Advanced Machine Learning and Deep Learning

Dr. Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

Deep Learning

What is Deep Learning?

- **What is Deep learning?**
- **Short (inaccurate) Answer:** A **wide** and **deep** ***Neural Network*** with many hidden layers and neurons.
- **General Answer:** A part of Machine Learning that tries to learn ***multiple levels of features or representations*** of the data. Higher level features are derived from lower level features to form a hierarchical representation (the input of each layer is the output from previous layer).

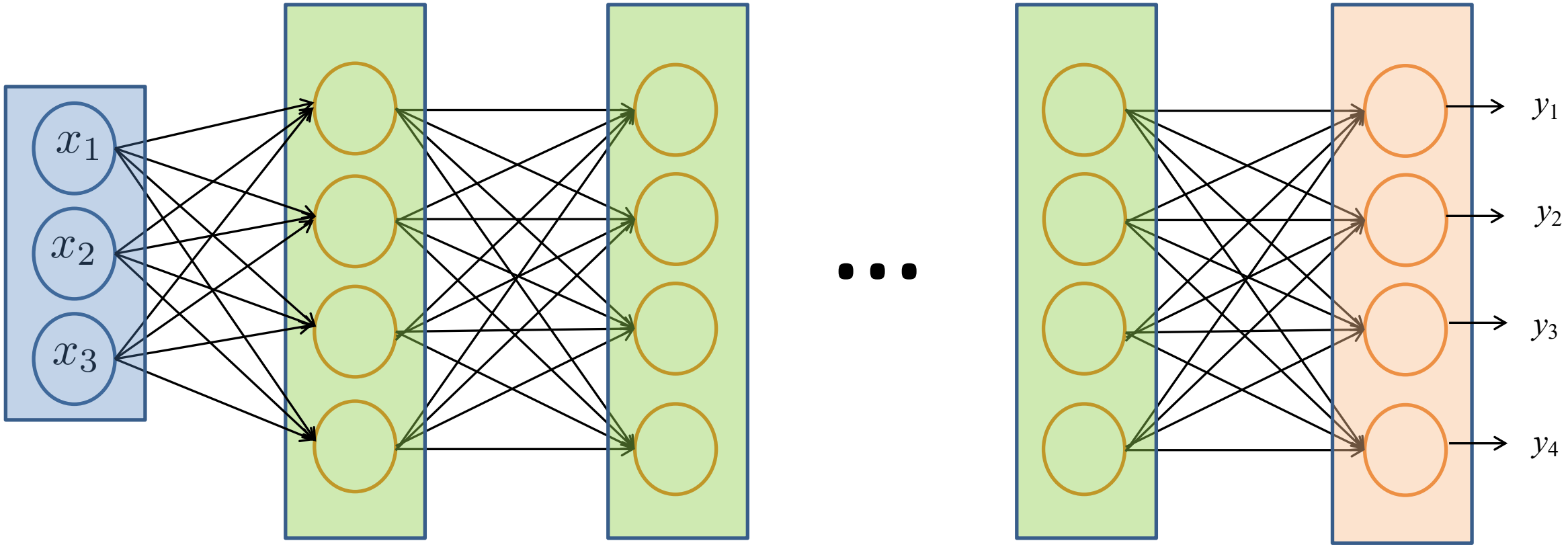
Deep Learning Methods

The main Deep Learning Methods:

- Multi-Layer Perceptron (MLP), Deep Neural Networks (DNN)
- Convolutional Neural Networks (CNN, aka ConvNet, RCNN, Fast RCNN, Faster RCNN, YOLO)
- Restricted Boltzmann Machines (RBM)
- Deep Belief Networks (DBN)
- Deep AutoEncoder
- Recurrent Neural Networks (RNN), LSTM
- Generative Adversarial Network (GAN)
- …

