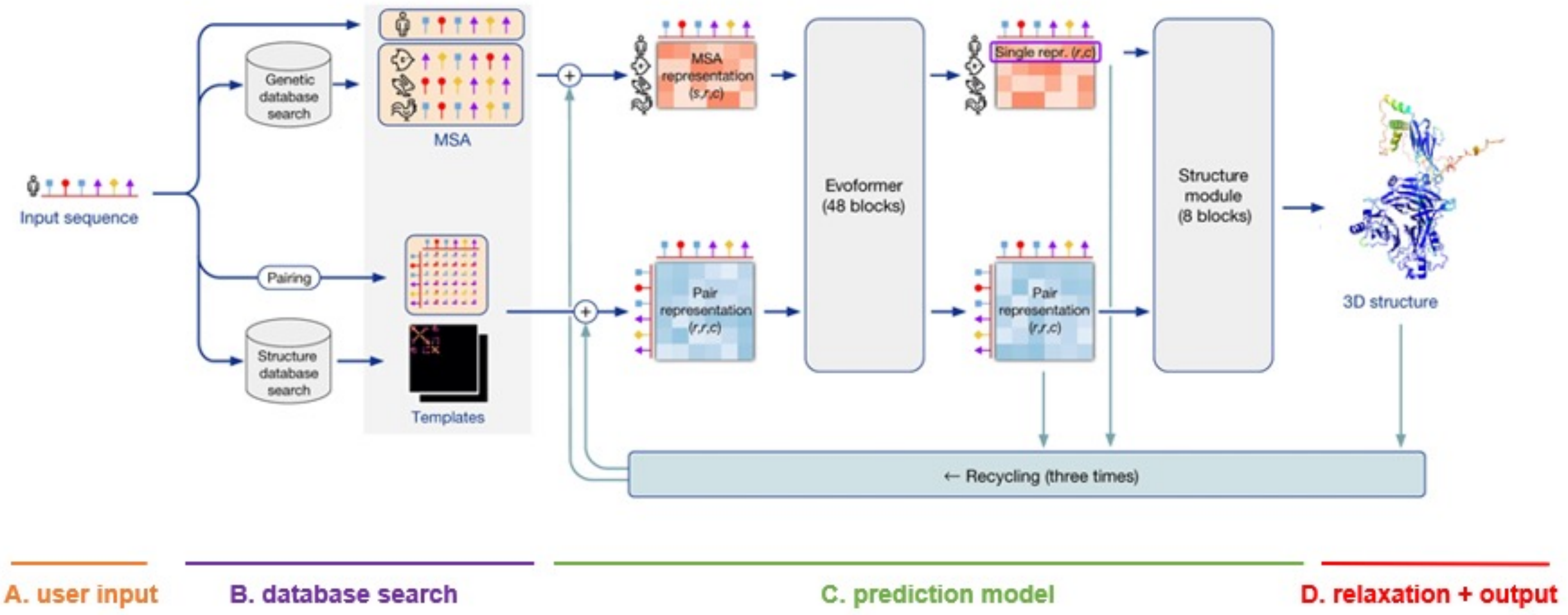
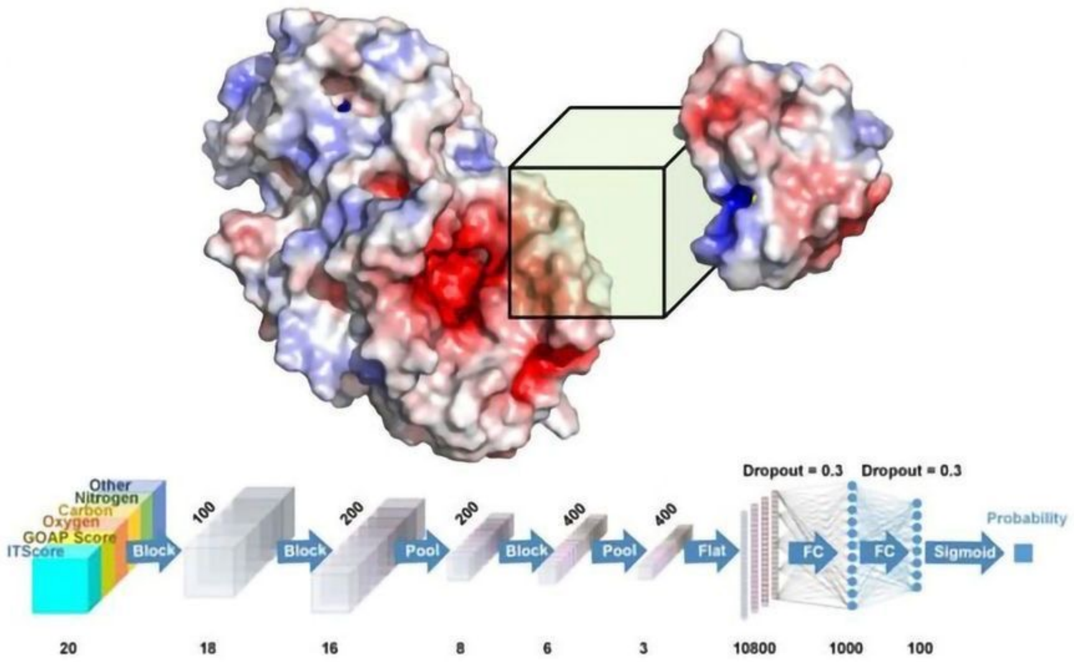


Illustration from [wiki]()

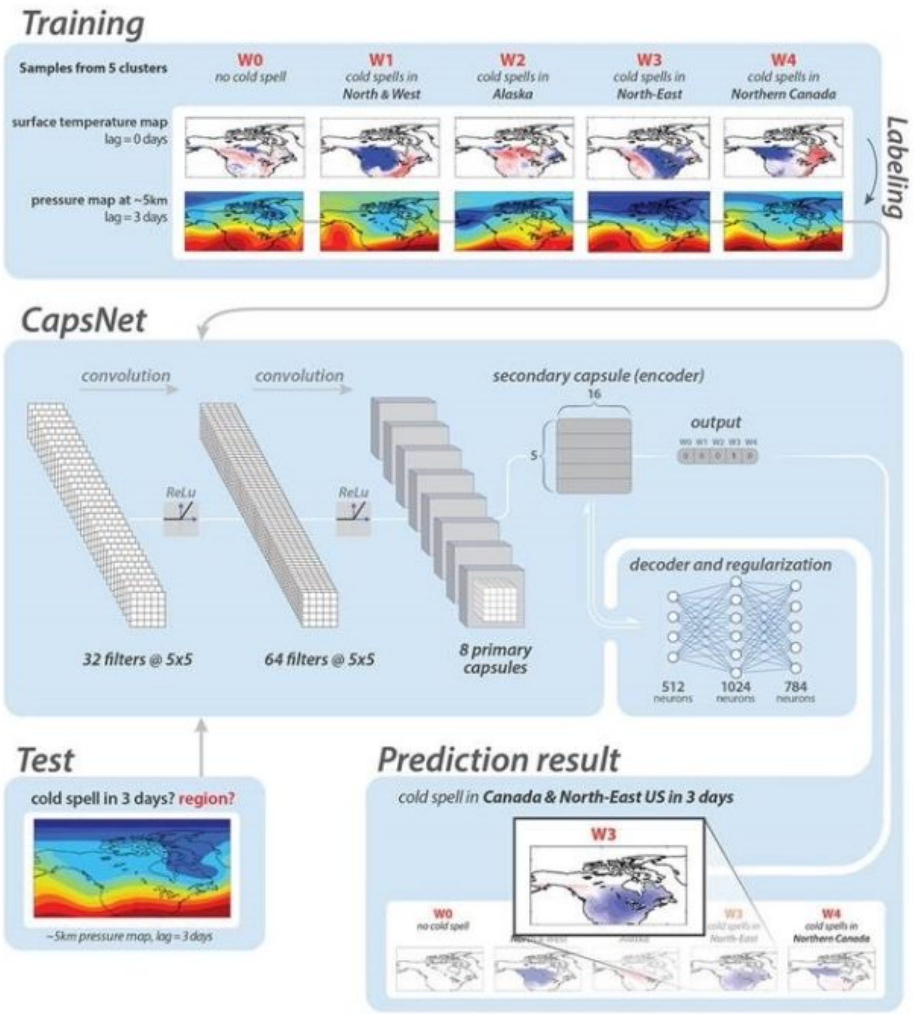
DL and Science

- Drug discovery



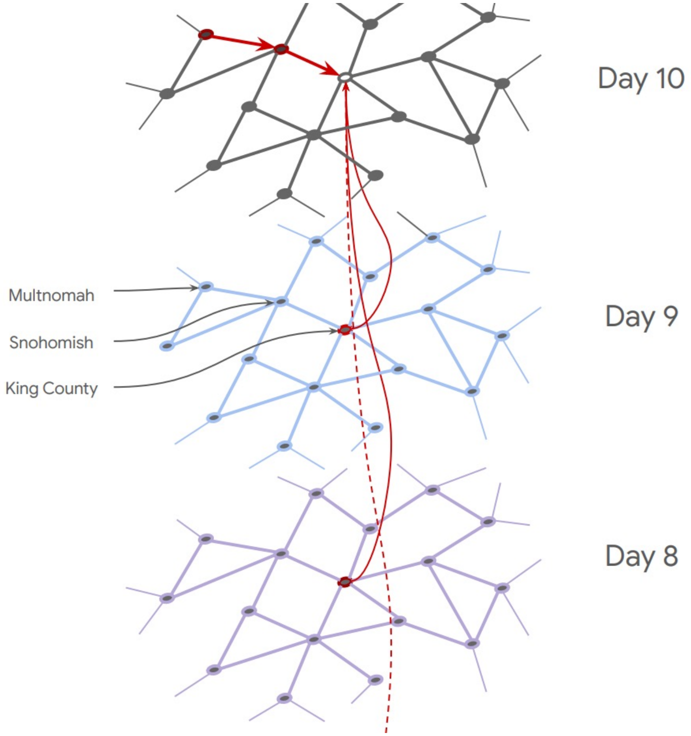
Illustrations from [here]() and [here]()

