Introduction to Machine Learning and Deep Learning

Yi-Lun Du（杜轶伦）
Shandong Institute of Advanced Technology

July 18-19, Qingdao

QCD and Nuclear Physics Summer School

Outline

- Overview introduction to machine/deep learning
- Machine learning methods
- Deep learning methods
  - Fully-connected neural network
  - Convolutional neural network

So-far Achievements

- Superhuman Game Playing—AlphaGo
- Near-human-level Chatbot—ChatGPT
- Near-human-level speech recognition, translation
- Near-human-level image recognition
- Near-human-level autonomous driving
- In science—AlphaFold…

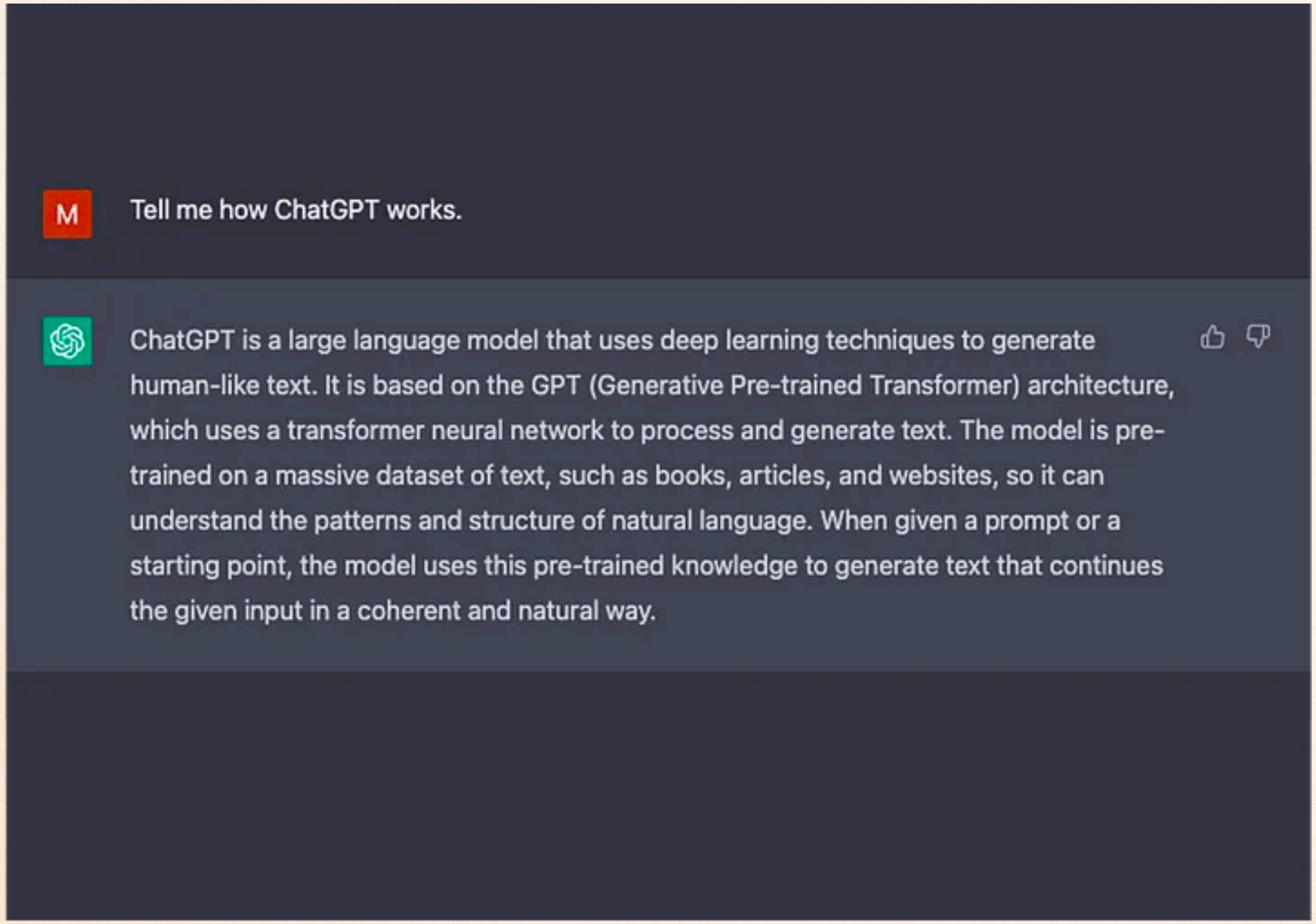
AlphaGo

AlphaGo 4 : Lee Sedol 1
Seoul, March 2016

AlphaGo Master 3 : Ke Jie 0
Wuzhen, May 2017

Google DeepMind, London
AlphaGo: Nature 529, 484-489 (28 January 2016)
Machine (deep) learning & Monte Carlo tree search techniques
AlphaGo Zero: Nature 550, 354-359 (19 October 2017)
Self-taught without human's knowledge

ChatGPT



- Write and debug computer programs, compose music, teleplays, fairy tales, and student essays; answer test questions (SAT), generate business ideas, write poetry and song lyrics, translate and summarize text…
- Refuse the counterfactual question or premise & reduce harmful and deceitful responses
- Remember a limited number of previous prompts in the same conversation

Object detection

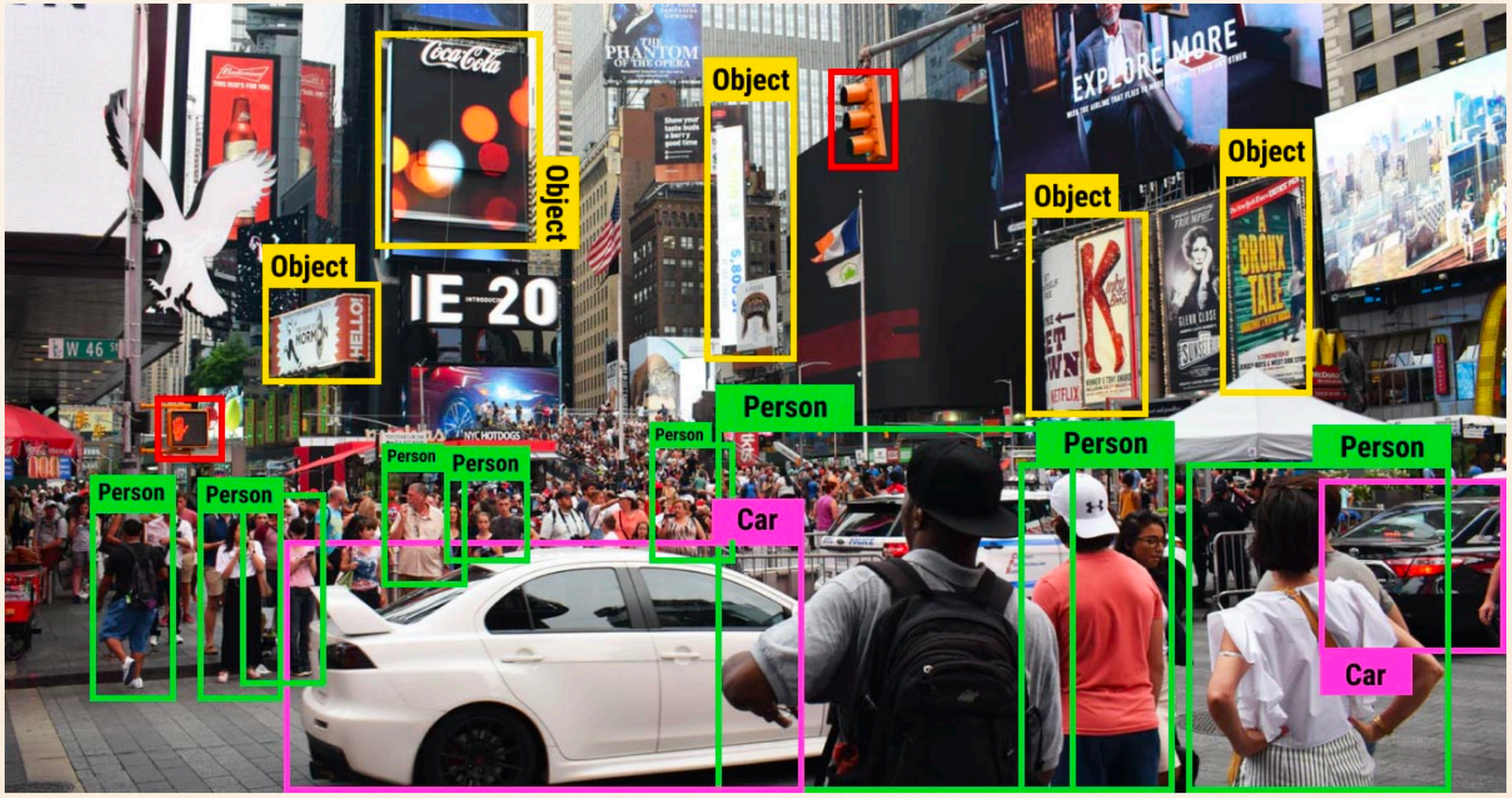
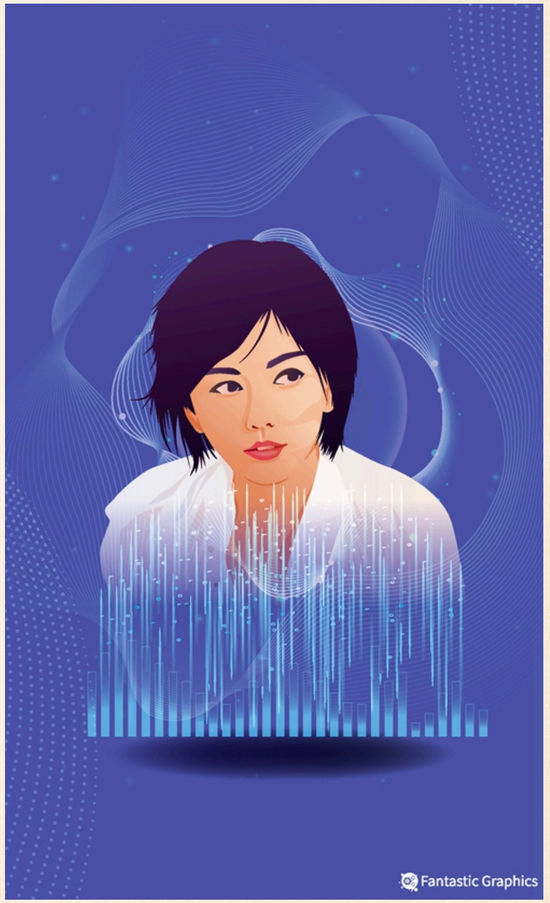


Figure from [The Startup Founder]

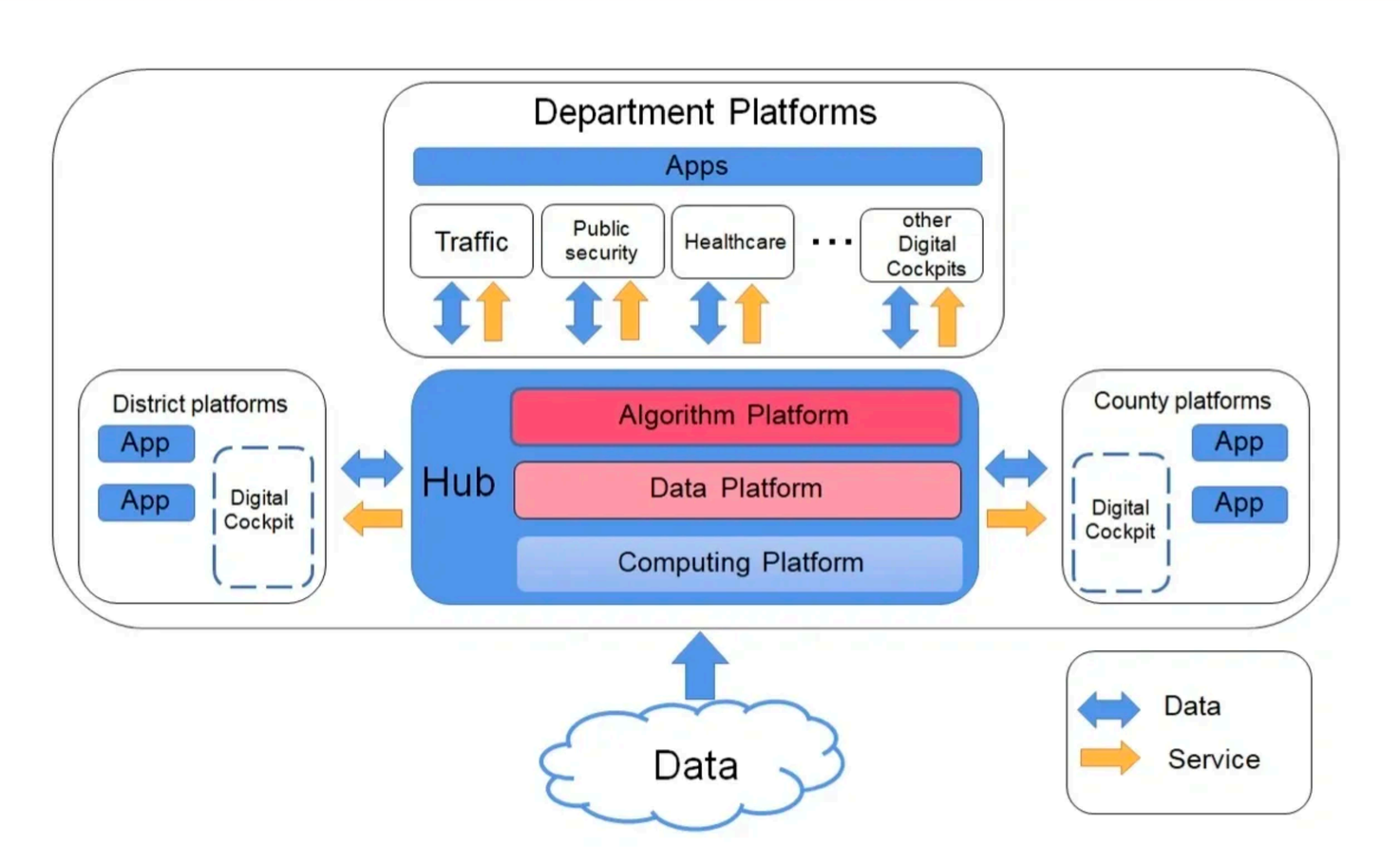
AI Art

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, arXiv:1508.06576



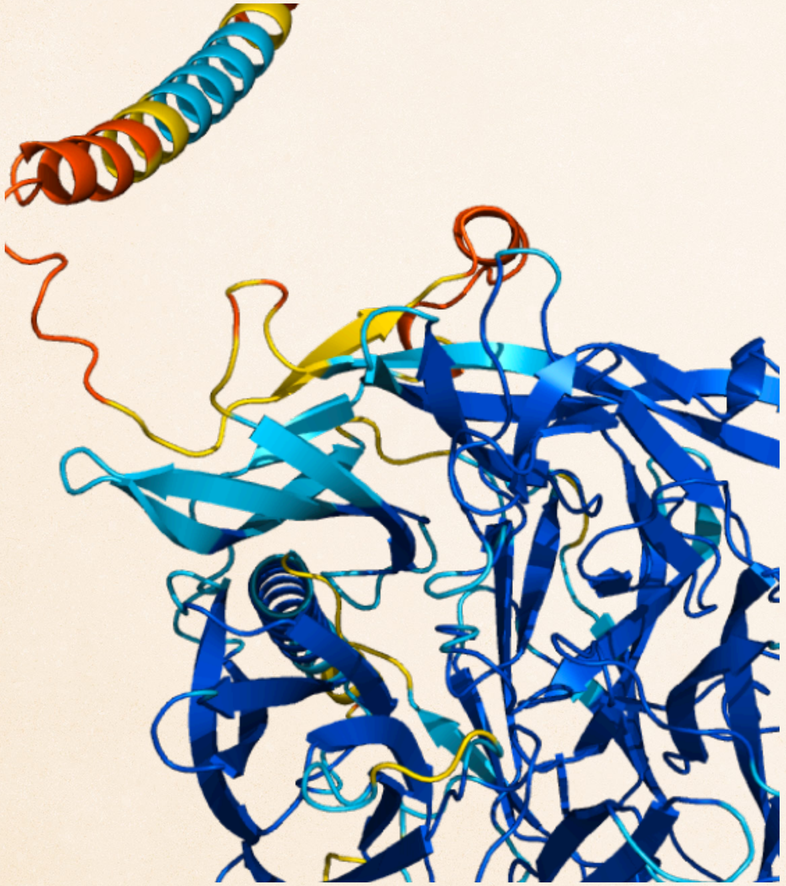
图片来源：中国青年报

Smart City



General view of the Hangzhou City Brain,
e.g., 10~15% speedup in city traffic

In Science: AlphaFold

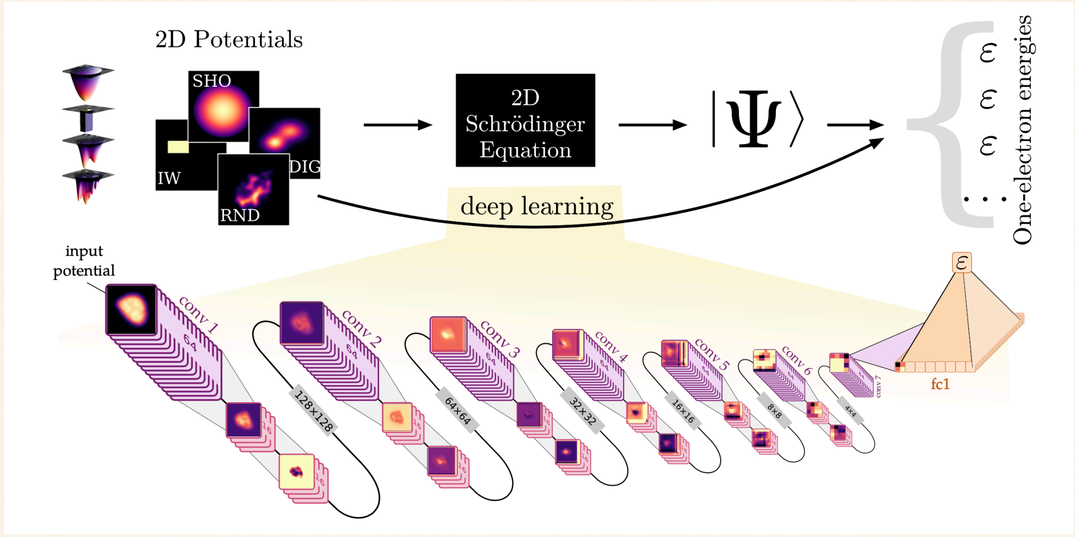


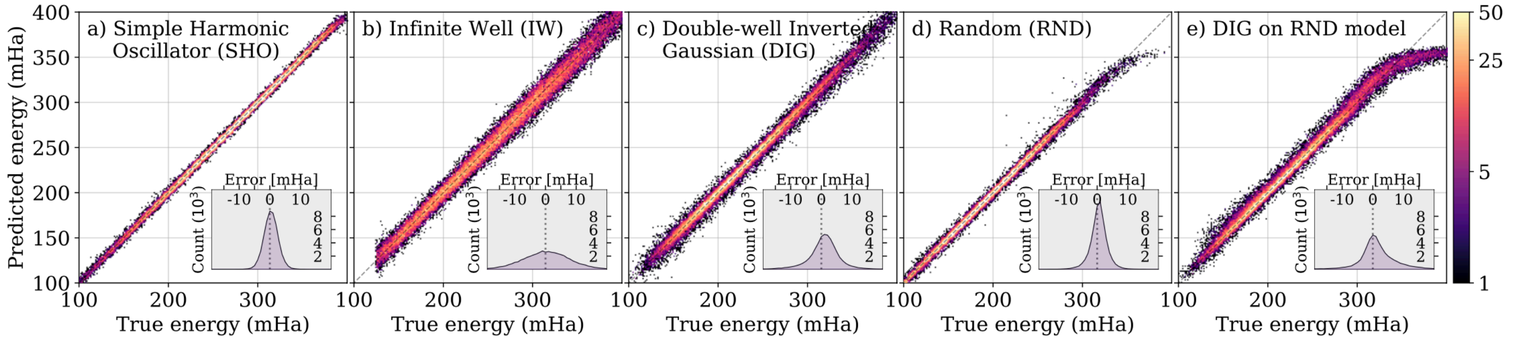
- It predicts a protein’s 3D structure from its amino acid sequence
- It regularly achieves accuracy competitive with experiment
- It provides open access to over 200 million protein structure predictions

Google DeepMind, Nature 596, 583–589 (2021)

Physics applications: quantum mechanics

Deep learning and the Schrödinger equation



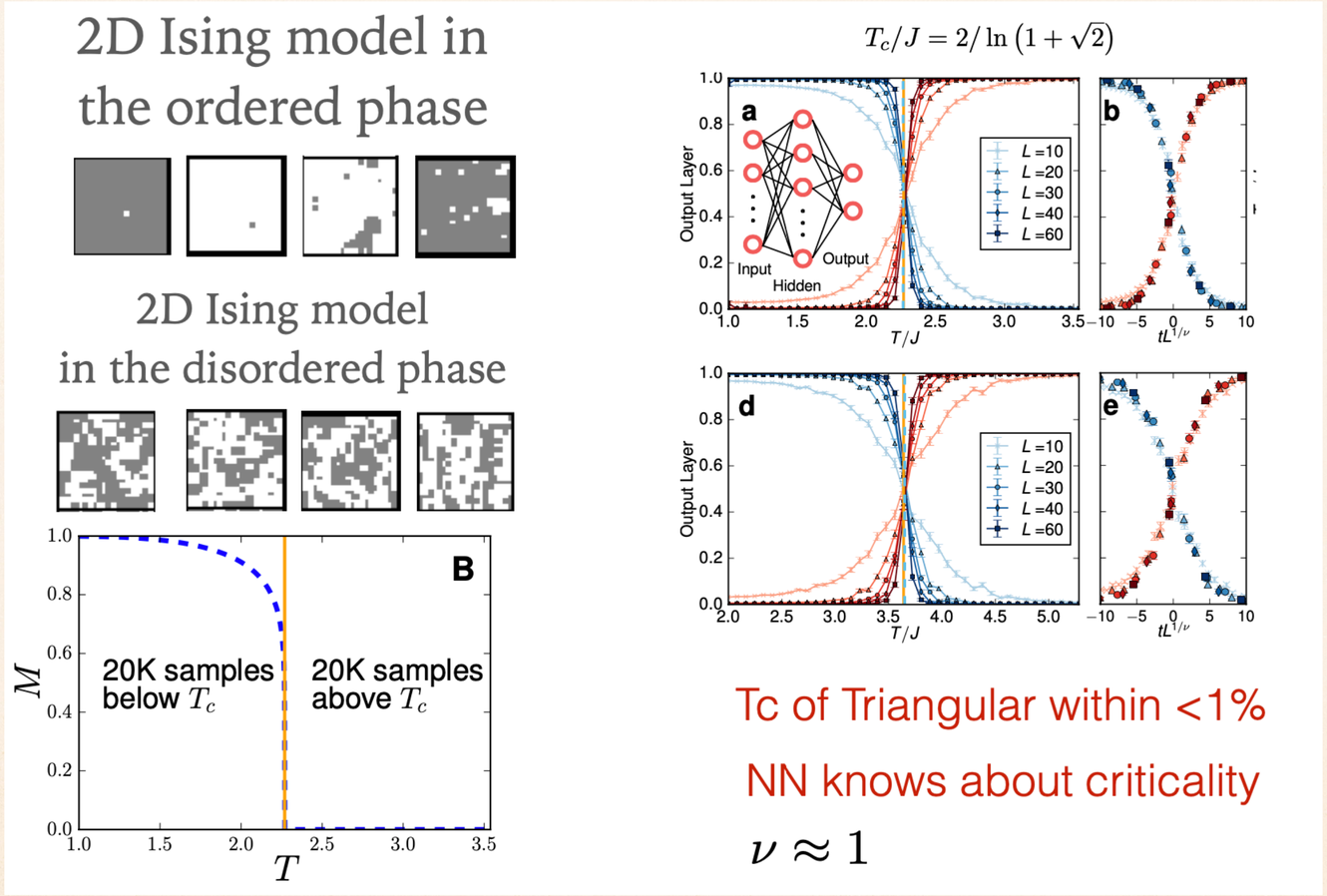


Kyle Mills, Michael Spanner, Isaac Tamblyn, Phys. Rev. A 96, 042113 (2017)

Physics applications: condensed matter physics

Machine learning phases of matter

$$H = -J\sum_{\langle ij\rangle} s_i s_j, \qquad s_i = \pm 1$$



Tc of Triangular within <1%

NN knows about criticality

$\nu \approx 1$

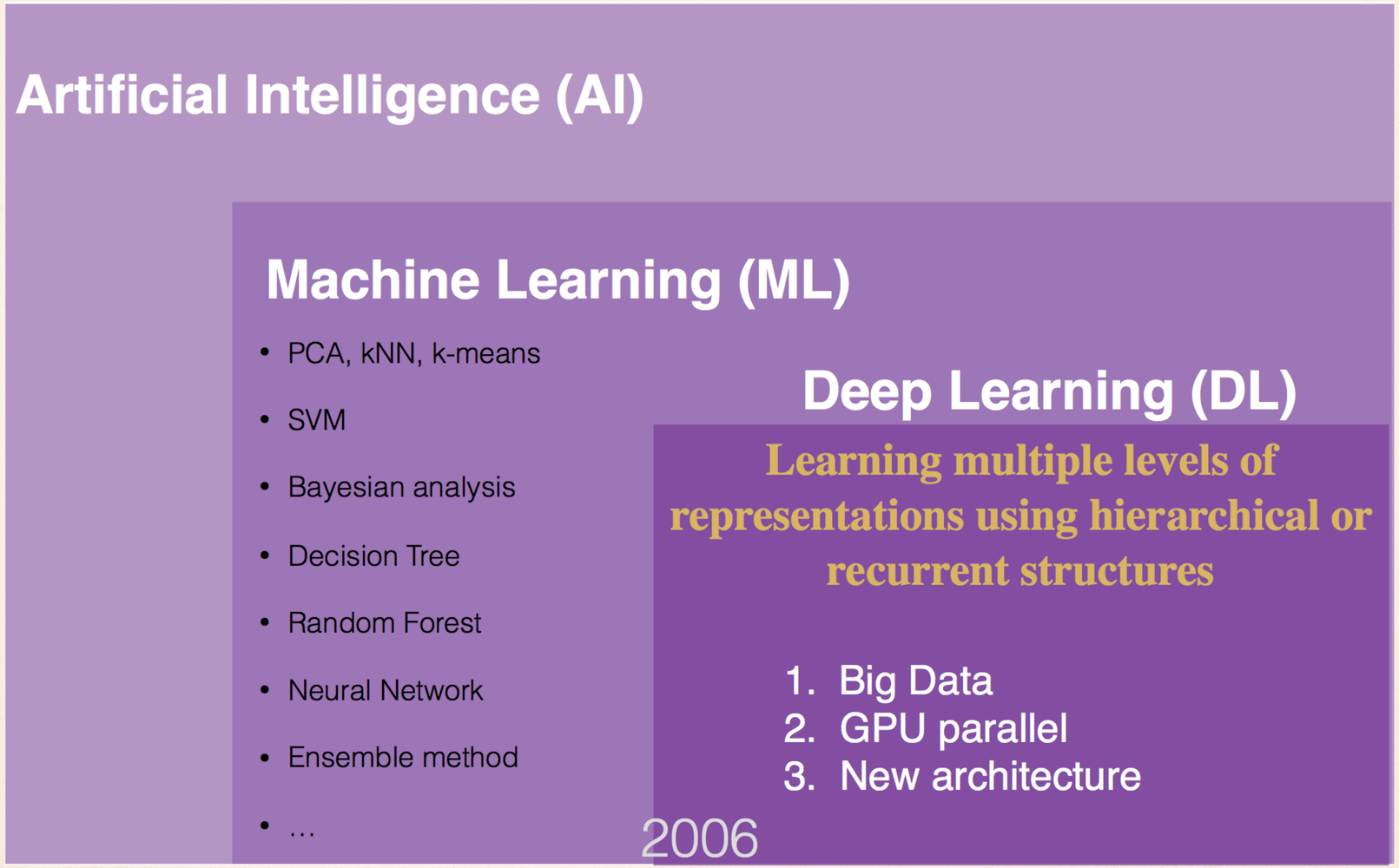
J. Carrasquilla and R. G. Melko, Nature Physics 13, 431-434 (2017)

Artificial Intelligence

- Wikipedia: Artificial intelligence (AI) is intelligence demonstrated by computers, as opposed to human or animal intelligence. "Intelligence" encompasses the ability to learn, reason, generalize, and infer meaning.

- The study of agents that perceive the world around them, form plans, and make decisions to achieve their goals. Its foundations include math, logic, philosophy, probability, neuroscience, and decision theory. It includes fields like machine learning, natural language processing, robotics, cognitive computing, computer vision…

What's ML, DL?



Geoffrey Hinton

Machine Learning

~looking for a function

- Subfield of AI : math principles and algorithms to achieve AI based on data.
- General definition: the field of study that gives computers the ability to learn without explicitly programmed. – Arthur Samuel, 1959
- Formal definition: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T as measured by P, improves with experience E. – Tom Mitchell 1997

The Curse of Dimensionality

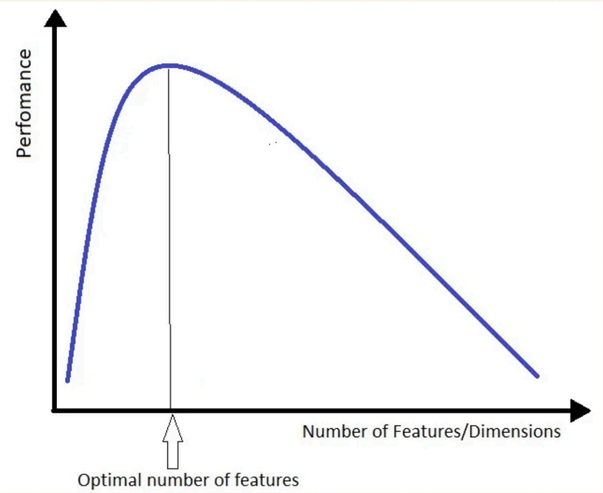


Figure from towards data science

- Many ML problems become exceedingly difficult when the number of dim. in the data is high. – # of possible distinct configs increases exponentially as the # of dim.
- Feature engineering is necessary but could be hard, incomplete and effort-consuming.
- How about handing over this part to machine as well?

