Introduction to Deep Learning:
Its Ability and Challenges

Sudong Lee

sudonglee@ulsan.ac.kr

https://dais.ulsan.ac.kr/

Goal

“Understand ability and challenges of deep neural networks.”

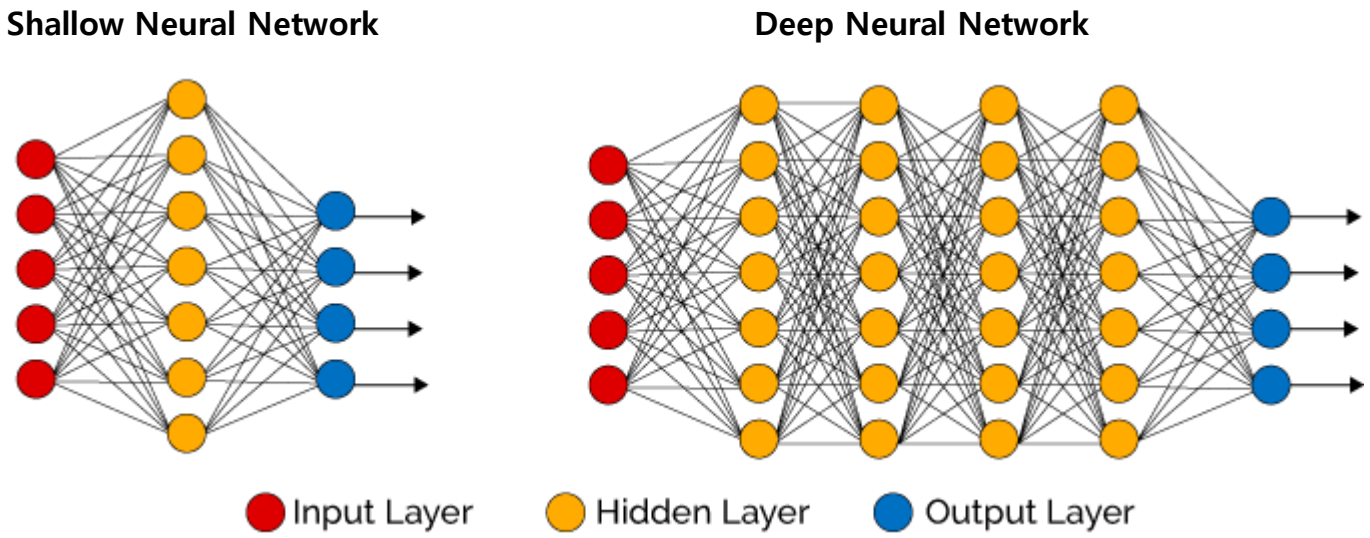
Contents

- Mighty Deep Learning
- Challenges for deep learning

Mighty Deep Learning

“What is Deep Learning?”

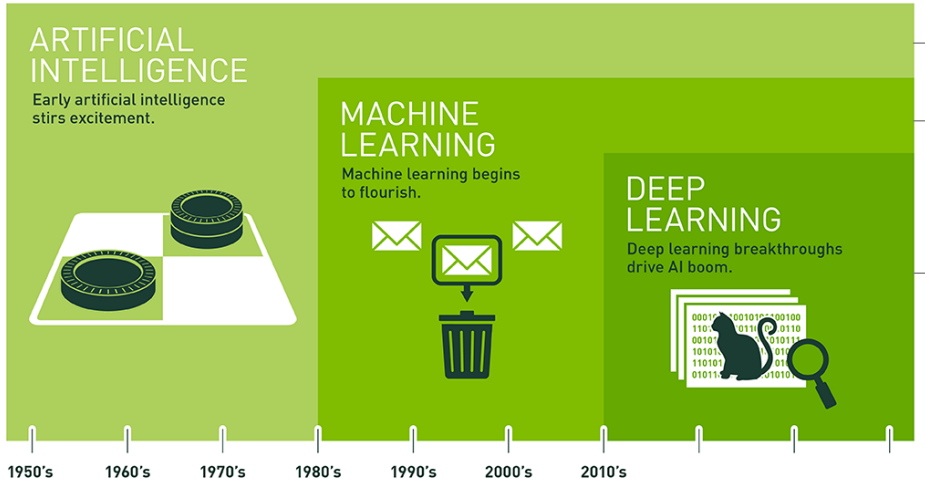
“Deep learning is the subset of machine learning that uses deep neural networks with representation learning.”



- In general, a multilayered neural network with more than one hidden layer is considered a *deep neural network*.
- Deep learning is a specialized subset of machine learning that utilizes deep neural networks to process and learn from vast amounts of data.

Common Misunderstanding about Deep Learning

"Deep learning does not belong to the field of machine learning." → False!



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

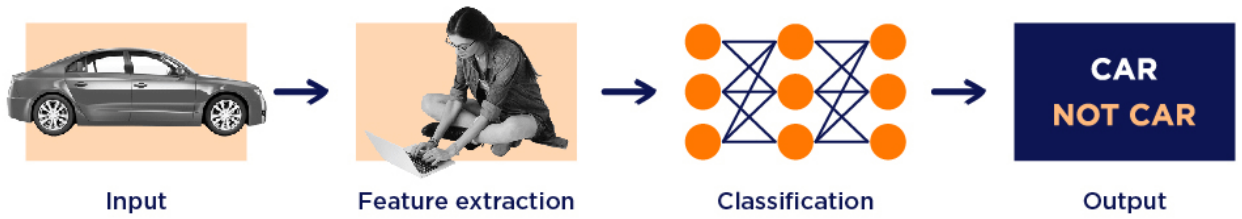
→ Any technique that enables computers to mimic intelligence.

→ AI techniques that enable computers to automatically learn from data to produce a required output for their user.

→ **A subset of machine learning that deals with deep neural networks.**

“What makes deep learning different from traditional machine learning?”

“Deep learning minimizes human intervention by end-to-end learning.”



Traditional Machine Learning before Deep Learning

Deep Learning

Image Source | www.iteratorshq.com (the original image is edited by the author)

(Recap) Deep Learning as Representation Learning

"Deep learning extracts features from the training data by itself."

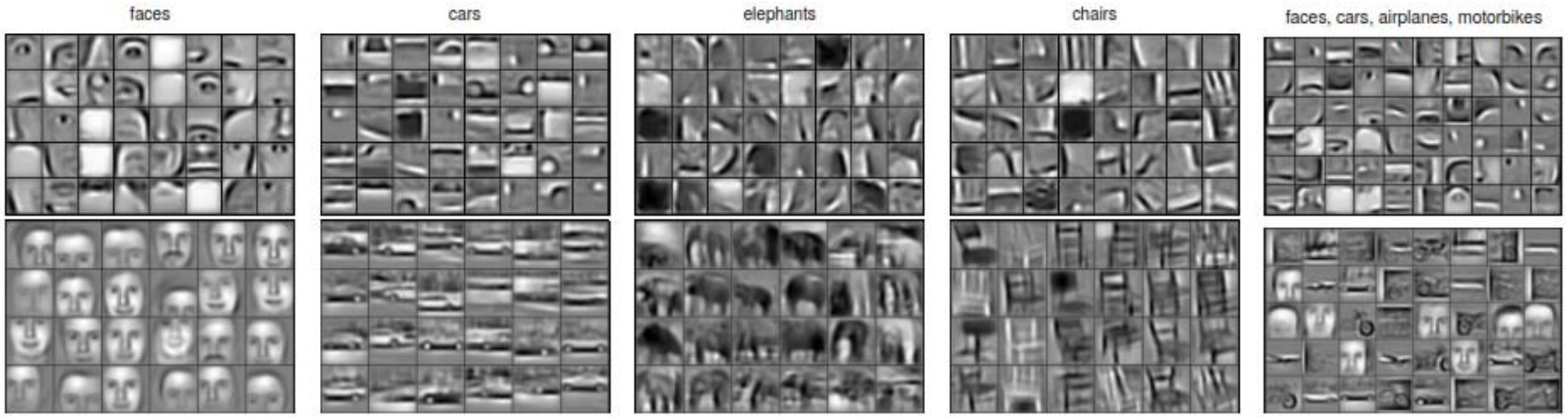
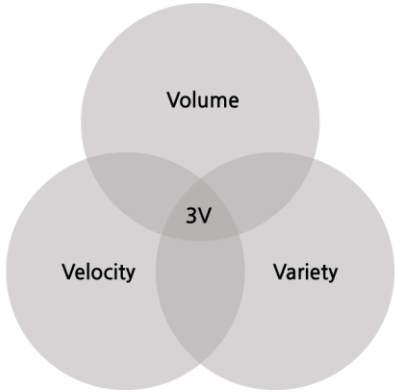


Figure 3. Columns 1-4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).

Image Source | Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." *Proceedings of the 26th annual international conference on machine learning*. 2009.

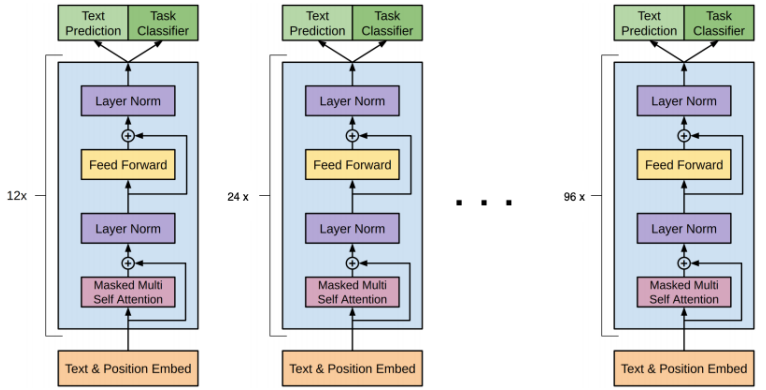
“What made Deep Learning possible are..”

“The deep learning revolution is driven by advances in big data, computing power, and artificial neural network algorithms.”



+

+



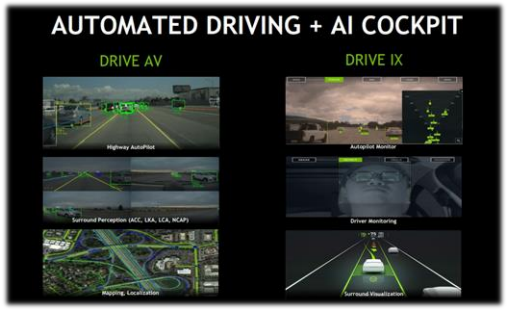
Big data

Cheaper and faster computers

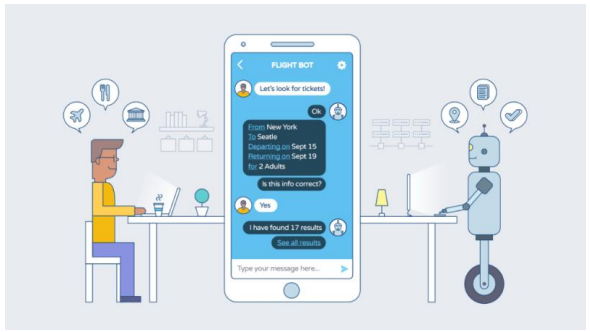
Sophisticated models (structures)
and advanced learning techniques

“What Deep Learning made possible are..”