- Weather forecasting

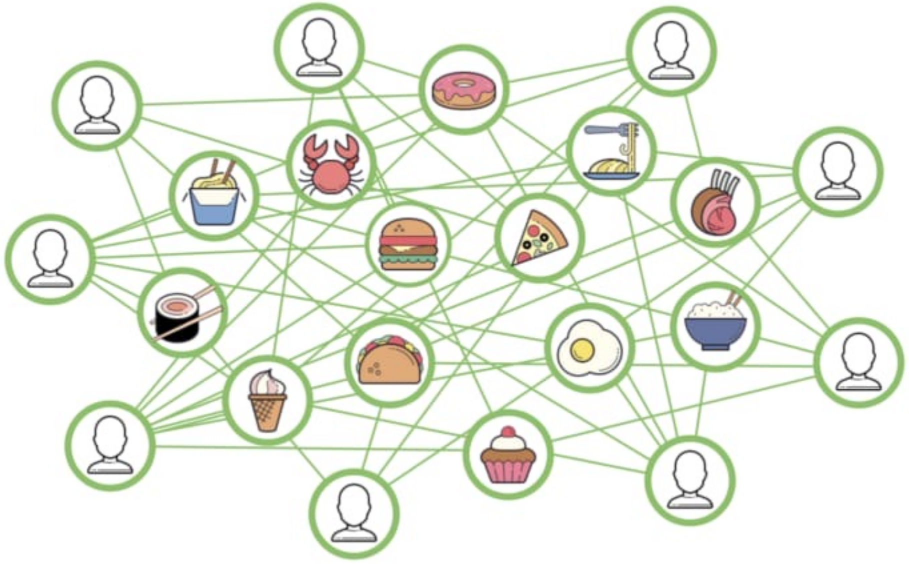


DL and Social Science

- Modeling spread of Covid



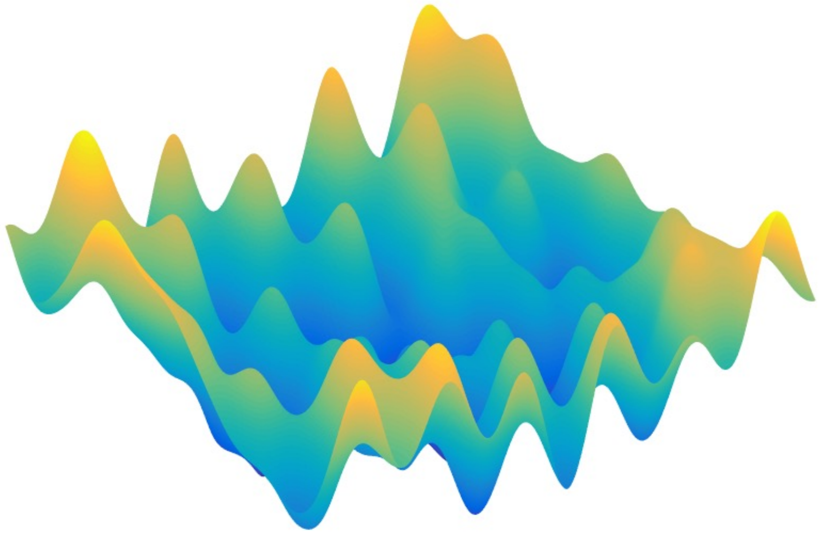
- Recommendation system

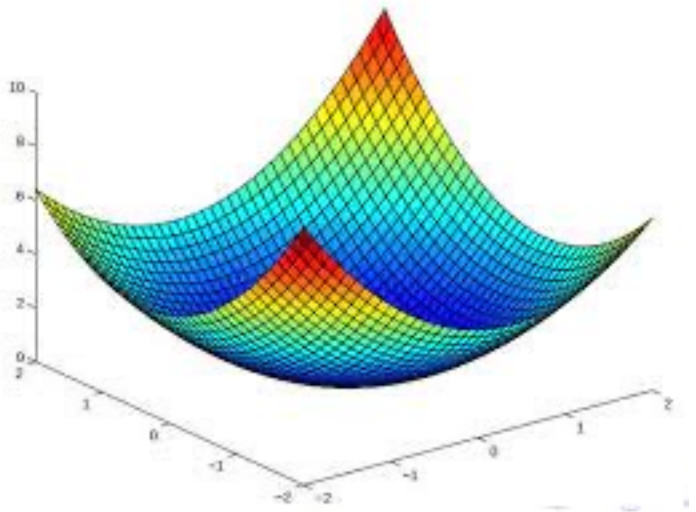


Illustrations from [here]() and [here]()

Open Problems

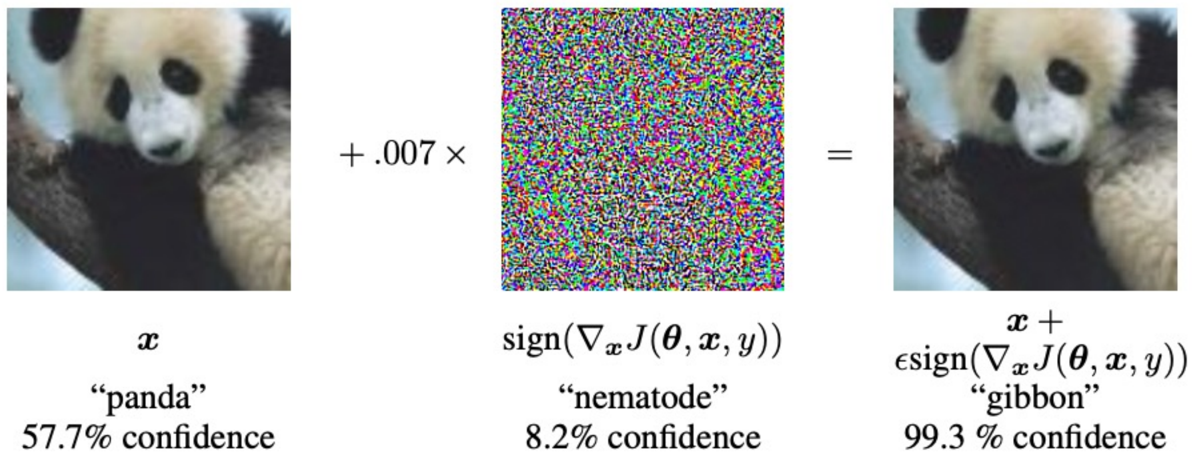
- Con-convex optimization
  - Training could land at somewhere sub-optimal
  - where we land at is non-deterministic due to
    - Randomness in Initialization
    - Randomness in feeding training data
    - Hyper parameters like learning rate
- As apposed to convex optimization
  - Unique or at least equally good minimums
  - Convergence Guarantees





Open Problems

- Generalization
  - Traditional wisdom: model with many parameters may overfit
  - Challenges traditional wisdom (Zhang et.al 2017)
- Robustness



Open Problems

- Interpretability
  - What feature is responsible
  - What training sample is responsible

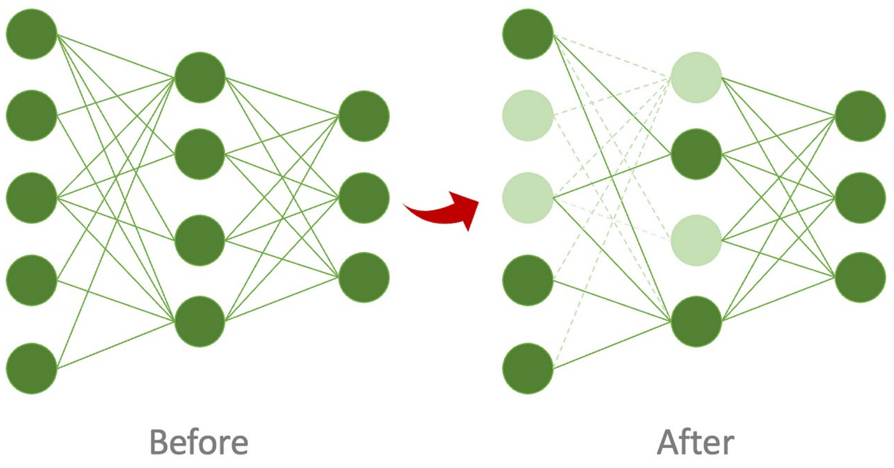
- Factuality, Hallucination

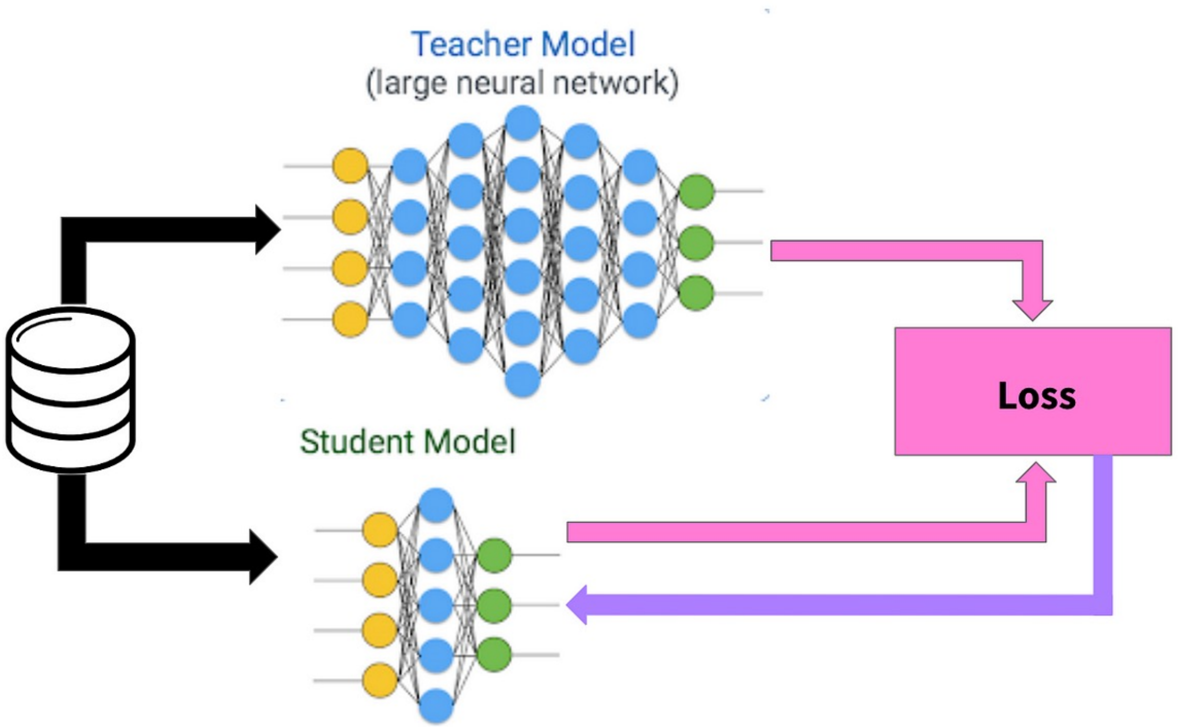
On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Authors: Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, Shmargaret Shmitchell Authors Info & Claims