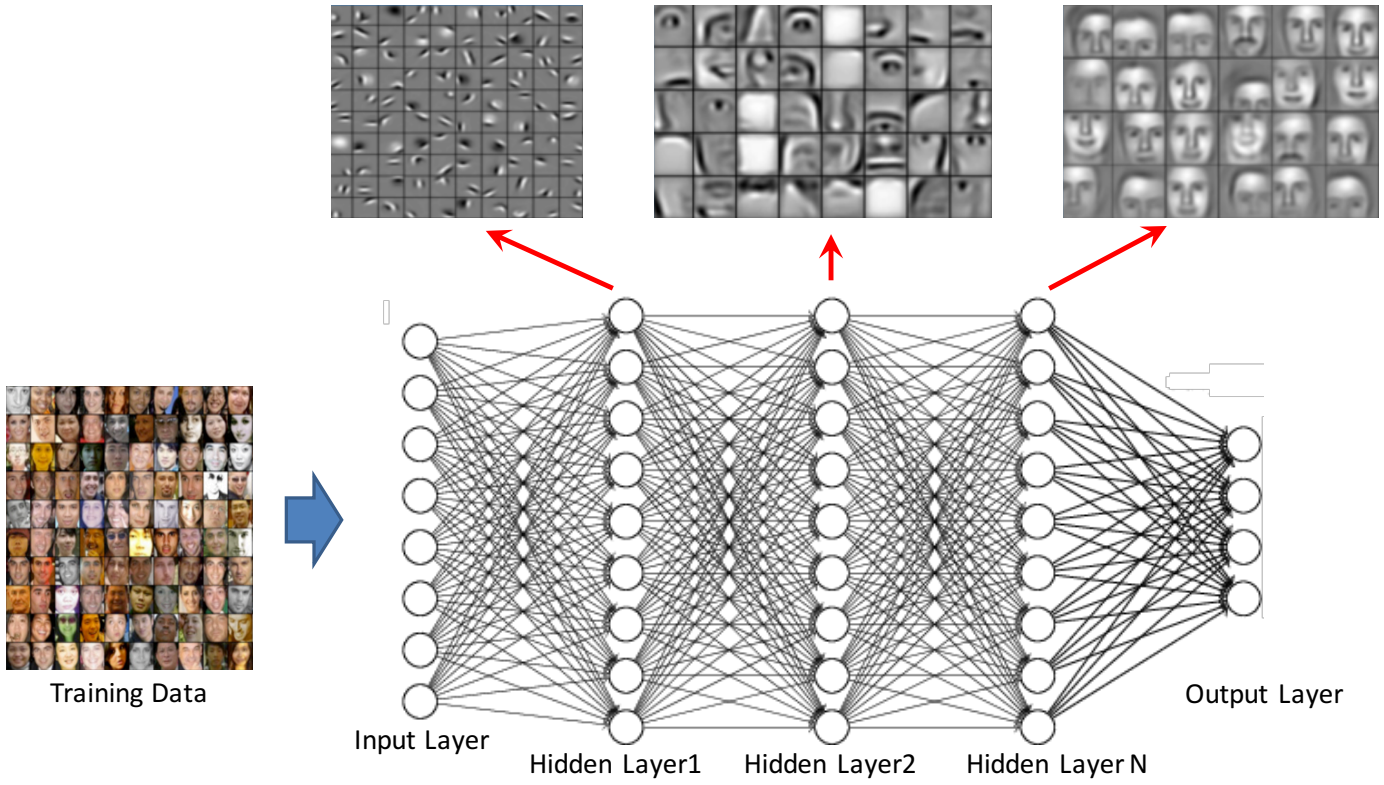
Deep Neural Networks (DNN)