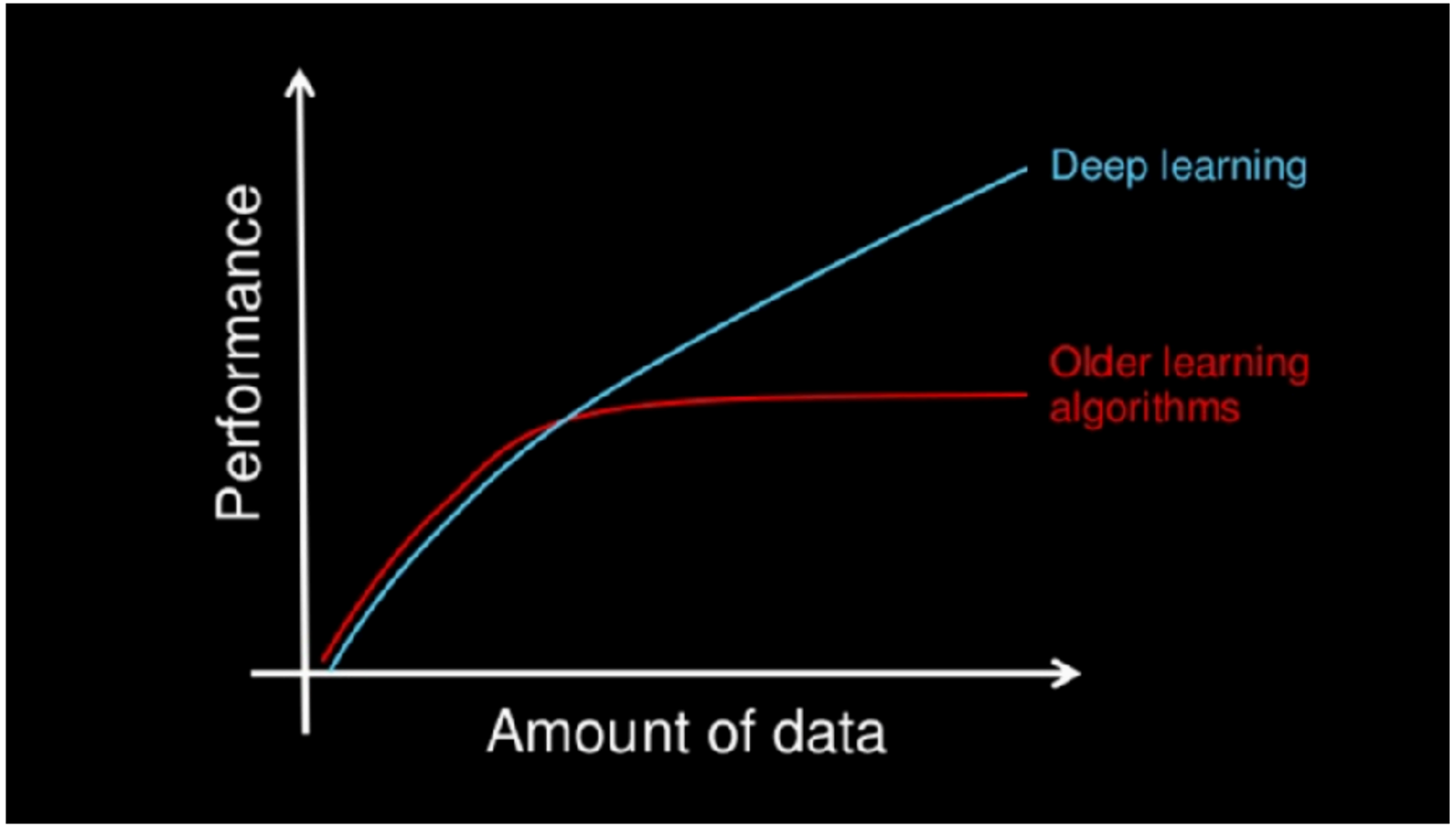
Deep Learning

- subset of ML: techniques for implementing ML in end-to-end way inspired by brains
- Feature engineering → Representation Learning
- Booming recently in theory and applications due to :
  - 1) Algorithm/architecture development
  - 2) Big-data
  - 3) Computing power (GPU)
- => Success of DL = Algorithm + Big-data + Computing-power

Universal Approximation Theorem (万能近似定理)

- Arbitrary *width* case: Multilayer feed-forward networks with as few as one hidden layer are universal approximators, provided that the network is given enough hidden units (Cybenko, 1989 (for sigmoid activation functions); Hornik et al., 1989).
  - Approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error
- It is not the specific choice of the activation function but rather the multilayer feed-forward architecture itself that gives neural networks the potential of being universal approximators (Hornik et al., 1991).
- Any continuous function on a closed and bounded subset of $\mathbb{R}^n$ is Borel measurable and therefore can be approximated by a NN.
- However, “universal” is just for exsistence but not for finding it out.
- Why we go to deeper?
  - Shallow NN might need (exponentially) more units;
  - Shallow NN overfit easier than deeper ones.
- Arbitrary *depth* case: the minimum width required for the universal approximation of $\mathbb{L}^p$ functions using feedforward neural networks with ReLU as activation functions (Park et al, 2021).

Machine learning VS deep learning



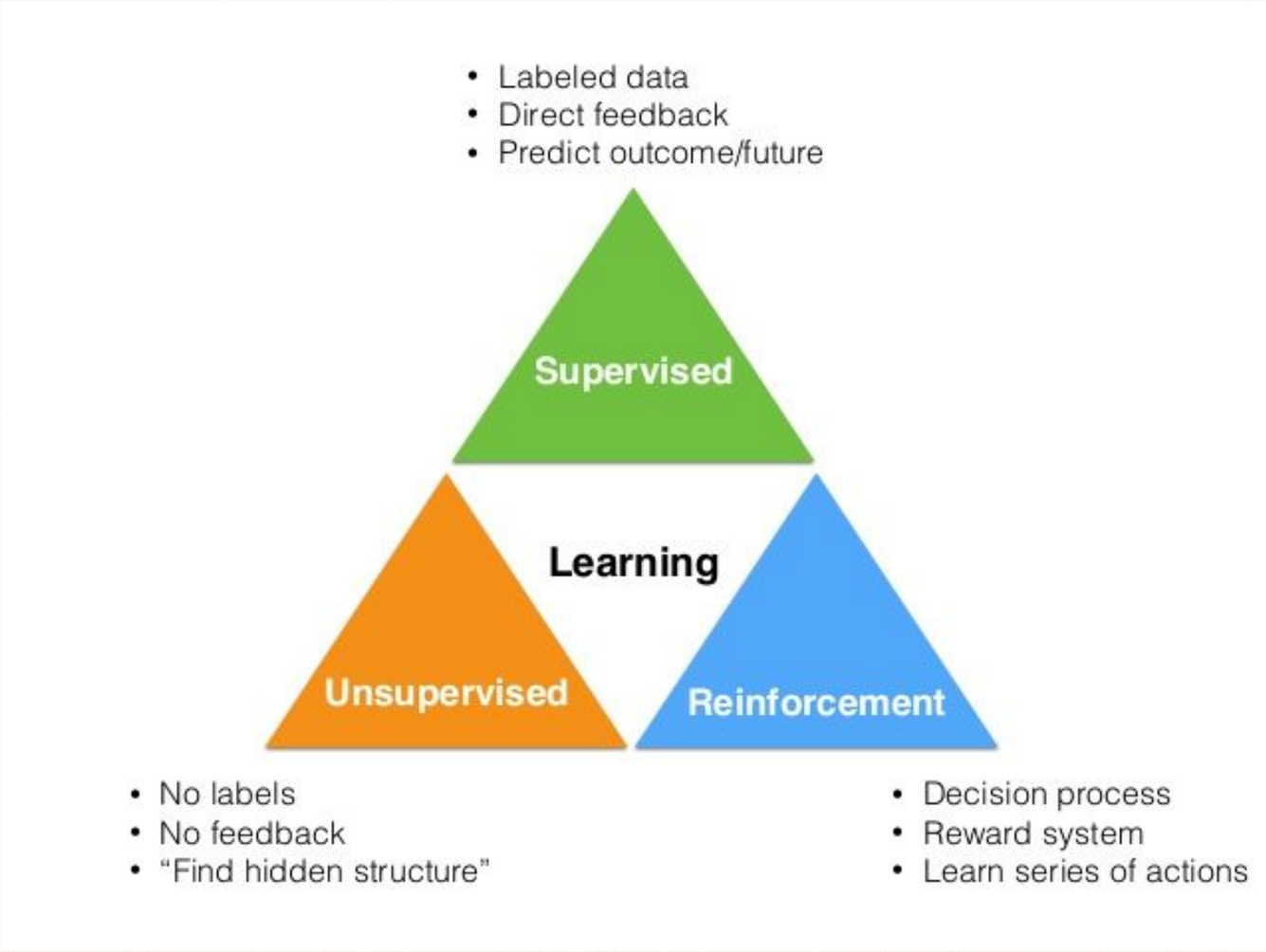
Credit: Andrew NG

Why it matters?

- Some tasks cannot be well defined except by examples
- Machine learning enables us to tackle tasks that are too difficult to solve with fixed programs written and designed by human beings.
- Data mining – find hidden correlations (knowledge)
- From a scientific and philosophical point of view, machine learning is interesting because developing our understanding of machine learning entails developing our understanding of the principles that underlie intelligence.

Types of ML/DL

—According to which kind of experience



Types of ML/DL

—According to tasks

- Classification
- Regression
- Structured learning
- Clustering
- Dimensionality Reduction
- Anomaly detection
- Generative Modeling
- …

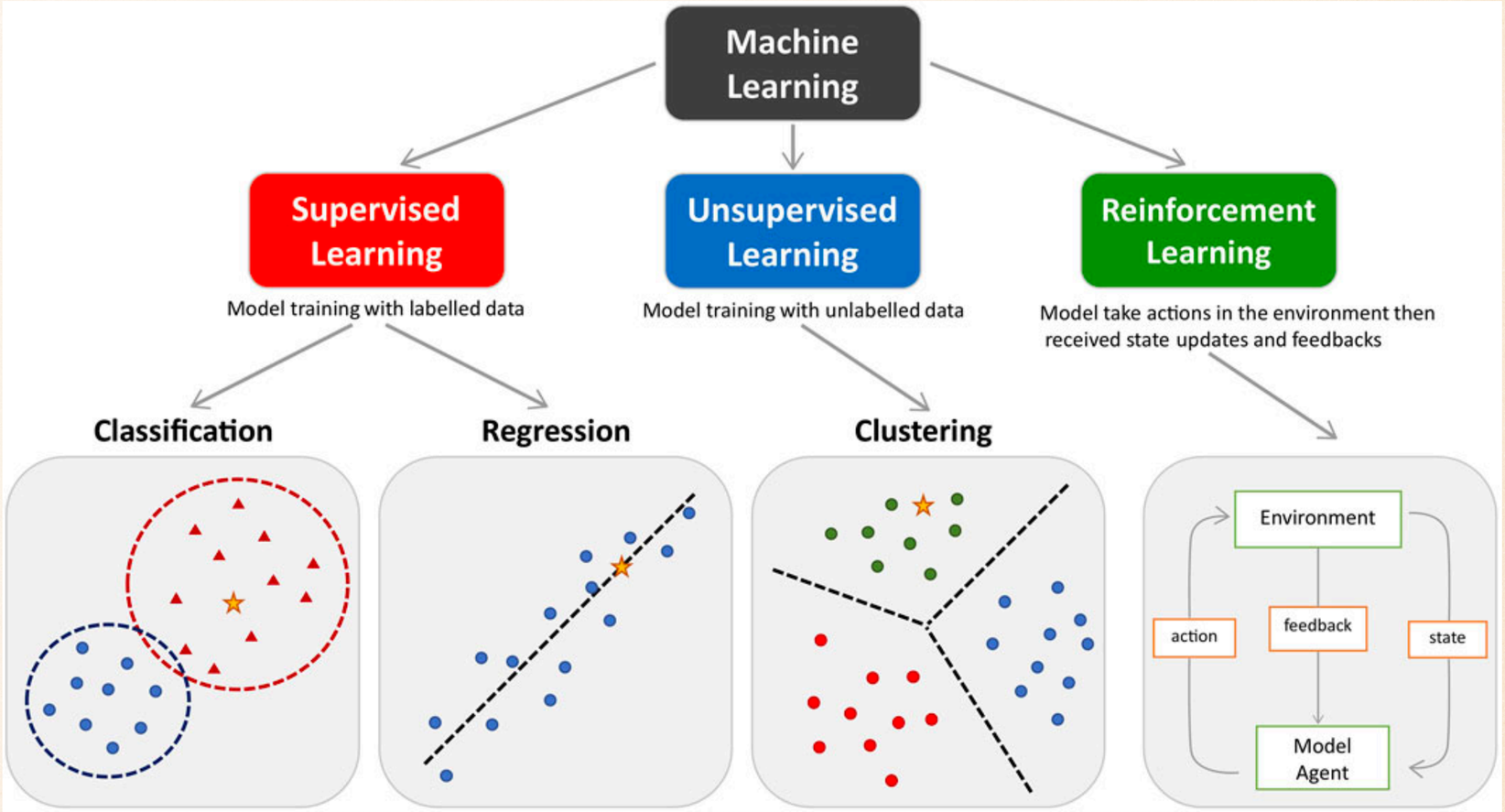


Figure from frontiersin

Main Challenges in ML/DL

- Insufficient amount of training data
- Non-representative training data
- Poor-quality data
- Irrelevant features (features engineering)

Open Source Python Libraries

- Tensorflow
- Keras
- Numpy
- Pandas
- Scikit-Learn
- …

Figure from towards data science

GPU Resources

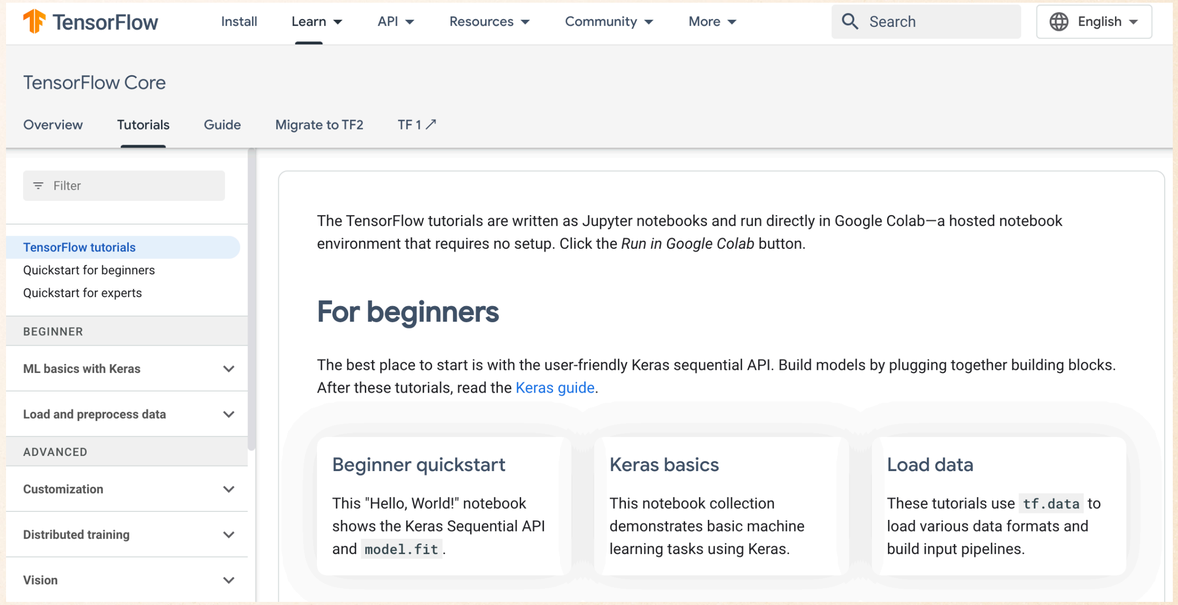
- Google Drive Colab, free but limited (1~2) GPU
- 腾讯云、阿里云、华为云…
- 工作站
- Otherwise use Jupyter notebook (Python) in your laptop

Reference books

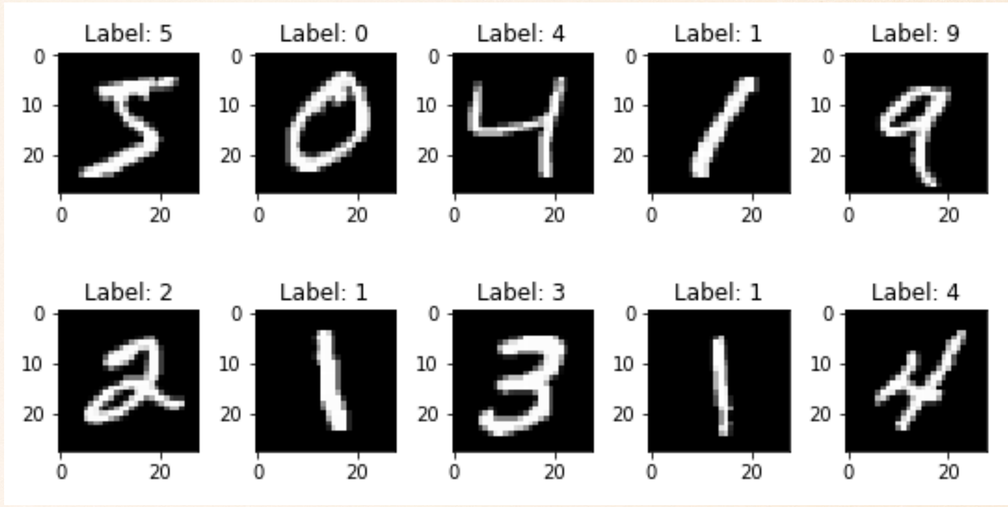
- Hands-On Machine Learning with Scikit-Learn and TensorFlow by **Aurélien Géron**
- Pattern Recognition and Machine Learning by **Christopher M. Bishop**
- Machine Learning by **Tom M. Mitchell**
- Deep Learning by **Ian Goodfellow, Yoshua Bengio, Aaron Courville**
- 机器学习（**周志华**著）

Useful Websites

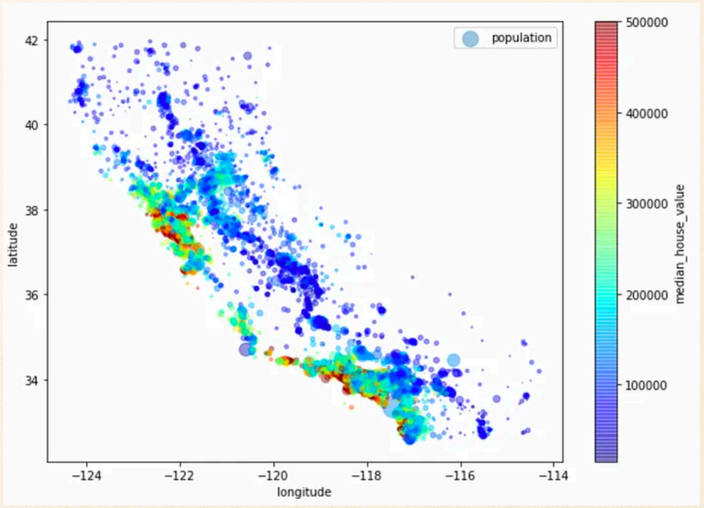
- Kaggle: https://www.kaggle.com/ (Datasets)
- Tutorials: https://www.tensorflow.org/tutorials
- https://medium.com
- https://towardsdatascience.com
- https://machinelearningmastery.com/
- https://www.jiqizhixin.com/ 机器之心



“Hello word” tasks



- Classification of cat/dog
- Classification of MNIST handwritten digits
- Prediction of California housing price



Introduction to Machine Learning and Deep Learning

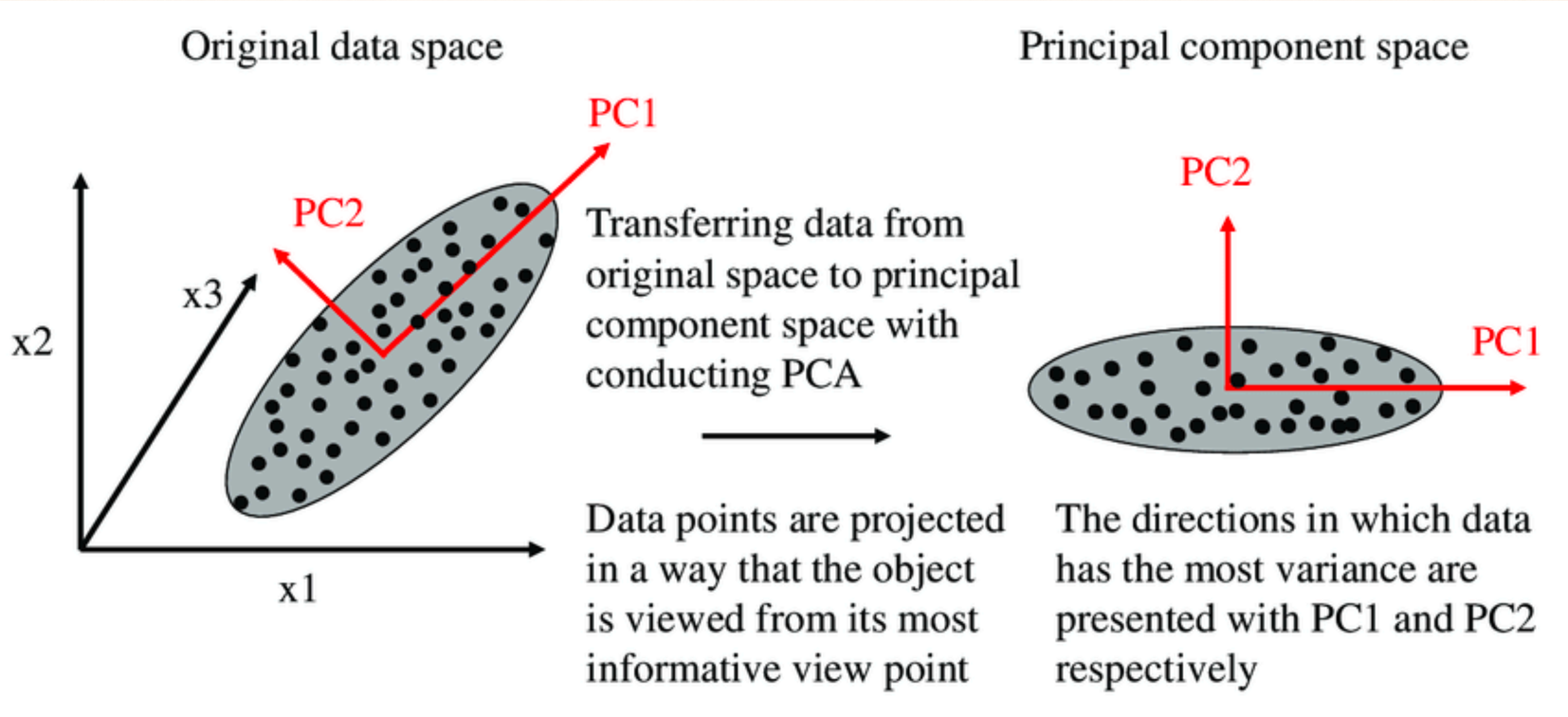
Yi-Lun Du（杜轶伦）
Shandong Institute of Advanced Technology

July 18-19, Qingdao
QCD and Nuclear Physics Summer School

ML method: Unsupervised Learning

- Dimensionality Reduction
  - Principle component analysis (PCA，主成分分析)
  - t-distributed stochastic neighbor embedding (t-SNE，随机近邻嵌入)
- Clustering
  - K-means clustering (K-平均算法)
  - k-nearest neighbors (K-近邻算法)

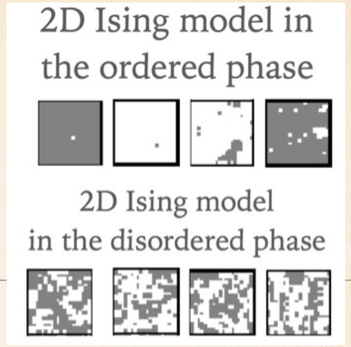
Principle component analysis (PCA，主成分分析)



- Linear dimensionality reduction

Figure from Infrastructures 4(3):53

PCA: Ising model



Discovering phase transitions with unsupervised learning

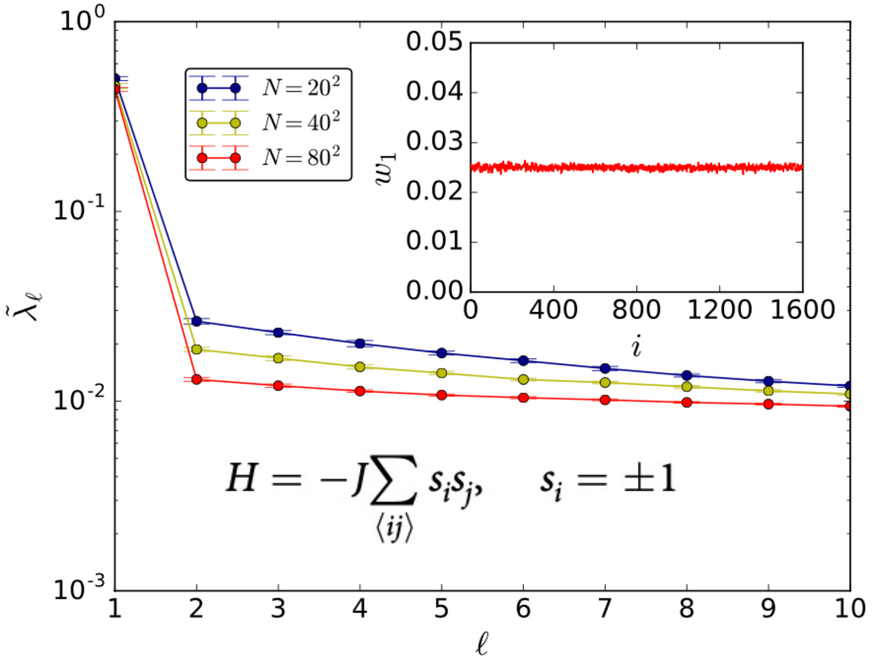


FIG. 1. The first few explained variance ratios obtained from the raw Ising configurations. The inset shows the weights of the first principal component on an $N = 40^2$ square lattice.

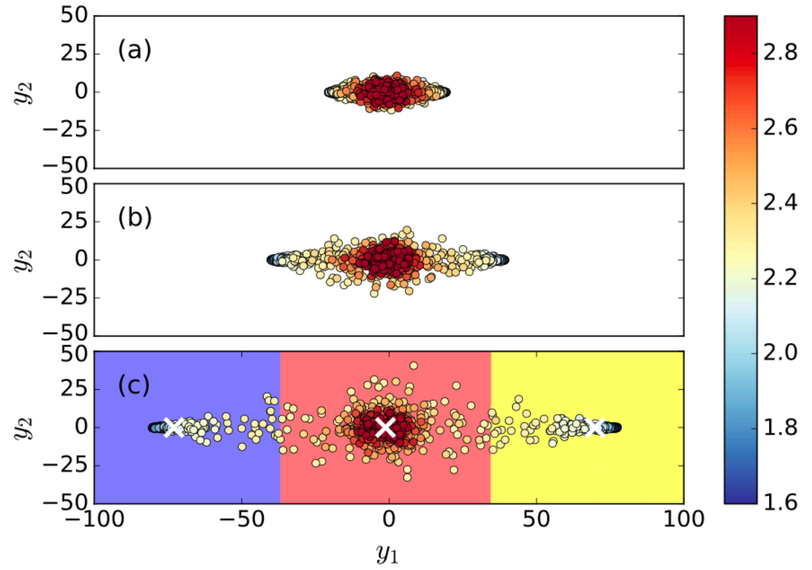
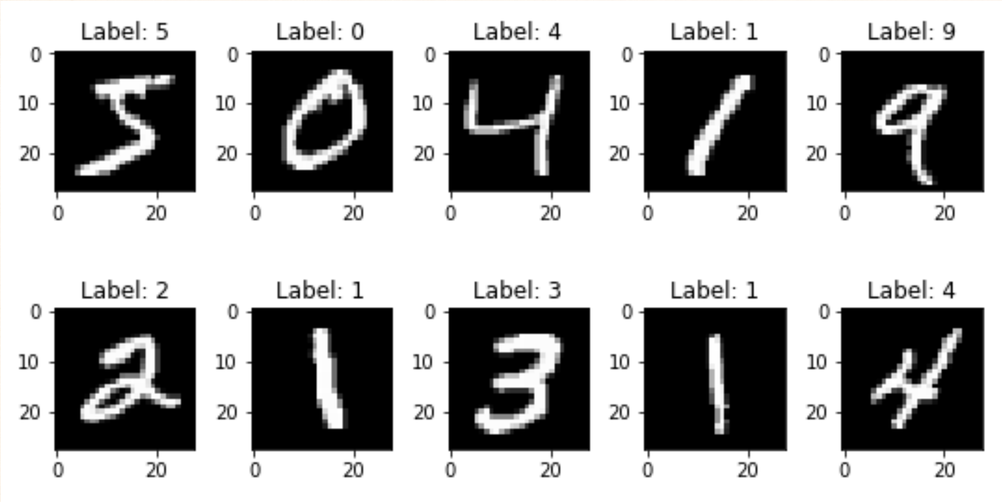


FIG. 2. Projection of the samples onto the plane of the two leading principal components. The color bar on the right indicates the temperature T/J of the samples. (a)–(c) are for $N = 20^2$, 40^2, and 80^2 sites, respectively. In (c) we perform k-means clustering to split the data into several phases. The white crosses denote the cluster centroids. Background colors indicate different predicted phases.