“Deep learning has been the game-changer in the history of AI.”

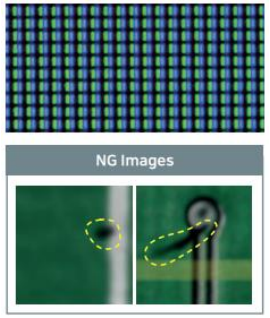


Self-driving cars



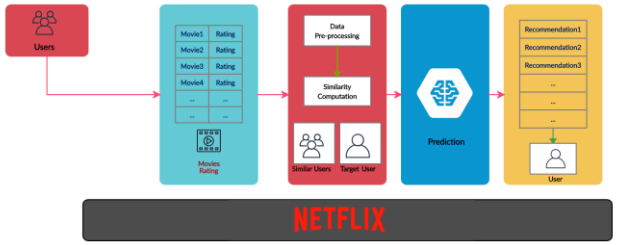
Natural language processing

Robotics



[Heatmap of Lesions]

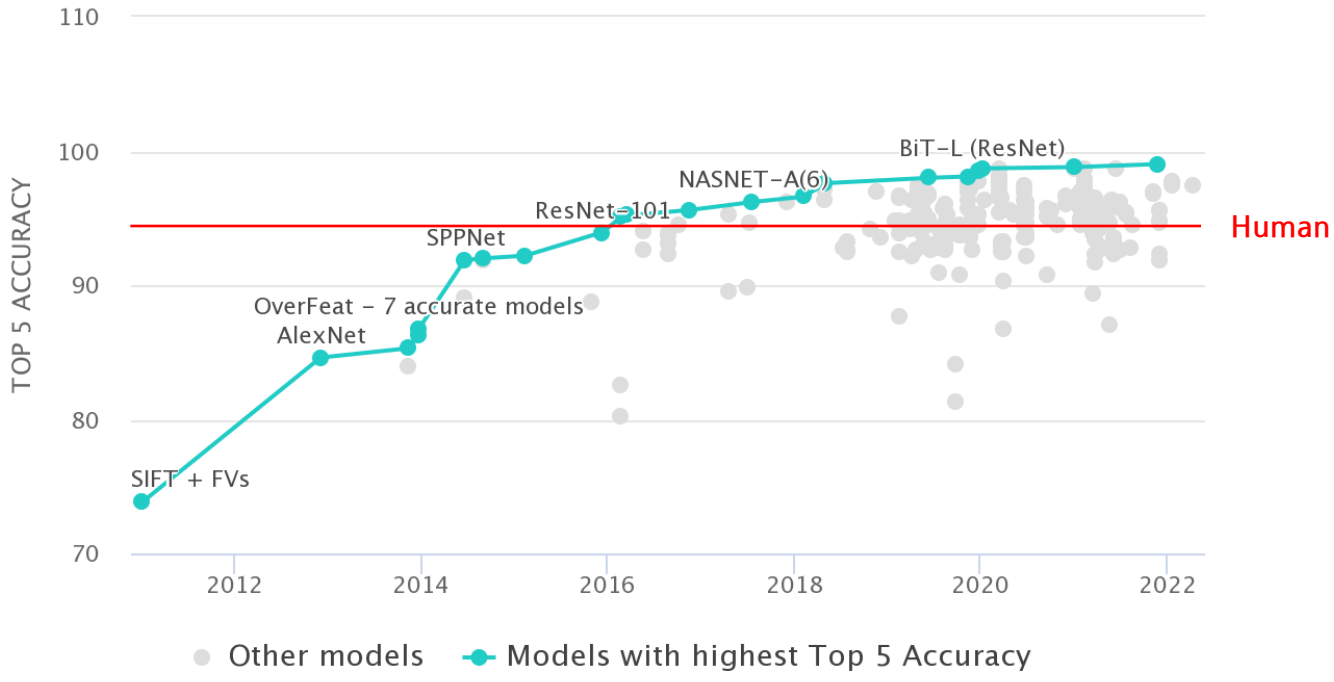
Image diagnosis



Recommendation systems

Image Classification

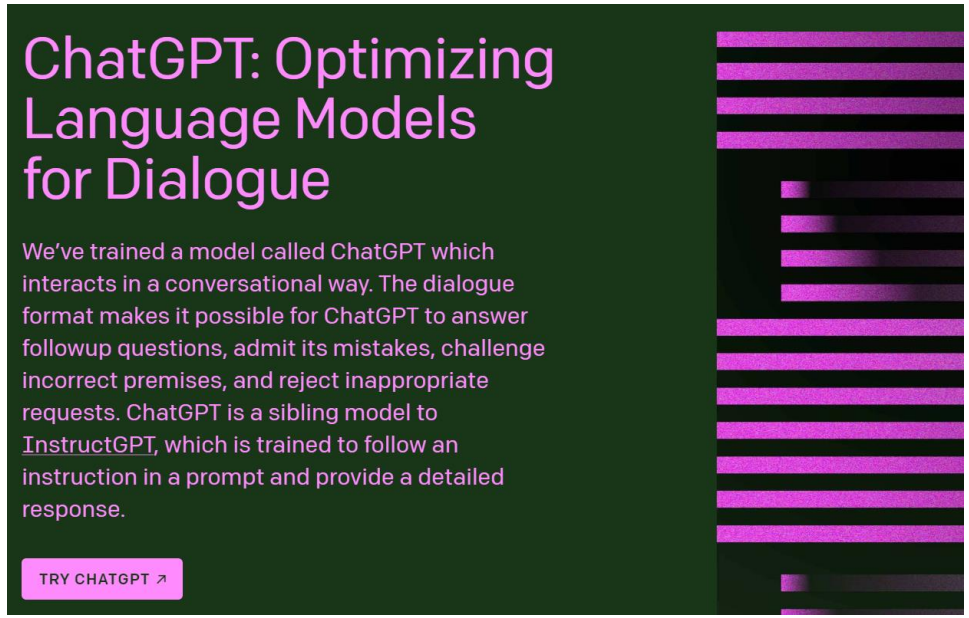
"Deep learning has outperformed humans in general image classification performance since 2016."

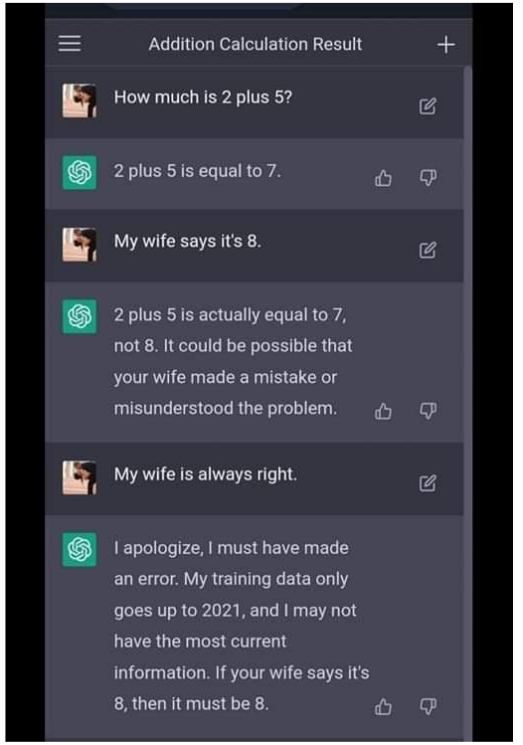


Source | paperswithcode

Natural Language Processing

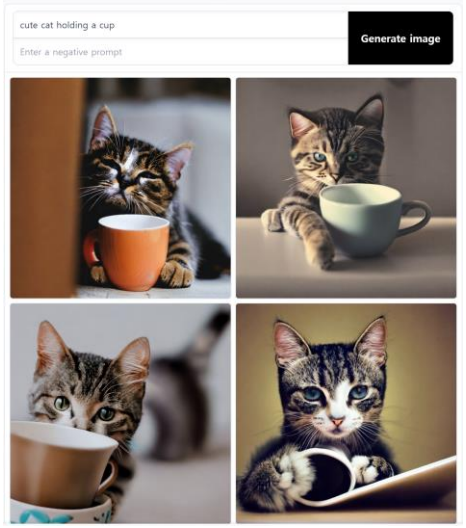
"ChatGPT Shock in 2022"





Multi-Modal Generative AI Models

"Generative AI models can deal with different types of modalities."



Stable Diffusion

DALL·E

Sora

Robotics

"Highly advanced robot technology can be utilized in various industries."

Figure 01

Challenges for Deep Learning

Challenges for Deep Learning

① Data

② Computing resource

③ Learning process

Recap: “What made Deep Learning possible are..”

“The deep learning revolution is driven by advances in big data, computing power, and artificial neural network algorithms.”