- **A *Deep Neural Network* usually includes several hidden layers:**

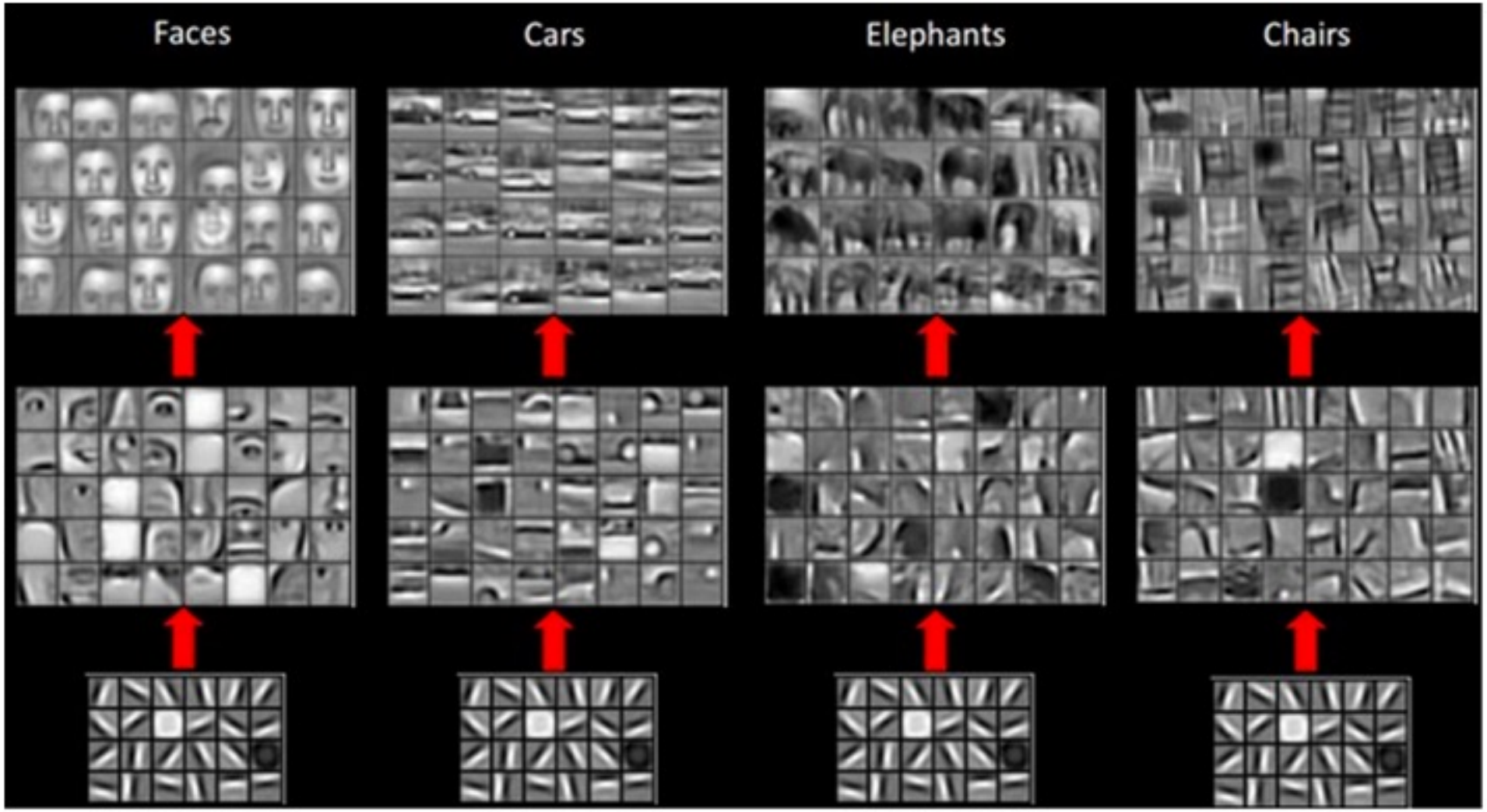


- A deep neural network consists of a hierarchy of layers, whereby each layer progressively extracts higher and higher-level features.
- Early layers look for very simple patterns (e.g. “edges” in a face recognition problem), middle layers combine simple patterns to find more complex patterns (e.g. eye, nose), and final layers try to combine previous patterns to find and recognize high level patterns (e.g. face)!

- Finally, the last layer makes a final decision based on the most advanced patterns find in the data!



Some Examples!