Lei Wang, PRB 94, 195105 (2016)

t-distributed stochastic neighbor embedding （t-SNE，随机近邻嵌入）

—reflect the similarities between the high-dimensional data in a 2-3 dim map



$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Locate y_i by minimizing KL loss

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- Non-linear dimensionality reduction

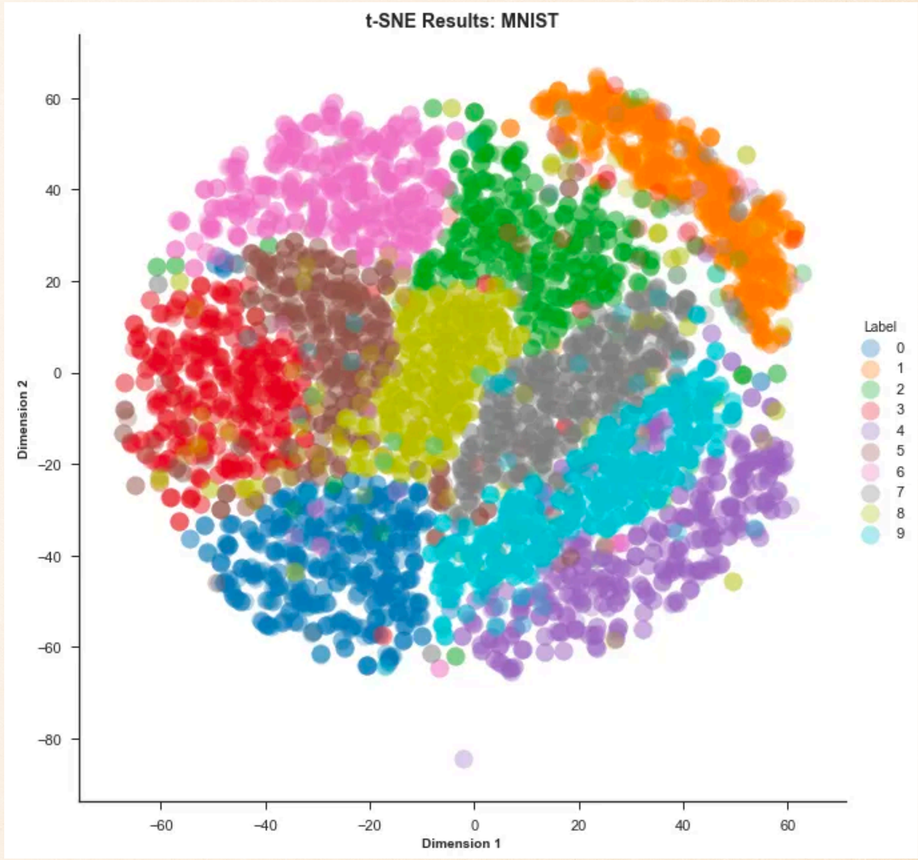
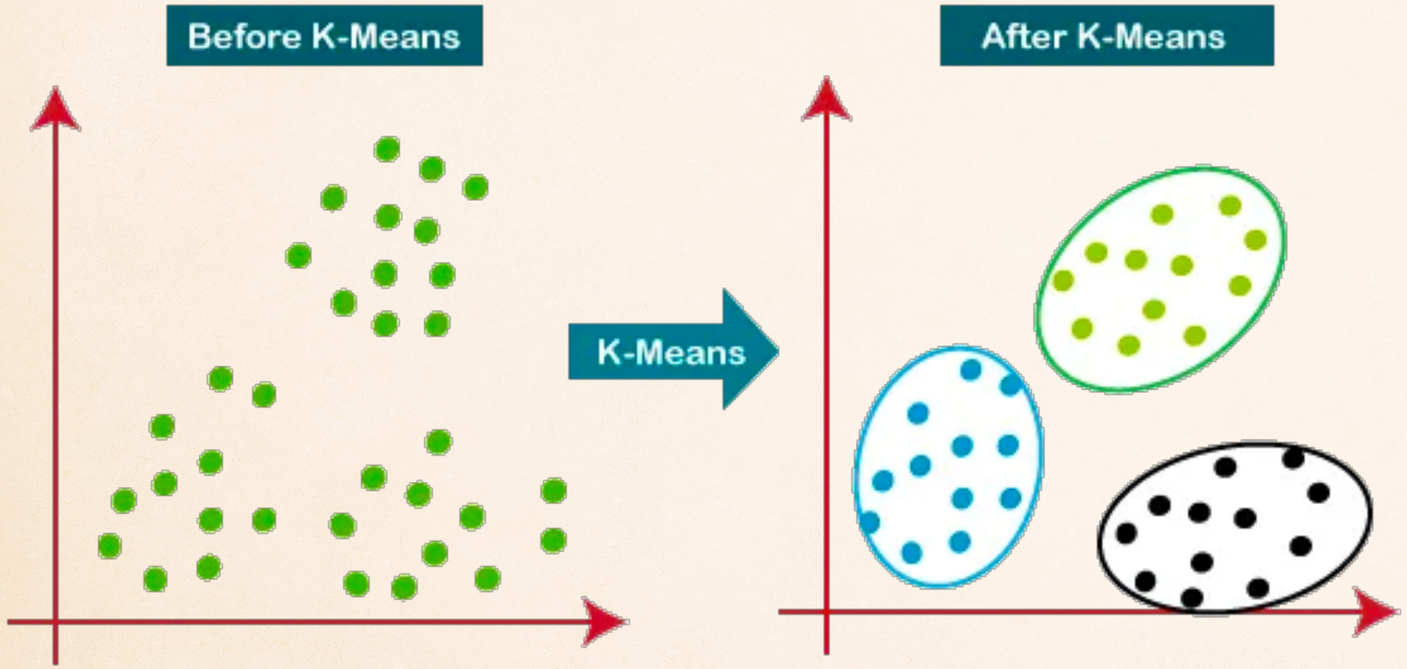
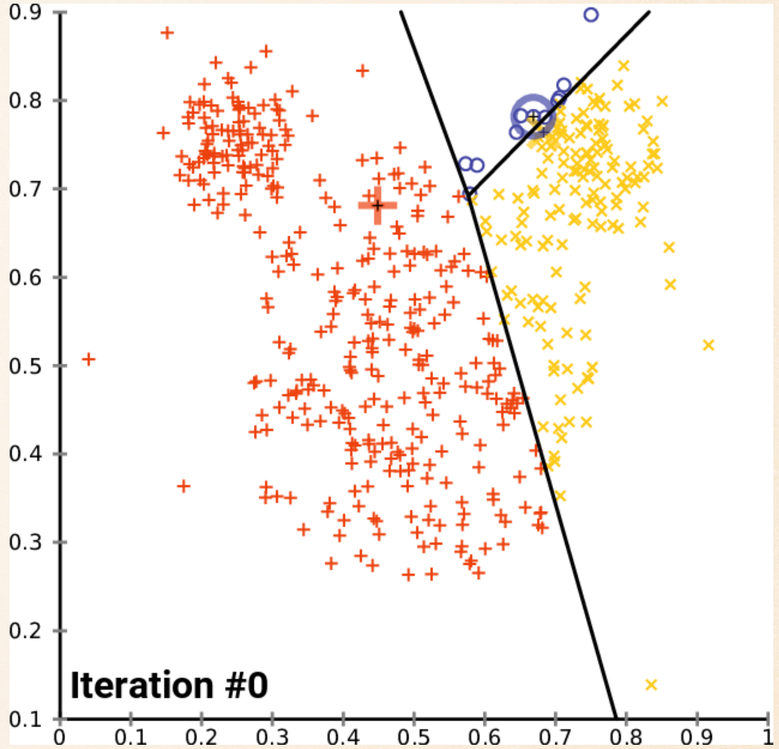


Figure from medium

K-means clustering （K-平均算法）

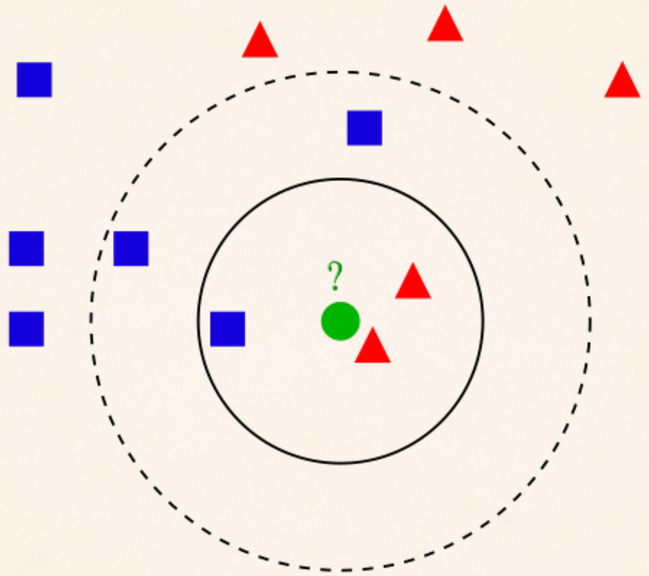




- Minimize the within-cluster sum of squares (to the mean) (WCSS)

Figure from medium & Wikipedia

k-nearest neighbors（kNN, k近邻算法）



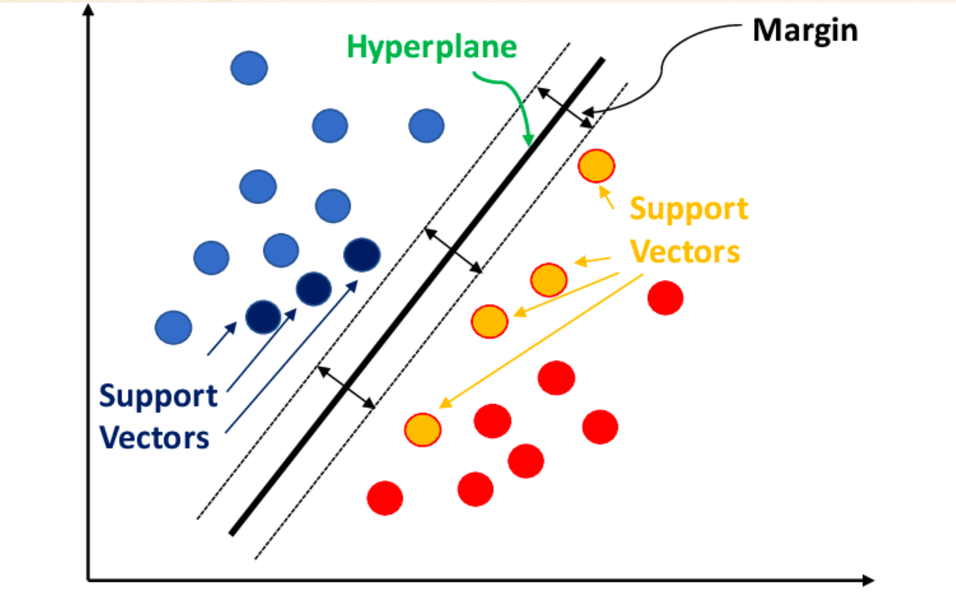
- Determine the class that the sample belongs to by the majority vote of its k-nearest neighbors
- How to choose the k value?
  - Avoid too small (overfitting by the noise) and too large (too simple)
  - Fine-tune k value by cross-validation

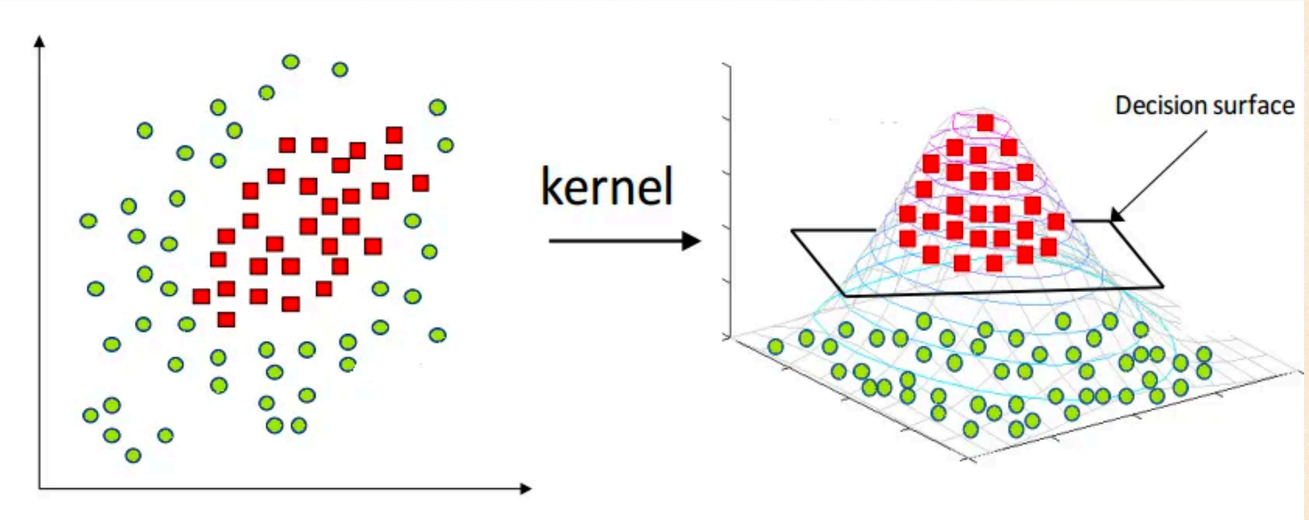
Figure from Wikipedia

ML method: Supervised Learning

- Support Vector Machine (SVM，支持向量机)
- Decision tree（决策树）
- Ensemble learning（集成学习）
  - Bagging
    - Random Forest（随机森林）
  - Boosting
    - Adaptive Boosting（自适应增强）
    - Gradient Boosting（梯度提升/增强）
    - Extreme Gradient Boosting (XG-Boost, 极端梯度提升)

Support Vector Machine (SVM，支持向量机)





- Classification: Find the broadest *street* between the (linear-separable) data
- For linear non-separable data, use kernels to transform the data to extra dim.

Figure from Chemical Engineering Science: X 2 (2019) 100023

Figure from medium

Support Vector Machine (SVM，支持向量机)

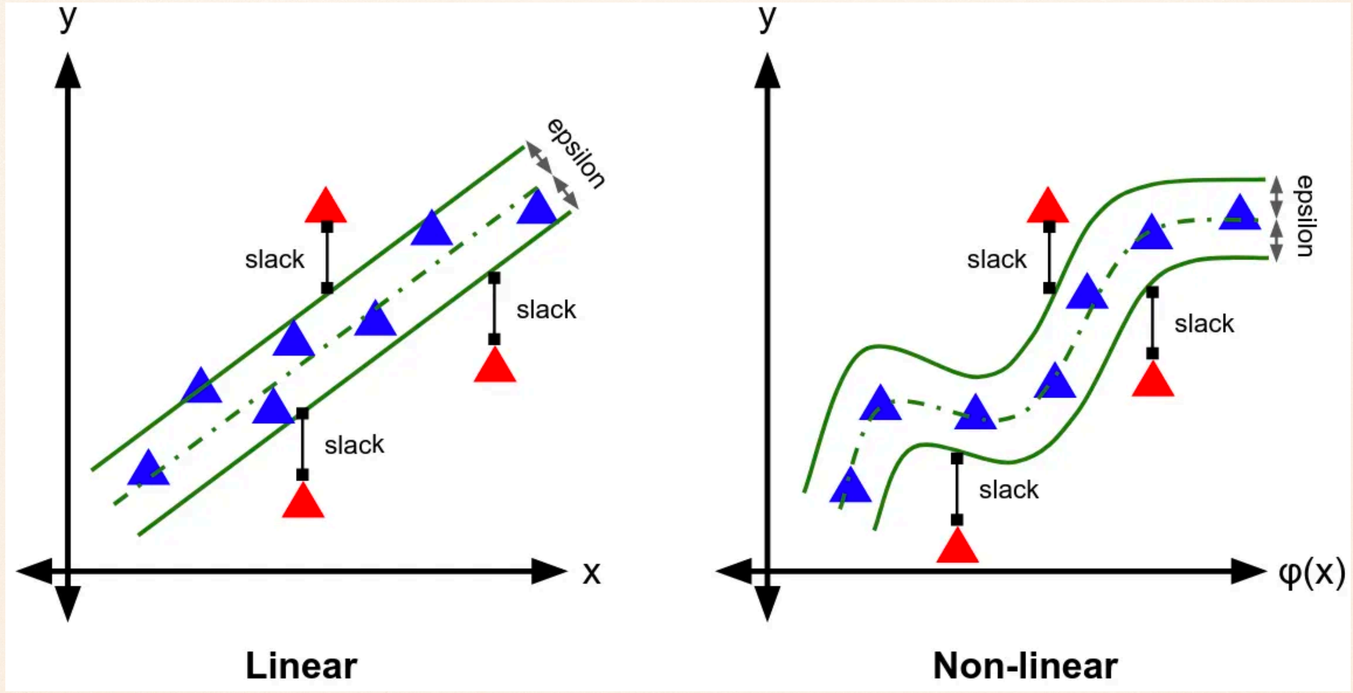


Figure from [medium]

- Regression: Switch to fit as much data as possible ***on*** the street with the width epsilon
- Epsilon gives a tolerable error of the model

Decision tree（决策树）

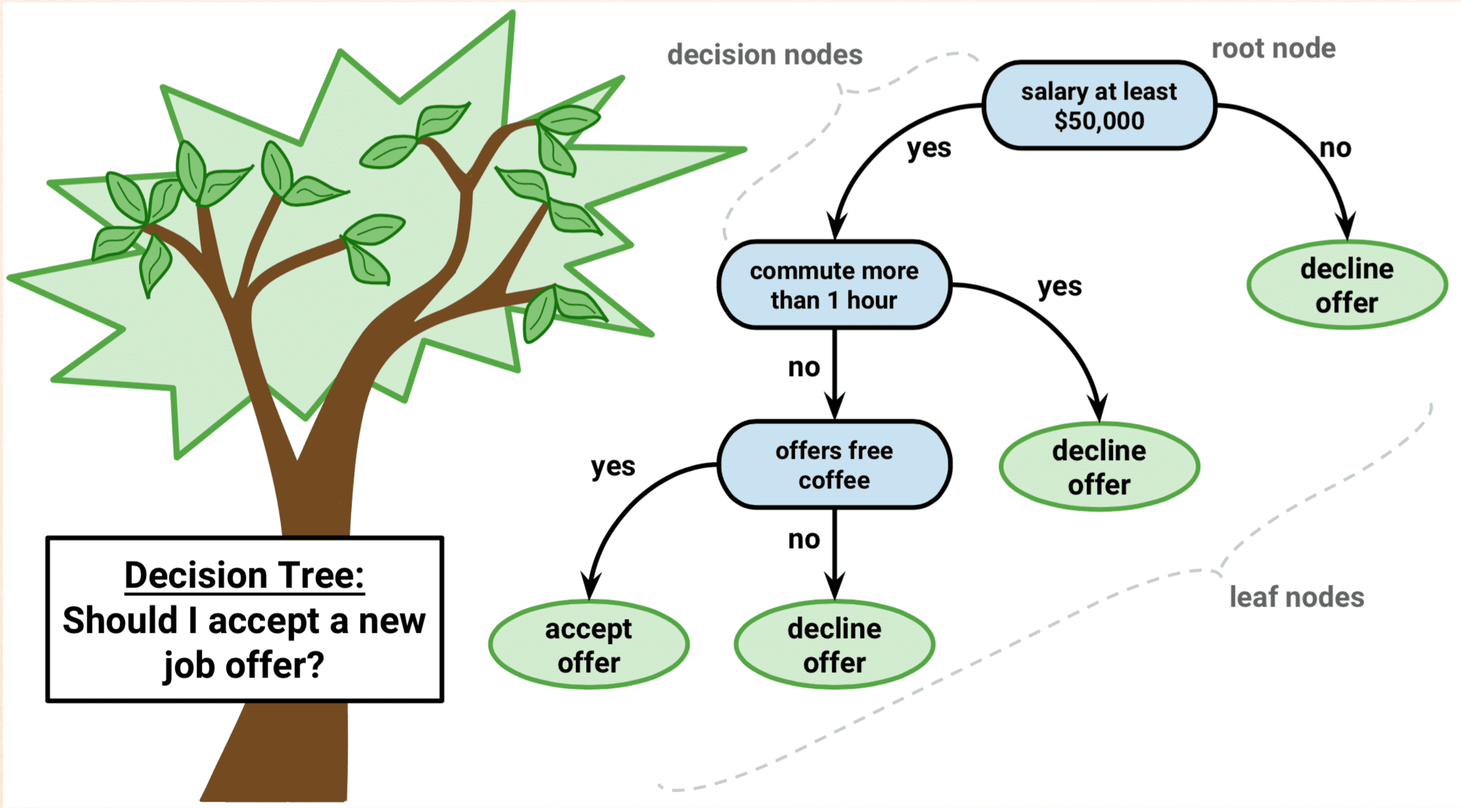


Figure from [regenerativetoday]

$$E = -\sum_{i}^{C} p_i \log_2 p_i$$

p_i: the proportion of the dataset made up of class i

- Use entropy (information gain) to determine the sequence of tree growing

Decision tree（决策树）

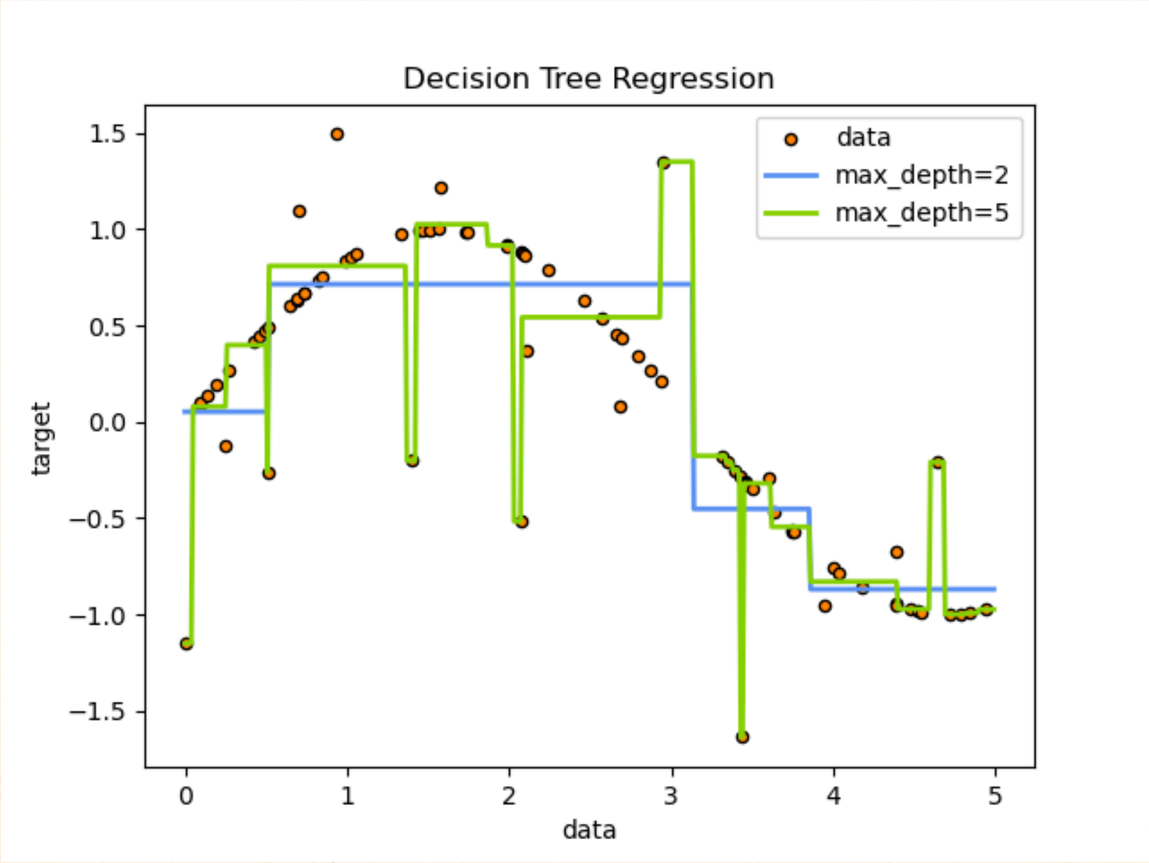
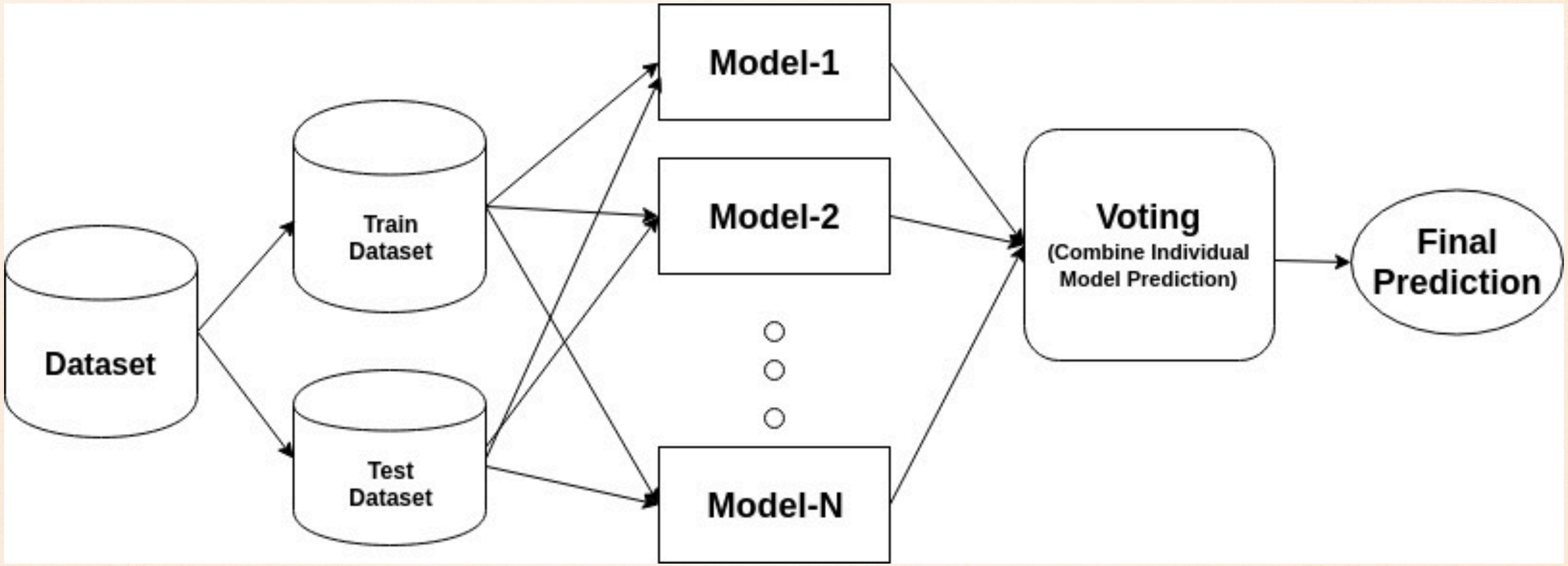


Figure from scikit-learn

Ensemble learning（集成学习）

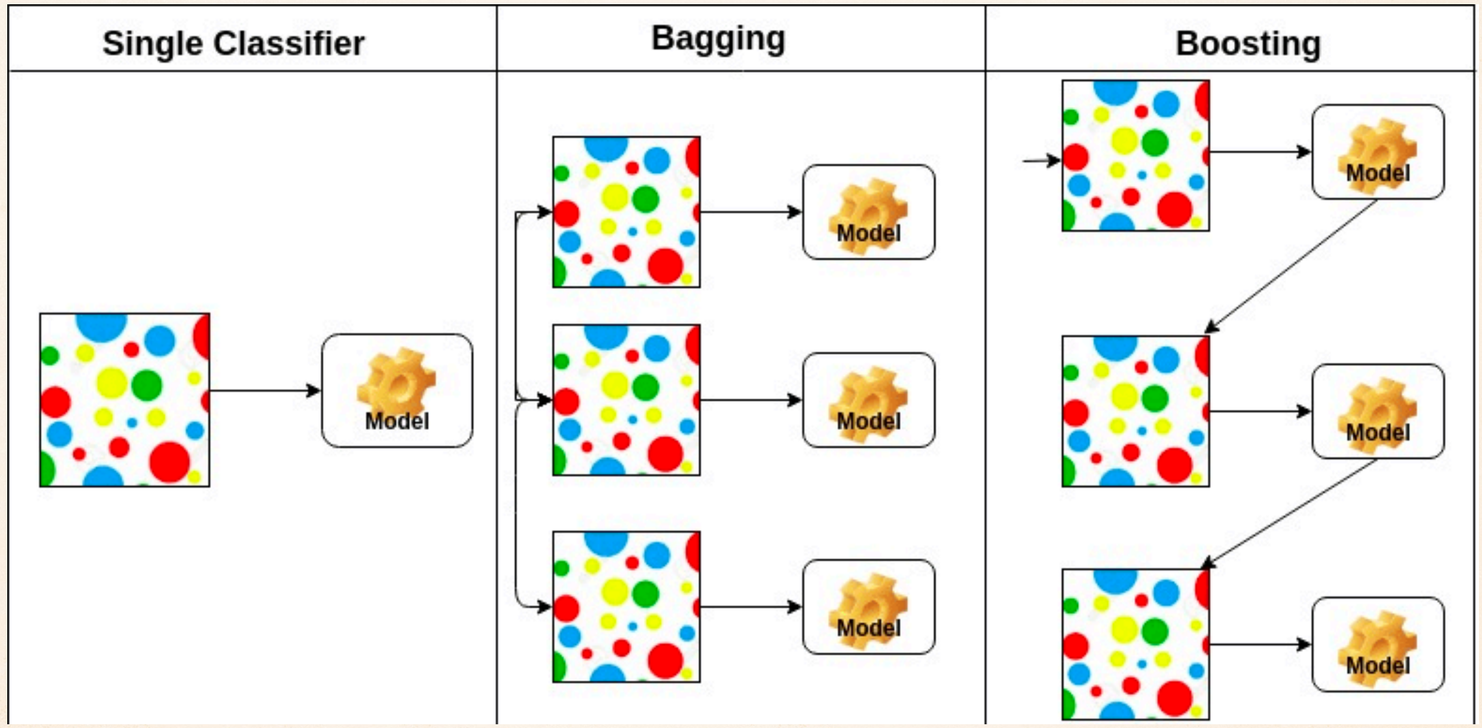
Figure from [machine learning geek]



- Train several different models and take the majority vote or average to get better and more robust results