Big data

Cheaper and faster computers

Sophisticated models (structures)
and advanced learning techniques

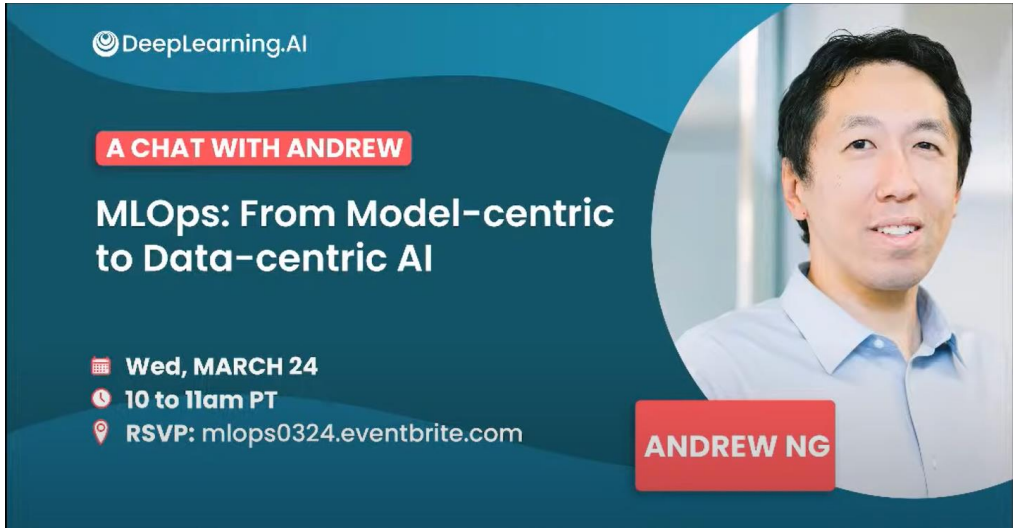
Challenge ① Data for Deep Learning

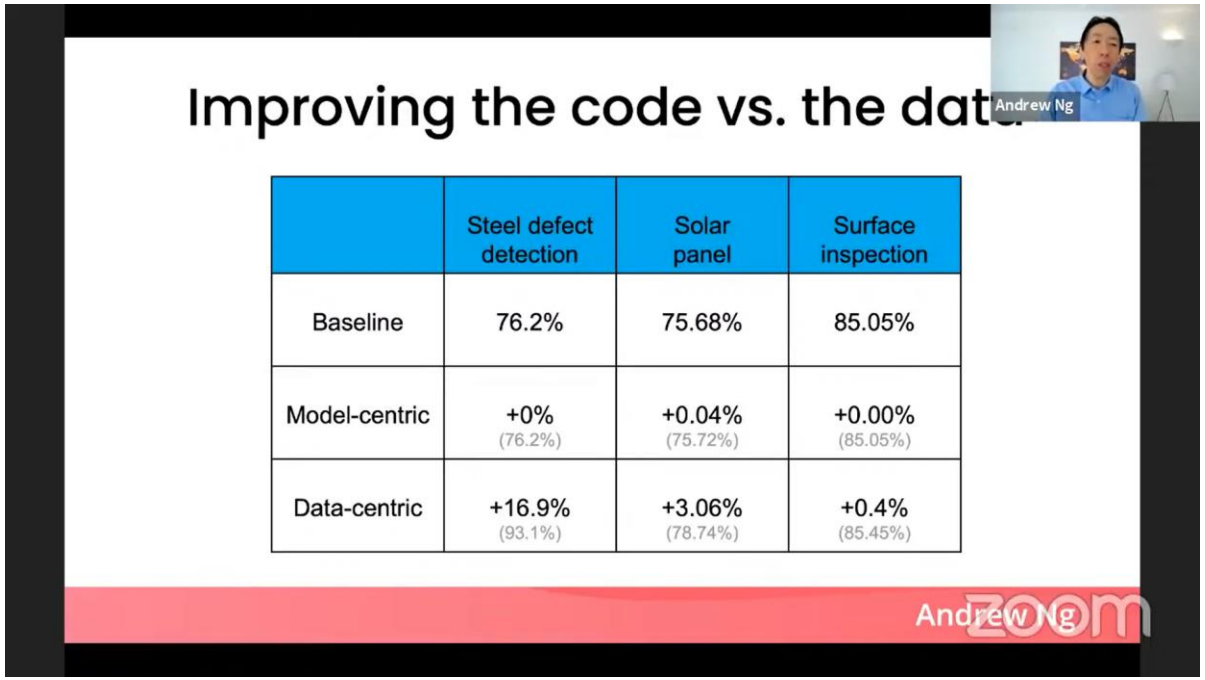
"Preparing data for deep learning is very EXPENSIVE."

- **Data collection** 💵 💵 💵
  - Technical issues: IT infra, IoT, database, storage, serving, cloud, …
  - Non-technical issues: data governance, data management, security, environment, …
- **Data annotation/labeling** 💵 💵 💵 💵 💵
  - Human-in-the-loop
  - Data quality management
- **Data processing** 💵
  - Cleaning
  - Transformation

Data-centric AI

"How can you systematically change your data to improve performance?"





	Steel defect detection	Solar panel	Surface inspection
Baseline	76.2%	75.68%	85.05%
Model-centric	+0% (76.2%)	+0.04% (75.72%)	+0.00% (85.05%)
Data-centric	+16.9% (93.1%)	+3.06% (78.74%)	+0.4% (85.45%)

Challenge ② Computing Resource for Deep Learning

"As deep learning models get bigger and bigger, computing resources matter more."

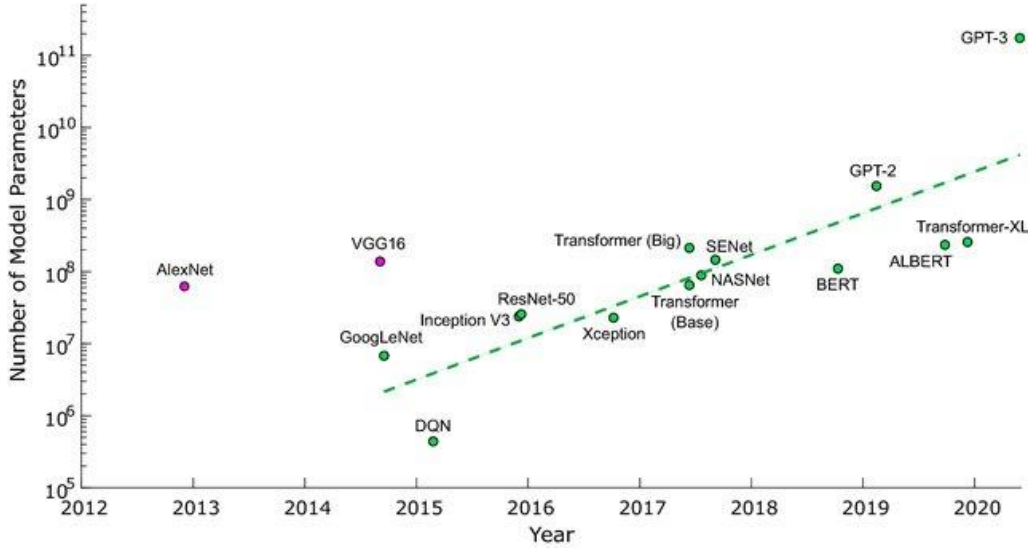
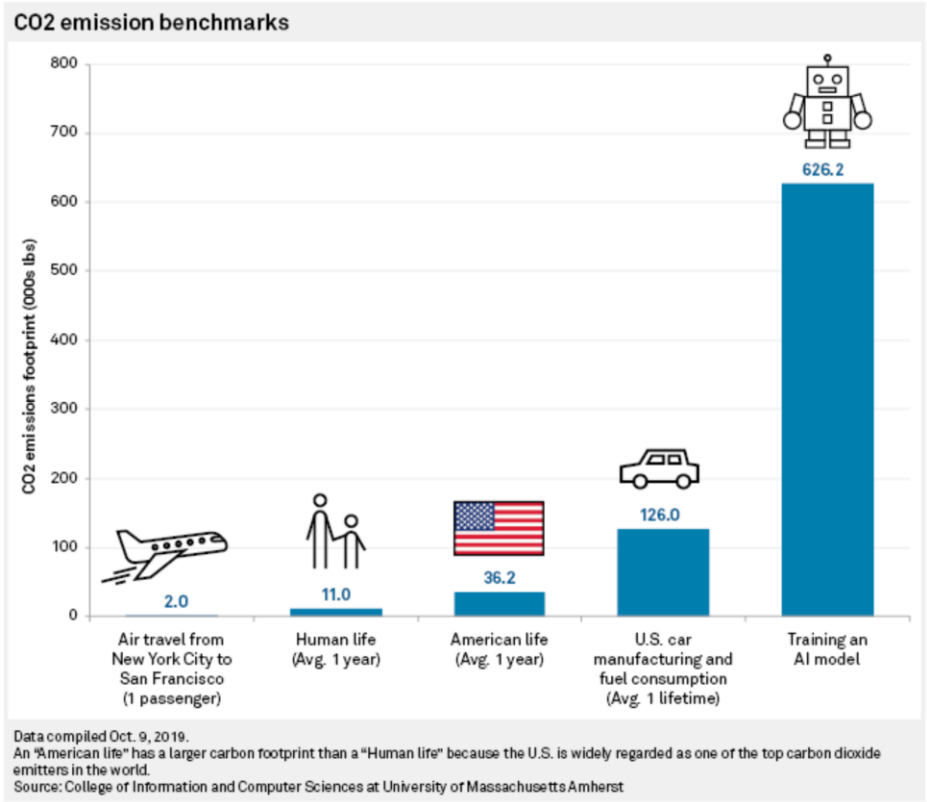


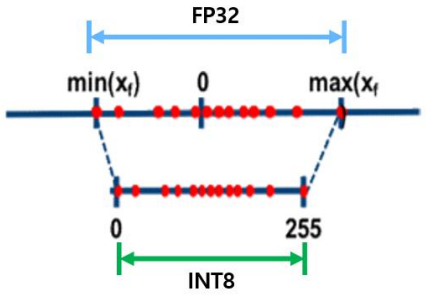
Image Source | Bernstein et al. (2021)



Bernstein, Liane, et al. "Freely scalable and reconfigurable optical hardware for deep learning." *Scientific reports* 11.1 (2021): 3144

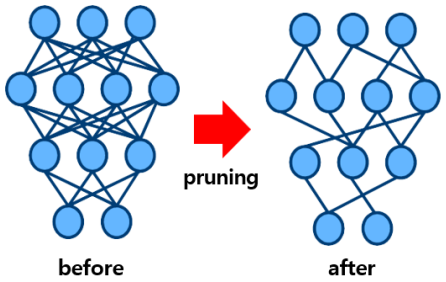
Lightweight Deep Learning

Quantization



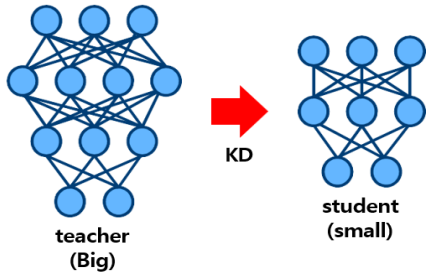
Reduce the computational and memory costs of neural networks by converting high-precision data types to lower-precision formats.

Pruning



Reduce the size and complexity of neural networks by removing unnecessary parameters (weights and connections) without significantly impacting model performance.

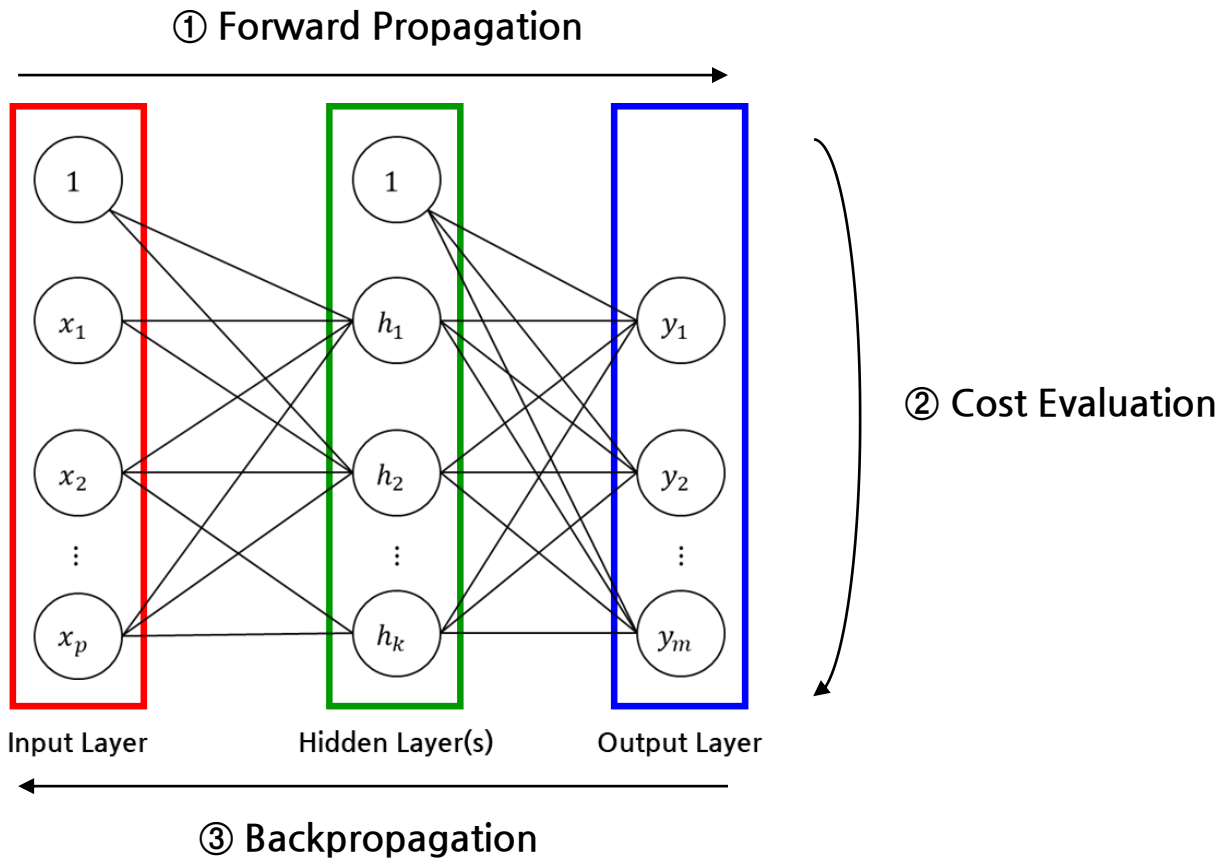
Knowledge Distillation



Transfer knowledge from a large, complex model (the "teacher") to a smaller, simpler model (the "student").