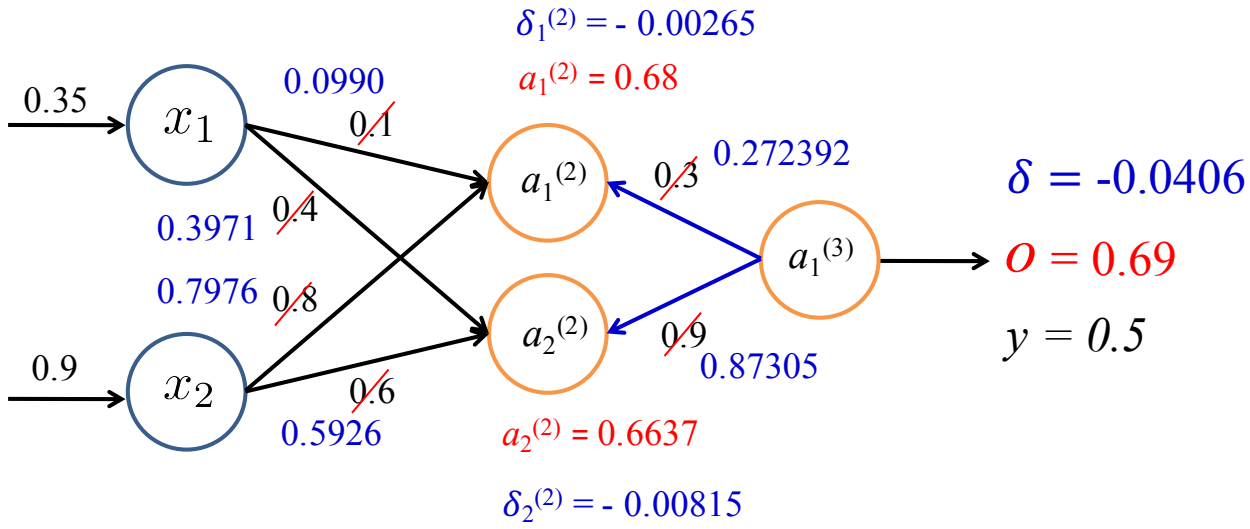


Deep Learning Challenges

- **Deep Learning Challenges:**
  1. **High Computational Complexity.**
  2. **Large number of parameters (Hyper-Parameters) for tuning. For many of them we need to do trial & error to find the best value.**
  3. **Need a large training dataset to be able to train the weights.**
  4. **The worst issue: Vanishing Gradient dramatically slows down the training process. Sometimes, the algorithm even fails in learning the weights.**

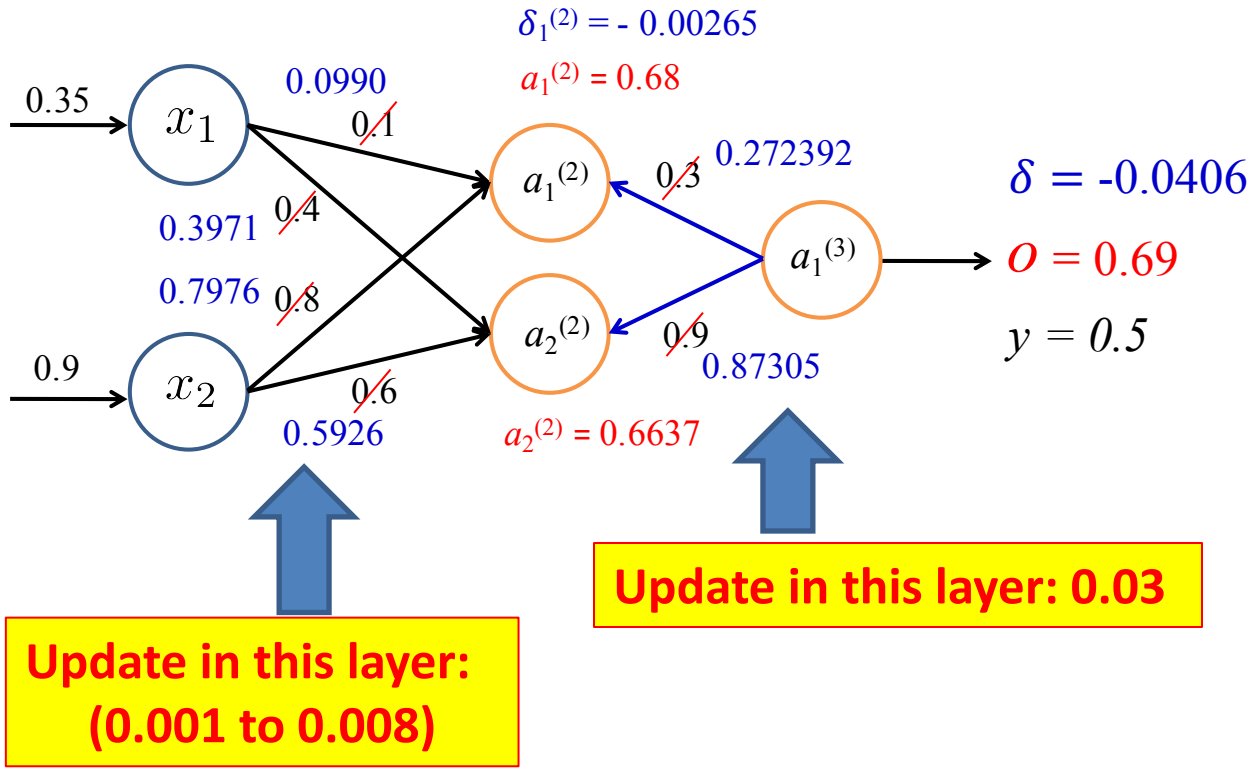
Vanishing Gradient Problem!