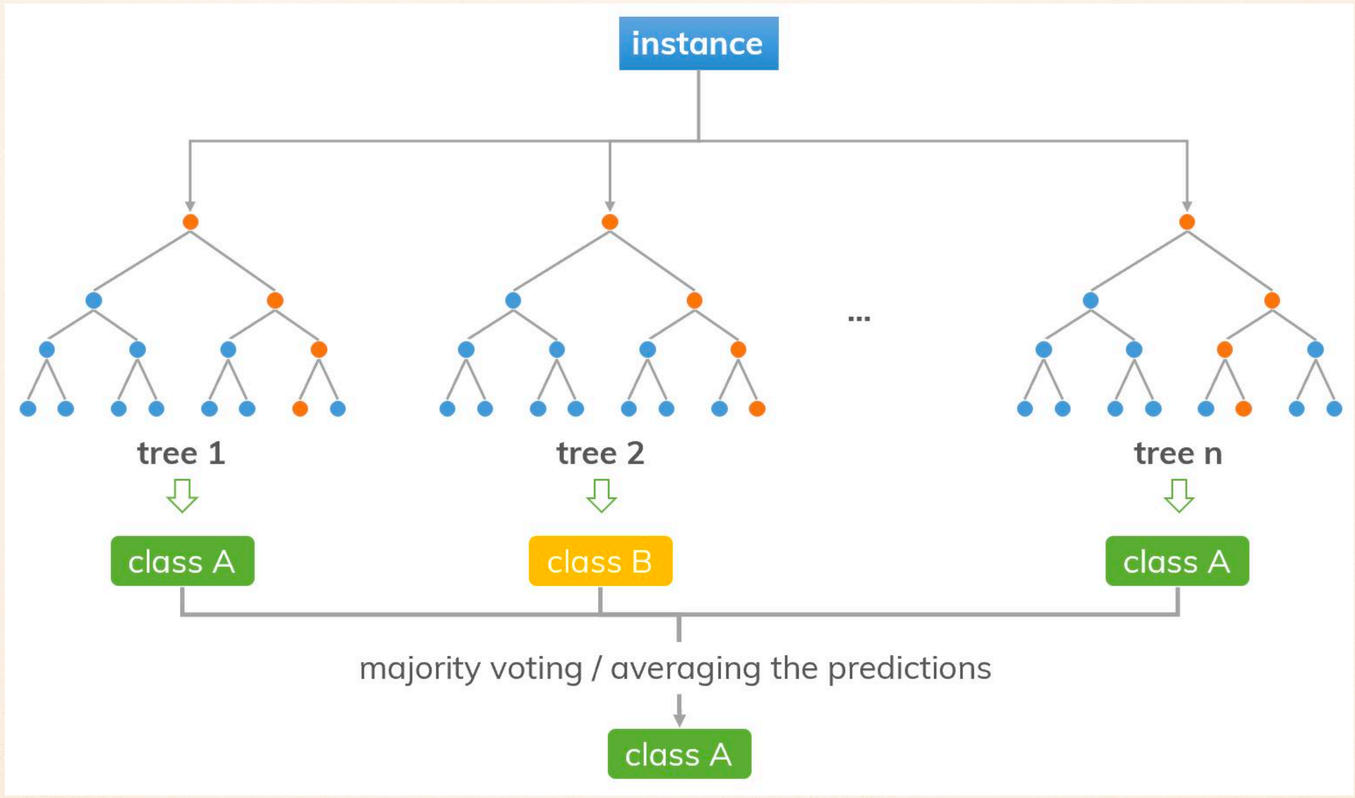
Bagging (Bootstrap aggregating 自助聚合，装袋) & Boosting（增强）

Figure from machine learning geek



- Bagging: Use the same training algorithm but train on different subsets of the training data in parallel
- Boosting: Combine several weak learners into a strong one by training the predictors sequentially

Random Forest（随机森林）

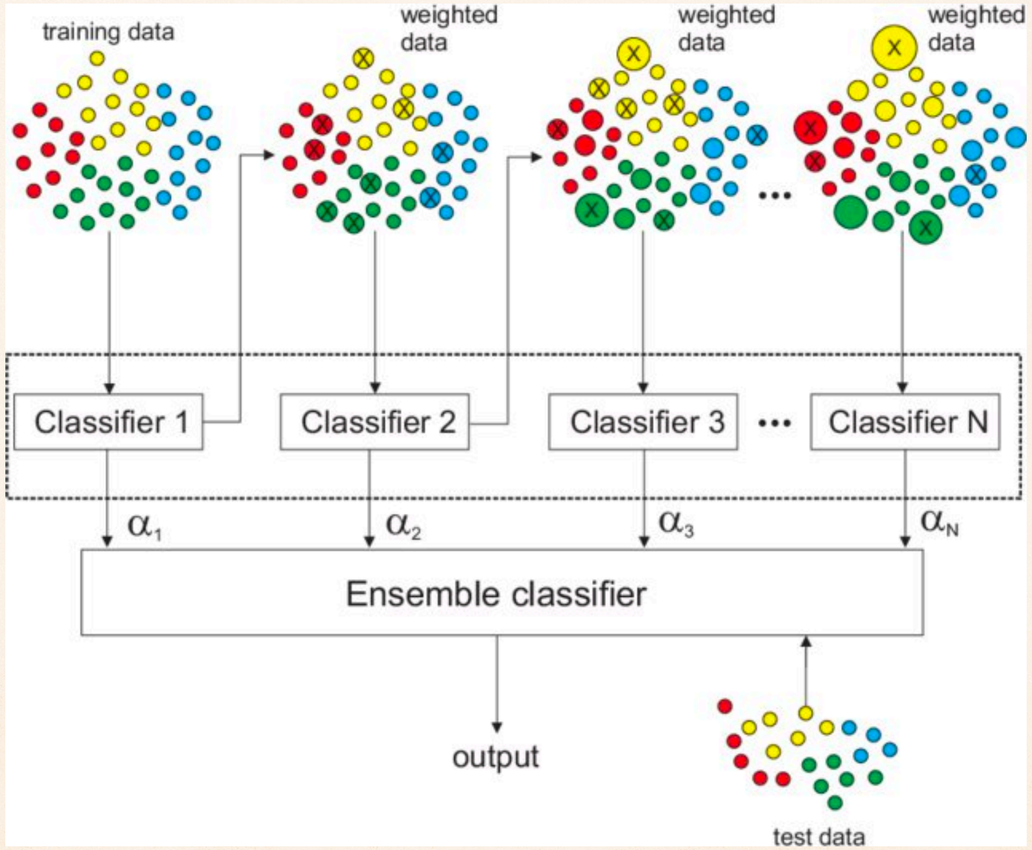


- Random Forests are an ensemble of decision tree, generally trained via the bagging method.

Figure from [dinhanhthi]

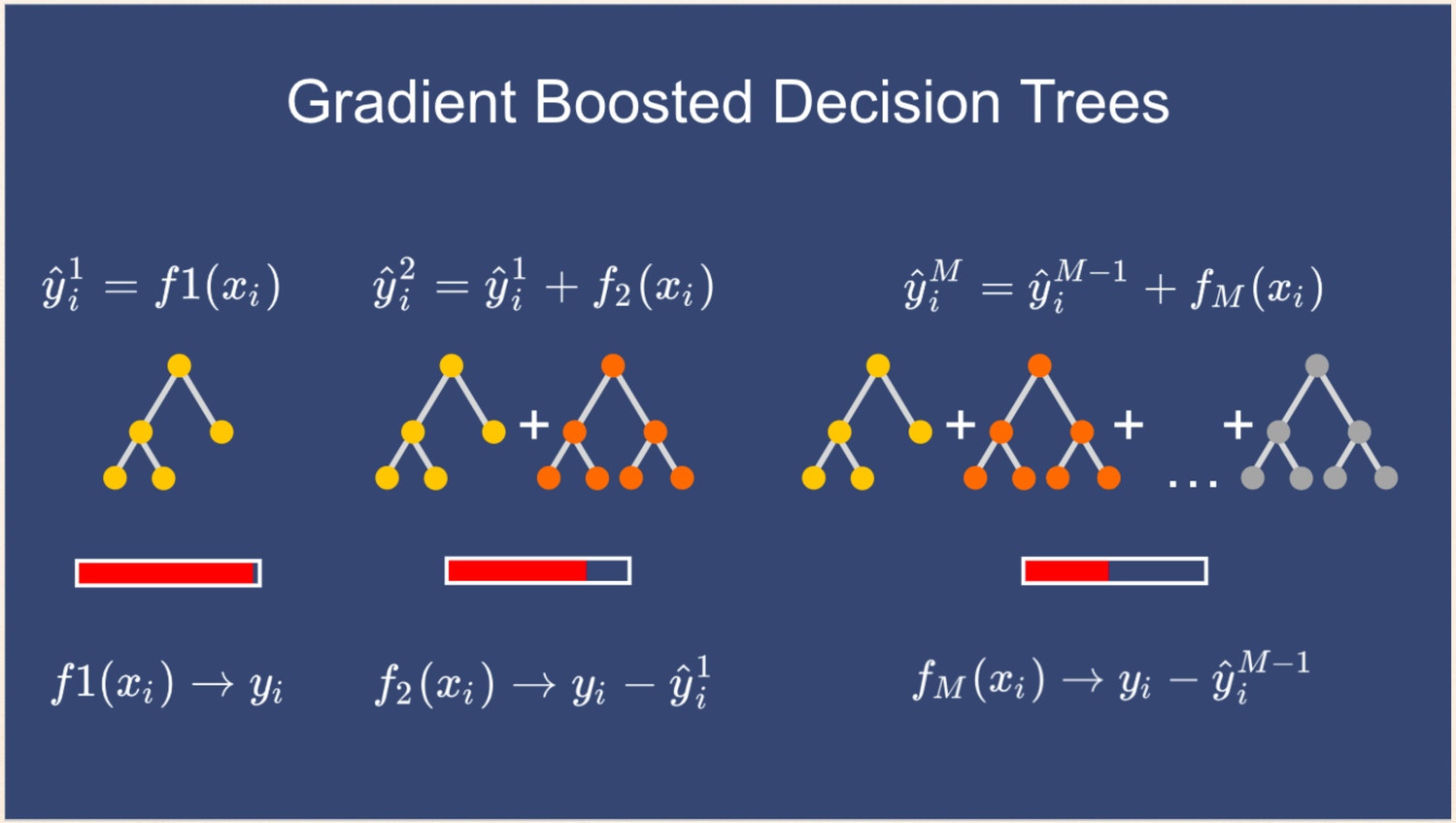
Adaptive Boosting（Ada Boost，自适应增强）

Figure from arXiv.2207.07580



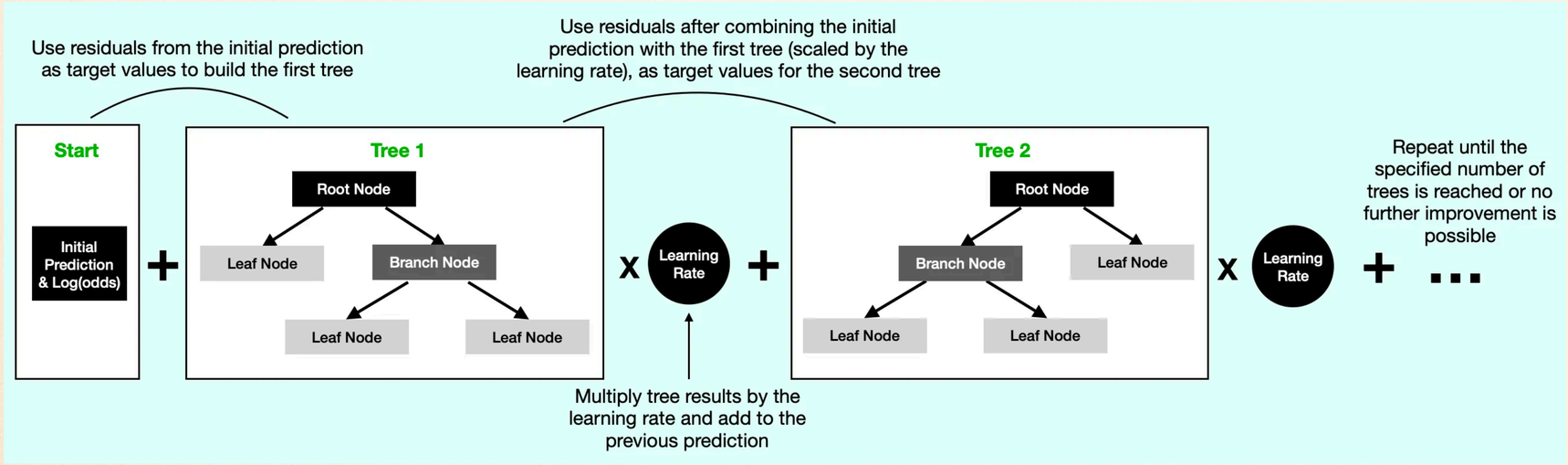
- Pay more attention on the under-fitted samples in training via increased weights

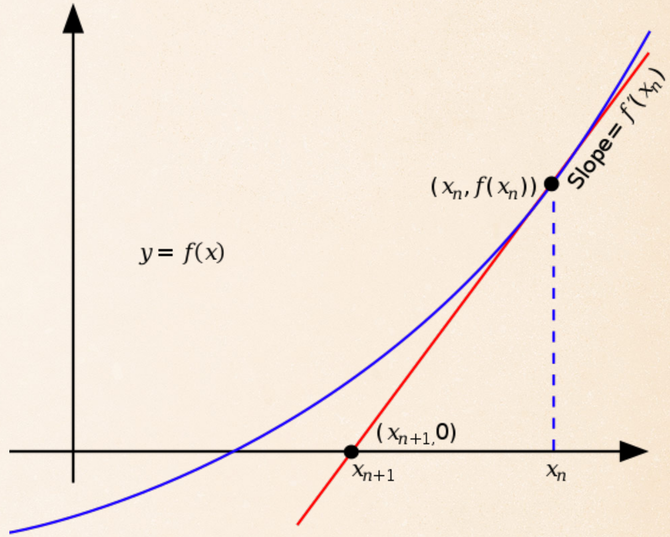
Gradient Boosting（梯度提升/增强）



- Fit the new predictors to the residual errors given by the previous ones.

eXtreme Gradient Boosting（XG-Boost，极端梯度增强）





- XGBoost works as Newton-Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order Taylor approximation is used in the loss function to make the connection to Newton-Raphson method.

- XGBoost is a very successful, very fast algorithm

California housing price

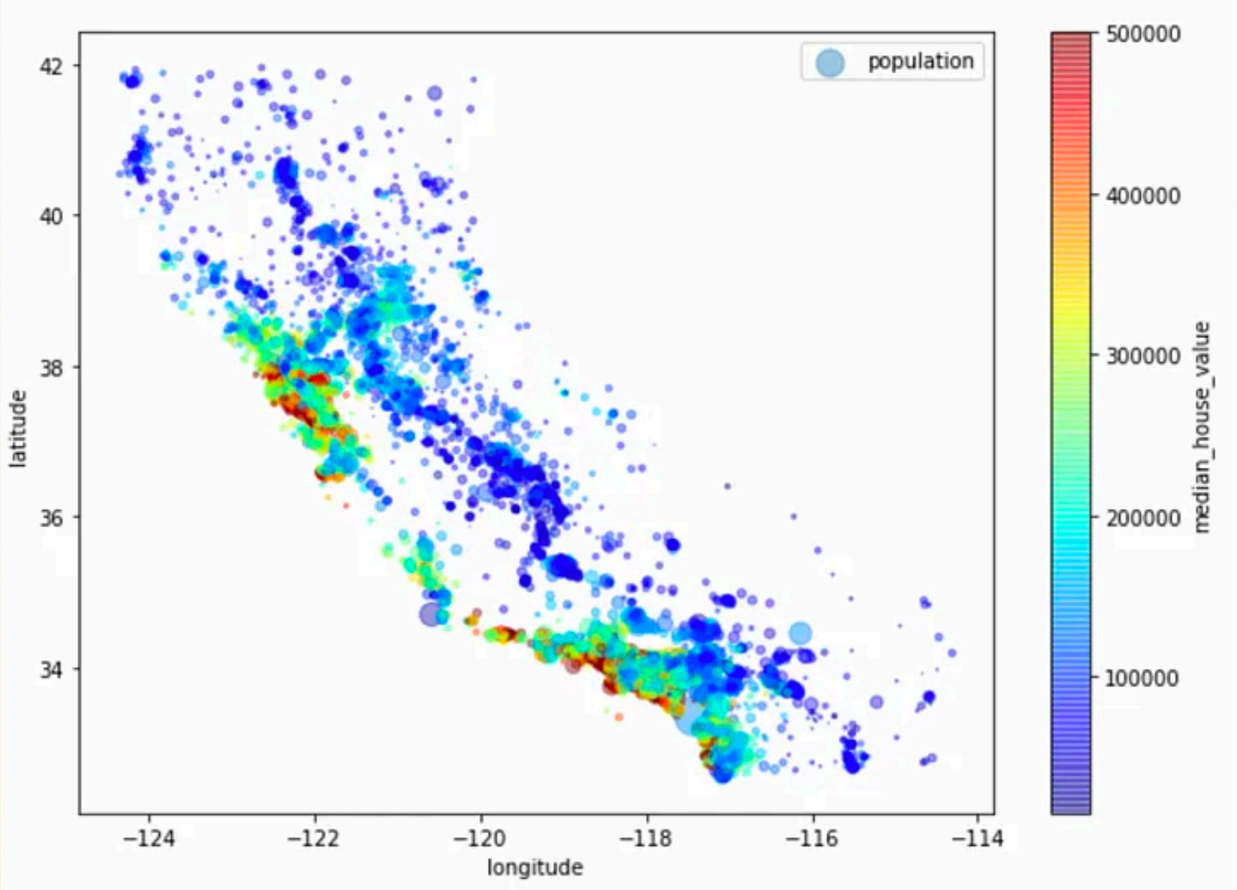


Figure from [medium]

How to start?

- Is there a current solution?
- What is the available data?
- Supervised or unsupervised?
- Classification or Regression?
- Select performance measure: loss

- Import/visualize the data
- Create a test set, 20~30% of the whole data.
- Take care of missing data.
- Scale the data
- Select & design the network
- Create a pipeline

Pre-analysis, data visualization & feature engineering

- longitude
- latitude
- housing median age
- total rooms
- total bedrooms
- population
- households
- median income
- ocean proximity
- total rooms/households
- total bedrooms/households
- population/households

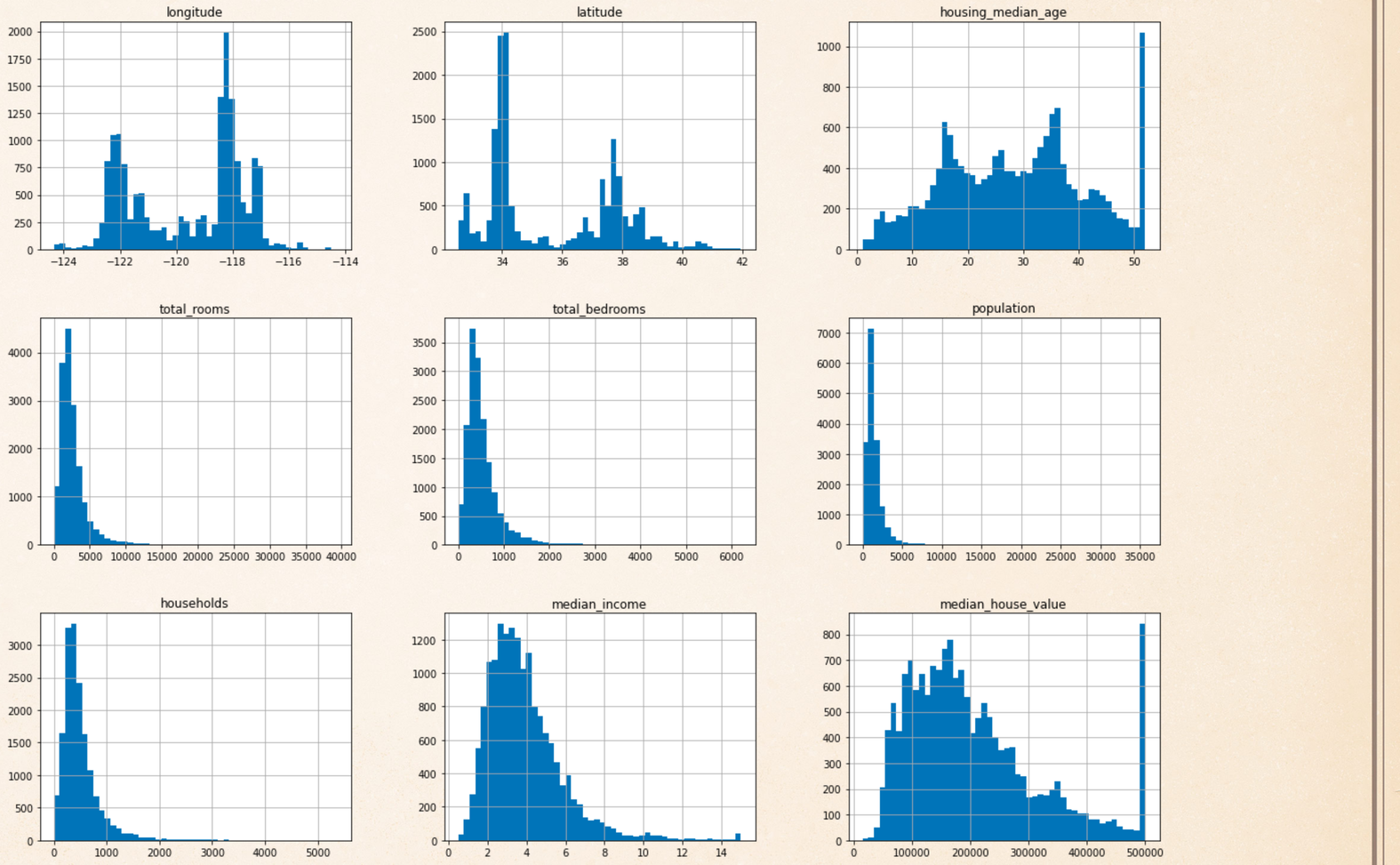


Figure from [medium]

Random shuffling and splitting

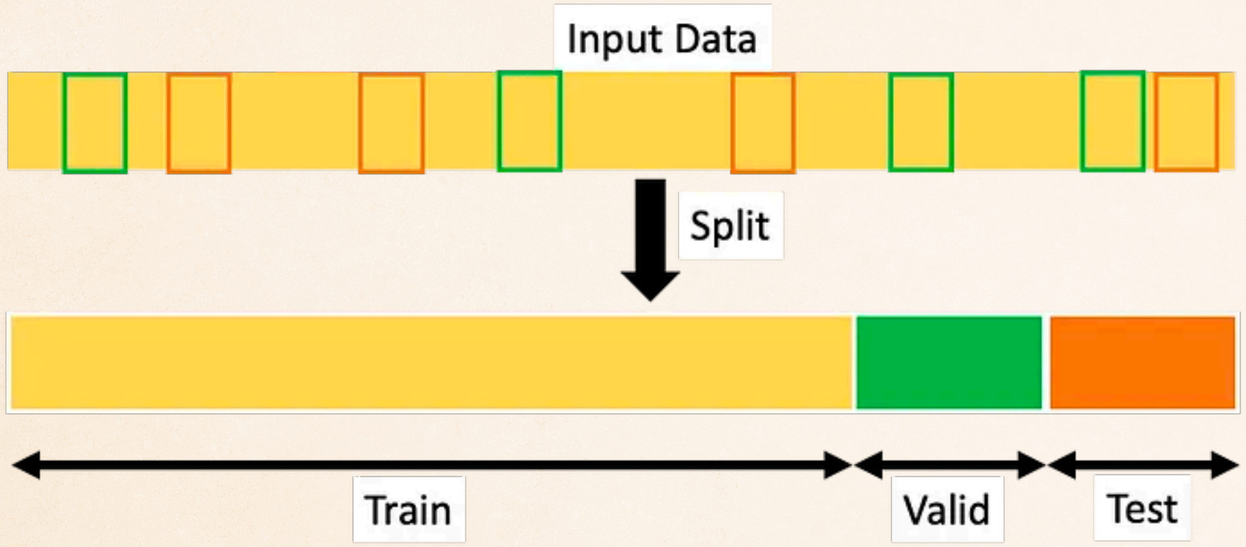


Figure from medium

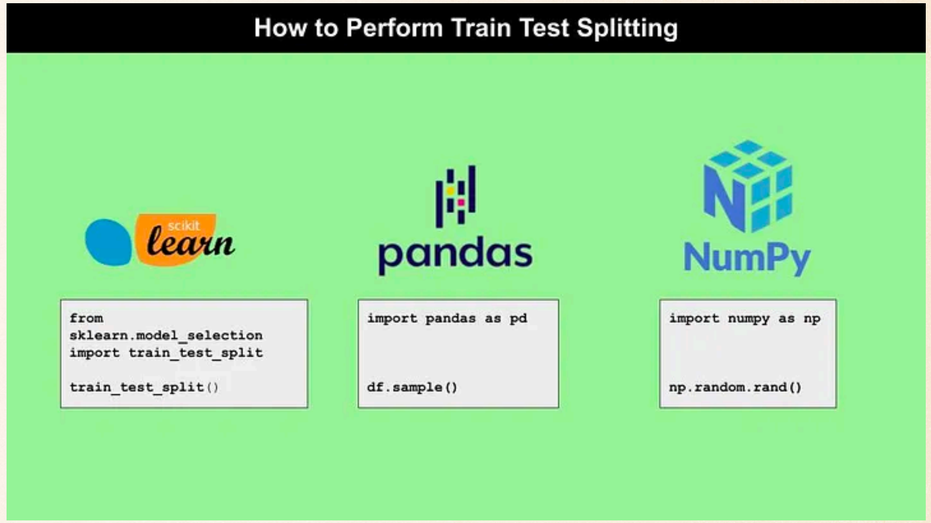


Figure from medium

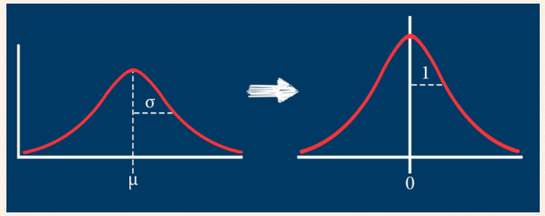
Data preprocessing

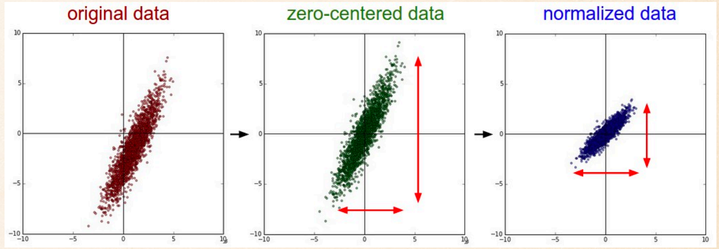
- Min-Max Normalization

$$\hat{X}[:,i] = \frac{X[:,i] - \min(X[:,i])}{\max(X[:,i]) - \min(X[:,i])}$$

- Z Normalization (Standardization)

$$\hat{X}[:,i] = \frac{X[:,i] - \mu_i}{\sigma_i}, (\mu_i = \frac{1}{N} * \sum_{k=1}^{N} X[k,i], \sigma_i = \sqrt{\frac{1}{N-1} * \sum_{k=1}^{N} (X[k,i] - \mu_i)^2})$$





- Unit Vector Normalization

$$\hat{X}[j,:] = \frac{X[j,:]}{||X[j,:]||}$$

The no-free-lunch-theorem

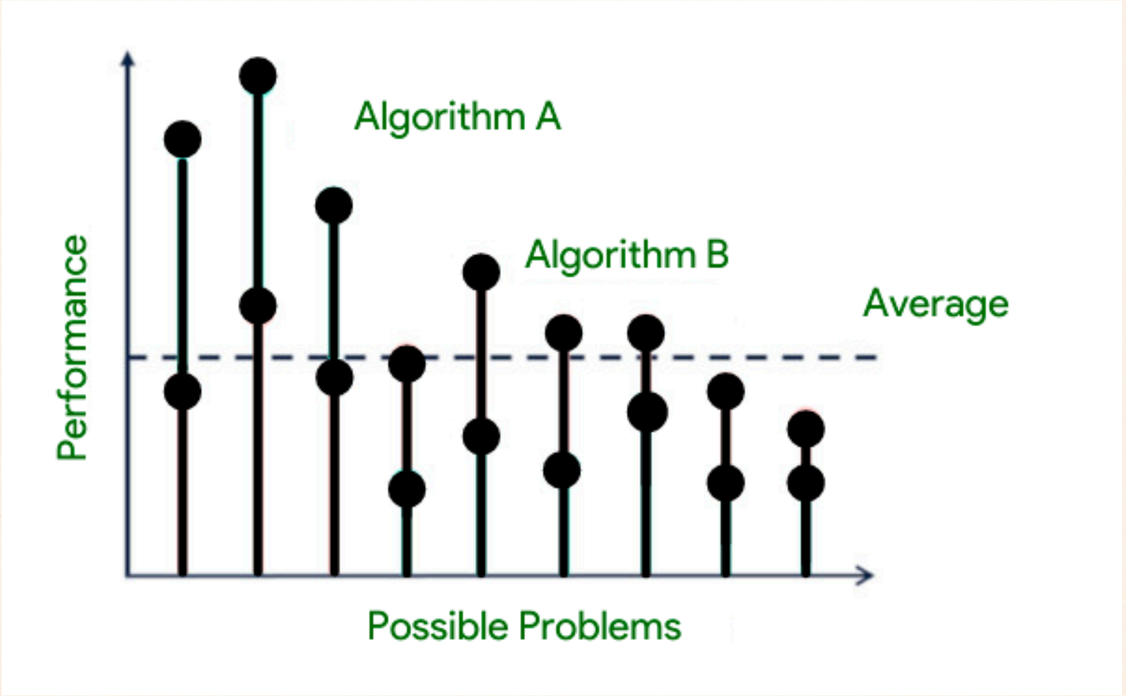


Figure from geeksforgeeks

- If you make no assumptions about the data, there is no reason to prefer one model over the other.
- Simply, there is no best model for all problems

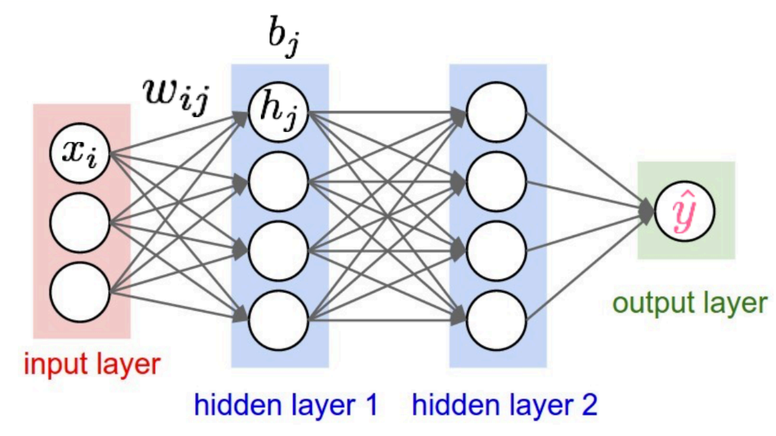
Introduction to Machine Learning and Deep Learning