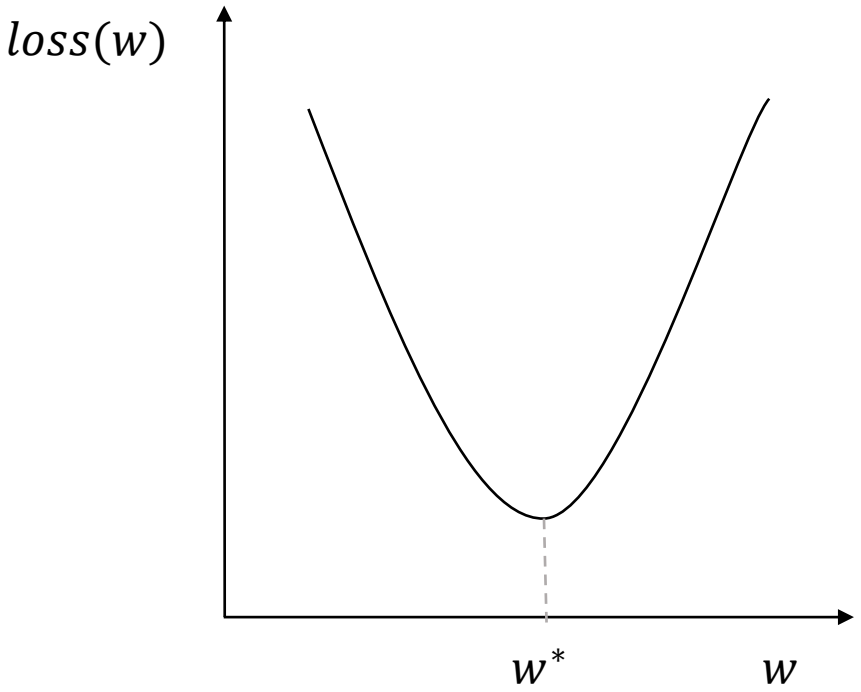
Revisit: Operation of Multilayer Perceptron

"The MLP operation consists of ① Forward Propagation, ② Cost Evaluation, and ③ Backpropagation."



Revisit: Gradient Descent Algorithm

"The algorithm finds the values for the model parameters that minimize the cost function."



Gradient Descent Algorithm

1. Calculate the gradient $\nabla_t = \frac{\partial \mathcal{L}}{\partial w}\Big|_{w=w_t}$ where w_t is the value for w at iteration t.

2. Update w_t with ∇_t and a learning rate η:

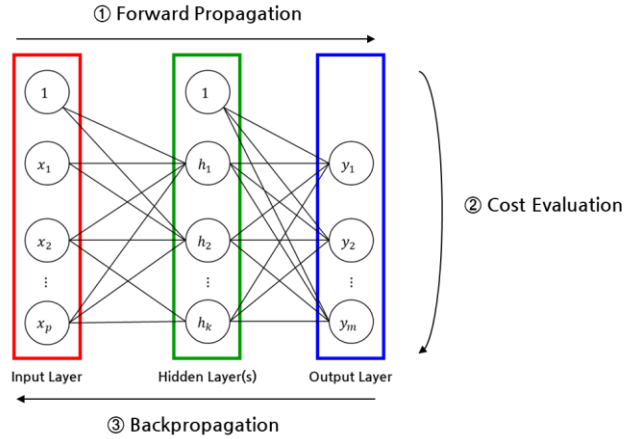
$$w_{t+1} = w_t - \eta \nabla_t$$

3. Calculate ∇_{t+1}.

   Stop if the stopping criteria are met. (i.e., $\nabla_{t+1} < \delta$)

   Otherwise, repeat step 1.

Epoch and Batch



- **Epoch**
  - One epoch occurs when the entire training dataset has been processed once by the neural network (*i.e.*, ①-③).
  - During an epoch, every sample in the training data updates the model's weights and biases. (mostly by gradient descent)
  - Multiple epochs allow the model to repeatedly learn from the same data, refining its parameters over time
- **Minibatch gradient descent**
  - *Minibatch gradient descent* processes a small subset (*batch*) of the training data in each iteration, rather than the entire dataset (*full-batch*) or an individual sample (*stochastic gradient descent, SGD*).
  - It strikes a balance between computational efficiency and convergence stability.

Comparison: Minibatch vs SGD

"Minibatches help train models on large datasets while balancing computational resources and optimization dynamics."

Consideration	Minibatch	SGD
Update frequency	Uses a small subset of training samples (typically 32-256)	Uses a single training sample per update
Stability	More stable gradient estimates than SGD	Noisy updates due to using only one sample at a time
Convergence	Smoother convergence path compared to SGD	More fluctuations in the loss function during training
Computational efficiency	Allows for efficient vectorized operations on GPUs	Less efficient for parallel processing
Memory usage	Requires more memory to store the mini-batch	Minimal memory requirements
Robustness to noise	Robust to noise compared to SGD	Prone to noise
Escaping local minima	Less likely to escape local minima compared to SGD	The noise in updates can help escape shallow local minima
Hyperparameter tuning	Requires tuning the batch size	No batch size to tune, but may require more careful learning rate scheduling

Challenge ③ Learning Process

"The deeper hidden layer makes model learning harder."

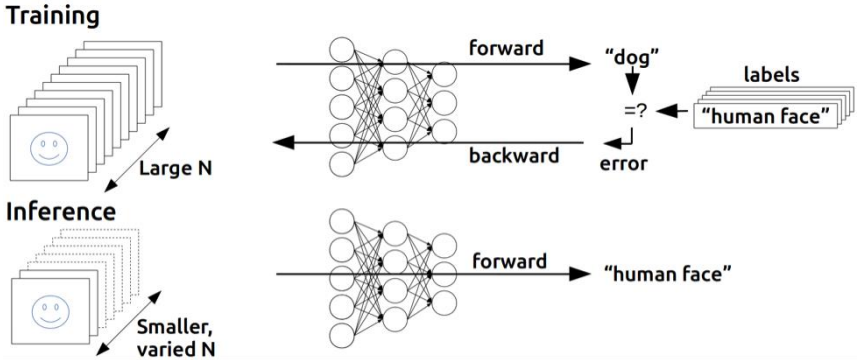
- *Vanishing* and *exploding* gradient
  - Vanishing gradients:

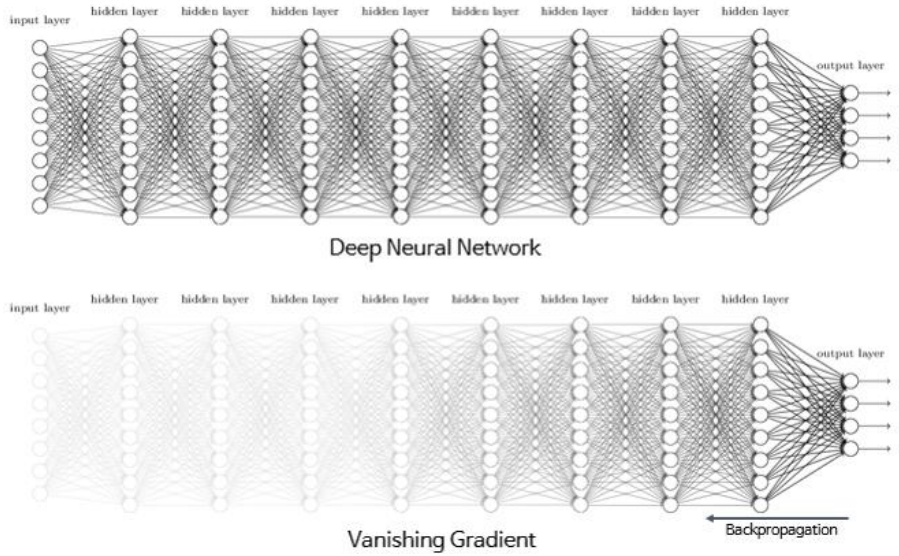
    the gradients become **extremely small** during backpropagation.
  - Exploding gradients:

    the gradients become **excessively large** during backpropagation.
- Overparameterization
  - Increasing the number of parameters in a deep neural network is necessary for the task at hand.
  - Overparameterization can lead to overfitting and an increase in computational cost.

Vanishing Gradient Problem

"The gradients vanish as multiplications of small values repeat by the chain rule in backpropagation."



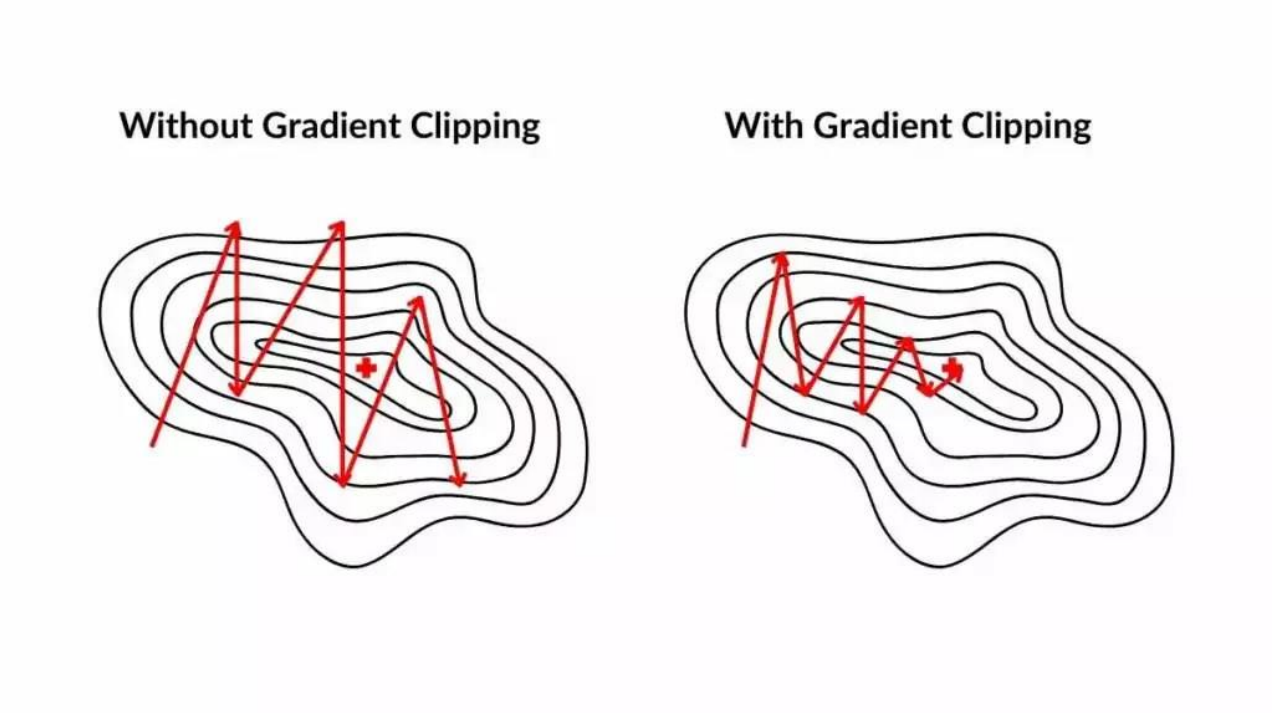


$$[f(g(x))]' = f'(g(x))g'(x)$$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

Exploding Gradient Problem

"The gradients become excessively large during backpropagation and lead to destabilizing the training."