- **Vanishing Gradient** is a big problem in deep neural networks training using gradient-based learning and backpropagation.
- **Vanishing Gradient was the main reason for deep learning failure in the past!**
- As you remember, in each iteration of backpropagation, each weight receives an **update** proportional to the error gradient computed by back-propagating of error.



$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta_i^{(l+1)}\ a_i^{(l)}$$

$$\delta_i^{(l)} = [\ \sum w_{ji}^{(l)} \delta_j^{(l+1)}\ ]\ a_i^{(l)}\ (1 - a_i^{(l)})$$



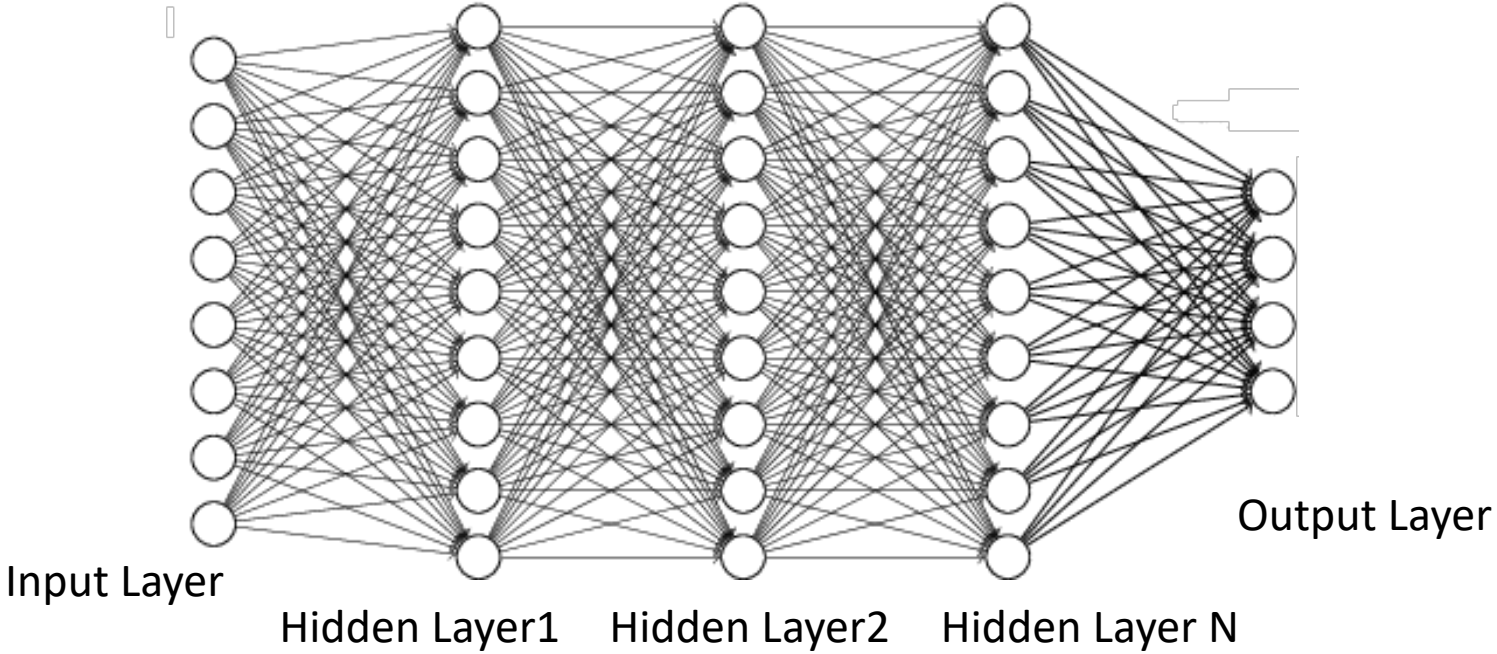
$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\, \delta_i^{(l+1)}\, a_i^{(l)}$$