Yi-Lun Du（杜轶伦）
Shandong Institute of Advanced Technology

July 18-19, Qingdao
QCD and Nuclear Physics Summer School

“hello world” example of deep neural network

Figure from [CS231N, Stanford]



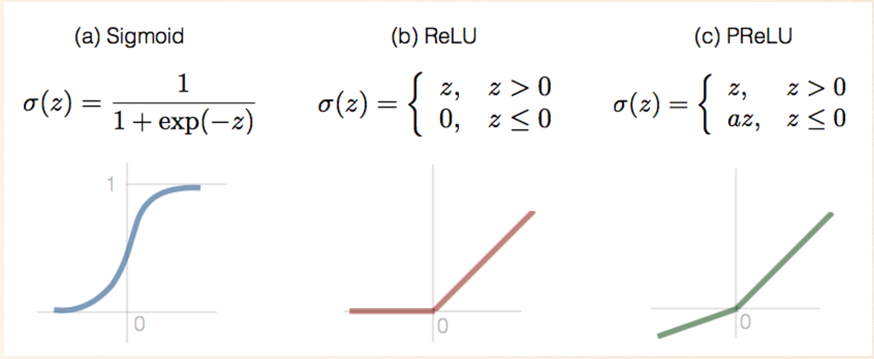
Fully Connected Network (FCN)

$$z_j = \sum_{i=1}^{N} x_i w_{ij} + b_j$$

Linear operations: rotating, boosting,... increasing or decreasing dimensions

$$l(\theta) = \sum_i (\hat{y}_i - y_i)^2$$

Mean square error (simplest loss function) with $\hat{y}_i$ the predicted value and y_i true value



$$h_j = \sigma(z_j)$$

Nonlinear activation function: correlation/links

$$\theta' = \theta - \epsilon \frac{\partial l(\theta)}{\partial \theta}$$

Gradient Descent for parameter update to minimize the loss function

How does deep neural network learn: Back-propagation

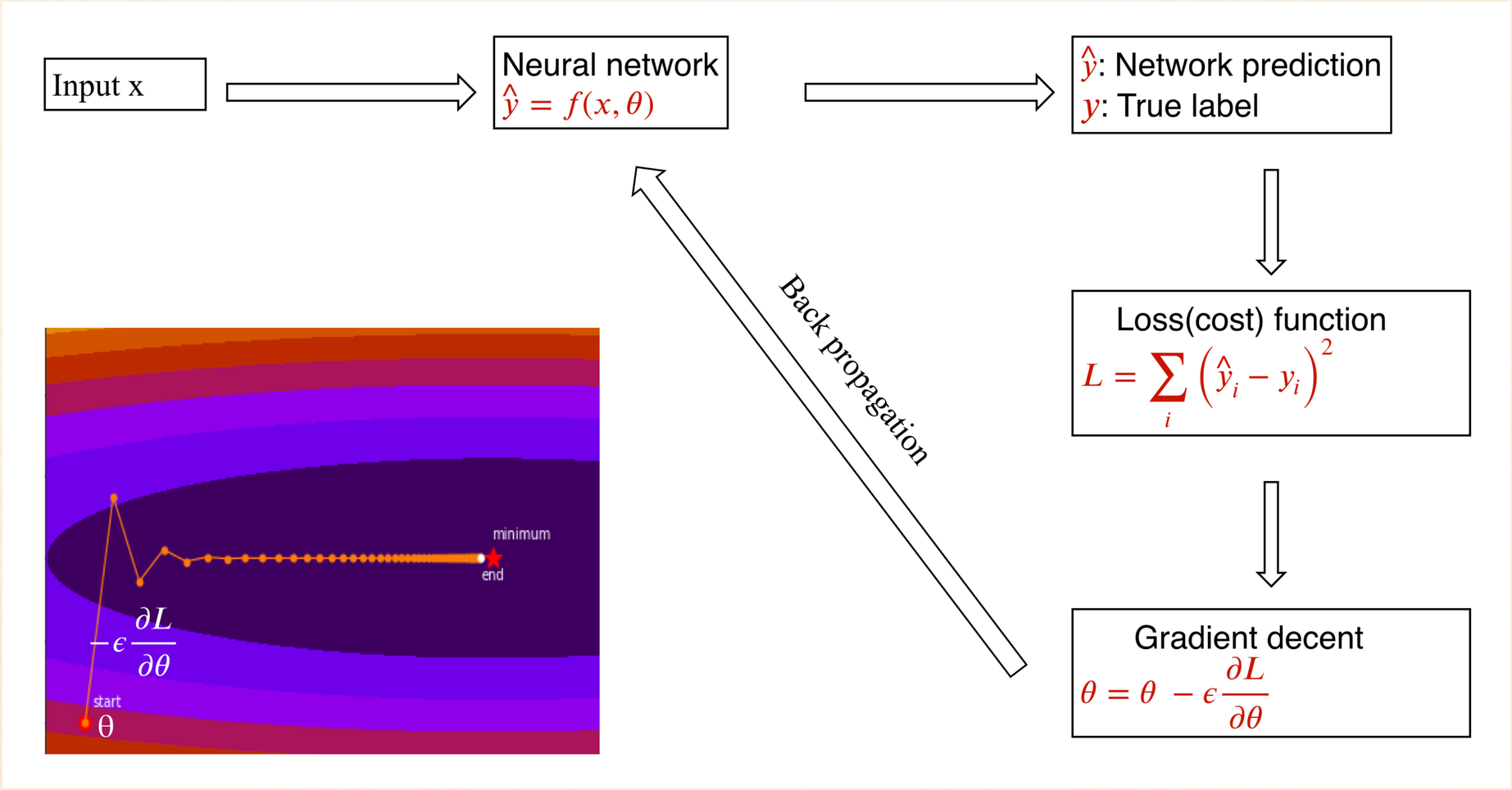


Figure from Longgang Pang

Forward Propagation - feed-forward

- The feed-forward procedure is to sequentially compute the following

$$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = f(z^{[l]})$$

until the final output $\hat{y} = a^{[L]} = f(z^{[L]})$

- **e.g.** for a 2-layer neural network with sigmoid activation functions, the compositional function it represents can be expressed as:

$$\hat{y} = f_{NN}(x;\theta) = \sigma(W^{[2]} \cdot \sigma(W^{[1]} \cdot x + b^{[1]}) + b^{[2]})$$

where θ are the model parameters: $\theta = \{W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}\}$

Credit: Kai Zhou

Backward propagation

- Using chain rule to calculate derivatives for weights and biases: $\Delta W_{ij}^{[l]} \to \Delta z_i^{[l]} \to \cdots \to \Delta L;\ \Delta b_i^{[l]} \to \Delta z_i^{[l]} \to \cdots \to \Delta L$

$$\frac{\partial L}{\partial W_{ij}^{[l]}} = \frac{\partial z_i^{[l]}}{\partial W_{ij}^{[l]}} \frac{\partial L}{\partial z_i^{[l]}} = a_j^{[l-1]} \frac{\partial L}{\partial z_i^{[l]}}$$

$$\frac{\partial L}{\partial b_i^{[l]}} = \frac{\partial z_i^{[l]}}{\partial b_i^{[l]}} \frac{\partial L}{\partial z_i^{[l]}} = \frac{\partial L}{\partial z_i^{[l]}}$$

Reminder: $z_i^{[l]} = \sum_j W_{ij}^{[l]} a_j^{[l-1]} + b_i^{[l]}$

where the sequence $\frac{\partial L}{\partial z_i^{[l]}}$ obey a recurrence relation between layer $[l]$ and layer $[l+1]$:

$$\frac{\partial L}{\partial z_i^{[l]}} = \frac{\partial a_i^{[l]}}{\partial z_i^{[l]}} \sum_{j=1}^{n_{l+1}} \frac{\partial z_j^{[l+1]}}{\partial a_i^{[l]}} \frac{\partial L}{\partial z_j^{[l+1]}} = f'(z_i^{[l]}) \sum_{j=1}^{n_{l+1}} W_{ji}^{[l+1]} \frac{\partial L}{\partial z_j^{[l+1]}}$$

$$a^{[l]} = f(z^{[l]})$$

Backward propagation

- Name $\frac{\partial L}{\partial z^{[l]}}$ to be the *Error* (contributing to loss) from l-th layer: $\delta^{[l]}$. (It's a vector of length n_l, and its i-th element is $\frac{\partial L}{\partial z_i^{[l]}} = \delta_i[l]$

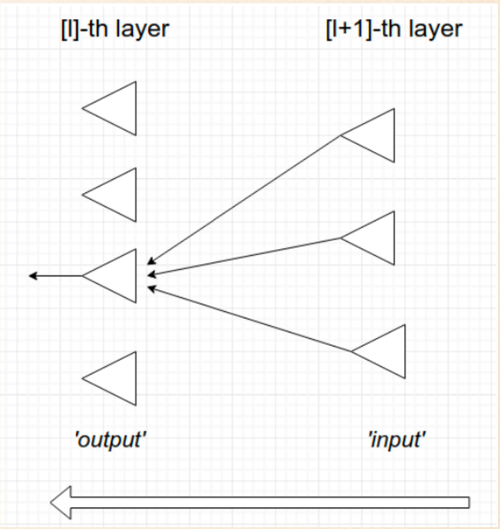
$$\frac{\partial L}{\partial W_{ij}^{[l]}} = a_j^{[l-1]} \delta_i^{[l]}$$

$$\frac{\partial L}{\partial b_i^{[l]}} = \delta_i^{[l]}$$

- The **weighted sum** of the error from $[l+1]$-th layer times activation derivative gives the error of $[l]$-th layer

$$\delta_i^{[l]} = f'(z_i^{[l]}) \sum_{j=1}^{n_{l+1}} W_{ji}^{[l+1]} \delta_j^{[l+1]}$$

- Like a new type of neuron - BP-neuron for calculating Error δ, with input $\delta^{[l+1]}$ and output $\delta^{[l]}$, where links are weights in forward-propagation. Afterward, the weighted sum multiply with $f'(z^{[l]})$- which acts like an amplifier



Backward propagation - Error propagation

- The error term in the output layer is: $$\delta_i^{[L]} = \frac{\partial L}{\partial z_i^{[L]}} = \frac{\partial a_i^{[L]}}{\partial z_i^{[L]}} \frac{\partial L}{\partial a_i^{[L]}} = \frac{\partial \hat{y}_i}{\partial z_i^{[L]}} \frac{\partial L}{\partial \hat{y}_i} = f'(z_i^{[L]}) \frac{\partial L}{\partial \hat{y}_i}$$

- Usually, we will find it's just (for MSE in regression, for Softmax/Sigmoid in classification) : $$\delta_i^{[L]} = \hat{y}_i - y_i$$

Backward propagation - Error propagation

- Then we can calculate the error in hidden layers via back-propagation layer by layer (from the output layer to the input layer) :

$$\delta^{[L]} \rightarrow \delta^{[L-1]} \rightarrow \delta^{[L-2]} \dots \rightarrow \delta^{[2]} \rightarrow \delta^{[1]}$$

with the recurrence relation :

$$\delta_i^{[l]} = f'(z_i^{[l]}) \sum_{j=1}^{n_{l+1}} W_{ji}^{[l+1]} \delta_j^{[l+1]}$$

or in the vectorized form :

$$\delta^{[l]} = f'(z^{[l]}) \cdot W^{[l+1]\top} \delta^{[l+1]} = W^{[l+1]\top} \delta^{[l+1]} \cdot f'(z^{[l]})$$

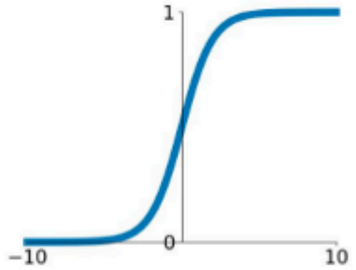
- Using $\partial L / \partial W_{ij}^{[l]} = a_j^{[l-1]} \delta_i^{[l]}$ and $\partial L / \partial b_i^{[l]} = \delta_i^{[l]}$, we can get the gradient w.r.t weights and biases from each layer's activation value $a^{[l]}$ and error terms $\delta^{[l]}$ in the backward direction. -- Vectorized form: $\partial L / \partial W^{[l]} = \delta^{[l]} a^{[l-1]T}$ and $\partial L / \partial b^{[l]} = \delta^{[l]}$

- Then the optimization can be done via gradient descent.

Activation functions

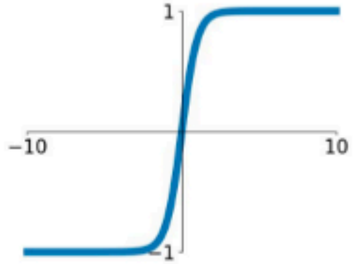
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



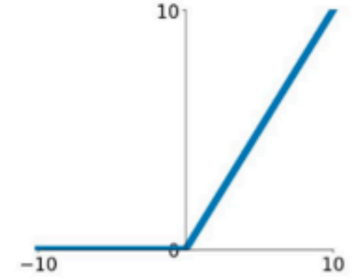
tanh

$$\tanh(x)$$



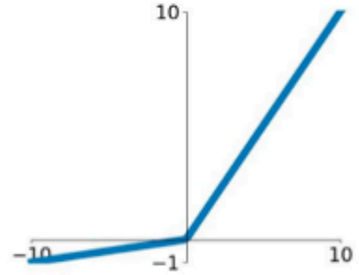
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

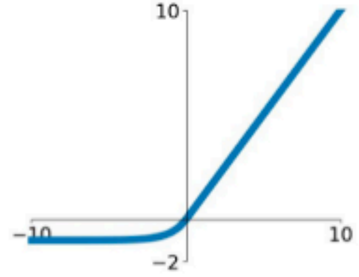
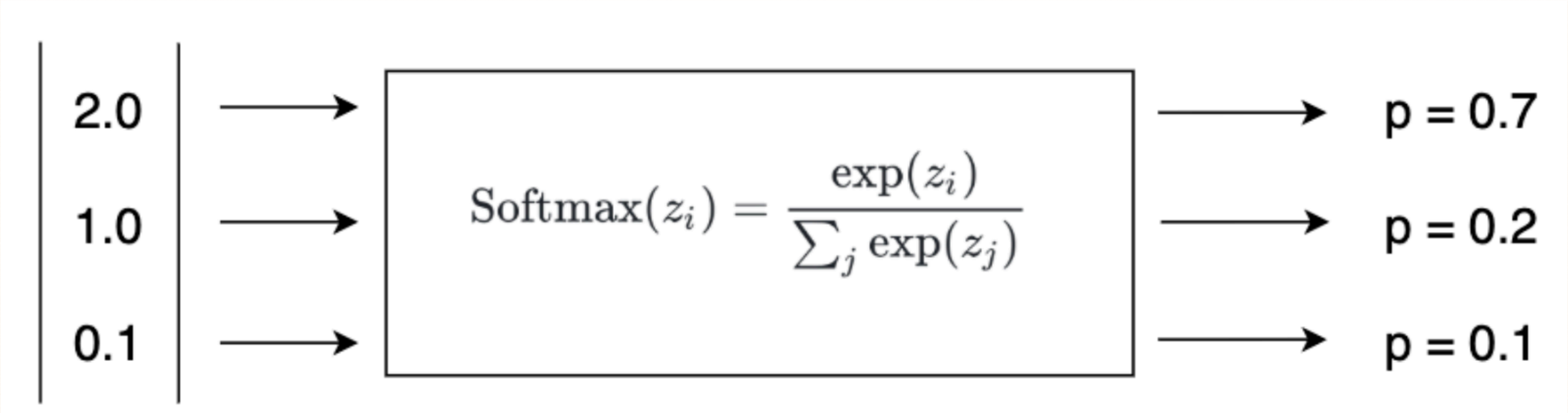


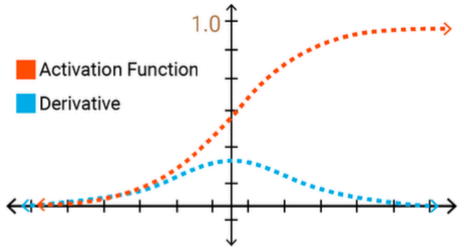
Figure from ResearchGate

Activation functions

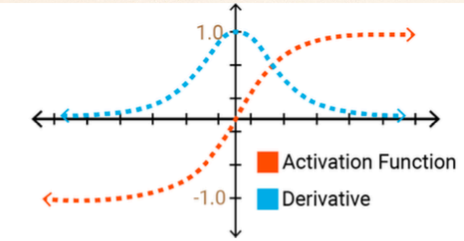


Vanishing/Exploding Gradients Problems

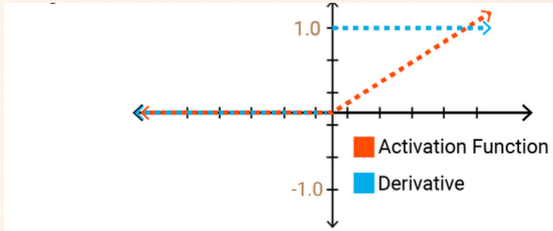
$$\delta_i^{[l]} = f'(z_i^{[l]}) \sum_{j=1}^{n_{l+1}} W_{ji}^{[l+1]} \delta_j^{[l+1]}$$



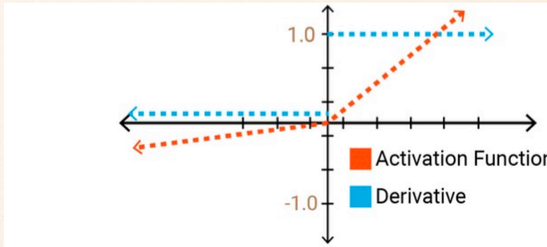
$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma'(z) = \sigma(z)(1-\sigma(z))$$



$$\tanh(z) = \sinh(z)/\cosh(z) = 2\sigma(2z) - 1$$
$$\tanh'(z) = 1 - \tanh^2(z)$$



$$ReLU(z) = \max(0, z), \quad ReLU'(z) = \frac{sign(z)+1}{2}$$



$$LReLU(z) = \max(\alpha z, z)(\alpha = 0.01) = \begin{cases} z, & z > 0 \\ \alpha z, & x \leq 0 \end{cases}$$

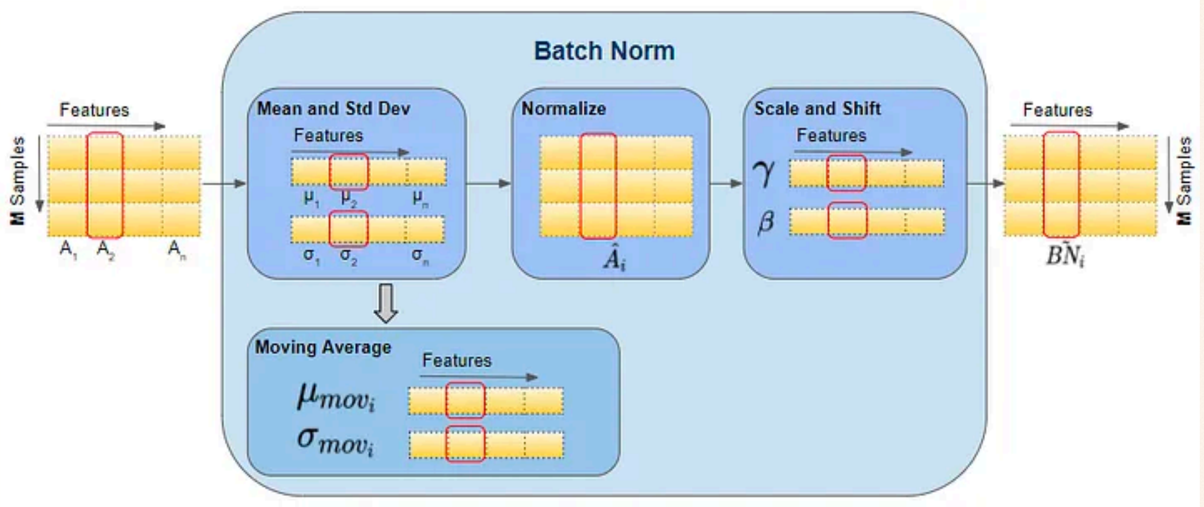
$$PReLU(z) = \max(\alpha z, z) = \begin{cases} z, & z > 0 \\ \alpha z, & x \leq 0 \end{cases}$$

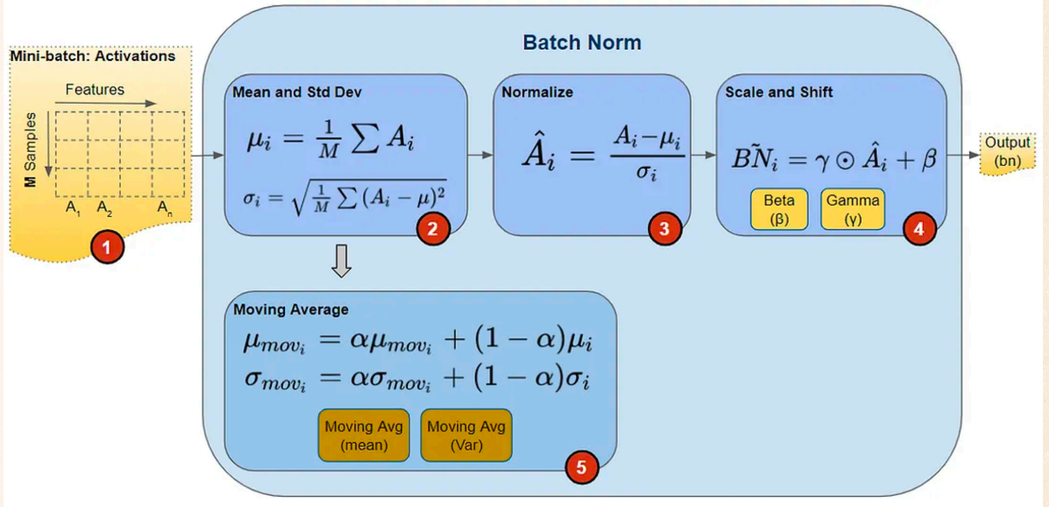
Credit: Kai Zhou

Kernel Initializer

- Glorot Uniform
- Glorot Normal $W_{ij} \sim N(0, \frac{1}{m}), m = (n_{input} + n_{output})/2$
- He Uniform
- He Normal $W_{ij} \sim N(0, \frac{2}{m}), m = n_{input}$
- Random Uniform
- Random Normal $W_{ij} \sim N(0,1)$

Batch normalization



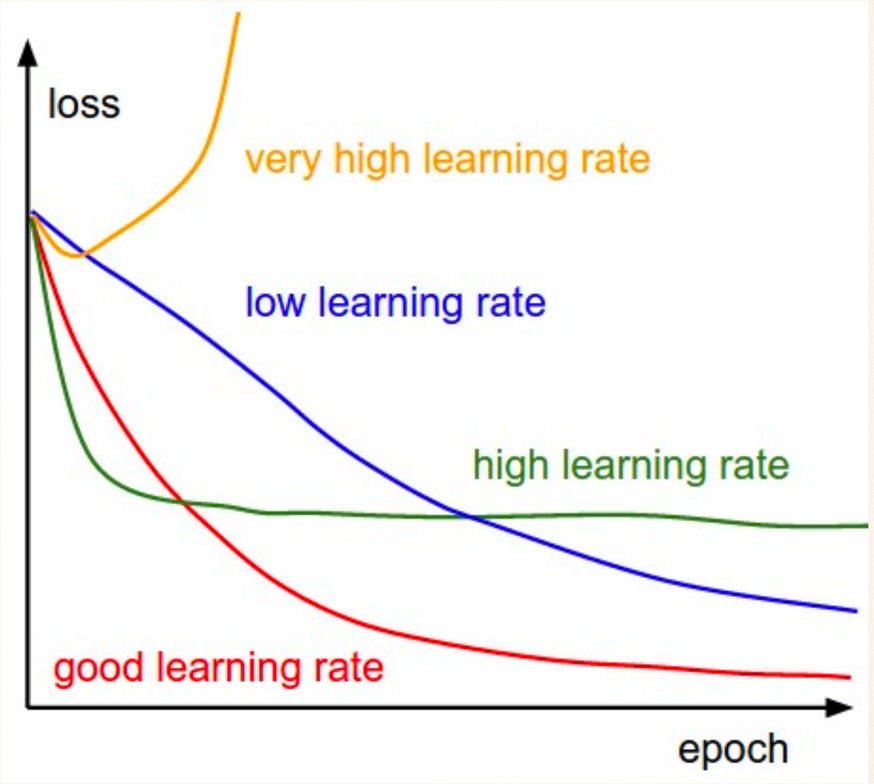


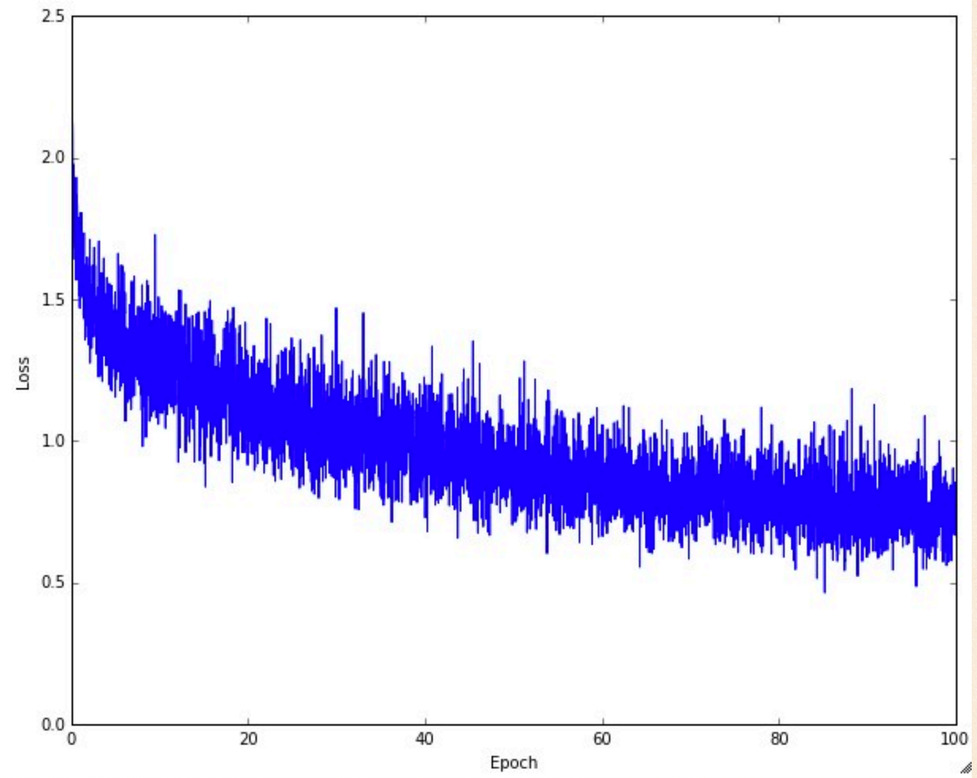
- At each layer, each feature in the input is normalized for the batch

Figure from [medium]

Learning Rate

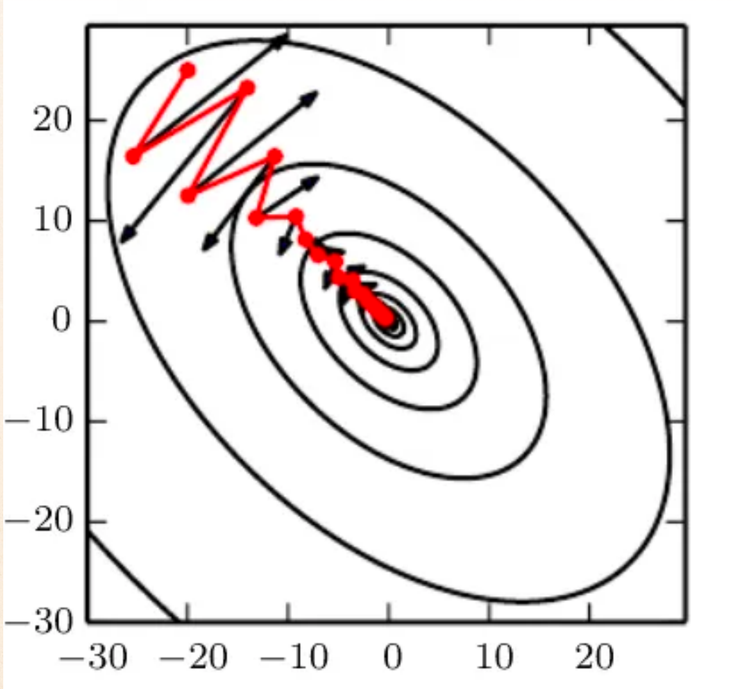
Figure from [CS231N, Stanford]





Faster Optimizers

- Momentum: cares about earlier gradients & pick up speed to be faster
- Nesterov Accelerated Gradient
- AdaGrad
- RMSProp
- Adam
- AdaMax
- Nadam
- Adam W



Serve as acceleration

Compute gradient estimate:

$$g = \frac{1}{m} \sum_i \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$$

Friction: α

Update velocity:

$$v = \alpha v - \varepsilon g$$

Update parameters:

$$\theta = \theta + v$$

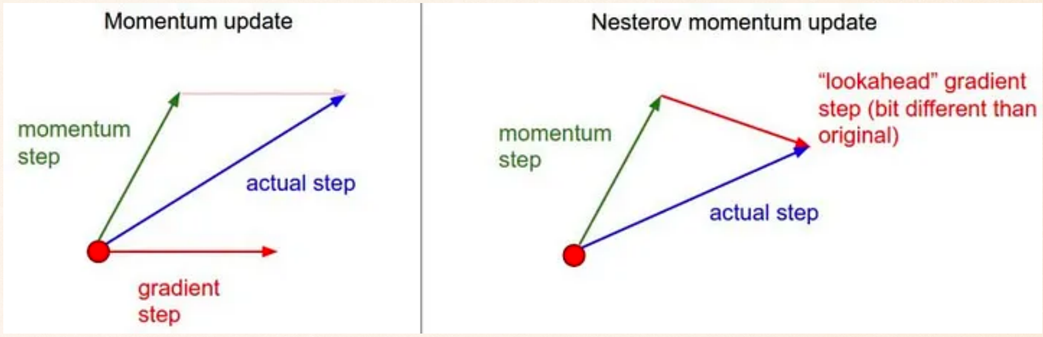
Figure from medium

Faster Optimizers

Figure from CS231N, Stanford

- Momentum
- Nesterov Accelerated Gradient
- AdaGrad
- RMSProp
- Adam
- AdaMax
- Nadam
- Adam W