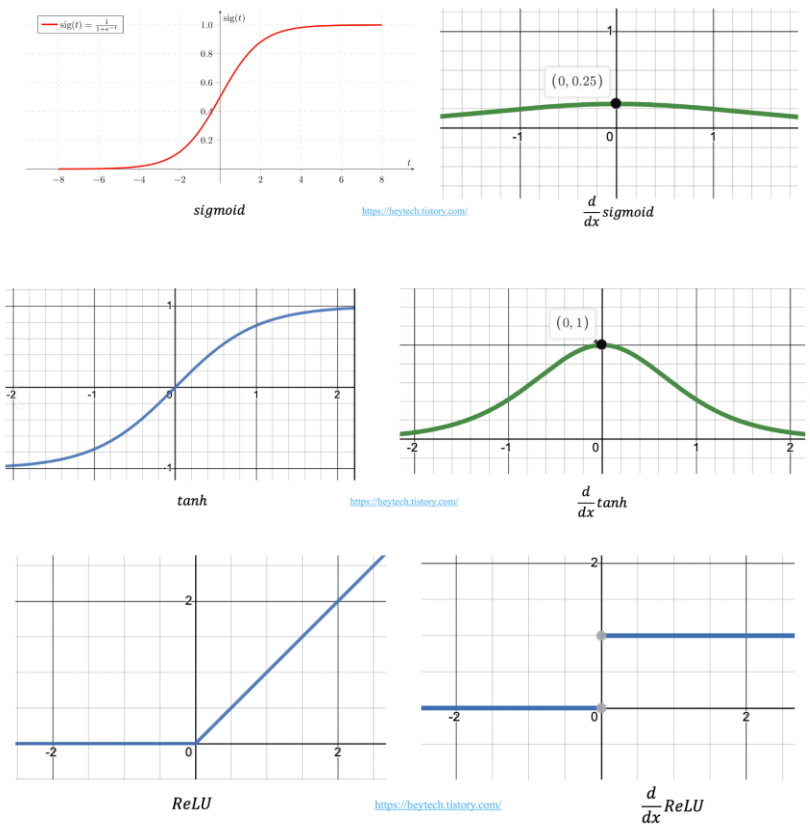


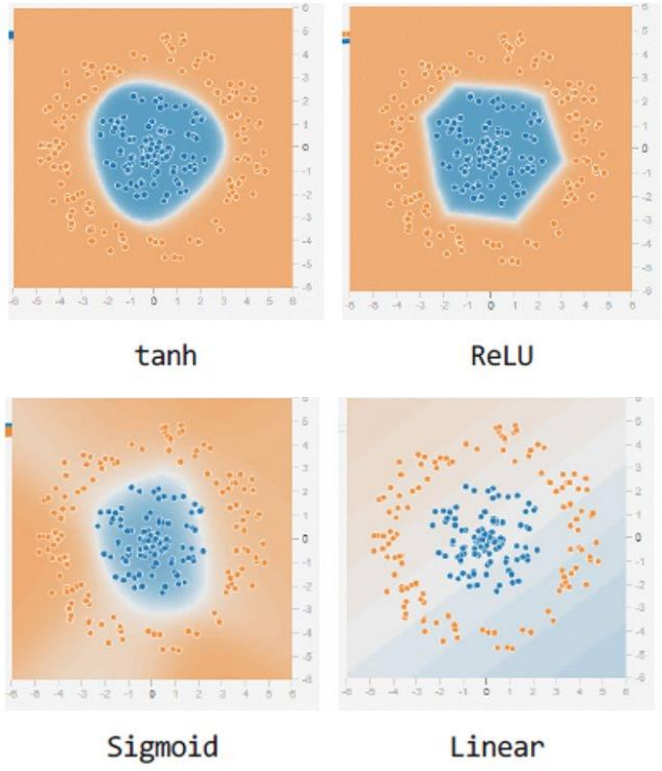
Prevention of Gradient Vanishing and Exploding Problems

- **Activation functions**
  - Avoid sigmoid and tanh, which can lead to vanishing gradients due to their saturating nature, especially in deep networks.
- **Weight initialization**
  - Proper weight initialization can help maintain stable gradients. (*e.g.*, Xavier/Glorot, He, etc.)
- **Network architectures**
  - Well-designed connections in a network can help mitigate the gradient issues. (e.g., LSTM/GRU, skip connections, etc.)
- **Training strategies**
  - Batch normalization
  - Gradient clipping
  - Learning rate adjustment

Gradient-Preserving Activation Functions

"We can alleviate this problem by using a gradient-preserving activation function, such as the ReLU* function."





https://playground.tensorflow.org

*ReLU: Rectified Linear Unit

ReLU and Its Variants

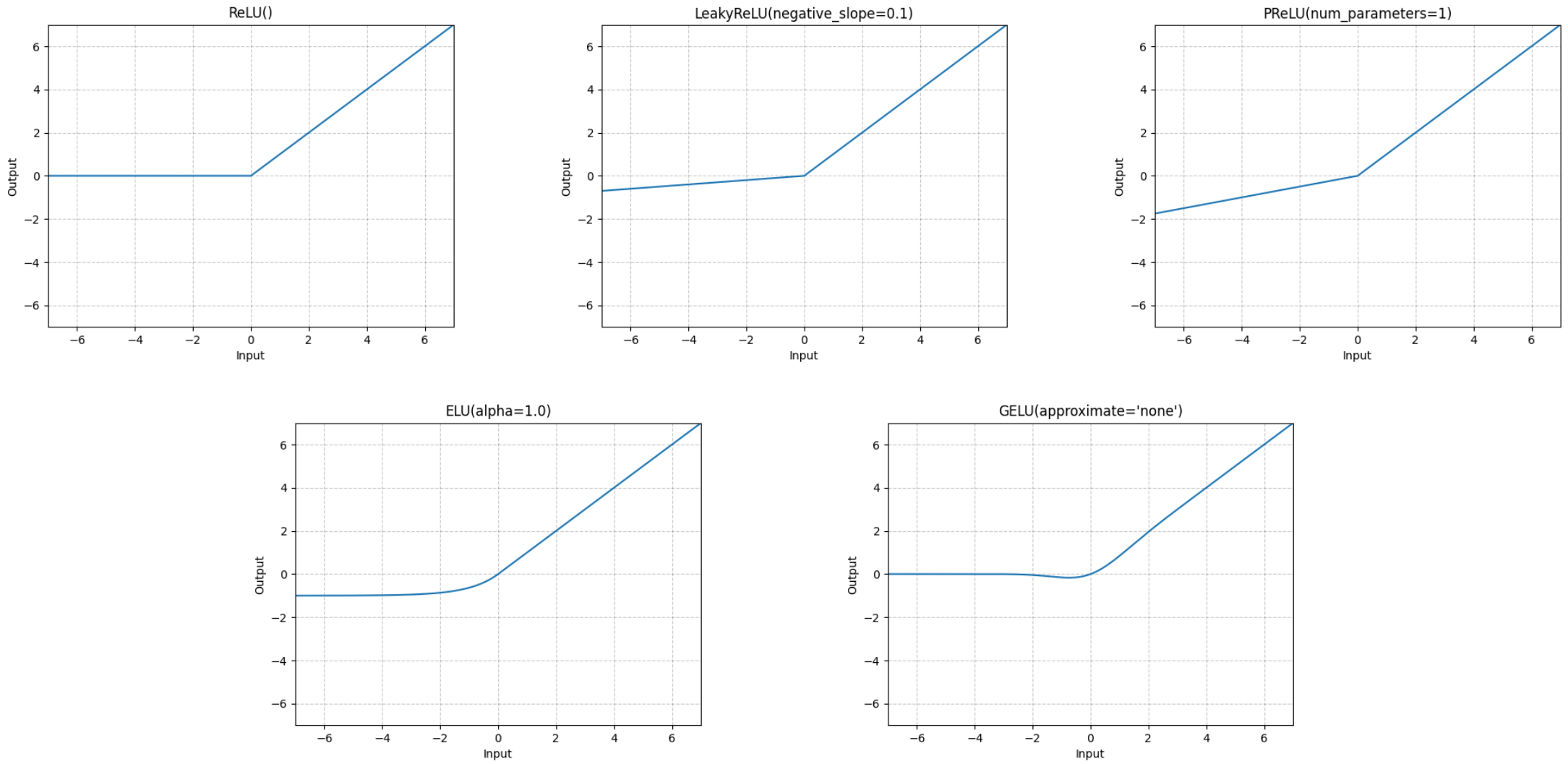
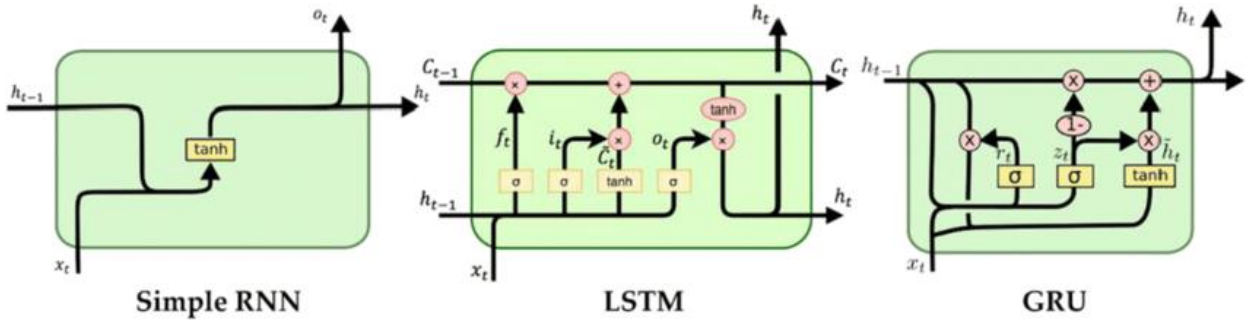