Open Problems

- Model efficiency



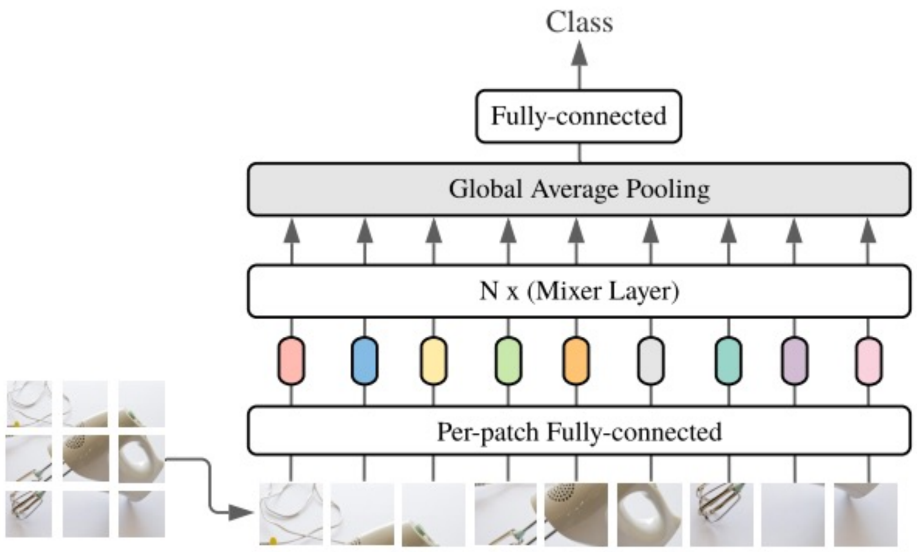


- Neural Architecture Search

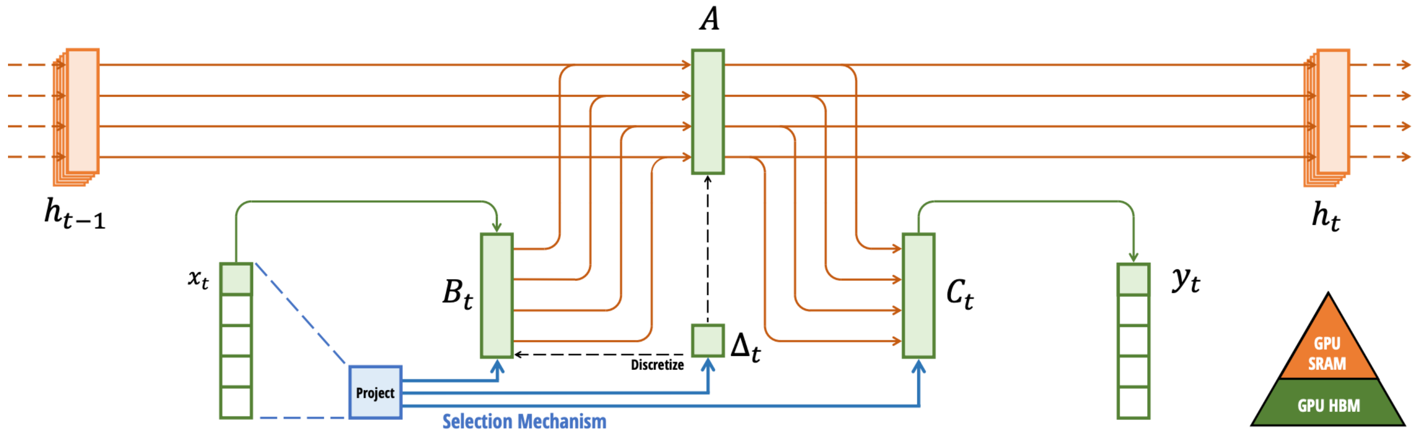
Illustration from [here]() and [here]()

Open Problems

- After transformers?

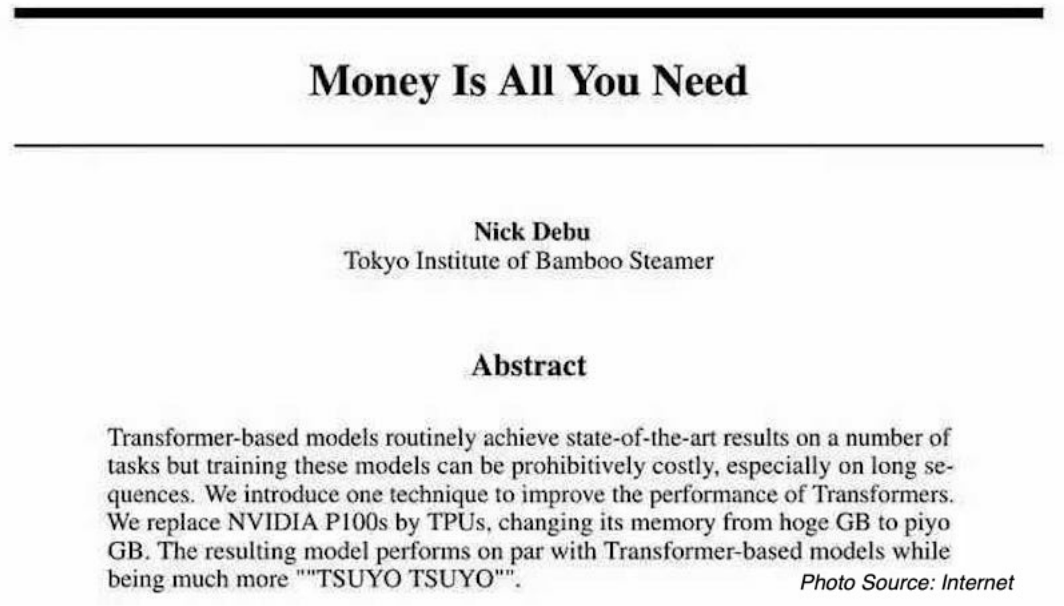


MLP mixer



Linear-time State Space Models

Criticism: rush for scaling, expensive


Money Is All You Need

Nick Debu
Tokyo Institute of Bamboo Steamer

Abstract

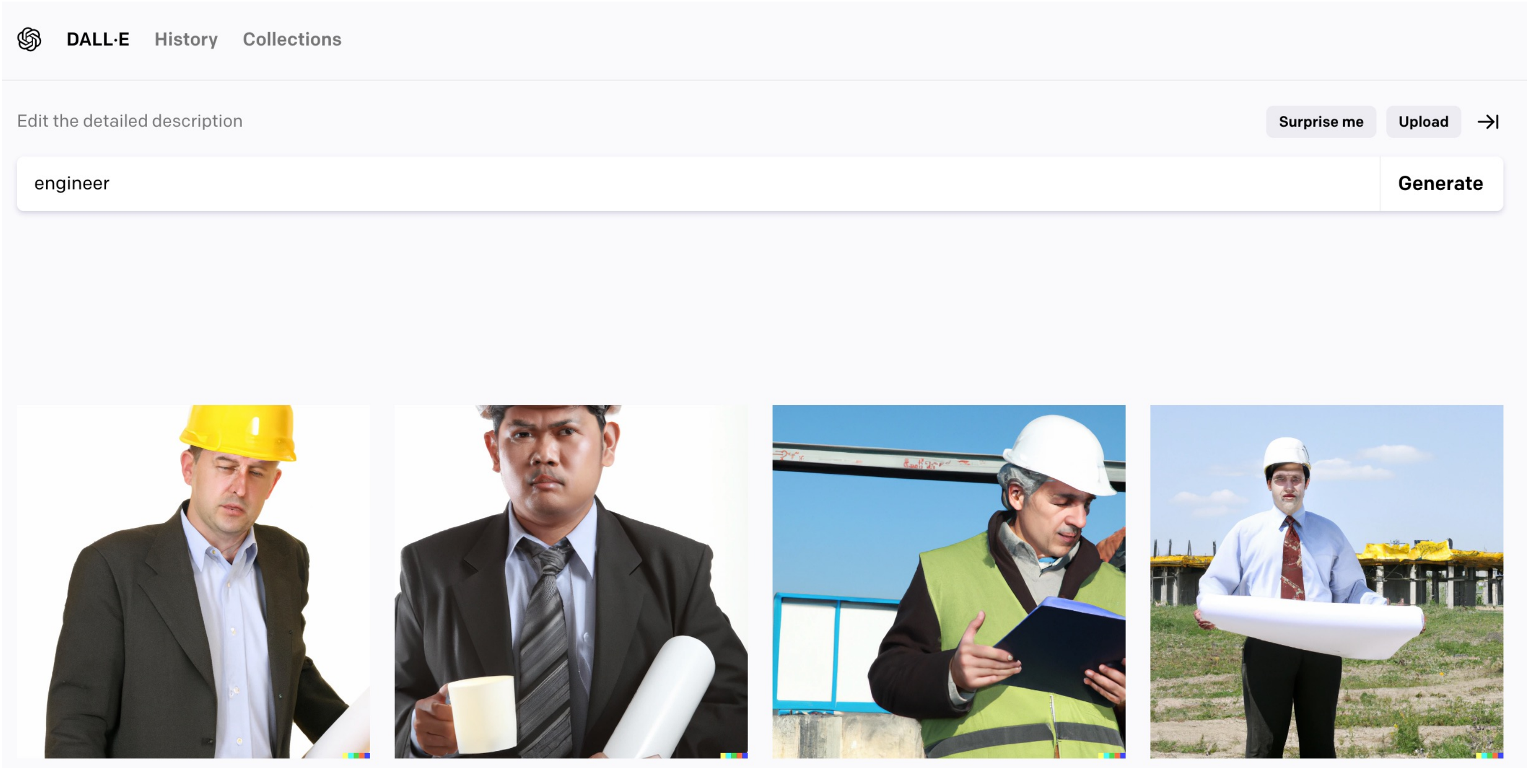
Transformer-based models routinely achieve state-of-the-art results on a number of tasks but training these models can be prohibitively costly, especially on long sequences. We introduce one technique to improve the performance of Transformers. We replace NVIDIA P100s by TPUs, changing its memory from hoge GB to piyo GB. The resulting model performs on par with Transformer-based models while being much more ""TSUYO TSUYO"".