Almost always faster than momentum



Apply an interim update:

$$\tilde{\theta} = \theta + v$$

Perform a correction based on gradient at the interim point:

$$g = \frac{1}{m}\sum_i \nabla_\theta L(f(x^{(i)};\tilde{\theta}), y^{(i)})$$

$$v = \alpha v - \varepsilon g$$

$$\theta = \theta + v$$

Momentum based on look-ahead slope

Figure from medium

Adaptive learning rate in different directions

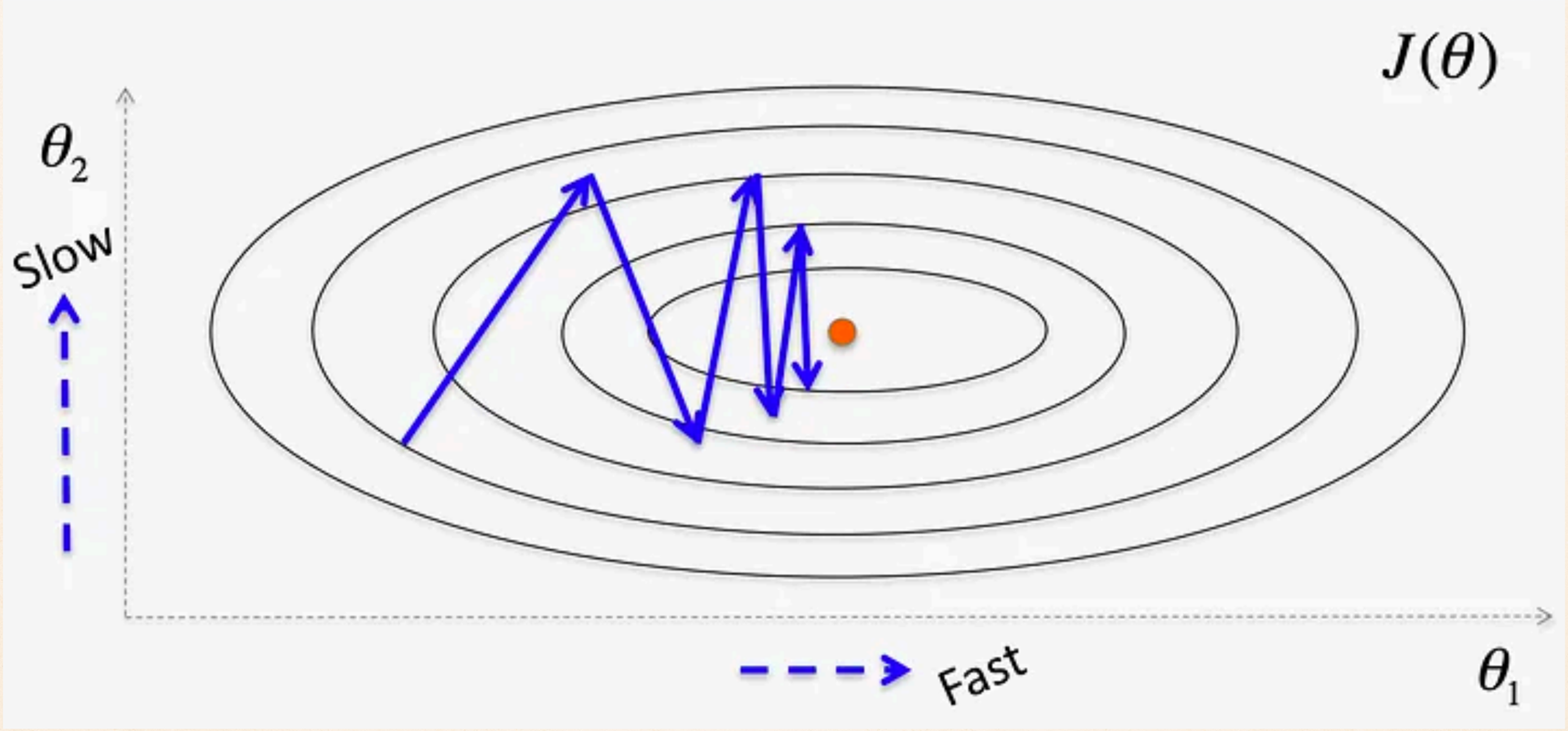


Figure from [medium]

Faster Optimizers

- Momentum
- Nesterov Accelerated Gradient
- AdaGrad (Adaptive Gradient)
- RMSProp (Root Mean Squared Propagation)
- Adam (~momentum+RMSProp)
- AdaMax
- Nadam (~Nesterov+adam)
- Adam W (~Adam+weight decay)

Accumulate squared gradients:

$$r_i = r_i + g_i^2$$

Update each parameter:

$$\theta_i = \theta_i - \frac{\varepsilon}{\delta + \sqrt{r_i}} g_i$$

Inversely proportional to cumulative squared gradient

$$r_i = \rho r_i + (1-\rho) g_i^2$$

$$\theta_i = \theta_i - \frac{\varepsilon}{\delta + \sqrt{r_i}} g_i$$

- AdaGrad slows down the gradient vector along the steepest dimension
- But it decays the learning rate, bad for neural network training
- Fixed by RMSProp

Figure from medium

Faster Optimizers

- Momentum
- NAG
- AdaGrad
- RMSProp
- Adam (adaptive momentum estimation)

Estimate first moment:

$$v_i = \rho_1 v_i + (1 - \rho_1) g_i$$

Estimate second moment:

$$r_i = \rho_2 r_i + (1 - \rho_2) g_i^2$$

Update parameters:

$$\theta_i = \theta_i - \frac{\varepsilon}{\delta + \sqrt{r_i}} v_i$$

Also applies bias correction to v and r

Works well in practice, is fairly robust to hyper-parameters

Figure from medium

Loss function

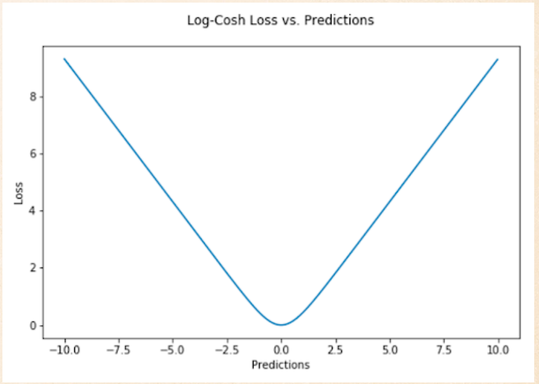
- Cross-entropy (For classification)
- Log loss
- Exponential Loss
- Hinge Loss
- Kullback Leibler Divergence Loss
- Mean Square Error (MSE, L2)
- Mean Absolute Error (MAE, L1)
- Huber Loss
- Logcosh Loss

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

$$L(y, y^p) = \sum_{i=1}^{n} \log(\cosh(y_i^p - y_i))$$



Save and load models

SavedModel format

The SavedModel format is another way to serialize models. Models saved in this format can be restored using `tf.keras.models.load_model` and are compatible with TensorFlow Serving. The SavedModel guide goes into detail about how to `serve/inspect` the SavedModel. The section below illustrates the steps to save and restore the model.

```
# Create and train a new model instance.
model = create_model()
model.fit(train_images, train_labels, epochs=5)

# Save the entire model as a SavedModel.
!mkdir -p saved_model
model.save('saved_model/my_model')
```

HDF5 format

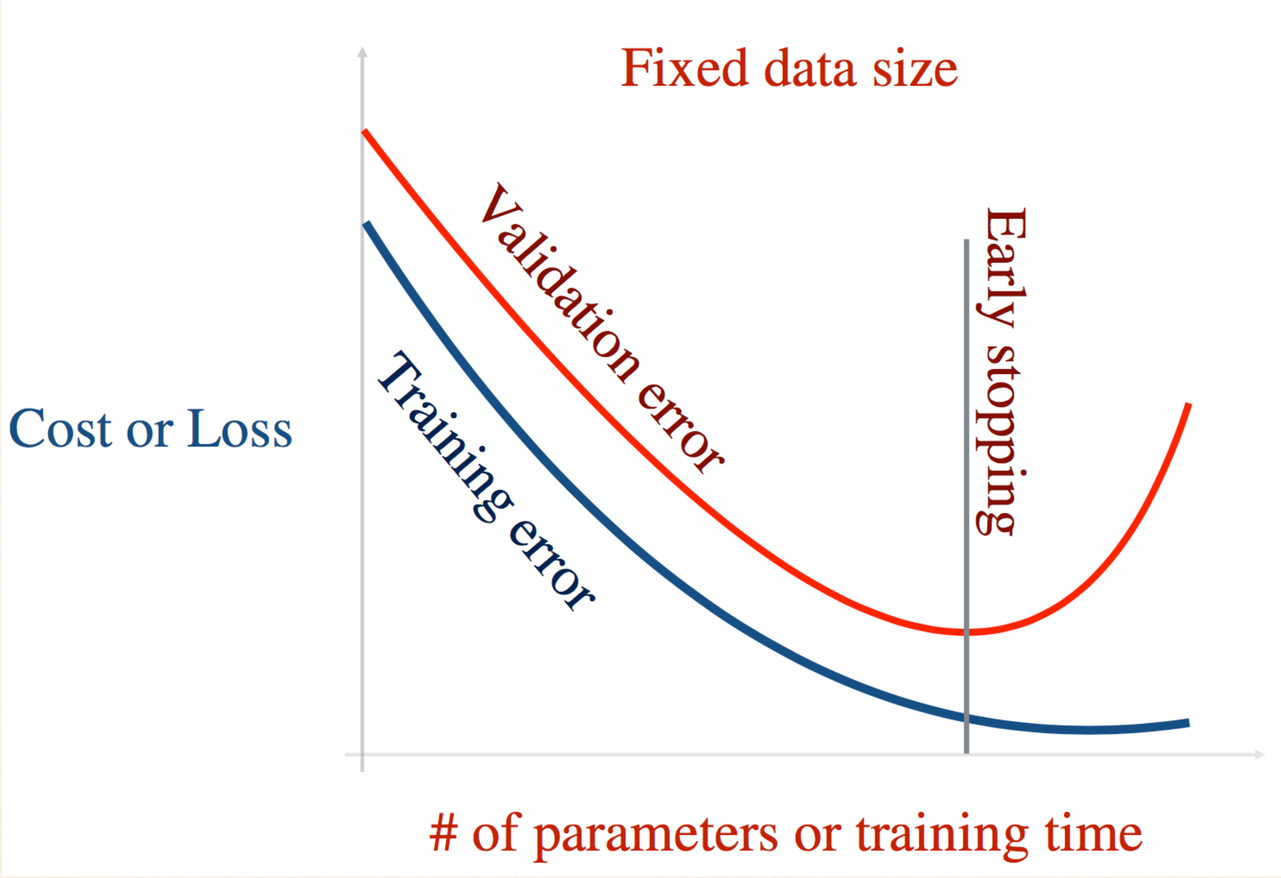
Keras provides a basic save format using the HDF5 standard.

```
# Create and train a new model instance.
model = create_model()
model.fit(train_images, train_labels, epochs=5)

# Save the entire model to a HDF5 file.
# The '.h5' extension indicates that the model should be saved to HDF5.
model.save('my_model.h5')
```

Figure from tensorflow

Overfitting problem in fully connected network



Too many parameters may easily overfit training dataset

Credit: Longgang Pang

Ways to reduce overfitting

- Early stopping
- Increase training dataset by
  - preparing more data
  - data augmentation (crop, scale, rotate, flip ...)
- Reduce number of parameters by
  - Dropout: randomly discarding neurons
  - Drop connect: randomly discarding connections
  - CNN: locally connected to a small chunk of neurons in the previous layer
  - Go deeper
- Regularization, weight decay…

Data augmentation

```
datagen = ImageDataGenerator(
        zoom_range=0.2,
        shear_range=0.2,
        rotation_range=40,
        fill_mode='nearest',
        horizontal_flip=True,
        preprocessing_function = image_contrast_adjusment)
```

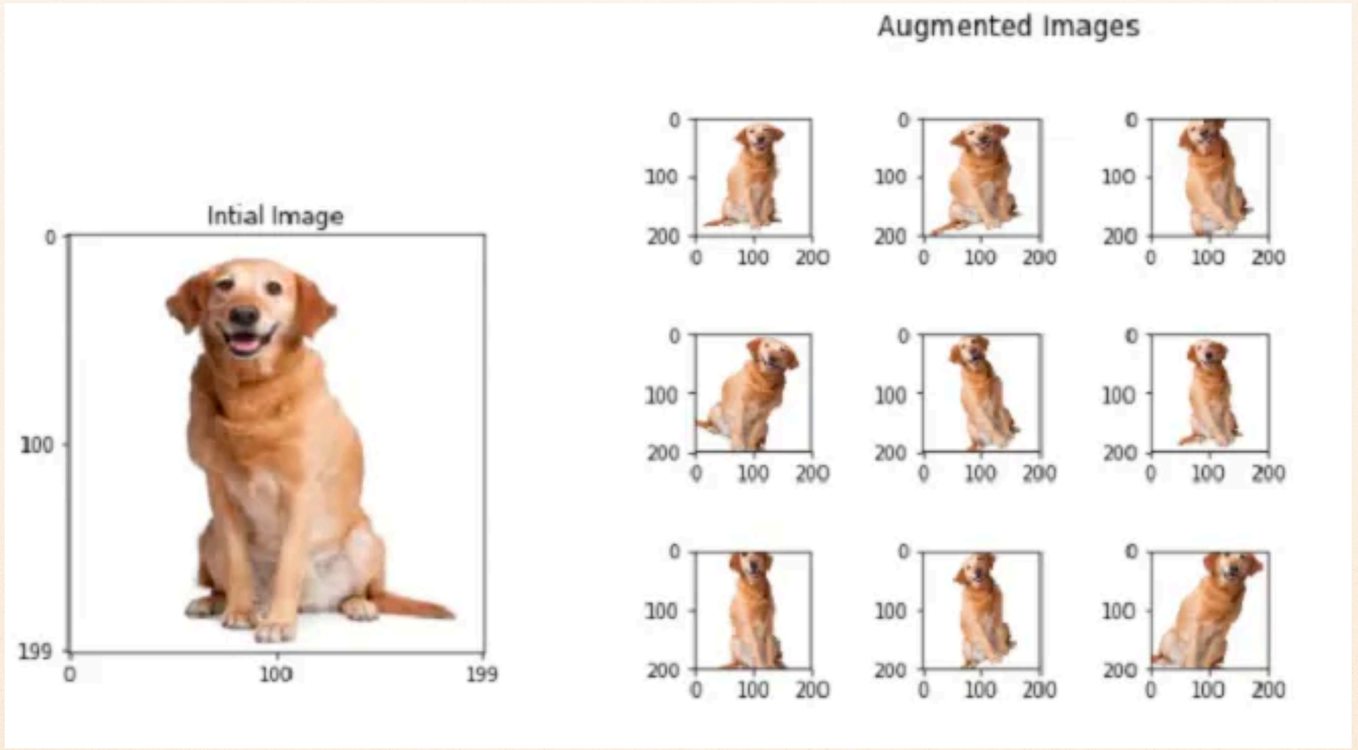
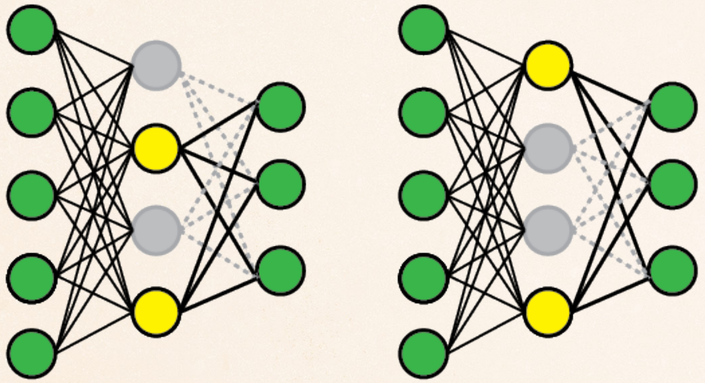
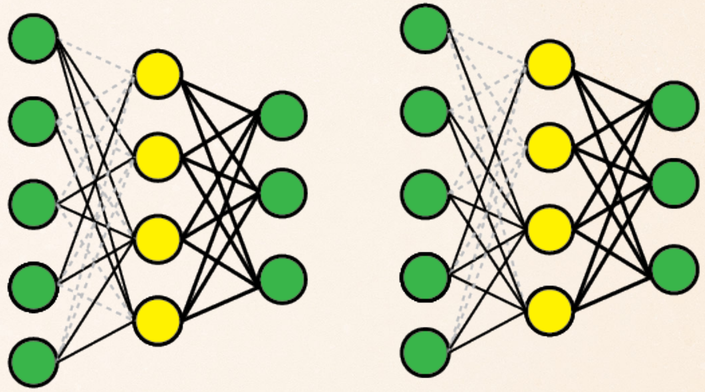


Figure from [medium]

Dropout and Drop Connect



Dropout



Drop Connect

Figures from StackExchange

L1 & L2 regularization

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda\sum_{i=1}^{N} \mid \theta_i \mid$$

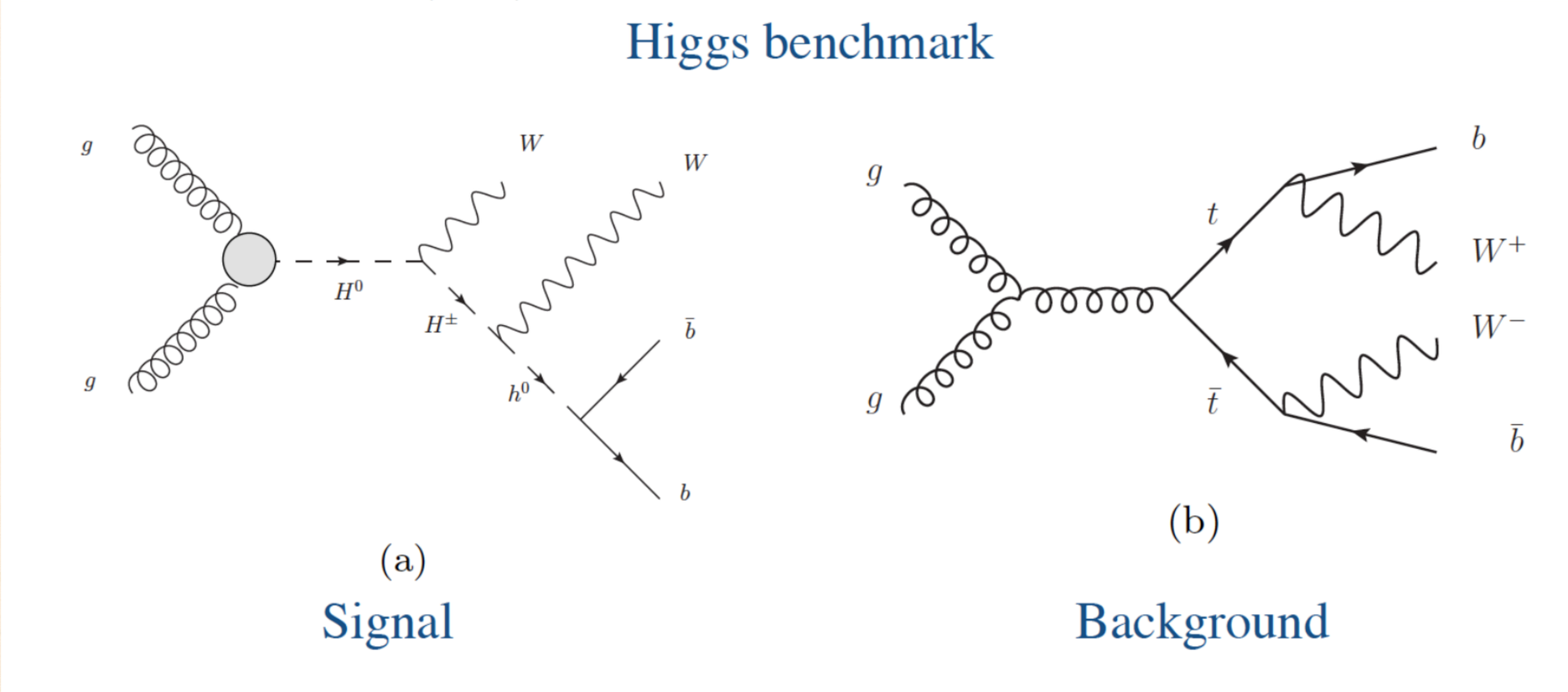
Leads to sparse model

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda\sum_{i=1}^{N} \theta_i^2$$

Constrain connection weights

Physics applications: particle physics

Searching for Exotic Particles in High Energy Physics with Deep Learning



8% improvement over the best approaches till publication

P. Baldi, P. Sadowski, and D. Whiteson, Nature Commun. 5,4308(2014)

Low-level and high-level features

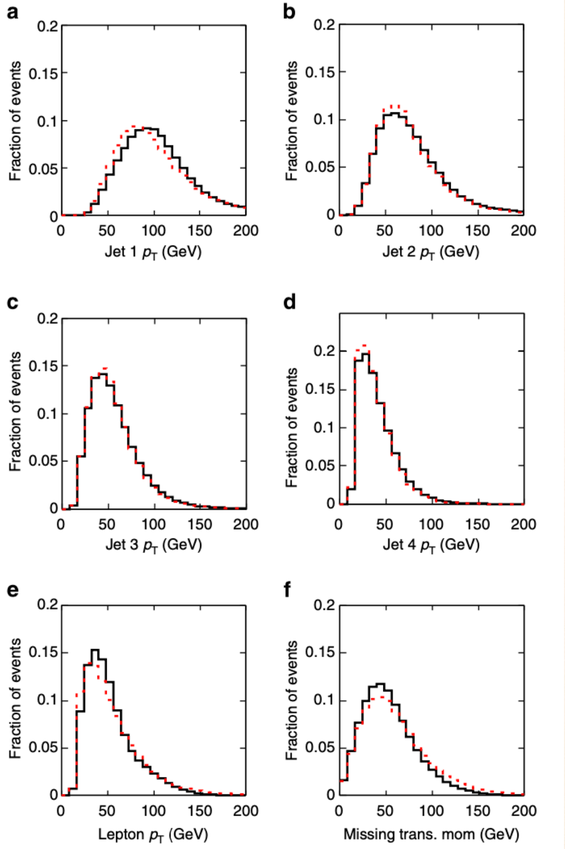


Figure 2 | Low-level input features for Higgs benchmark. Distributions in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) benchmark events. Shown are the distributions of transverse momenta (p_T) of each observed particle (**a**–**e**) as well as the imbalance of momentum in the final state (**f**). Momentum angular information for each observed particle is also available to the network, but is not shown, as the one-dimensional projections have little information.

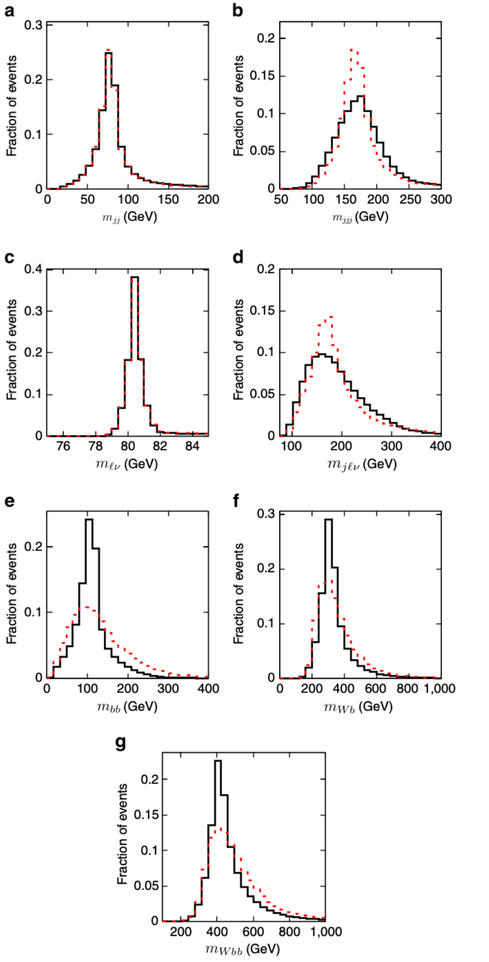


Figure 3 | High-level input features for Higgs benchmark. Distributions in simulation of invariant mass calculations in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) events.

Performance: area under the ROC curve (AUC)

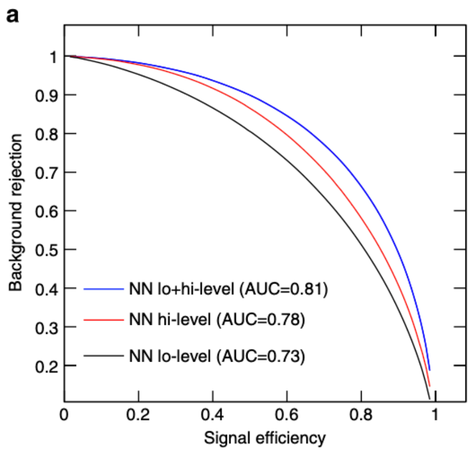


Figure 7 | Performance for Higgs benchmark. For the Higgs benchmark, comparison of background rejection versus signal efficiency for the traditional learning method (**a**) and the deep learning method (**b**) using the low (lo)-level features, the high (hi)-level features and the complete set of features.

- Shallow NN fails to independently discover the discriminating power of the high-level features from low level features.
- High-level features do not capture all the information contained in the low-level features.
- Deep NN automatically discover the insight contained in the high-level features from low level features.
- Deep NN finds additional separation power beyond what is contained in the high-level features compared with shallow NN.

Introduction to Machine Learning and Deep Learning

Yi-Lun Du（杜轶伦）
Shandong Institute of Advanced Technology

July 18-19, Qingdao
QCD and Nuclear Physics Summer School

Why we need convolution neural network?