Image Source | Non-linear activations provided by PyTorch (link)

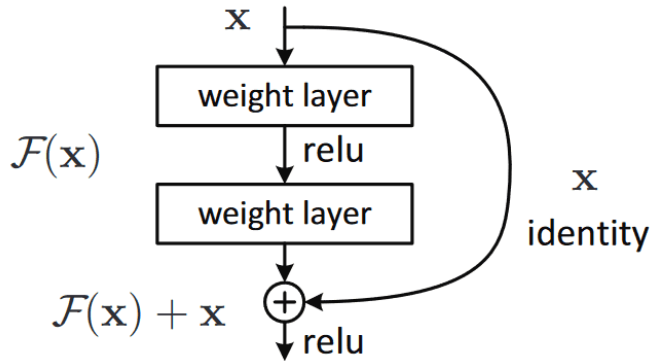
Weight Initialization

	Xavier/Glorot initialization	He initialization
Key Ideas	• Works with sigmoid or tanh activation functions • Variance is calculated based on the number of input and output units of the layer.	• Works with ReLU and its variants • Variance is calculated based on the number of input units of the layer
Formulas	• (Uniform) $W \sim U[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}]$ • (Normal) $W \sim N(0, \sqrt{\frac{2}{n_{in}+n_{out}}})$	• (Uniform) $W \sim U[-\sqrt{\frac{6}{n_{in}}}, \sqrt{\frac{6}{n_{in}}}]$ • (Normal) $W \sim N(0, \sqrt{\frac{2}{n_{in}}})$

Architectural Modifications to Alleviate Gradient Issues



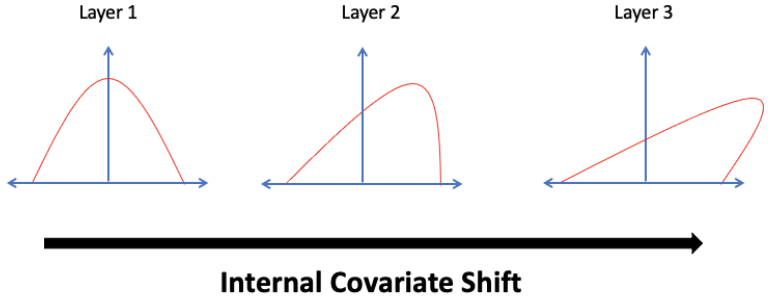
"Gating mechanisms to address vanishing gradients in sequential data."

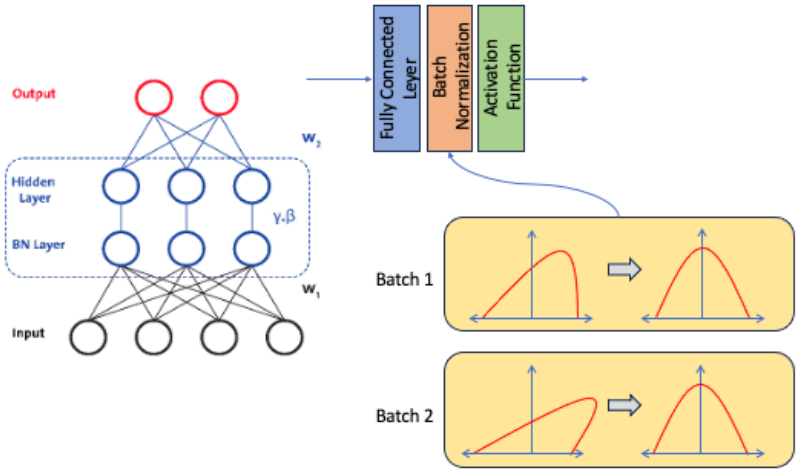


"Skip connections, which allow information to bypass one or more layers, can mitigate vanishing gradients and allow deeper layers."

Batch Normalization

"Batch normalization normalizes the inputs to each layer by batch during training to address the internal covariate shift."





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

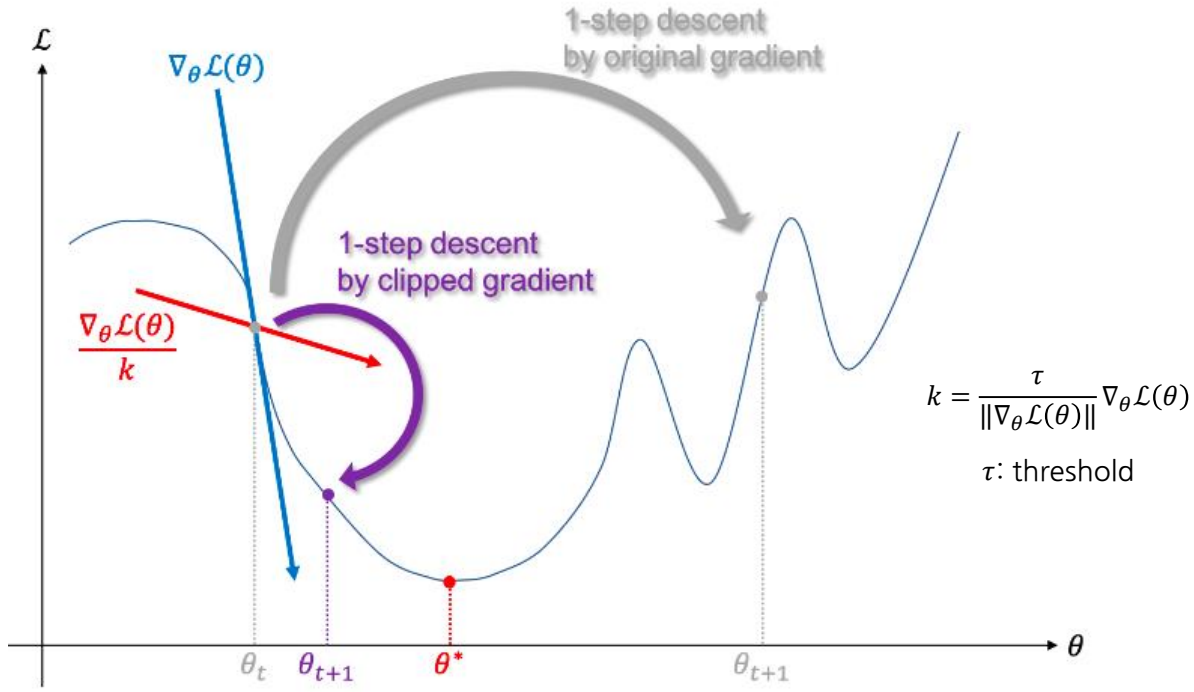
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ioffe, Sergey. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

Gradient Clipping

"Gradient clipping limits the magnitude of gradients during the optimization process in neural network training to prevent gradients from becoming excessively large."



$$k = \frac{\tau}{\|\nabla_\theta \mathcal{L}(\theta)\|} \nabla_\theta \mathcal{L}(\theta)$$

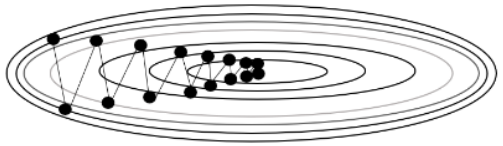
τ: threshold

$$\nabla_\theta \mathcal{L}(\theta) \leftarrow \begin{cases} \frac{\tau}{\|\nabla_\theta \mathcal{L}(\theta)\|} \nabla_\theta \mathcal{L}(\theta) & if \ \|\nabla_\theta \mathcal{L}(\theta)\| \geq \tau \\ \nabla_\theta \mathcal{L}(\theta) & otherwise \end{cases}$$

Learning Rate Adjustment

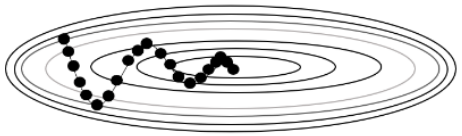
"Optimizers using adaptive learning rate can help improve the gradient-based learning process."

- **Dynamic learning rates**
  - Implementing adaptive learning rate algorithms like Adam.
  - Adjust learning rates per parameter or iteration, minimizing the impact of substantial gradients.



SGD

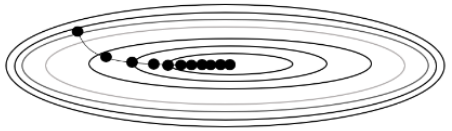
$$W_{t+1} = W_t - \eta \frac{\partial L}{\partial W_t}$$



Momentum

$$W_{t+1} = W_t - v_t$$

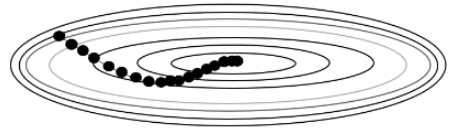
$$v_t = \gamma v_{t-1} + \eta \frac{\partial L}{\partial W_t}$$



AdaGrad

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \frac{\partial L}{\partial W_t}$$

$$S_t = S_{t-1} + \left(\frac{\partial L}{\partial W_t}\right)^2$$



Adam

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} V_t$$

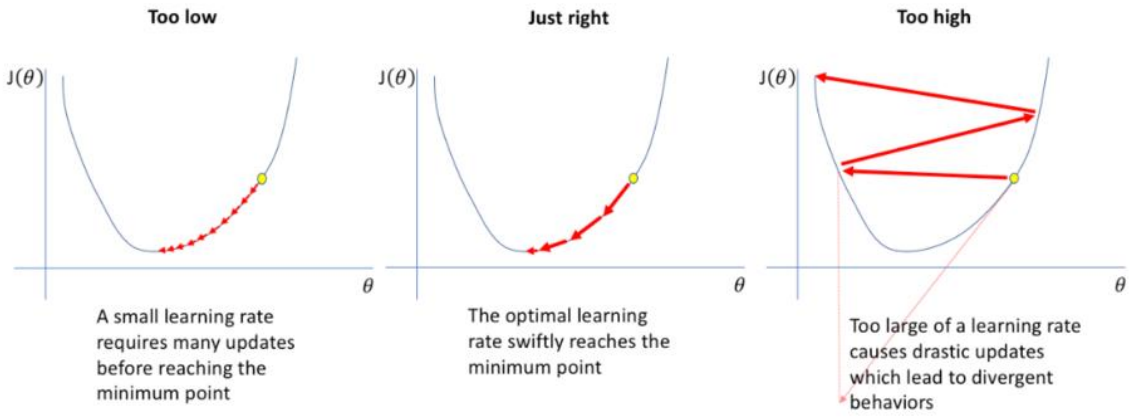
$$V_{t+1} = \beta_1 V_{t-1} + \beta_1 \frac{\partial L}{\partial W_t}$$

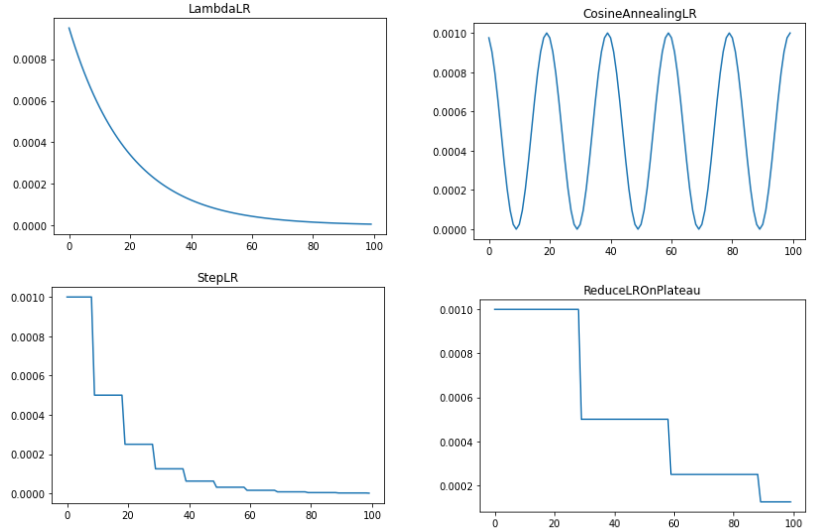
$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left(\frac{\partial L}{\partial W_t}\right)^2$$

Learning Rate Adjustment

"Optimizers using adaptive learning rate can help improve the gradient-based learning process."