Photo Source: Internet

Images from this [page]

Social Impact: Bias and Fairness

Stereotypes: generate more images of one gender/ethnic group



Example from this [article]

Social Impact: Legal Issues

- “deepfaked” celebrities

A viral video that appeared to show Obama calling Trump a 'dips---' shows a disturbing new trend called 'deepfakes'

Kaylee Fagan Apr 17, 2018, 1:48 PM PDT Share Save

https://www.youtube.com/watch?v=cQ54GDm1eL0

- Use of data

AP WORLD U.S. POLITICS SPORTS ENTERTAINMENT BUSINESS SCIENCE FACT CHECK ODDITIES HEALTH VIDEO CLIMAT

Israel-Hamas war Can Trump still run for president? NYE store hours Missouri beat Ohio Kathy

BUSINESS

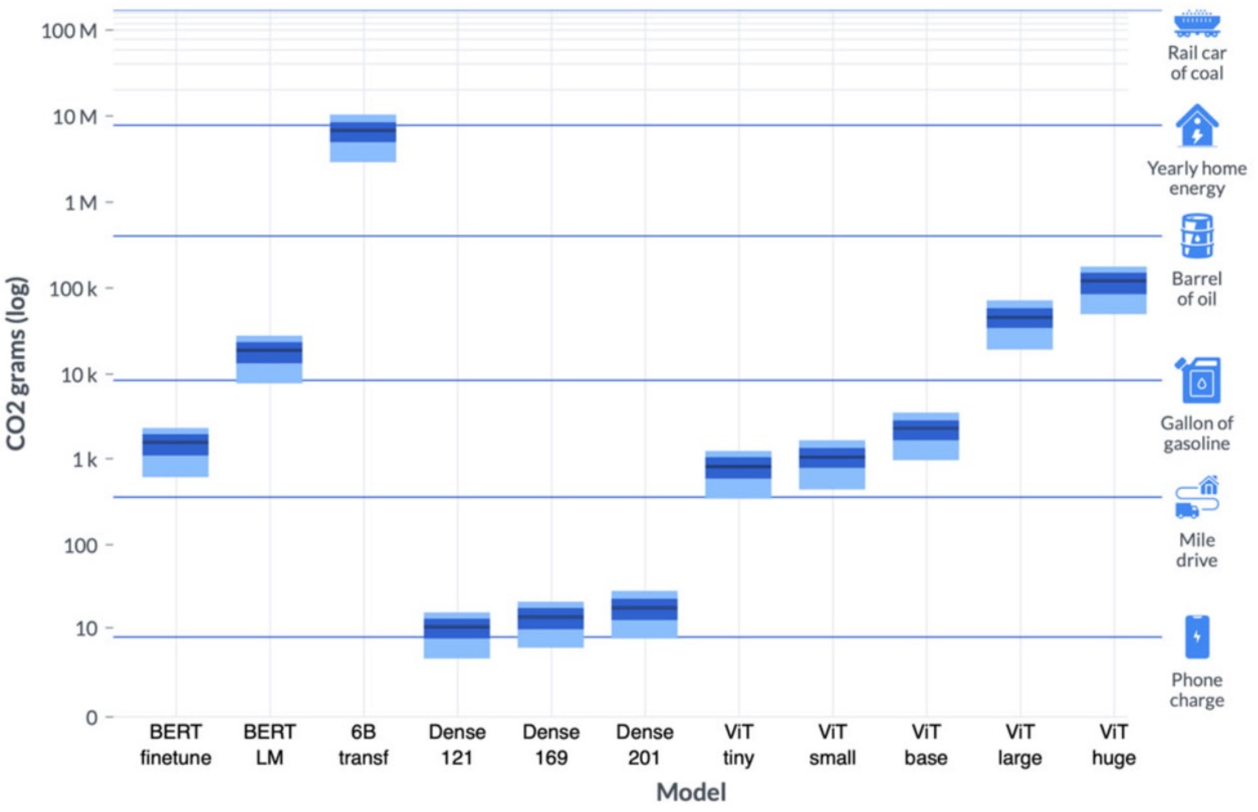
The New York Times sues OpenAI and Microsoft for using its stories to train chatbots

See article [here]() and [here]()

Social impact: Carbon footprint

Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]



CO2 Relative Size Comparison

See article here and here

Agenda

- About this class
- Deep Learning nowadays
- Programming
- Math

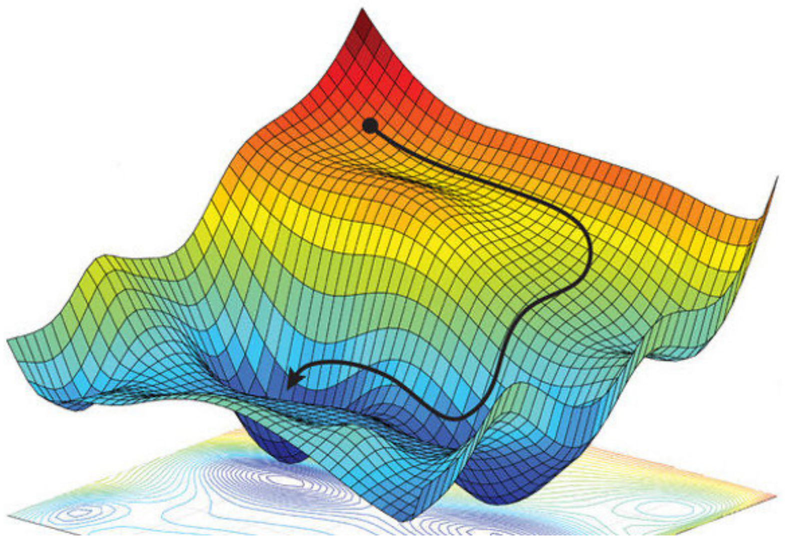
PyTorch: Tensors

- Installation: pip3 install torch
- The most basic object: torch.Tensor
  - Very much like NumPy Arrays, but also supports:
  - GPU acceleration (torch.Tensor.to(GPU_id))
  - auto-grad (torch.Tensor.backward)

```
[>>> import torch
[>>> x=torch.tensor([1.0, 2.0], requires_grad=True)
[>>> y=torch.tensor([3.0, 4.0], requires_grad=True)
[>>> z=torch.sum(x*y)
[>>> x
tensor([1., 2.], requires_grad=True)
[>>> y
tensor([3., 4.], requires_grad=True)
[>>> z
tensor(11., grad_fn=<SumBackward0>)
[>>> z.backward()
[>>> x.grad
tensor([3., 4.])
[>>> y.grad
tensor([1., 2.])
```

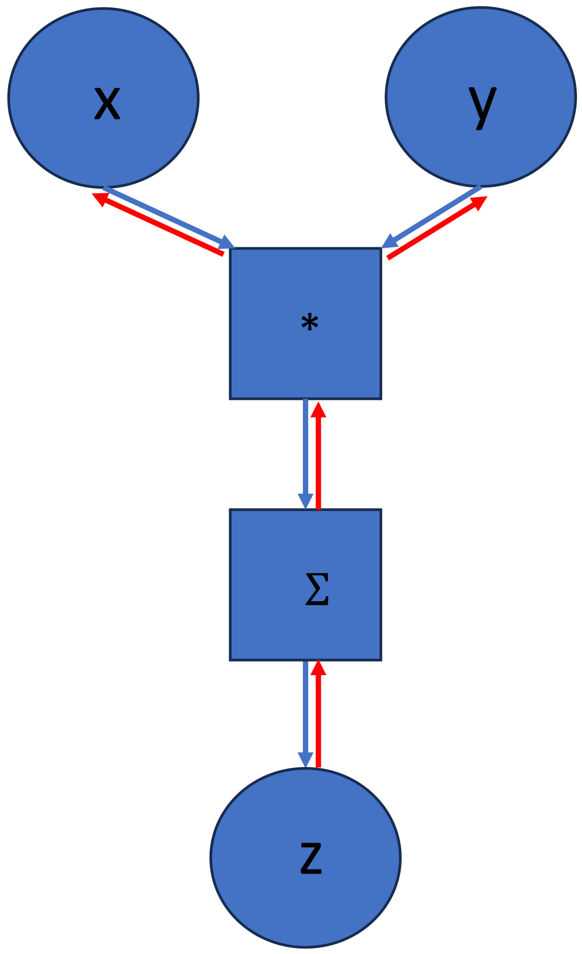
Why auto-grad?

- Deep nets are trained by minimizing a loss function with Stochastic Gradient Descent (SGD)
- Deriving gradients by hand (will revisit this) is infeasible

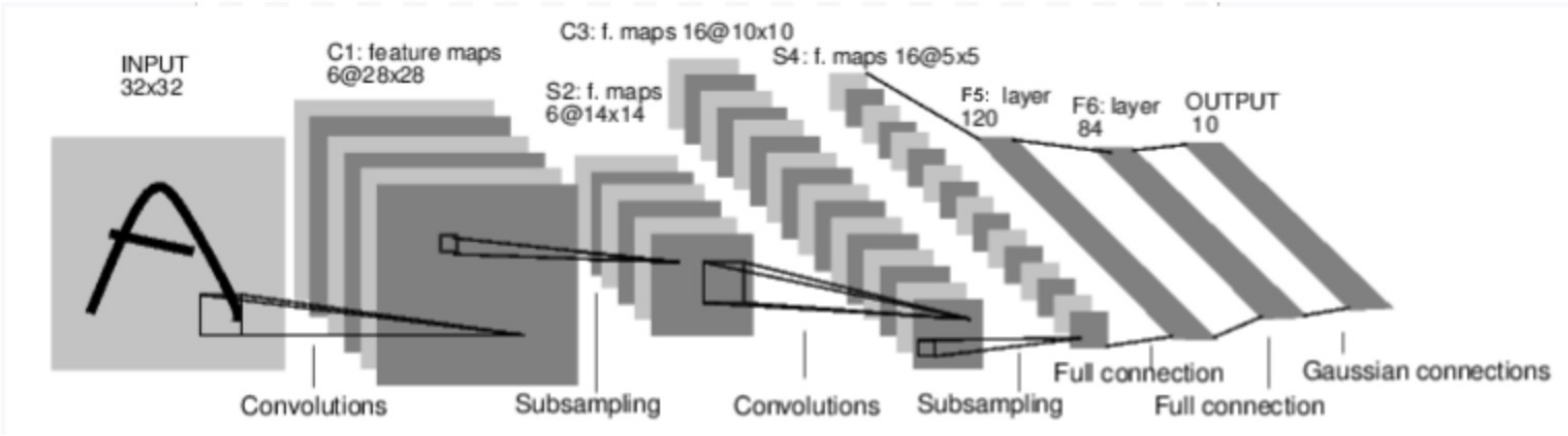


PyTorch: Computation Graph

- z=torch.sum(x*y)
- When we call z.backward()
  - Populates a field .grad
  - Avoids deriving gradient by hand



Example



ConvNet example from [official tutorial page]

How we implement

- Create a class inheriting torch.nn.Module
- implement at least two functions
  - __init__(self, **kwargs):
    defines all necessary model parameters, variables
  - forward(self, inputs: torch.Tensor):
    creates computation graph

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

Example from [official tutorial page]

PyTorch: Loss and Optimizer

- Cross-entropy loss: ℓ(data label, predicted label)
  - Off-the-shelf, inherits torch.nn.Module

```
criterion = nn.CrossEntropyLoss()
```

- Gradient based optimizer
  - Off-the-shelf

```
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

PyTorch: Training

```
for epoch in range(2):  # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:    # print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