- Simple flatten of image/audio/video kind of discard the 2D or 3D structural information inside the data -- like some local invariances which exist in practice and can help efficiently recognize images.
- Imagine input image resolution is 100x100x3 ('3' is for RGB color channels), in fully connected network each neuron in the first hidden layer would have 100x100x3=30,000 independent connections (weight value) -- too many parameters easily induce overfitting and slow learning of the model.
- Inspired from biological brain's visual cortex (视觉皮层) study, especially the concept of receptive field (感受视野) -- many neurons react only to visual stimuli (视觉刺激) located in a small region of the visual field. This further induces hierarchical representation

Convolution neural network: example

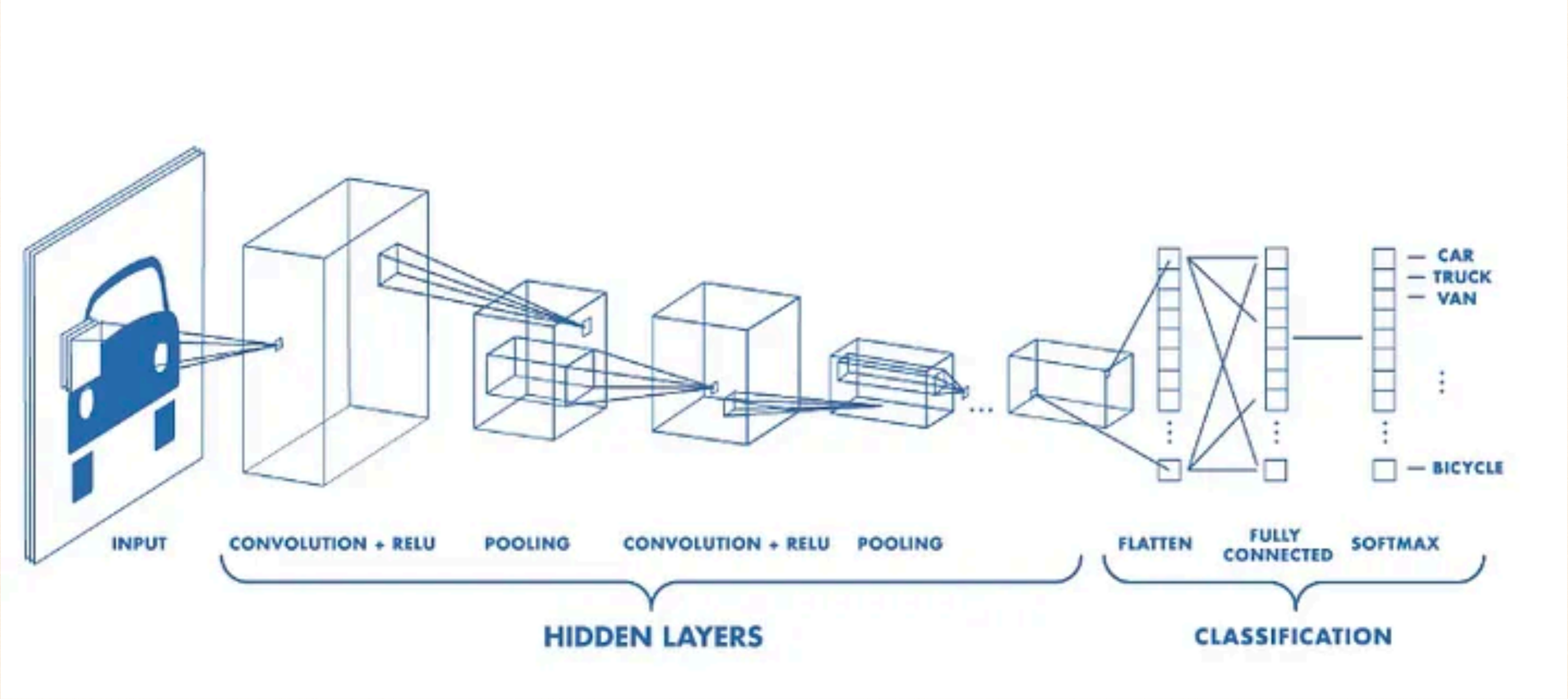
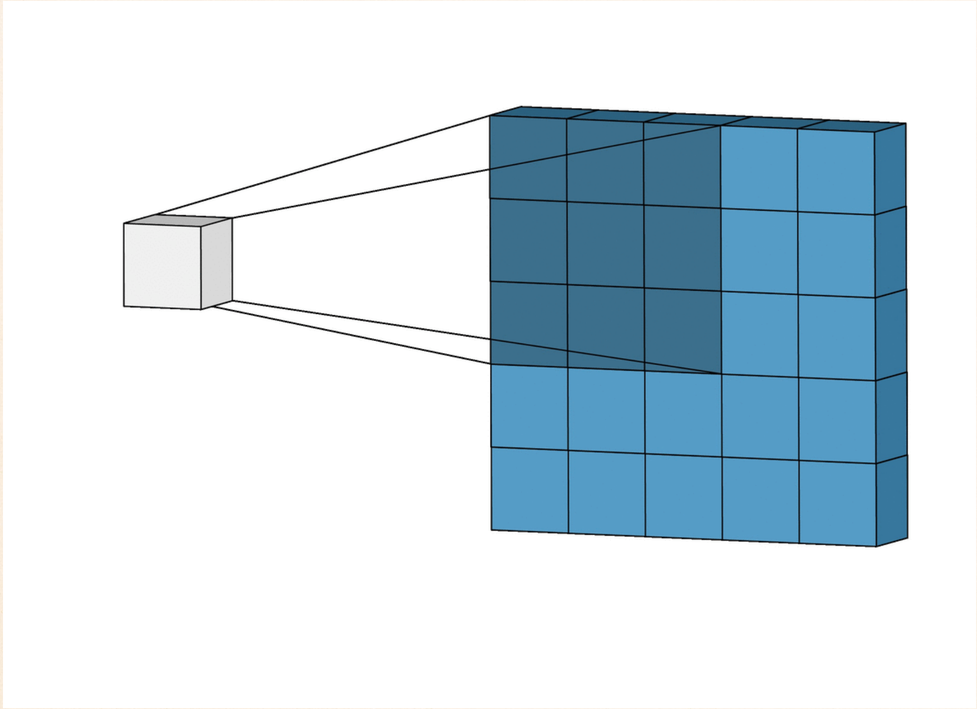


Figure from MathWorks

Convolution Operation



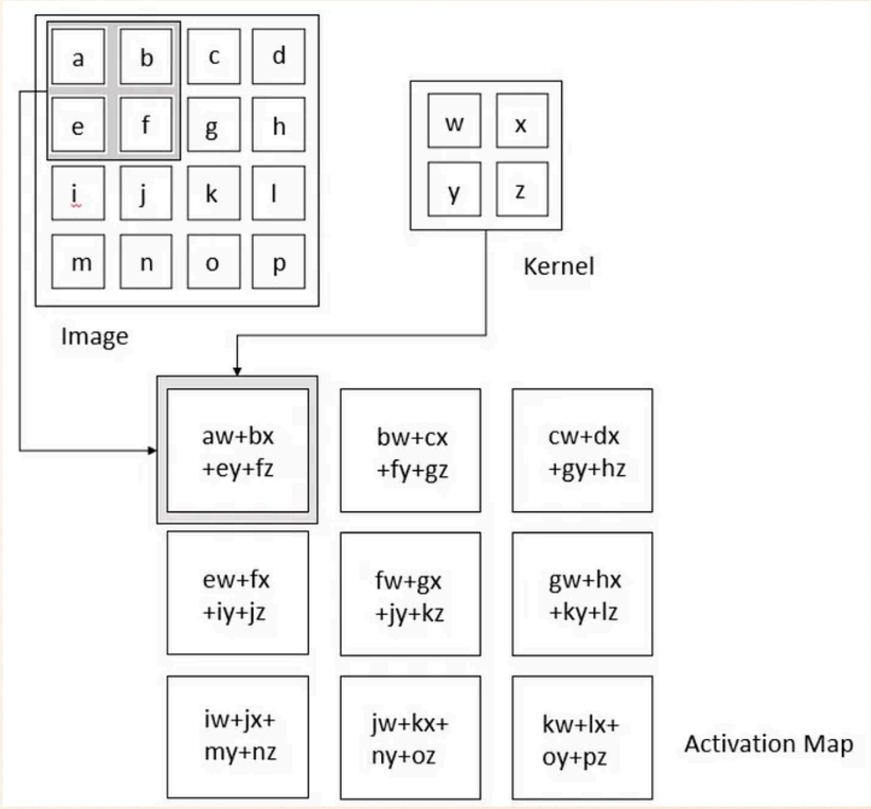
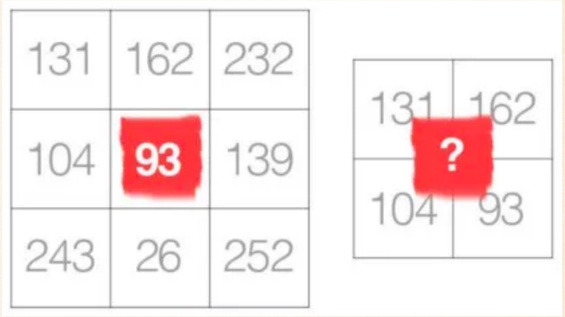


Figure from [medium]

Kernel size

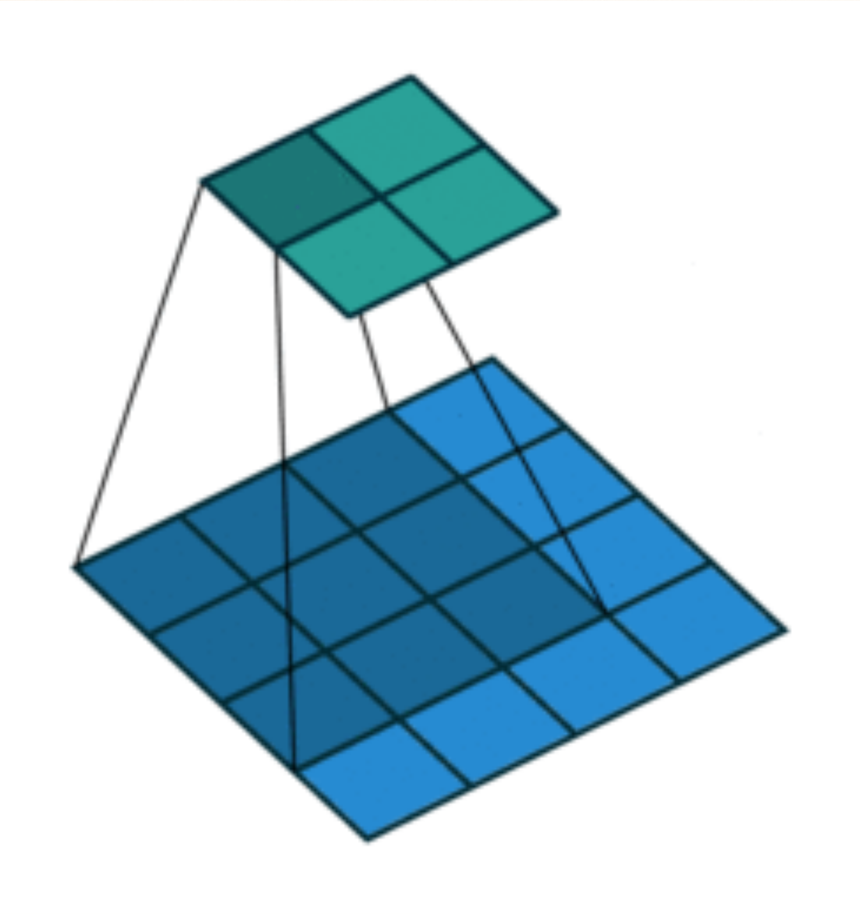


- Most popular choice: 3*3
- Even-sized kernels 2*2, 4*4 are not preferred due to the symmetry
- 1*1 kernel does not try to capture the information from the neighborhood pixels
- Large sizes 5*5, 7*7, etc., are in general more time-consuming

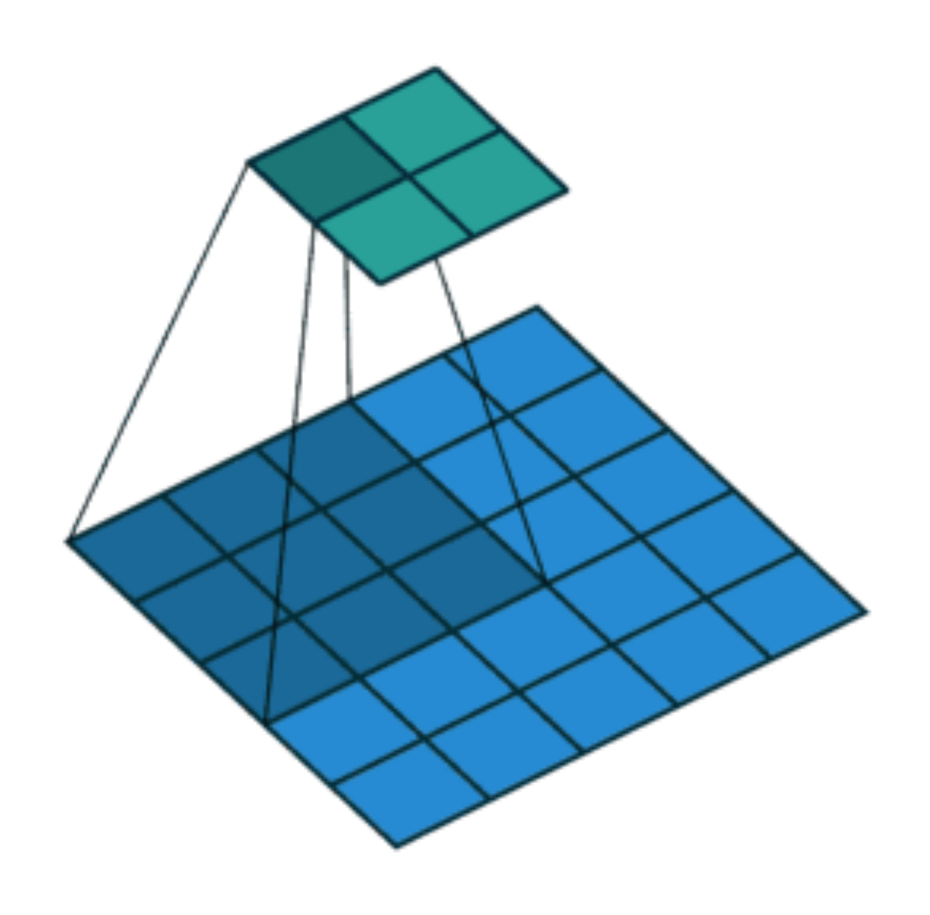
Figure from medium

Stride （步长）

$$W_{out} = \frac{W - F}{S} + 1$$

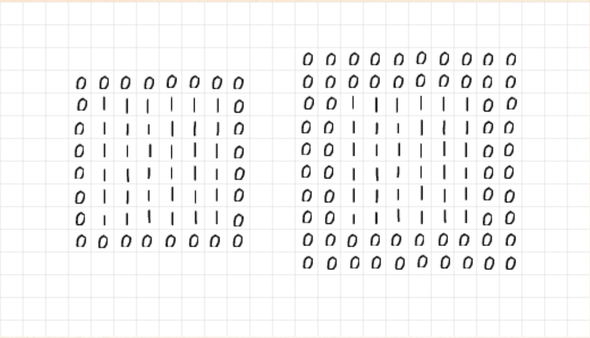


Stride = (1,1)



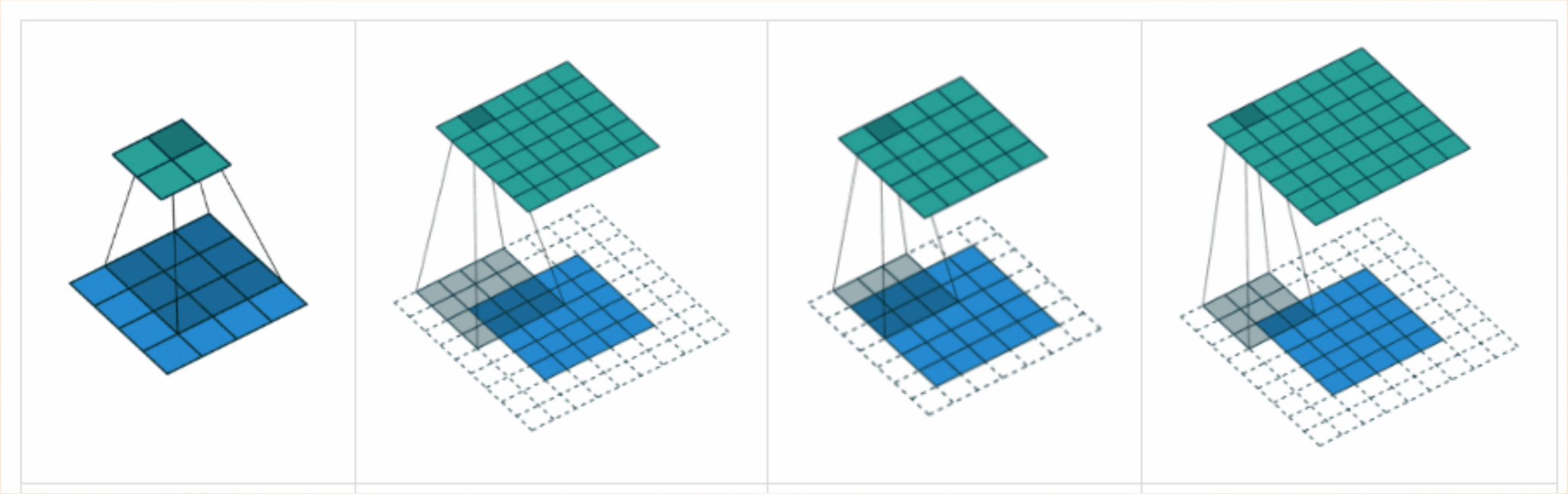
Stride = (2,2)

Figures from MathWorks



Padding（填充）

$$W_{out} = \frac{W - F + 2P}{S} + 1$$



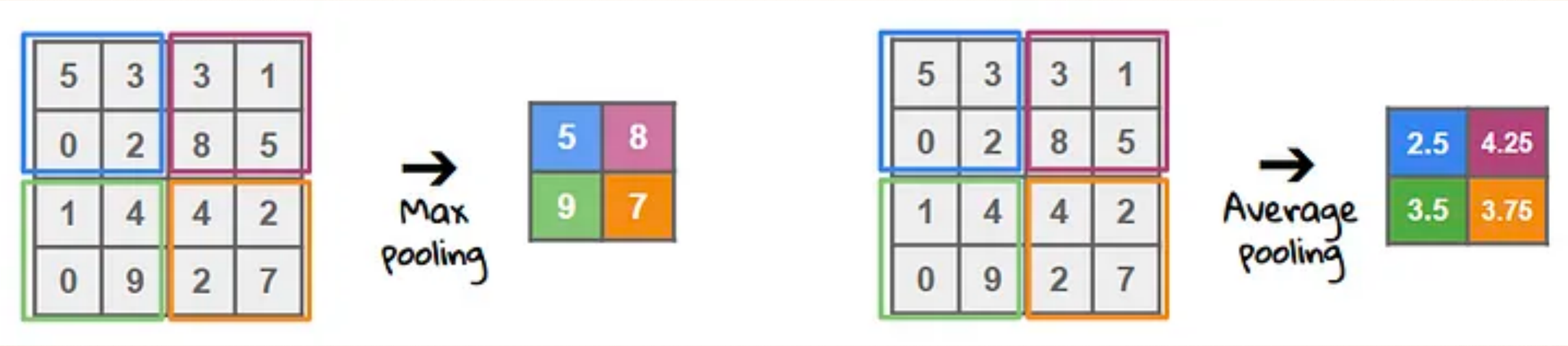
valid

same

full

Figure from medium

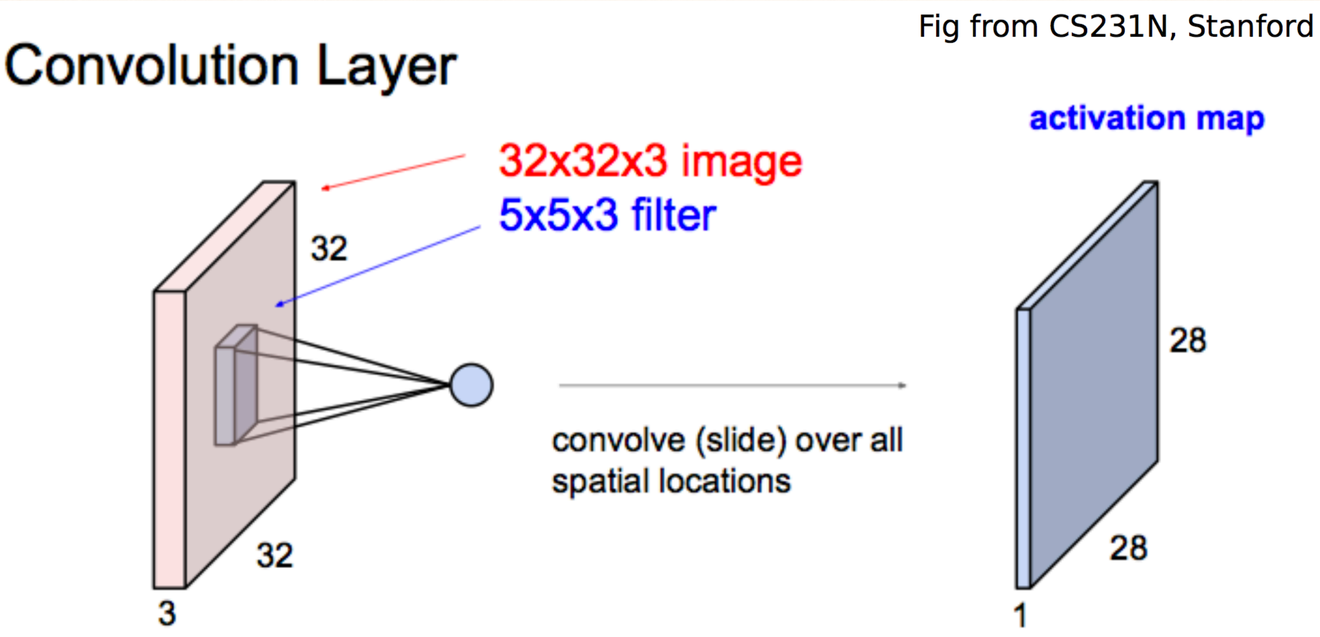
Pooling（池化）



- Max pooling generally performs better than Average one since it preserves only the strongest signal

Figure from [medium]

3D Convolution



Advantage: scaling, rotating, translation invariant features can be learned since only subregion is connected to the filter/kernel which scan the whole input to feel the 2d structure and local statistics, and weight shared.

Flattening

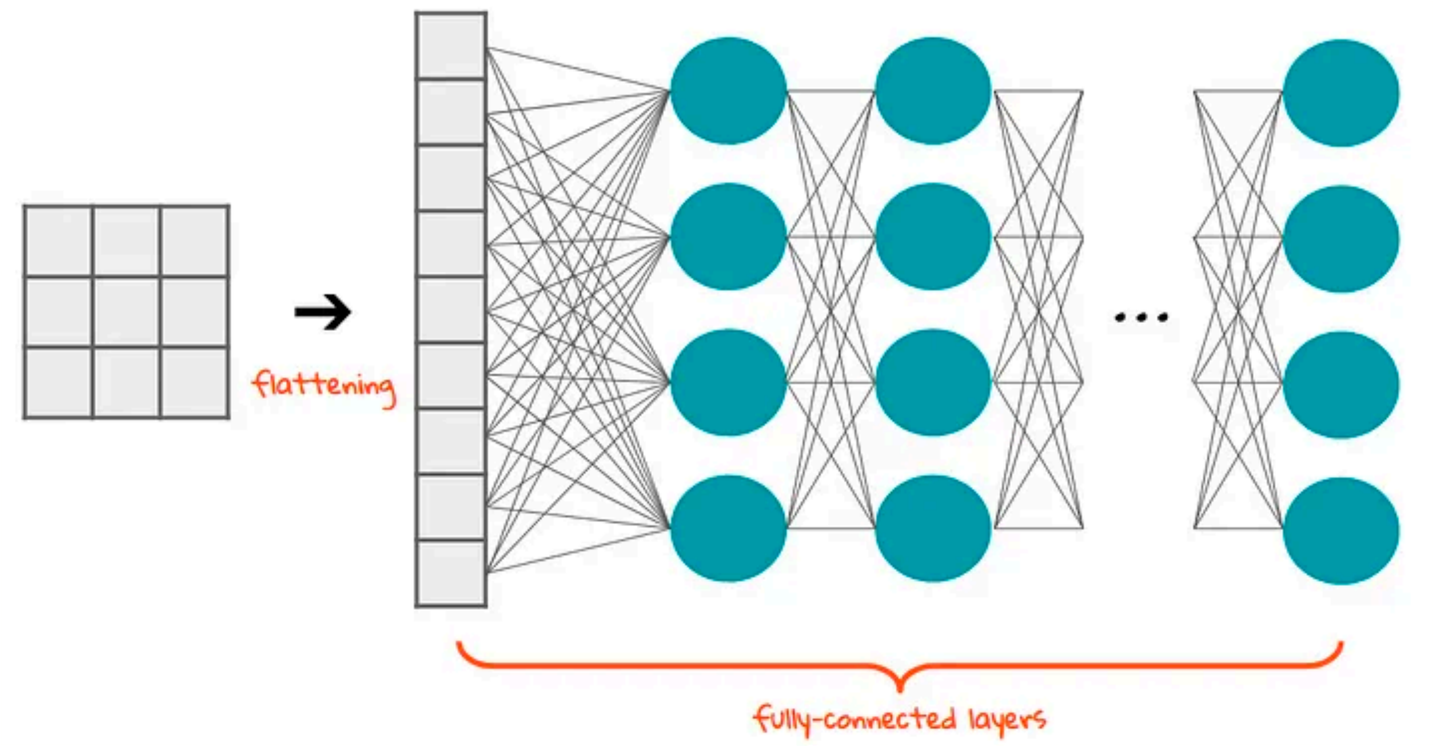


Figure from medium

How to understand CNN?

25	223	196	249	60	.....	47	126	121	237

25	2	1	44
223	7	6	60
196	8	2	148
249	1	3	40
60	7	1	154
59	1	7	213
214	7	3	163
89	182	219	13
74	146	113	72
89	18	244	85
1	4	8	97
3	4	2	121
2	1	2	131
7	6	8	47
3	5	5	126
7	6	8	121
5	3	1	237

86	4	8	184
252	3	8	40
34	7	7	163
105	2	3	69
56	3	8	175
126	1	2	178
163	8	4	142
22	222	74	180
163	158	204	253
245	98	85	180
1	5	1	98
4	2	8	90
7	8	1	235
2	1	3	217
3	6	5	97
6	5	8	79
8	8	5	133

1	0.3

=SUMPRODUCT(B2:C2,$G$5:$H$5)

63.2		
20	252.9	5.4
55.9	36.1	9.1
23.7	105.6	2.9
60.5	56.9	5.4
55.4	126.3	1.6
46.6	165.4	9.2
128	88.6	244.2
279.9	210.4	219.2
139	274.4	123.5
30.4	2.5	5.3
35	4.6	4.4
71.5	9.4	8.3
68.1	2.3	1.9
34.1	4.8	7.5
31.7	7.5	7.4
44.9	10.4	9.5

28.6	40.4	920.8
40.2	40.3	200.8
38.4	35.7	815.7
20.5	15.2	345.3
20.6	40.3	875.8
17.6	10.1	890.2
56.3	20.8	710.4
1112.2	392.2	907.4
806.3	1035.8	1285.4
514.5	434.8	908.5
25.1	5.5	490.1
10.4	40.2	450.8
40.7	5.8	1175.1
5.2	15.1	1085.3
30.3	25.6	485.5
25.6	40.5	395.8
40.8	25.8	665.5

(0.1,5)

86	4	8	184
252	3	8	40
34	7	7	163
105	2	3	69
56	3	8	175
126	1	2	178
163	8	4	142
22	222	74	180
163	158	204	253
245	98	85	180
1	5	1	98
4	2	8	90
7	8	1	235
2	1	3	217
3	6	5	97
6	5	8	79
8	8	5	133

1	0.3
0.5	2

219.2	23.9	147.2
283.9	22.9	349.5
92.6	16.1	195.4
139.6	20.4	377.7
121.9	9.9	417.5
223.8	13.6	341.4
620.4	268.2	443.6
486.1	731.2	736
528.9	438.2	682.4
284.9	128	335.5
8.5	22.3	214.4
24.1	10.4	505.5
12.4	14.8	507
15.8	14.9	264.6
17.8	26	196.1
27.5	21.4	300.2

Dishashree Gupta, ⟨⟨Architecture of Convolutional Neural Networks (CNNs) demystified⟩⟩

CNN vs FCN in image recognition

- Local features are broken in flattened representation
- FCN: converted to class of input data and improve the Generalizablity - “firewall”
- 2D/3D filters with certain weights scan the whole picture to capture different features in neighborhood: edges, specific color, blurring noise...
- Convolutional and Pooling layer: reduce No. of dimension and parameters
- The deeper layer, the more abstract features learned

Visualizing Deep Neural Network Decisions

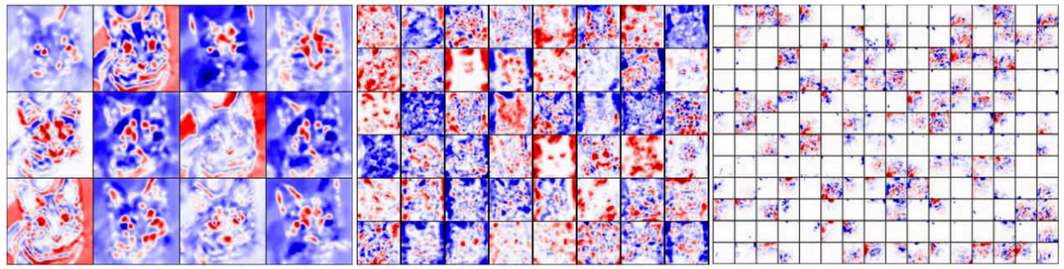


Figure 5: Visualization of feature maps from thee different layers of the GoogLeNet (l.t.r.: "conv1/7x7_s2", "inception_3a/output", "inception_5b/output"), using conditional sampling and patch sizes $k = 10$ and $l = 14$ (see alg. 1). For each feature map in the convolutional layer, we first evaluate the relevance for every single unit, and then average the results over all the units in one feature map to get a sense of what the unit is doing as a whole. *Red* pixels activate a unit, *blue* pixels decreased the activation.

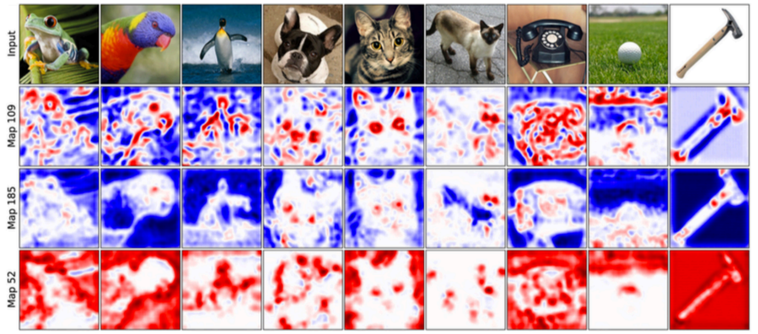
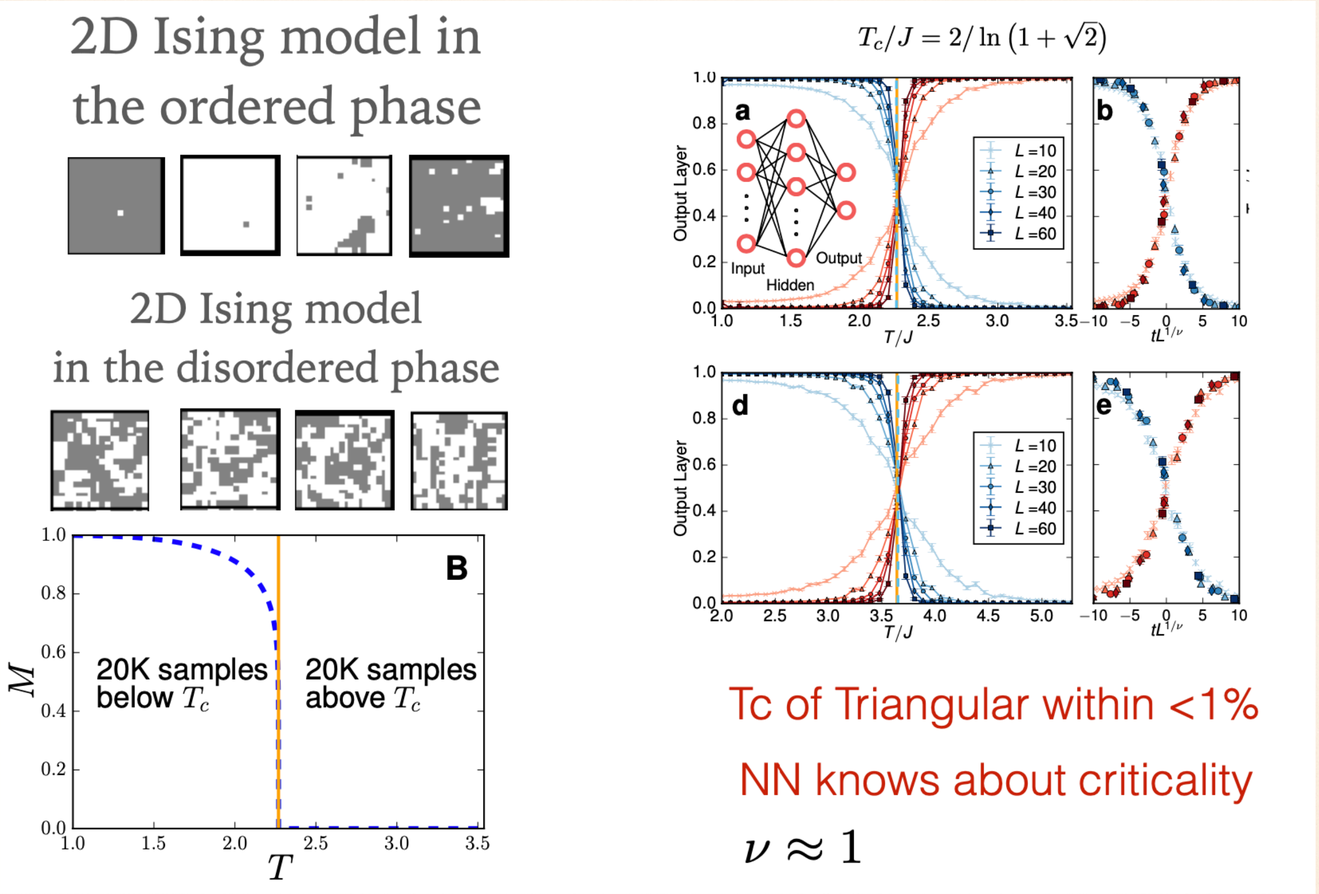


Figure 6: **Visualization of three different feature maps**, taken from the "inception_3a/output" layer of the GoogLeNet (from the middle of the network). Shown is the average relevance of the input features over all activations of the feature map. We used patch sizes $k = 10$ and $l = 14$ (see alg. 1). *Red* pixels activate a unit, *blue* pixels decreased the activation.