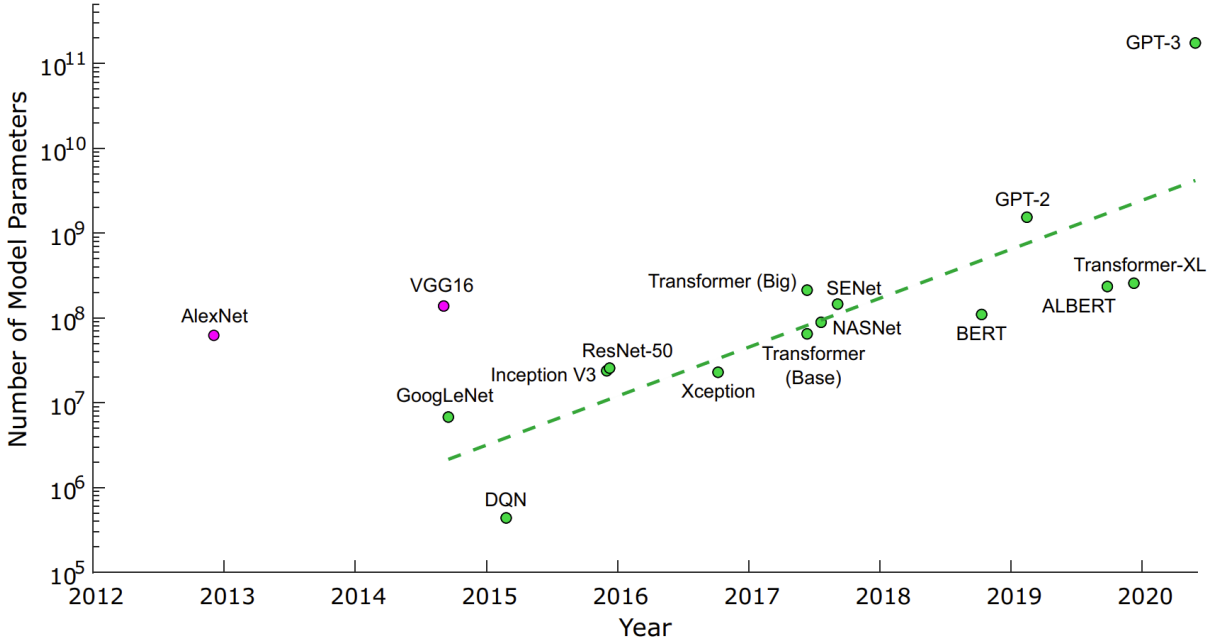
- **Learning rate scheduler**
  - Learning rate schedulers dynamically adjust the learning rate during training based on predefined rules or the model's performance.
  - The general approach is to start with a higher learning rate and gradually decrease it as training progresses.

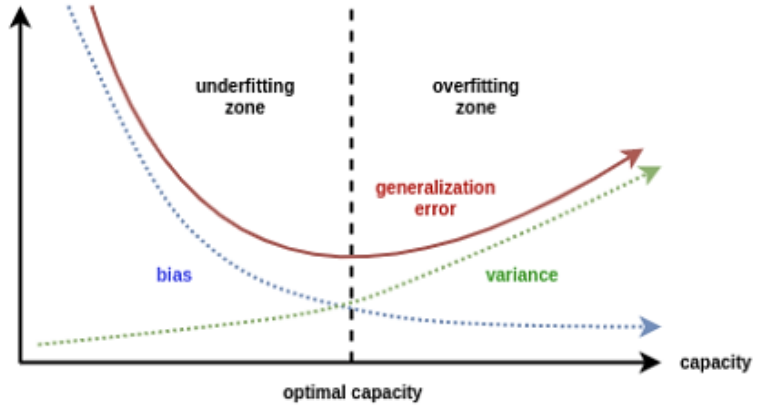




Overparameterization

"Overparameterization can lead to overfitting."





Bernstein, Liane, et al. "Freely scalable and reconfigurable optical hardware for deep learning." Scientific reports 11.1 (2021): 3144.

Overparameterization

1. Ad-hoc learning techniques for overfitting prevention

- Early Stopping: stop learning before overfitting occurs
- Weight regularization: regularization for penalizing overfitting of weights
- Dropout: randomly deactivate a part of hidden neurons

2. Data Augmentation

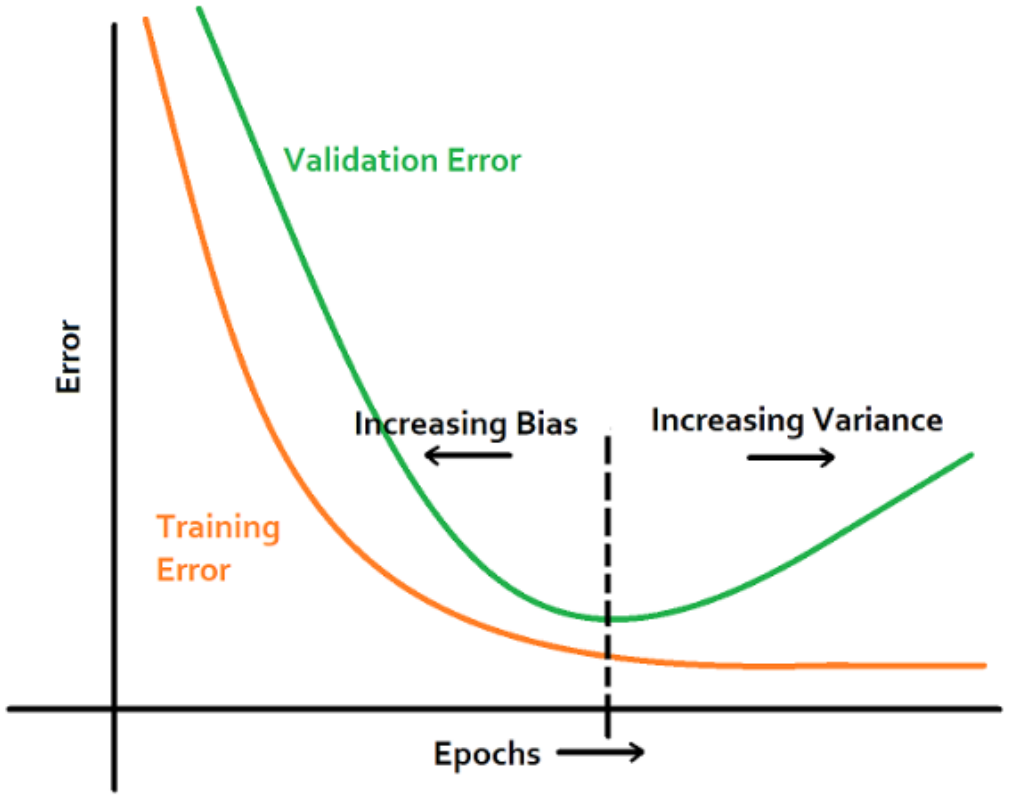
Increase the number of training samples by transforming the training dataset

3. Transfer Learning

Apply a pre-trained model from one task to a related task.

Early Stopping

"Early stopping of training when the model's performance on the validation data starts to degrade can prevent overfitting."



Weight Regularization

"Regularization discourages learning a more complex model, so as to avoid the risk of overfitting."

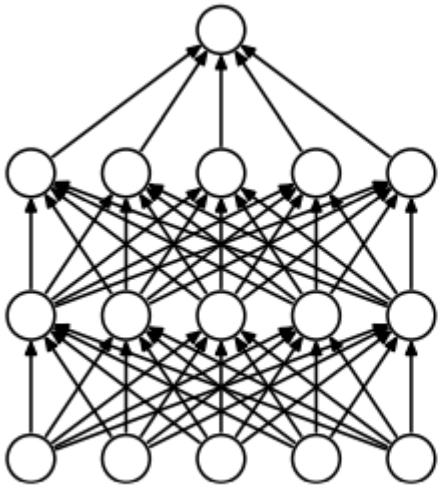
- The loss function

$$L(\boldsymbol{w}) = \ell(\boldsymbol{w}) + \alpha\|\boldsymbol{w}\|$$

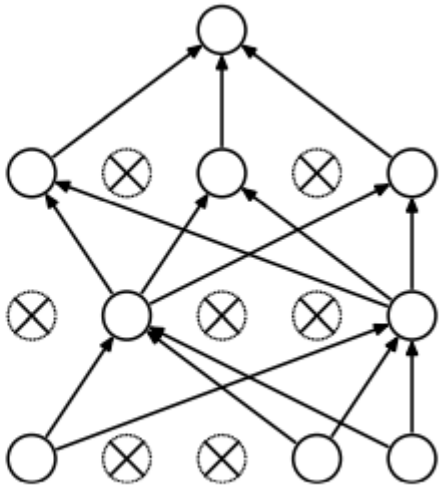
  - $\ell(\boldsymbol{w})$: the prediction error. ($e.g., MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$)
  - Regularization: $\|\boldsymbol{w}\|$ ($\|\cdot\|$: a norm)
    - L_p-norm: $\|\boldsymbol{w}\|_p = \left(\sum_{j=1}^{m}|w_j|^p\right)^{\frac{1}{p}}$
    - Penalty term α ($\alpha > 0$) : A user-determined hyperparameter that determines the degree of regularization.