Don’t forget this! Otherwise gradient will be accumulated

Calls the forward() function

Auto-grad happens here

Update deep net’s parameters

Agenda

- About this class
- Deep Learning nowadays
- Programming
- Math

Linear algebra: vectors

- n dimensional vector $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$
- If $\boldsymbol{x}$ is real (common in deep learning), we can write $\boldsymbol{x} \in \mathbb{R}^n$
- Inner (dot) product between n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\boldsymbol{x} \cdot \boldsymbol{y} \equiv \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$$

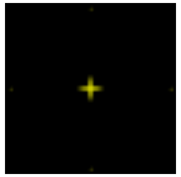
- We call $\boldsymbol{x}$ and $\boldsymbol{y}$ *orthogonal* if $\boldsymbol{x} \cdot \boldsymbol{y} = 0$

Linear algebra: vector norms, distances

- L^p norm for $p \in \mathbb{R}, p \geq 1$

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}$$

- For $0 < p < 1$, the above is not a norm
- L^0 norm: number of non-zero elements in $\boldsymbol{x}$
- Unit norm ball for p in 0.1 to 2
- L^p distance between $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$: $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p$
- Cosine similarity: $\frac{\boldsymbol{x}_1 \cdot \boldsymbol{x}_2}{\sqrt{\boldsymbol{x}_1 \cdot \boldsymbol{x}_1} \cdot \sqrt{\boldsymbol{x}_2 \cdot \boldsymbol{x}_2}}$



(from wiki)

Linear algebra: matrices

- Matrix multiplication

$\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, then $\boldsymbol{AB}$ is $m \times p$:

$$(\boldsymbol{AB})_{i,j} = \sum_{k=1}^{n} \boldsymbol{A}_{ik} \boldsymbol{B}_{kj}$$

- diagonal matrix: off-diagonal elements are all zero
- Identity matrix $\boldsymbol{I}$: diagonal matrix with all 1's on its diagonal
- Inverse of a square matrix $\boldsymbol{A}$: $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$
- Transpose of a matrix: $\boldsymbol{A}^{T}$

Special Matrices

- Orthogonal matrix: a square matrix $\boldsymbol{A}$ such that
$$\boldsymbol{A}^T = \boldsymbol{A}^{-1}$$
- Symmetric matrix: a square matrix $\boldsymbol{A}$ such that
$$\boldsymbol{A} = \boldsymbol{A}^T$$
- Toeplitz matrix: a square matrix where each diagonal is constant

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

Special Matrices

- Circulant matrix: a Toeplitz matrix, each row being a circulant shift of the preceding

$$\begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

- Givens rotation matrix (we restrict to 2D): the operator that rotates a 2D vector counter clockwise by θ

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

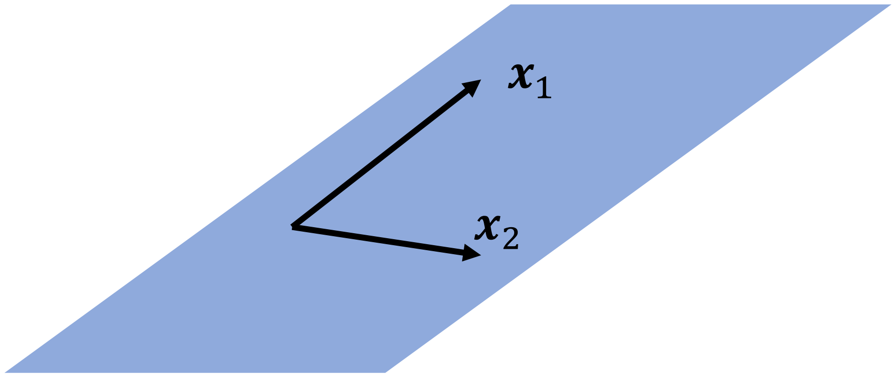
Linear dependency

- Consider a set of vectors $[\boldsymbol{x}_1, \dots \boldsymbol{x}_n]$, where $\boldsymbol{x}_i \in \mathbb{R}^m$
- **Linearly independent:** none of them can be written as the linear combination of the rest
  - Question: what if $n > m$
- span$(\boldsymbol{x}_1, \dots \boldsymbol{x}_n)$ is the set of all linear combinations of $\boldsymbol{x}_i$'s,

$$\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i \,, \forall \alpha_i \in \mathbb{R}$$

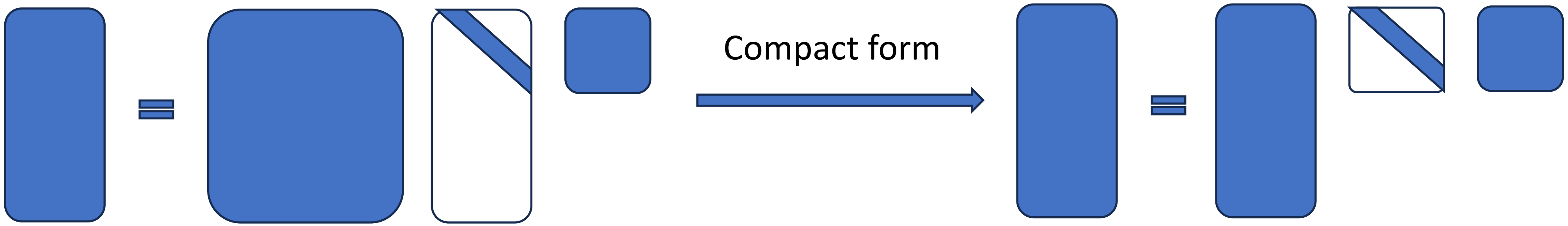
Subspace

- Consider $n < m$, and Linearly independent $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
- span($\boldsymbol{x}_1, \ldots \boldsymbol{x}_n$) defines a subspace of $\mathbb{R}^m$
- $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ is a basis for the subspace
- **Intrinsic** dimension is n, e.g., $m = 3, n = 2$, 2D plane in $\mathbb{R}^3$



Singular Value Decomposition (SVD)

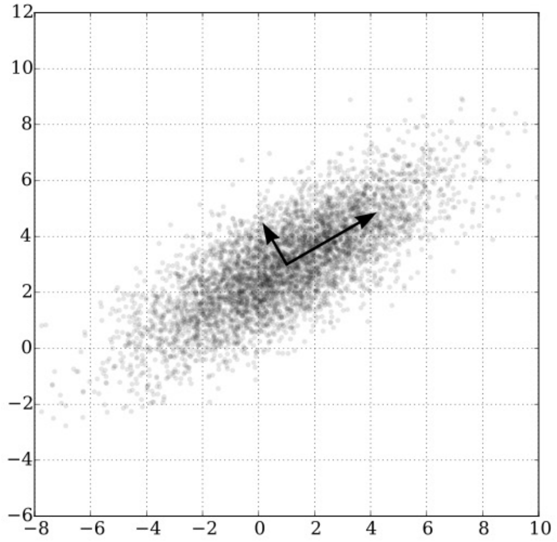
- Any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ can be factorized into $\boldsymbol{A} = \boldsymbol{U \Sigma V}^T$
- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V} \in \mathbb{R}^{n \times n}$ orthogonal
- $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min(m,n)})$, $\sigma_i \geq 0$ in descending order
- Illustration



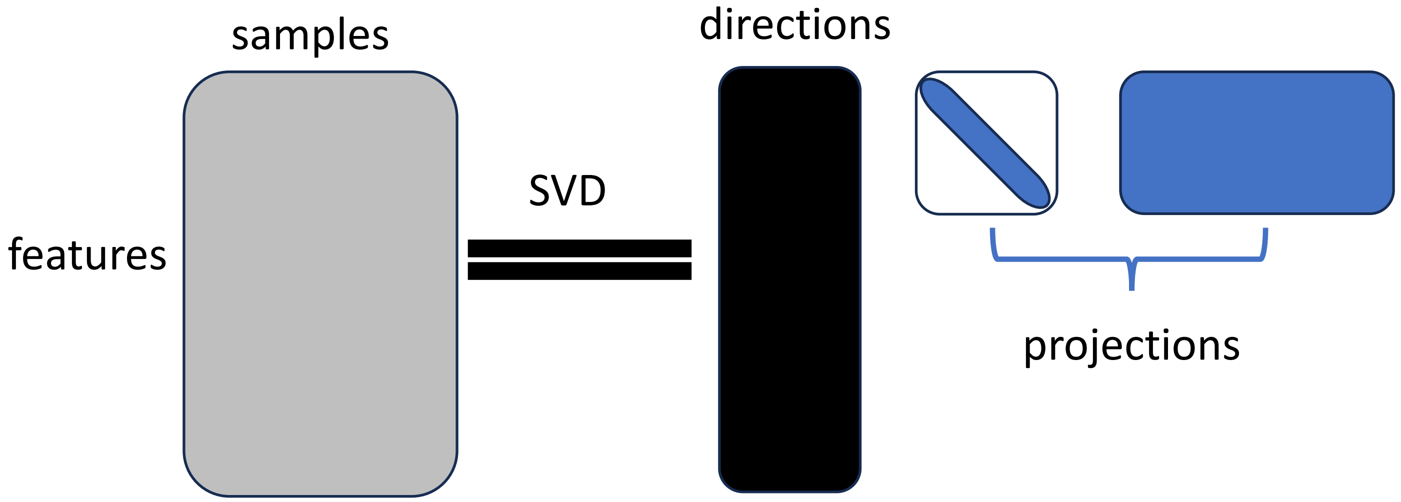
Practical Meaning of SVD

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T = \sum_{i=1}^{\min(m,n)} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$, $\sigma_i \geq 0$ in descending order
- Consider multiplying $\boldsymbol{A}$ to a vector $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{Ax}$:
  - $\boldsymbol{x}$ along $\boldsymbol{v}_1$ amplified the most, by σ_1
  - $\boldsymbol{v}_2$: the orthogonal direction amplified the most, by σ_2
  - $\boldsymbol{v}_3$: the orthogonal (to $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$) direction amplified the most, by σ_3
  - …
  - If $\sigma_i = 0$, then $\boldsymbol{x}$ along $\boldsymbol{v}_i$ is nulled
- How to get the basis from linearly dependent vectors?
  - Collect them as columns of $\boldsymbol{A}$, run SVD
  - The $\boldsymbol{u}_i$'s such that $\sigma_i > 0$

Principal Component Analysis (PCA)



- For i=1, …, reduced_dimension:
  - Find next orthogonal direction with biggest variance
  - Project feature to this direction
- Easily solved by singular value decomposition (SVD)



Norm

- 1-norm: $\|\boldsymbol{A}\|_1 = \max_j \sum_i \left|\boldsymbol{A}_{i,j}\right|$
- ∞-norm: $\|\boldsymbol{A}\|_\infty = \max_i \sum_j \left|\boldsymbol{A}_{i,j}\right|$
- Spectral norm $\|\boldsymbol{A}\|_2 = \sigma_1$
- Frobenius norm $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j} {\boldsymbol{A}_{i,j}}^2} = \sqrt{\sum_i \sigma_i^2}$
- Nuclear-norm $\|\boldsymbol{A}\|_* = \sum_i \sigma_i$
- Rank$(\boldsymbol{A})$: number of non-zero σ_i's