Luisa M Zintgraf, Taco S Cohen, Tameem Adel, Max Welling, arXiv:1702.04595

Physics applications: condensed matter physics

Machine learning phases of matter



Tc of Triangular within <1%

NN knows about criticality

$\nu \approx 1$

J. Carrasquilla and R. G. Melko, Nature Physics 13, 431-434 (2017)

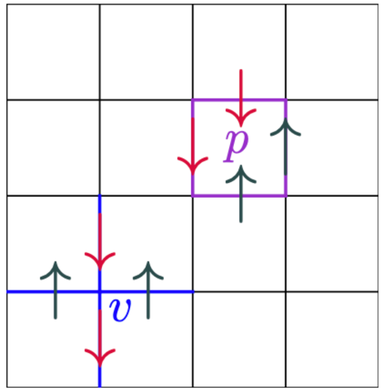
Topological phase

Wegner's Ising gauge theory:

$$H = -J \sum_p \prod_{i \in p} \sigma_i^z$$

F.J. Wegner, J. Math. Phys. 12 (1971) 2259

(Kogut Rev. Mod. Phys. 51, 659 (1979))



The ground state is a highly degenerate manifold with exponentially decaying spin–spin correlations.

The grandmother of all lattice models for topological quantum computation

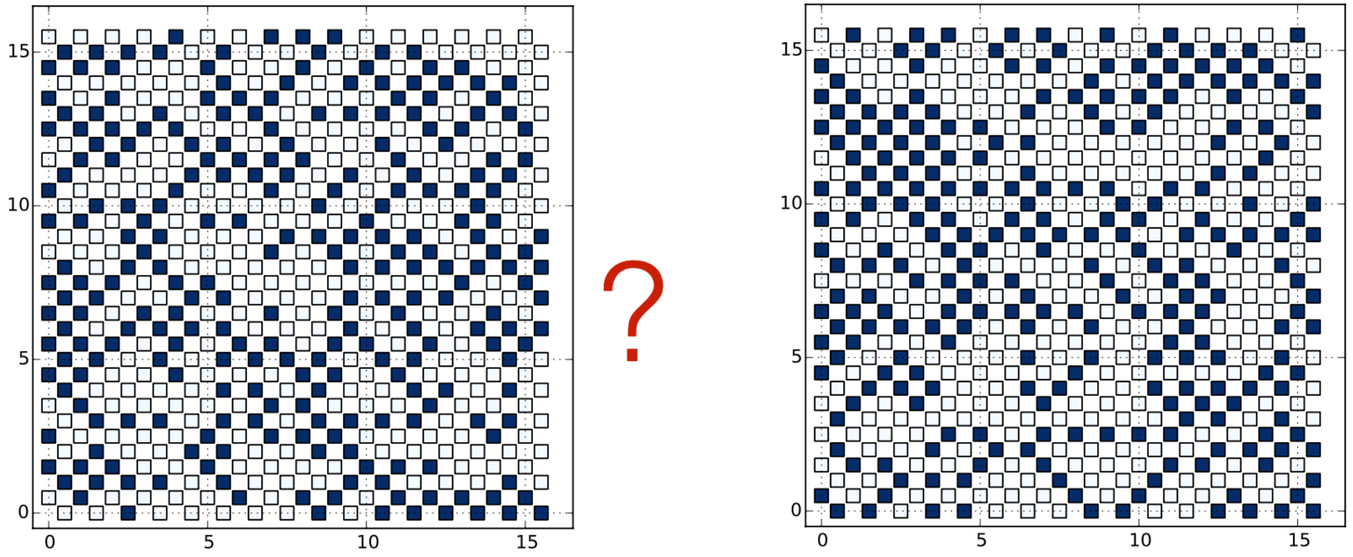
Ground state is a classical disordered topologically ordered phase

Castelnovo and Chamon Phys. Rev. B 76, 174416 (2007)

T=0 high temperature phase *T=infinity*

No conventional order parameter

For two configurations



high-temperature state

Ground state

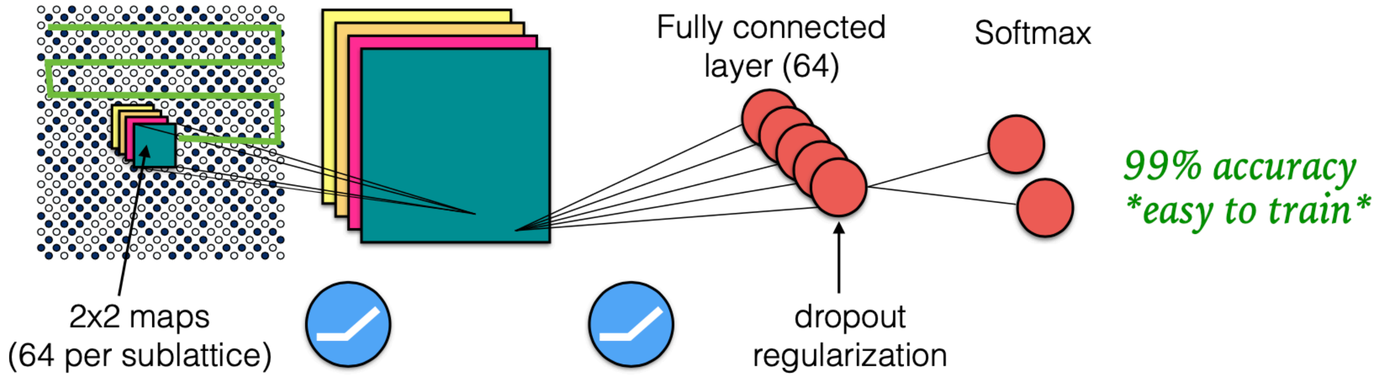
Feedforward NN are difficult to apply to this problem and lead to 50% accuracy

Physics applications: condensed matter physics

ISING GAUGE THEORY

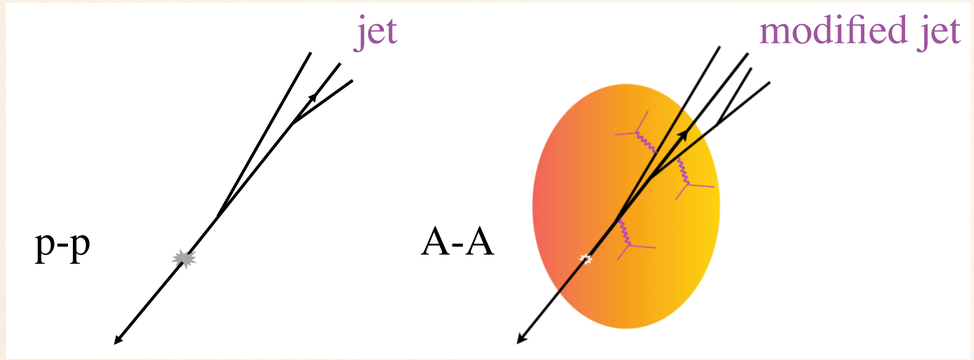
F.J. Wegner, J. Math. Phys. 12 (1971) 2259

$$H = -J \sum_p \prod_{i \in p} \sigma_i^z$$



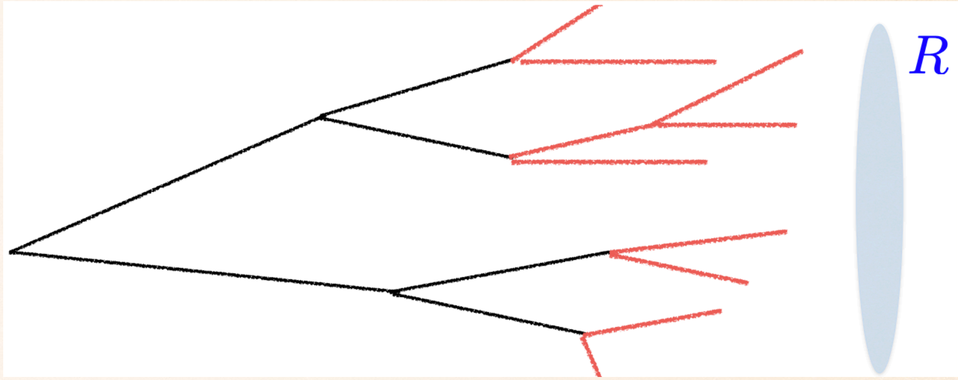
The picture we draw for what the CNN is using to distinguish the phases is that of the detection of satisfied local constraints. In few words, the neural network figures out the energy and uses it to classify states

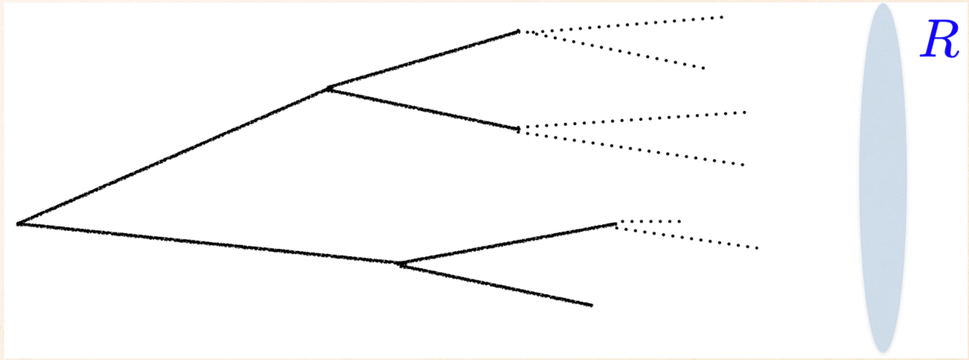
QGP and Jet Modifications



By J. Brewer

- Jets are powerful probes to QGP
  - QGP: AA collisions at $\sqrt{s}$ with different centralities
  - Jets (dijets, γ/Z-jets): initiator's flavor, energy, position, direction and jet substructures developed at early stages
  - Jet-medium interactions
- Towards more precise probes & jet tomography with ML



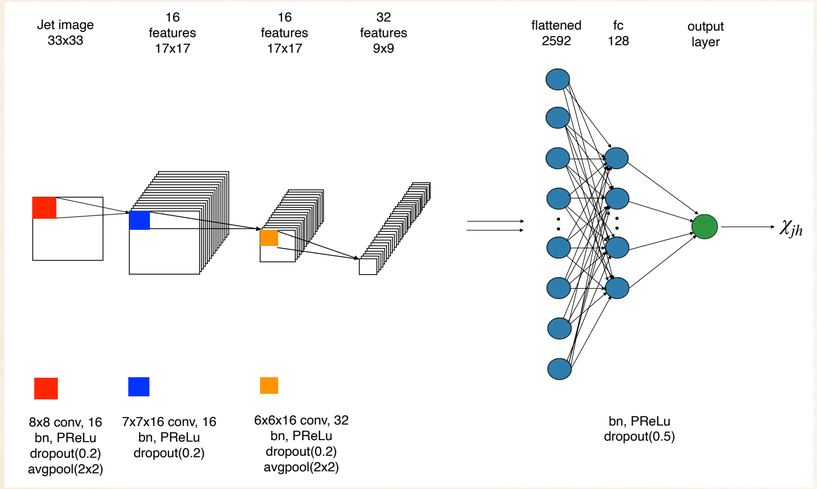


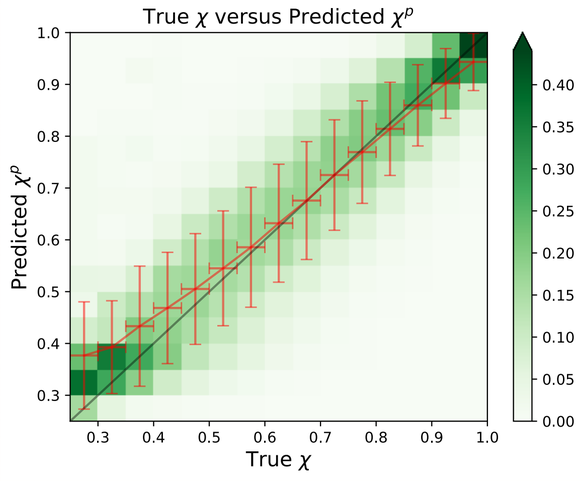
By D. Pablos

Hypethetical twin jets in vacuum/medium

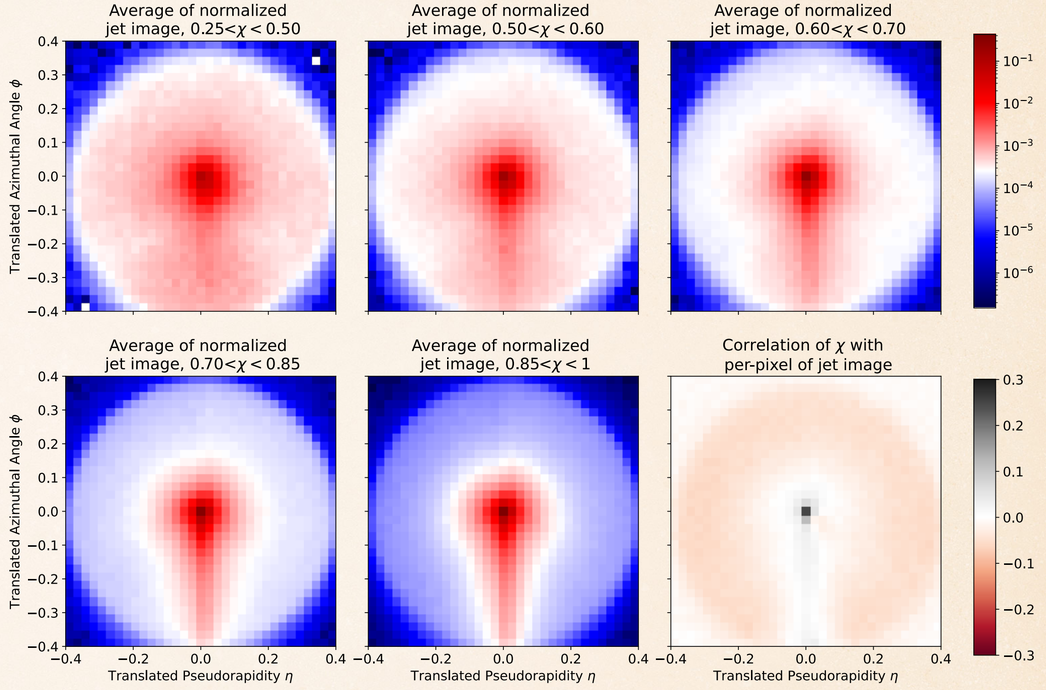
Prediction of Jet Energy Loss

YLD, D. Pablos and K. Tywoniuk, JHEP03(2021)206





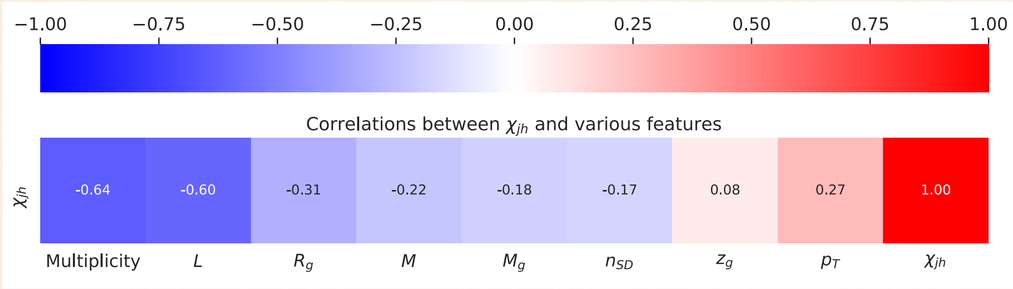
Hybrid model



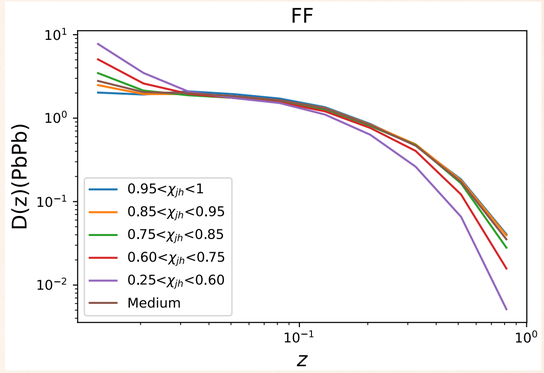
- Prediction of jet energy loss between the twin jets
- Jet quenching increases the number of soft particles at large angles
- Well predicted for a wide range of χ

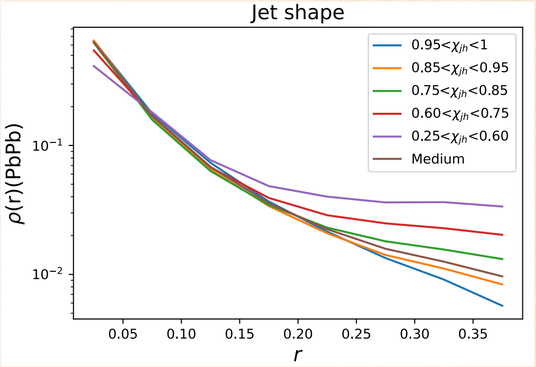
Prediction with jet observables & Interpretability

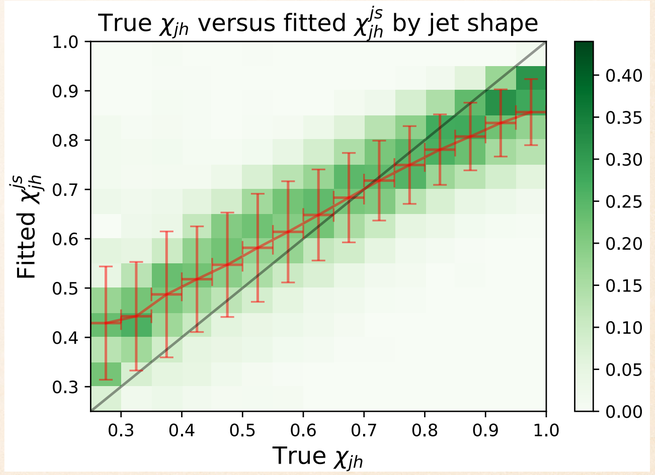
YLD, D. Pablos and K. Tywoniuk, JHEP03(2021)206



Input (size)	Output	Network	Loss
FF (10)	χ_{jh}	FCNN	0.0058
Jet shape (8)	χ_{jh}	FCNN	0.0033
FF, jet shape (18)	χ_{jh}	FCNN	0.0032
FF, jet shape, features (25)	χ_{jh}	FCNN	0.0028
Jet image & FF, jet shape, features (25)	χ_{jh}	API: CNN&FCNN	0.0028







- Jet shape outperforms jet FF.
- Motivates construction from jet shape by 17-parameter fitting: Still a bit worse than CNN
- Jet observables recover the performance by jet image with equivalent predictive power: Interpretability!

RNN,循环神经网络

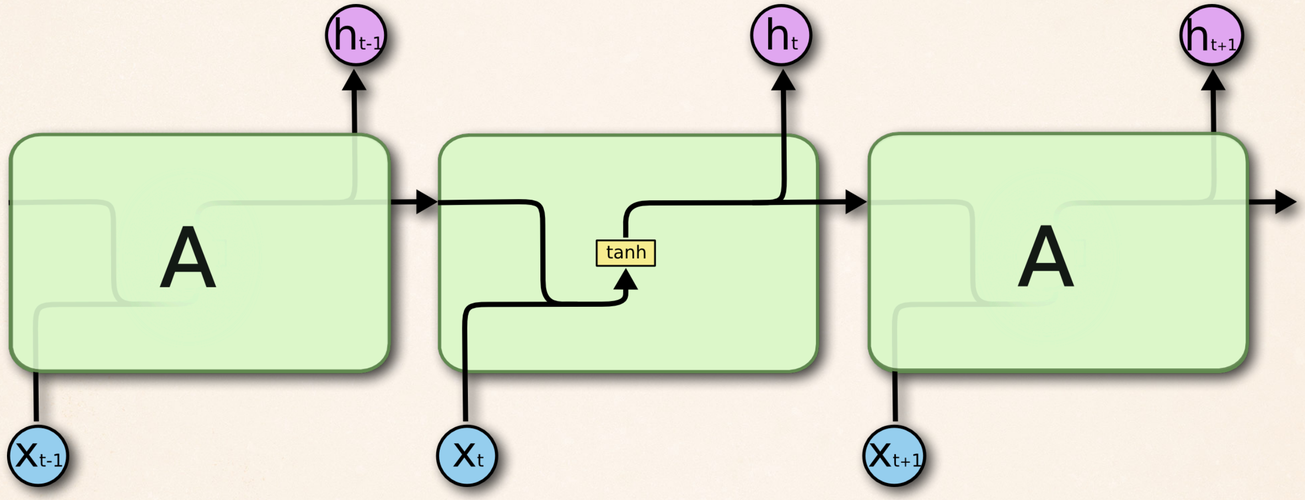


Figure from [medium]

LSTM,长短期记忆网络

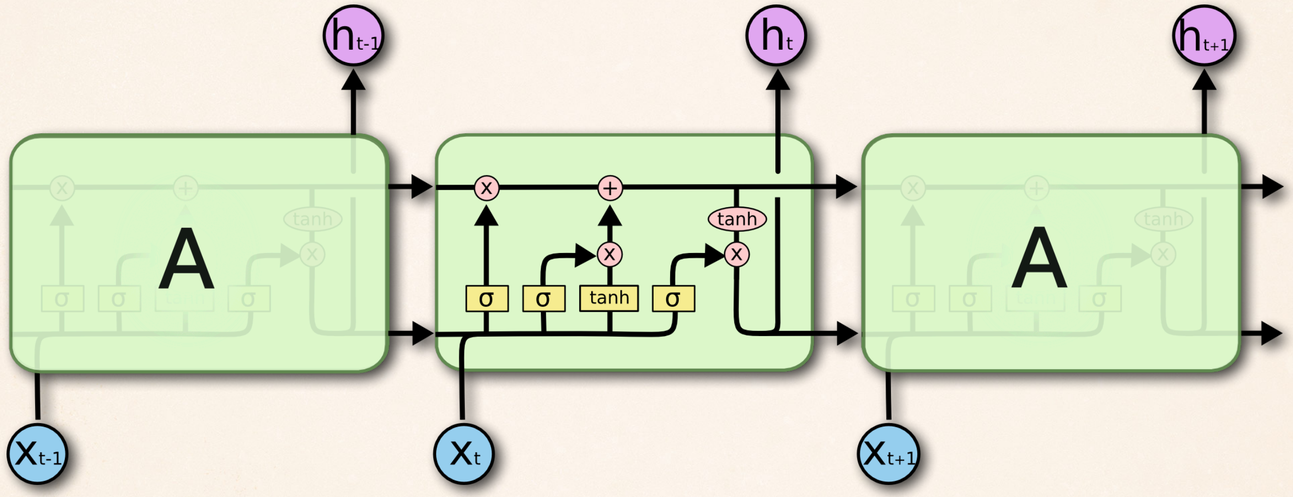
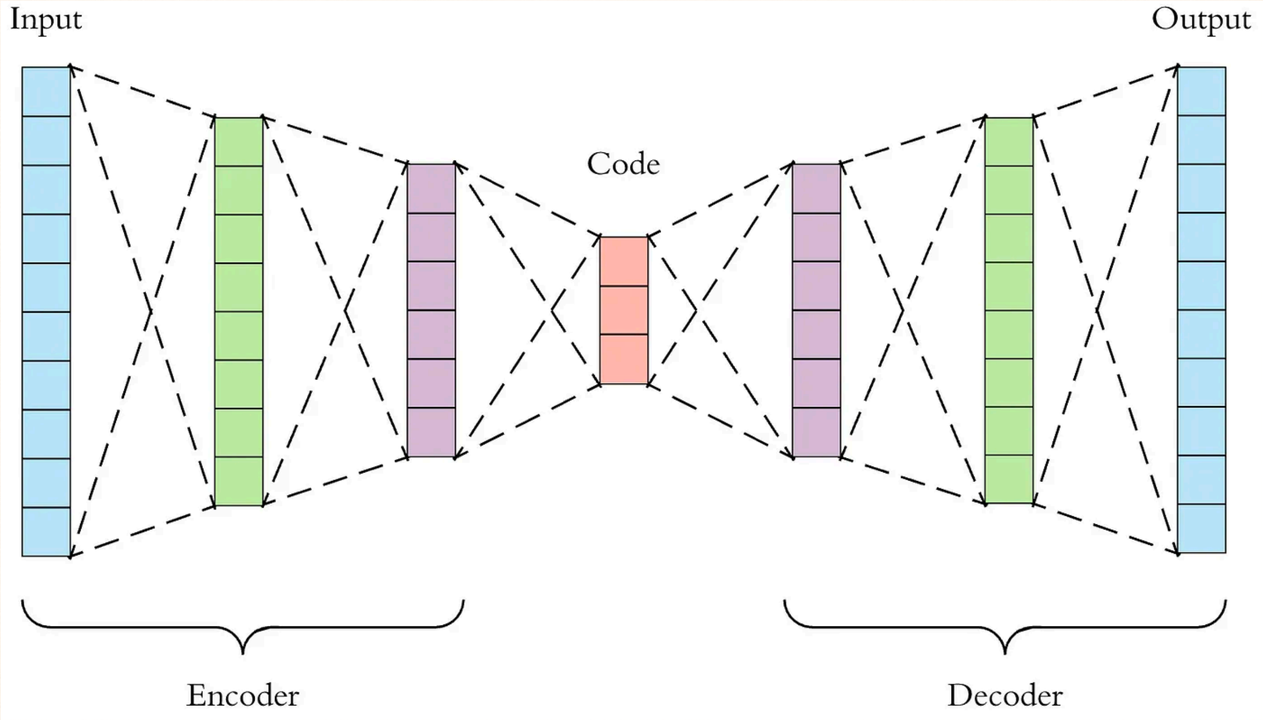


Figure from [medium]

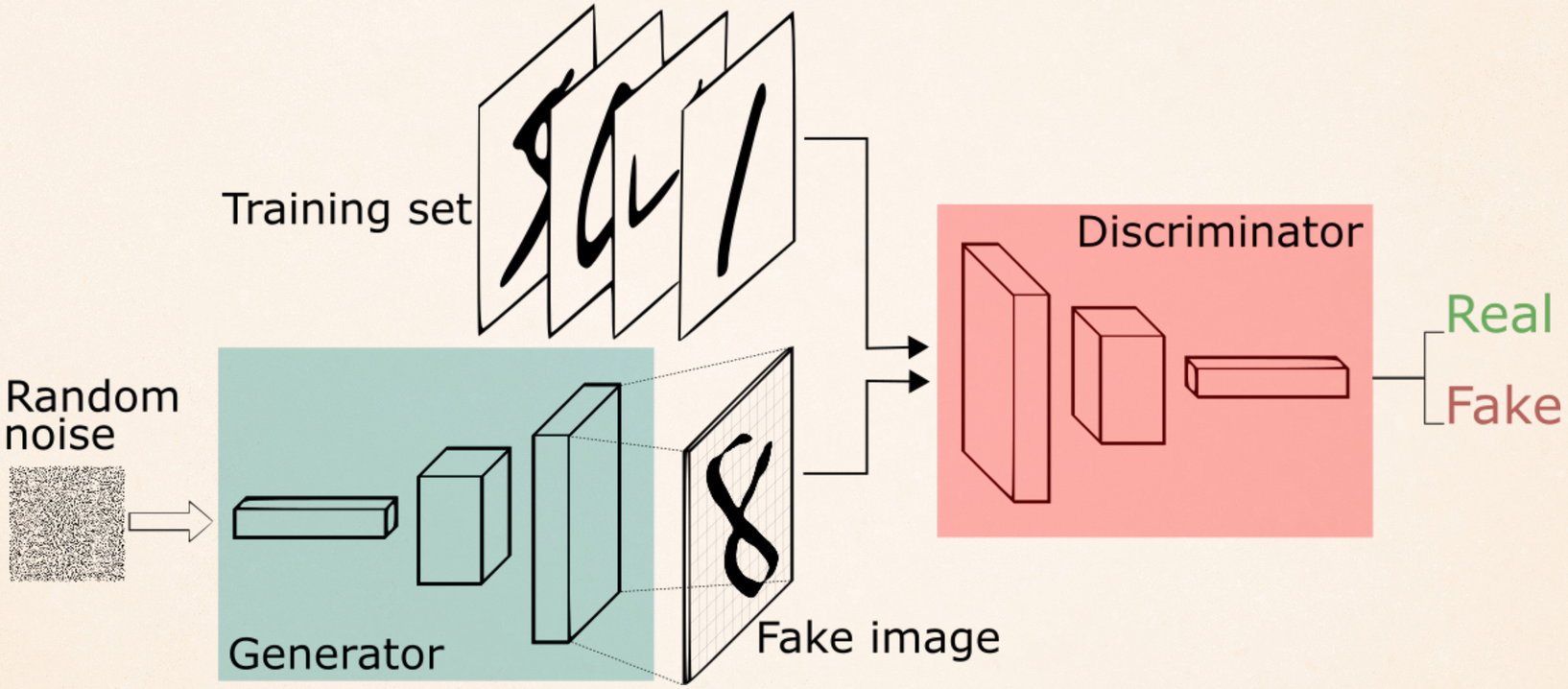
Auto Encoder（自编码器）



- Extract features by minimizing the reconstruction error using bottleneck network

Figure from medium

Generative adversarial network (GAN，生成对抗网络)



- Generator + Discriminator

Figure from github