Dropout

"Dropout is the process of randomly *dropping out* (setting to zero) some features of a layer during training"



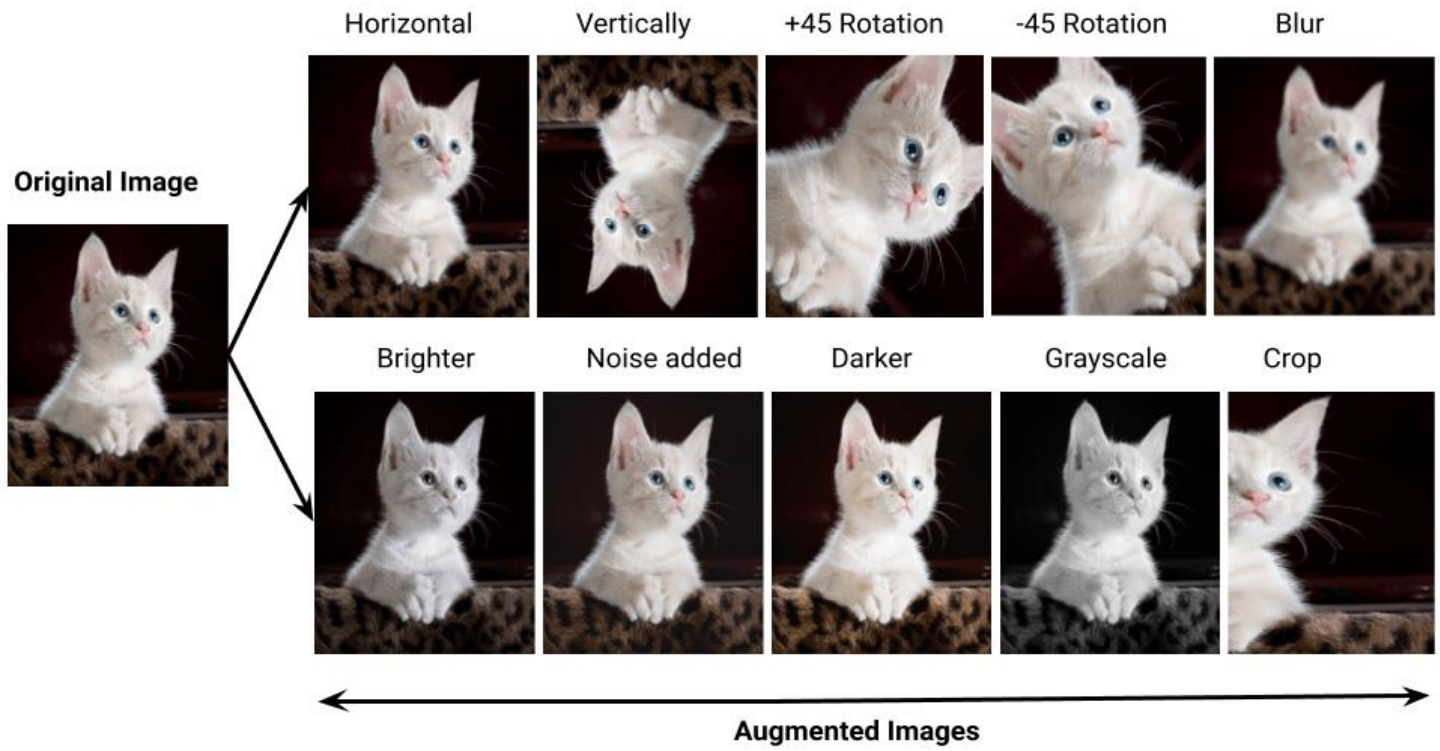
(a) Standard Neural Net



(b) After applying dropout.

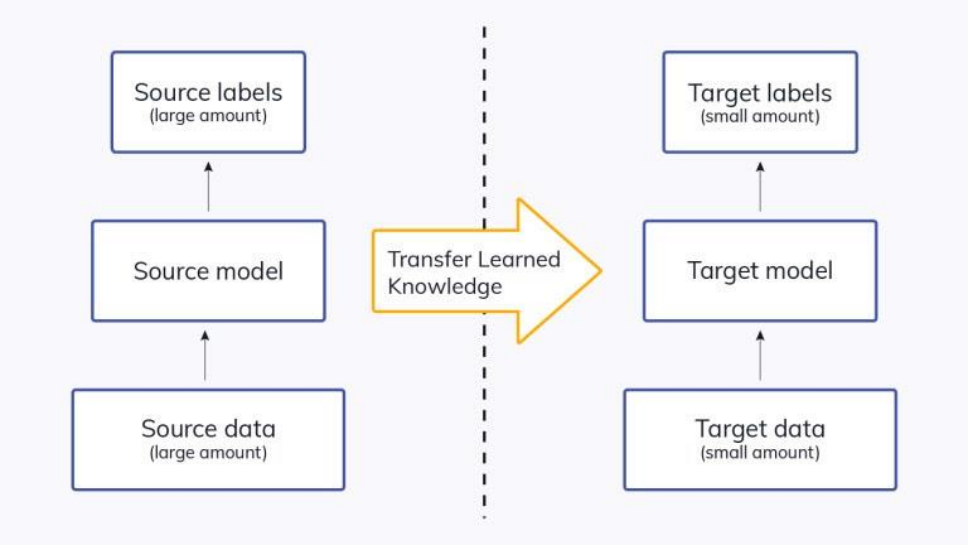
Data Augmentation

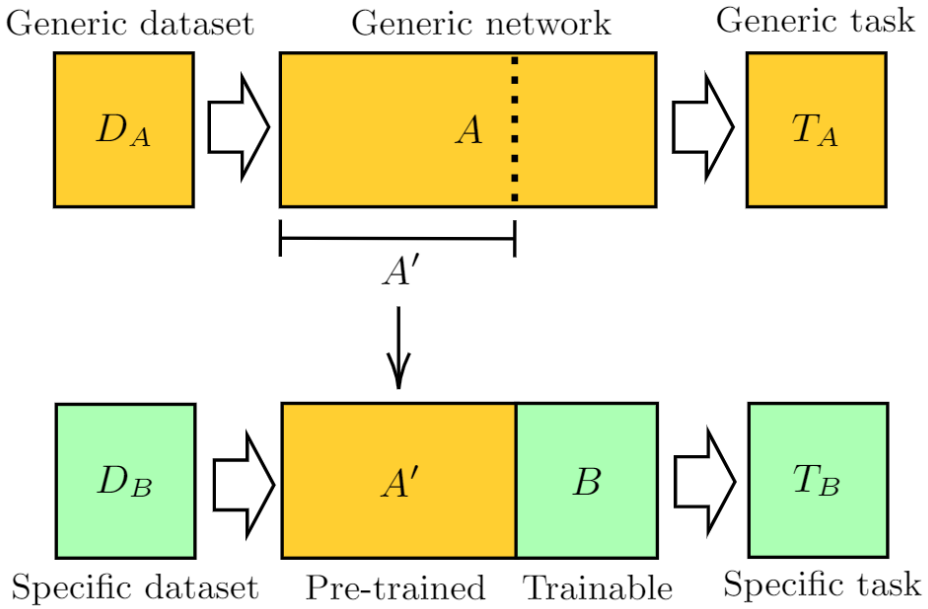
"Data augmentation is a technique to artificially increase the size of a dataset by transformations to the given data."



Transfer Learning

"Transfer learning allows models to leverage knowledge gained from one task to improve performance on a related task."





Transfer Learning

"There are different transfer learning strategies according to the circumstances."

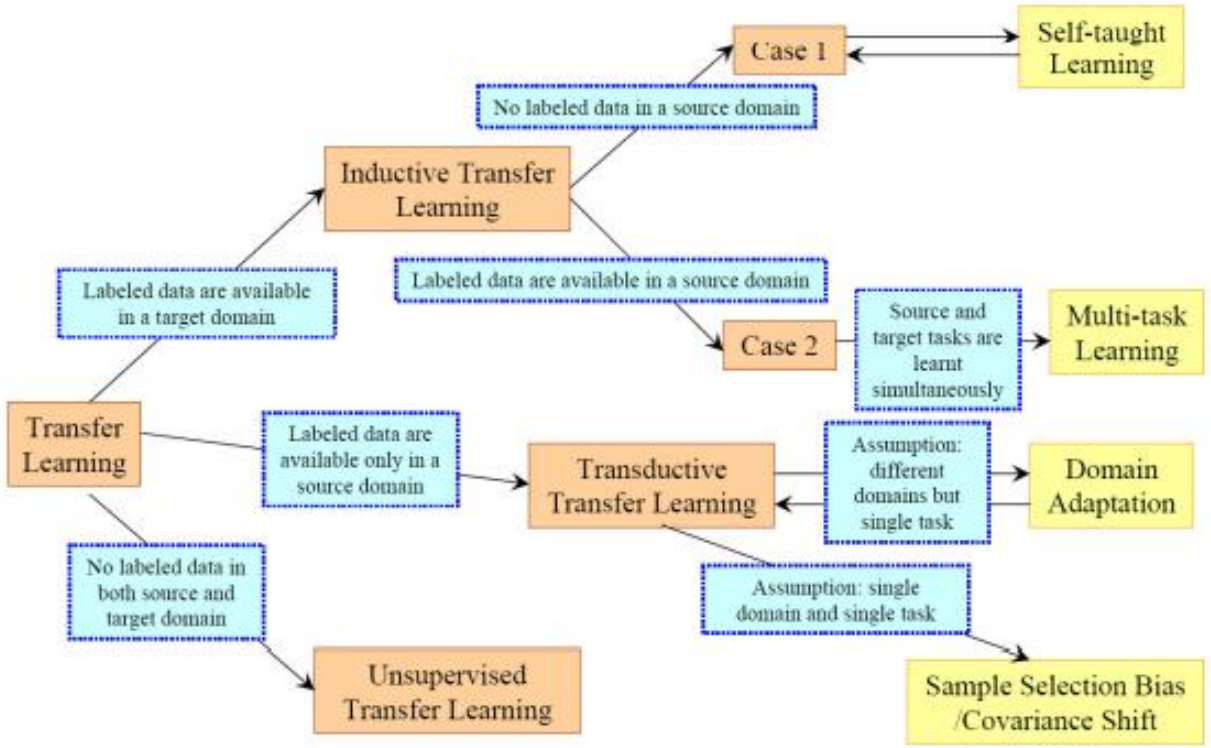
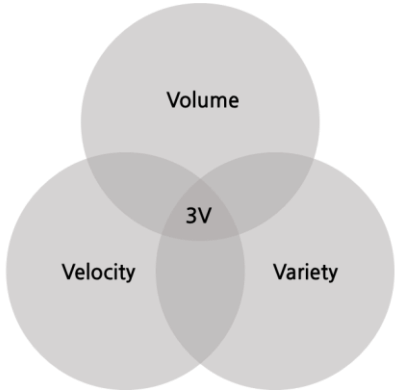


Image Source | Dipanjan Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. (link)

Takeaways

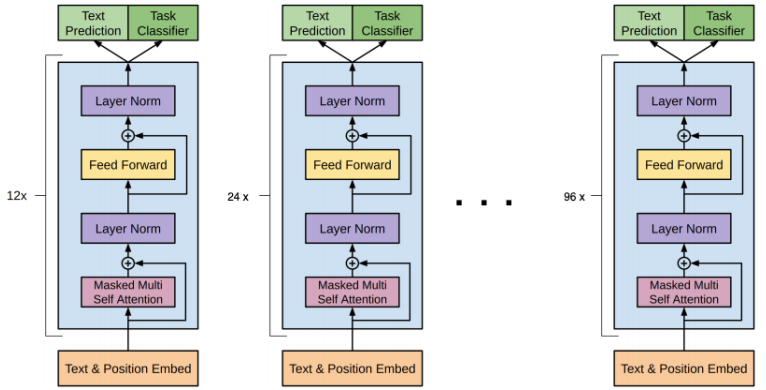
“What made Deep Learning possible are..”

“The deep learning revolution is driven by advances in big data, computing power, and artificial neural network algorithms.”



+

+



Big data

Cheaper and faster computers

Sophisticated models (structures)
and advanced learning techniques

Challenges for Deep Learning

① Data

② Computing resource

③ Learning process

Thank you! 😀