Eigen Value Decomposition (EVD)

- Square matrix $\boldsymbol{A} \in \mathbb{R}^{m \times m}$, decompose $\boldsymbol{A} = \boldsymbol{U\Lambda U}^{-1}$
- $\boldsymbol{\Lambda} = diag\{\lambda_1, \dots, \lambda_m\}$ in descending order
- Symmetric $\boldsymbol{A}$ (common for this class), then
  - $\boldsymbol{U}$ is orthogonal, and $\boldsymbol{A} = \boldsymbol{U\Lambda U}^T$
- When all $\lambda_i \geq 0$
  - we call $\boldsymbol{A}$ positive semi definite (PSD), denoted as $\boldsymbol{A} \succcurlyeq 0$
  - EVD coincides with SVD

Illustration from [Chapter 2 of deep learning]

Linear Least Squares

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

- $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{x} \in \mathbb{R}^n, m \geq n$
- $\boldsymbol{x}^* = \boldsymbol{A}^\dagger \boldsymbol{b}$, where $\boldsymbol{A}^\dagger = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T$ is pseudo-inverse
- What if $m < n$?
  - More than one solution
  - Usually impose some **regularizer** on $\boldsymbol{x}$, e.g., L^1 or L^2 norm
  - $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|^2 \Rightarrow \boldsymbol{x}^* = (\boldsymbol{A}^T \boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{A}^T \boldsymbol{b}$ (ridge regression)
  - $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|_1$ (Lasso)

Gradient and Hessian

- Many learning problems aim at

$$\min_{\boldsymbol{w}} \sum_i \ell(\boldsymbol{w}, \boldsymbol{x}_i)$$

- Many solvers will involve gradient

$$\nabla \ell(\boldsymbol{w}, \boldsymbol{x}_i) = \frac{\partial \ell}{\partial \boldsymbol{w}}$$

- Sometimes we will also talk about 2nd order gradient (Hessian)

$$\boldsymbol{H} = \frac{\partial^2 \ell}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}$$

Matrix Calculus

- Calculate derivative of function defined on matrices/vectors
- $f: \mathbb{R}^m \longmapsto \mathbb{R}$: ∇f is $\mathbb{R}^m$
- $f: \mathbb{R}^{m \times n} \longmapsto \mathbb{R}$: ∇f is $\mathbb{R}^{m \times n}$
- $f: \mathbb{R}^m \longmapsto \mathbb{R}^n$: ∇f is $\mathbb{R}^{m \times n}$ (Jacobian)
- Hessian:
  - we typically deal with $f: \mathbb{R}^m \longmapsto \mathbb{R}$ only
  - $m \times m$ matrix
- I often check this wikipage

Chain rule

- For an L-layer network, $\ell(\boldsymbol{w}, \boldsymbol{x})$ is a composite function, i.e.,

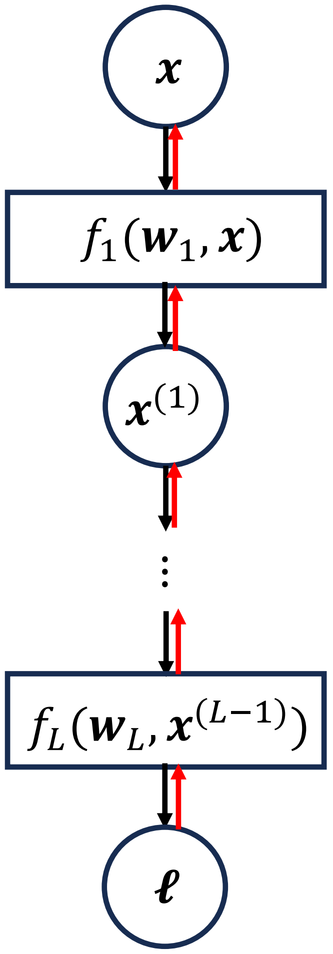
$$\boldsymbol{x}^{(1)} = f_1(\boldsymbol{w}_1, \boldsymbol{x})$$
$$\boldsymbol{x}^{(2)} = f_2(\boldsymbol{w}_2, \boldsymbol{x}^{(1)})$$
$$\vdots$$
$$\ell = f_L(\boldsymbol{w}_L, \boldsymbol{x}^{(L-1)})$$

- Chain rule and back-propagation:

$$\frac{\partial \ell}{\partial \boldsymbol{x}^{(l)}} = \frac{\partial \ell}{\partial \boldsymbol{x}^{(l+1)}} \cdot \frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}}$$

$$\frac{\partial \ell}{\partial \boldsymbol{w}_l} = \frac{\partial \ell}{\partial \boldsymbol{x}^{(l)}} \cdot \frac{\partial \boldsymbol{x}^{(l)}}{\partial \boldsymbol{w}_l}$$

Jacobian

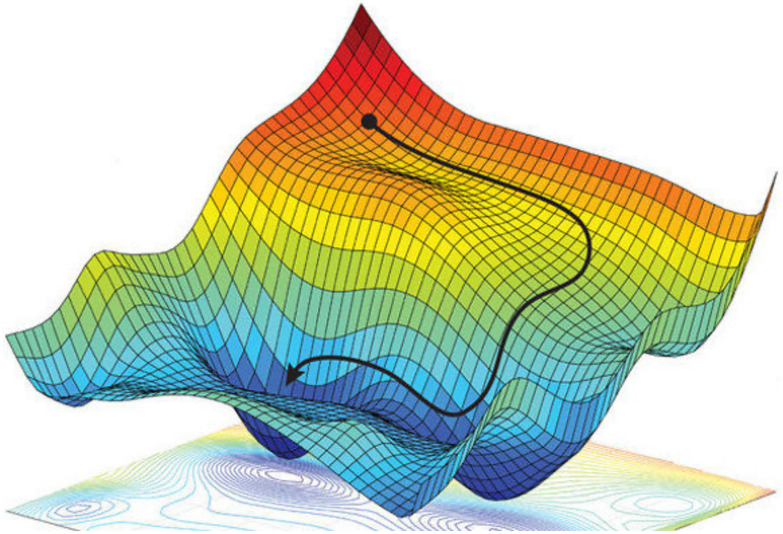


Gradient Descent (GD)

For step $s = 0, 1, \ldots$

$$\boldsymbol{w}(s+1) = \boldsymbol{w}(s) - \eta \nabla \ell(\boldsymbol{w}(s)),$$

till $\nabla \ell(\boldsymbol{w}(s)) \approx 0$ (or max steps reached, or loss reduction is tiny)



Stochastic Gradient Descent (SGD)

- we want to $\min_{\boldsymbol{w}} \sum_{i=1}^{N} \ell(\boldsymbol{w}, \boldsymbol{x}_i)$
- If N is big, Compute $\nabla \ell(\boldsymbol{w}, \boldsymbol{x}_i)$ for all i's can be costly
- So we sample a mini-batch of i's each step
- Starting from $\boldsymbol{w}(0)$, for step $s = 0, 1, \ldots$

$$\boldsymbol{w}(s+1) = \boldsymbol{w}(s) - \eta \sum_{i \in \mathcal{B}} \nabla \ell(\boldsymbol{w}(s), \boldsymbol{x}_i) \, ,$$

  till stopping criteria met
- "noisy" steps

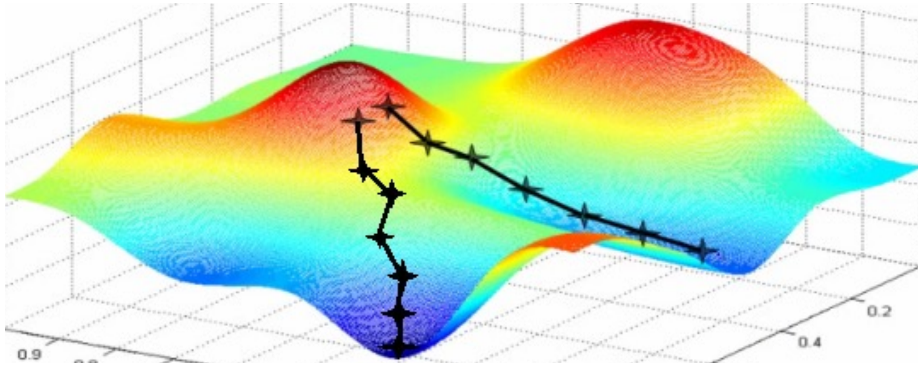


Illustration from this [page]

Probability Distribution

- Discrete random variable
  - Probability Mass Function (PMF): $P(X = x_i)$
  - $P(X = x_i) \geq 0$ and $\sum_i P(X = x_i) = 1$
- Continuous random variable
  - Probability Density Function (PDF): $p(x)$
  - $p(x) \geq 0$ and $\int_x \; p(x) = 1$
- The above can be generalized to random vectors
- Cumulative Distribution Function (CDF) for real valued random variable
  - $P(x) = \mathrm{Pr}(X \leq x)$
  - Continuous case: $p(x) = \frac{dP(x)}{dx}$

Expectation, Variance, Covariance

- Expectation $\mathbb{E}_{X\sim p}[f(X)] = \int f(x)p(x)dx$
- Variance $Var_{X\sim p}[f(X)] = \mathbb{E}_{X\sim p}[(f(X) - \mathbb{E}_{X\sim p}[f(X)])^2]$
- Covariance

$$Cov\big(f(X), g(Y)\big) = \mathbb{E}[(f(X) - \mathbb{E}[f(X)])(g(Y) - \mathbb{E}[g(Y)])]$$

- Covariance matrix of random vector $\boldsymbol{x} \in \mathbb{R}^n$

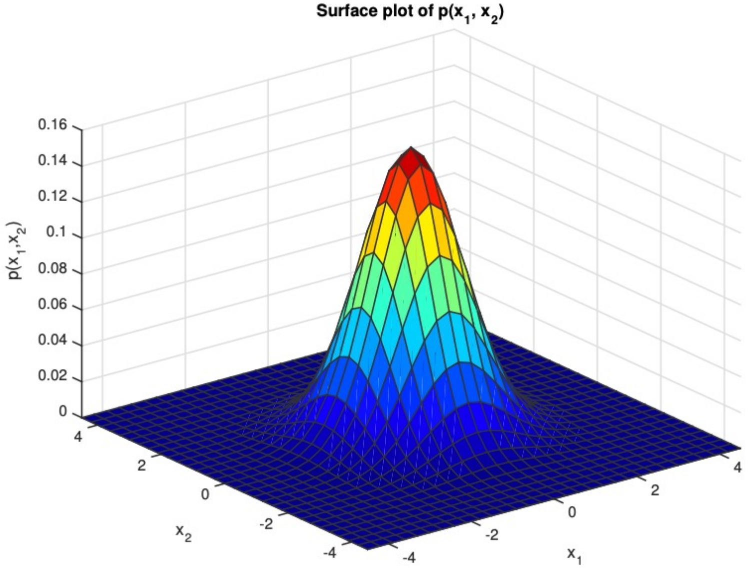
$$Cov(\boldsymbol{x})_{\,i,j} = Cov(x_i, x_j)$$

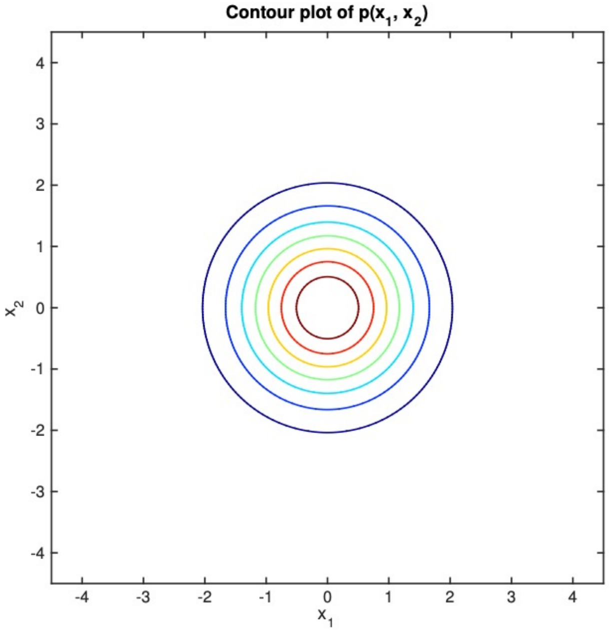
$$Cov(\boldsymbol{x})_{i,i} = Var(x_i)$$

(multivariate) Gaussian Distribution

- $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \in \mathbb{R}^n$, where $\boldsymbol{\Sigma} \succcurlyeq \mathbf{0}$
- $p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$
- Example: 2D-spherical Gaussian
  - $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
  - Correlation coefficient

$$\rho = \frac{\sigma_{12}}{\sqrt{\sigma_{11} \cdot \sigma_{22}}} = 0$$

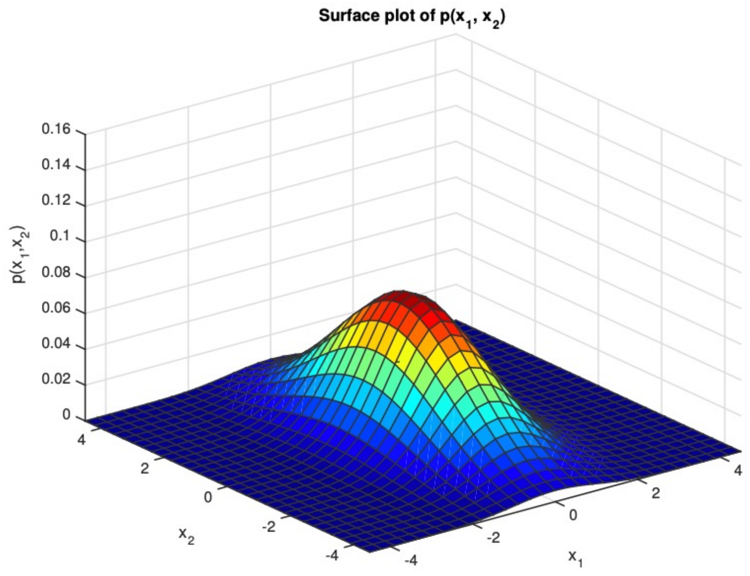


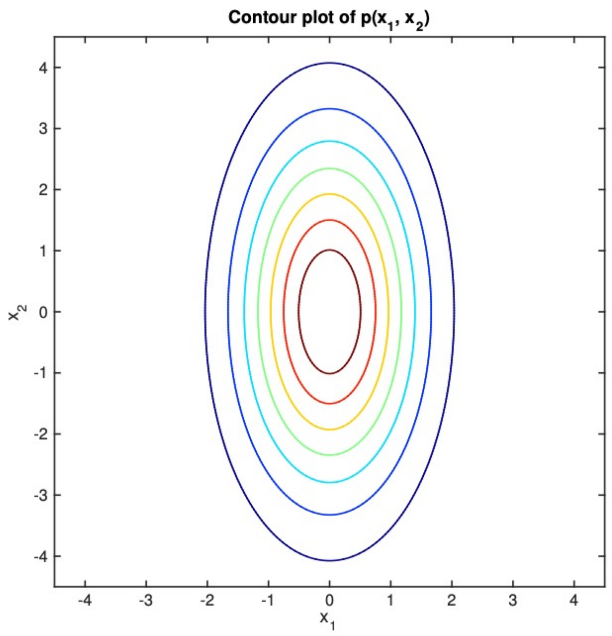


Plots from these slides

(multivariate) Gaussian Distribution

- Example:
  - $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$
  - $\rho = 0$

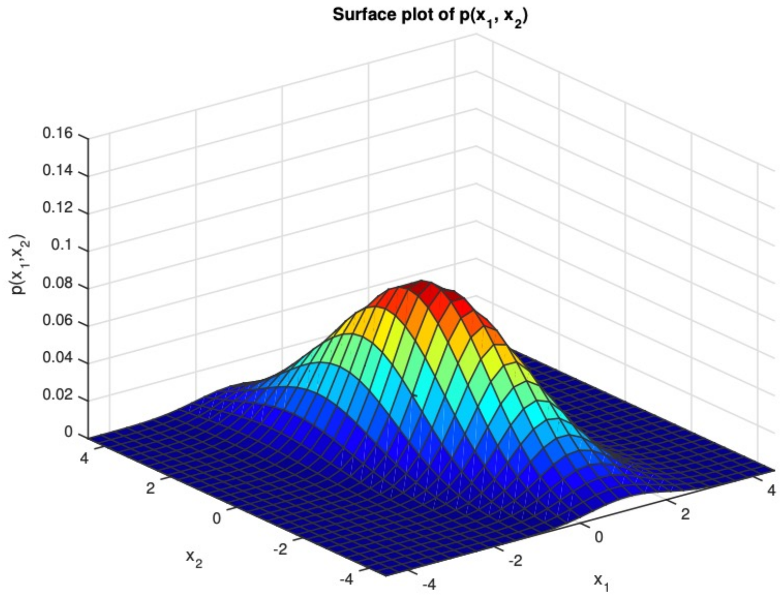


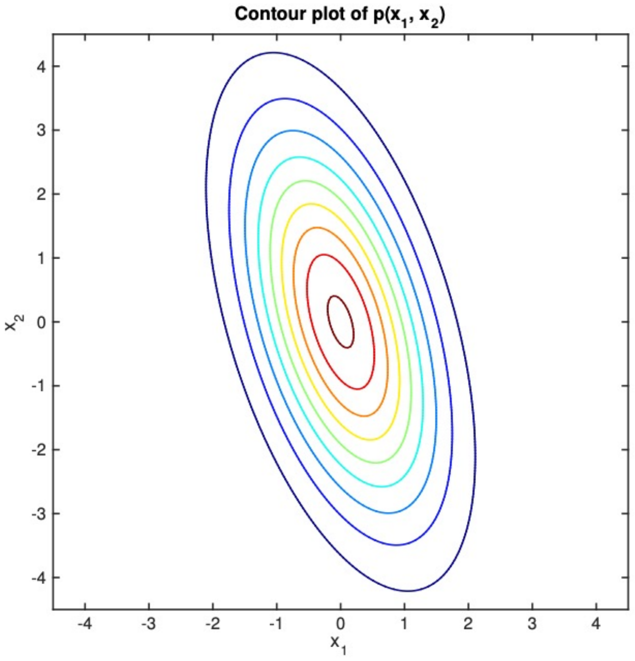


Plots from these [slides]()

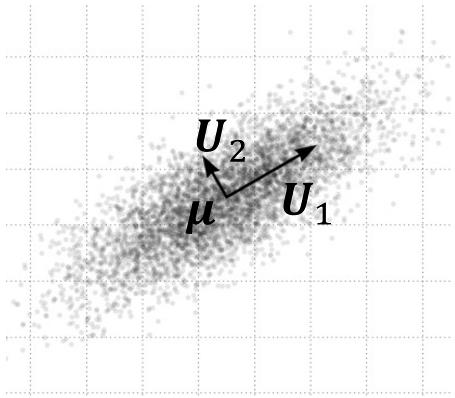
(multivariate) Gaussian Distribution

- Example
  - $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix}$
  - $\rho = -0.5$





- More generally, contour ellipsoid (centered at $\boldsymbol{\mu}$) in $\mathbb{R}^n$
  - Let eigen decomposition $\boldsymbol{\Sigma} = \boldsymbol{U \Lambda U}^T$
  - Axes along columns $\boldsymbol{U}_i$'s. Axis length related to λ_i's



Plots from these slides

(multivariate) Gaussian Distribution

- How do we estimate the $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ given data $\boldsymbol{x}_i$'s?
- Maximum Likelihood Estimator (MLE)

$$(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}) = \arg\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \Longrightarrow$$

$$\widehat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i,$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^T$$

Conditional Probability

- $P(Y = y|X = x) = \frac{P(X=x,Y=y)}{P(X=x)}$
- "Lazy" expression: $p(y|x) = \frac{p(x,y)}{p(x)}$
- Bayes Rule

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)}$$

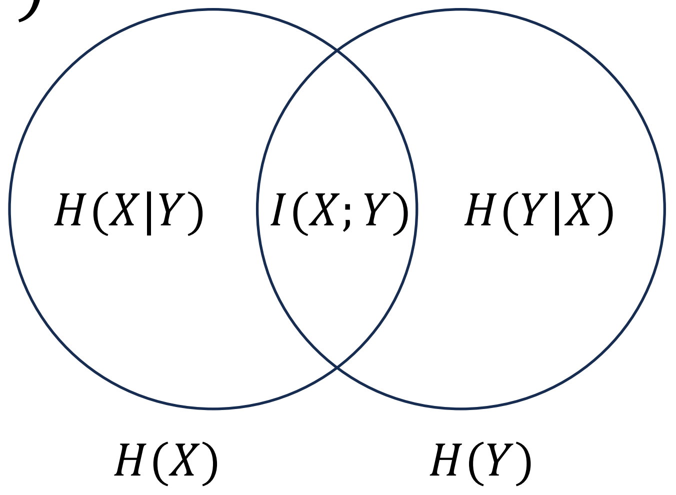
- **Marginalize** $p(x) = \int p(x, y)dy$
- Independence: $p(x, y) = p(x)p(y)$

Entropy

- Shannon Entropy: $H(X) = \mathbb{E}_{X\sim p}[-\log p(X)]$
  - sometimes denoted as $H(p)$
  - bits to encode a random variable, given its pdf
- Cross Entropy $H(p,q) = \mathbb{E}_{X\sim p}[-\log q(X)]$
  - Bits to encode a random variable, according to a different pdf, $q(\cdot)$
  - Need more bits than $H(p)$
- KL divergence $KL(p||q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] = H(p,q) - H(p)$
  - Asymmetric $KL(p||q) \neq KL(q||p)$
  - In supervised learning, we often have p as ground-truth label, and q as model prediction
  - Why not the other way?

Mutual Information

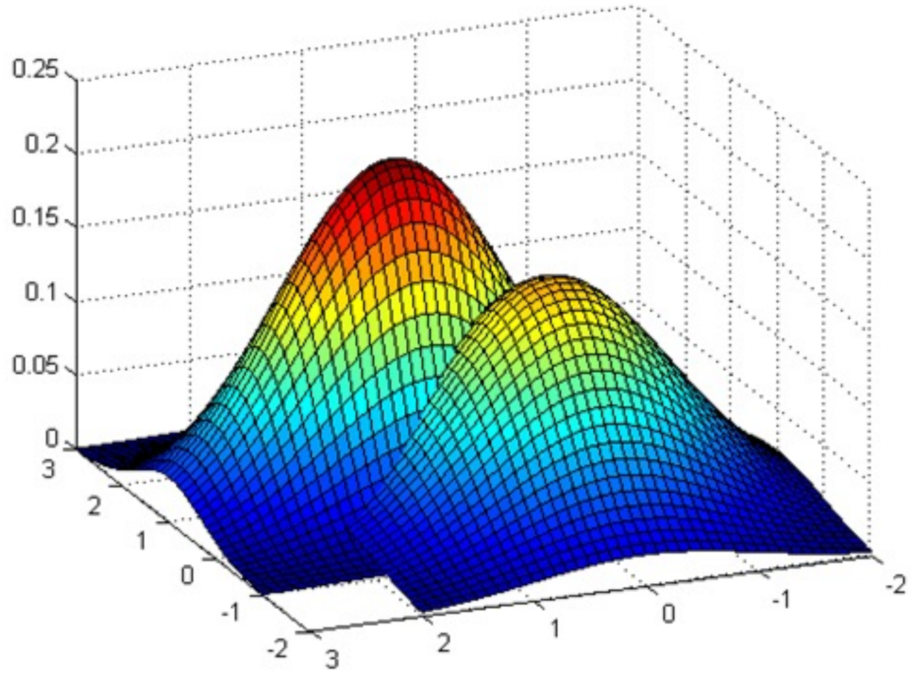
- Joint entropy $H(X, Y) = \mathbb{E}_{x,y\sim p(x,y)}[-\log p(x, y)]$
- Conditional Entropy $H(Y|X) = \mathbb{E}_{x,y\sim p(x,y)}[-\log p(y|x)]$
    - How much uncertainty left in Y given X
- $I(X; Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$
  $= H(X) + H(Y) - H(X, Y)$
- $I(X; Y) = KL(p(X, Y)||p(X)p(Y))$
- $I(X; Y) = 0$: X and Y are independent



more Math, and classical v.s. DL

Gaussian Mixture Model (GMM)

- Modeling
  - multiple speakers (speaker identification)
  - Pronunciations (speech recognition)
  - Human faces (face identification)
  - …
- latent variable $c \sim p(c)$
- Component pdf: $p(\boldsymbol{x}|c) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
- $p(\boldsymbol{x}) = \sum_{c=1}^{C} p(c) p(\boldsymbol{x}|c)$



Pdf of a Gaussian mixture with 2 components

MLE for GMM

- Maximum Likelihood Estimator for $\{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\}$'s
- Via Expectation-Maximization (EM) algorithm:
- Initialize $\{\boldsymbol{\mu}_c(0), \boldsymbol{\Sigma}_c(0)\}$ for $c = 1, \ldots C$. Denoted as $\boldsymbol{\theta}(0)$
- for step $s = 0,1, \ldots$
  - E-step: estimate $p(c|\boldsymbol{x}_i, \boldsymbol{\theta}(s))$ for all i's, then get expected log-likelihood

$$\sum_i \mathbb{E}_{c \sim p(c|\boldsymbol{x}_i, \boldsymbol{\theta}(s))}[\log p(\boldsymbol{x}_i, c|\boldsymbol{\theta})]$$

  - M-step: $\boldsymbol{\theta}(s+1) \leftarrow$ maximizer of the above

Simple application of EM: k-means

- Assume $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$ where σ^2 is some constant that we don't care
- We only need to estimate cluster centers $\boldsymbol{\mu}_c$, for $c = 1, \dots, C$
- Randomly initialize guessed $\{\boldsymbol{\mu}_c(0)\}_{c=1}^{C}$.
- For step $s = 0,1, \dots$
  - E-step: calculate cluster assignment for each sample $\boldsymbol{x}_i$:

$$p(c|\boldsymbol{x}_i, \{\boldsymbol{\mu}_c(s)\}_{c=1}^{C})$$

    We may simplify by assuming 1-hot assignment
  - M-step: update centers $\{\boldsymbol{\mu}_c(s+1)\}_{c=1}^{C}$ based on the assignment

GMM v.s. DL

- GMM for speaker identification

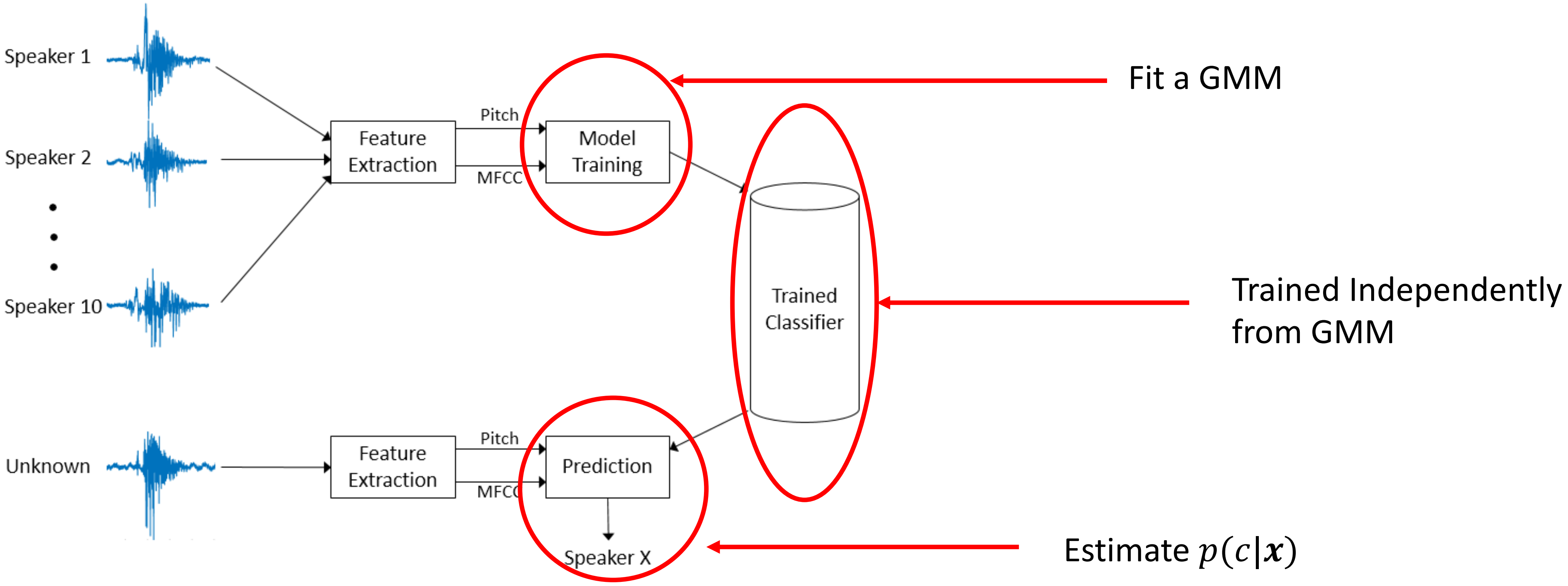
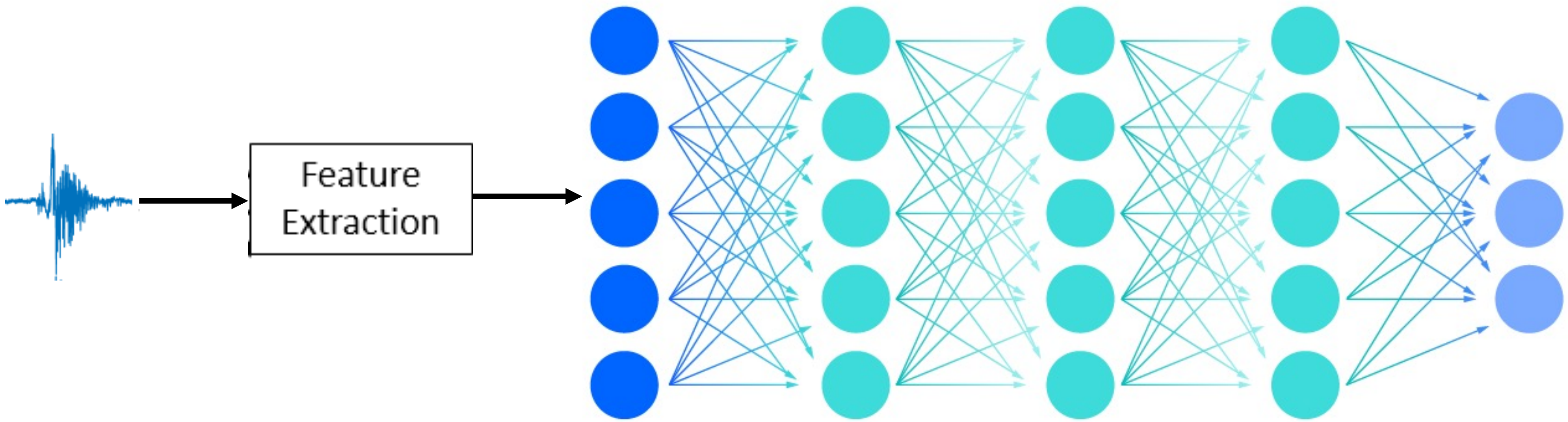


Illustration from this page

GMM v.s. DL

- DL for speaker identification
- Jointly trained model and classifier



GMM vs DL

Methods	Datasets	Overall Accuracy (%)
Proposed Deep GRU	Aishell-1	98.96%
	Aishell-1 with Gaussian white noise added	91.56%
MFCC-GMM [10]	Aishell-1	86.29%
	Aishell-1 with Gaussian white noise added	55.29%

So why DL approach outperforms?

Table from this paper