$$\delta_i^{(l)} = [\sum w_{ji}^{(l)} \delta_j^{(l+1)}]\, a_i^{(l)}\, (1 - a_i^{(l)})$$

- This makes the **update step size** **decrease exponentially** when we move backward towards the front layers (because when we move backward we keep multiplying the gradient error to a small number in each layer!).
- Some activation functions such as Sigmoid or tanh makes the problem even worse.

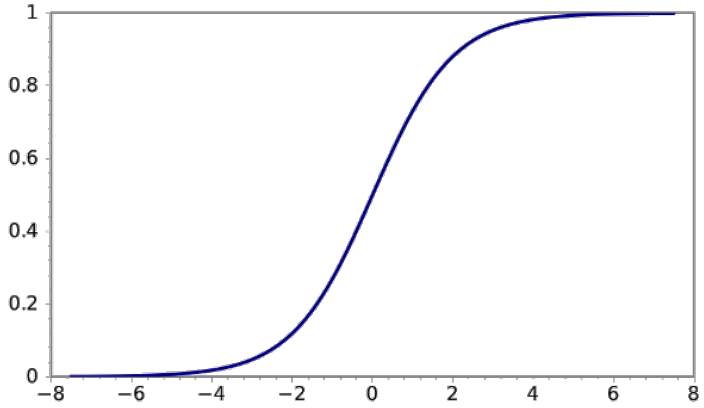
Vanishing Gradient Problem!

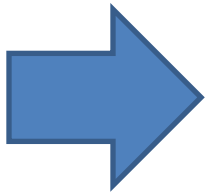
- Thus, for a **deep neural network** with many hidden layers, the **update step size will be very small in front layers** in each iteration of backpropagation. Thus, if we start with random initial weights, it will take **LONG** time for the first layers to learn the weights! (remember that first layers are very important, because they are responsible to find the basic patterns that will be used in next layers).

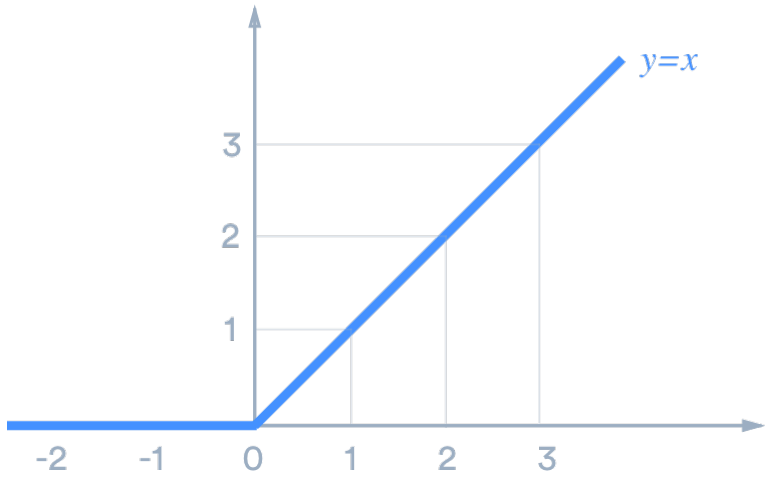


ReLU: Rectified Linear Unit as Activation Function

- Using ReLU (Rectified Linear Unit) as a non-linear activation function (instead of Sigmoid or tanh) can improve the Vanishing Gradient problem a little bit by reducing the gradient decay rate in backpropagation.
- Still, if we start with random initial weights, it will take a long time for the first layers to learn the weights!







Sigmoid function:

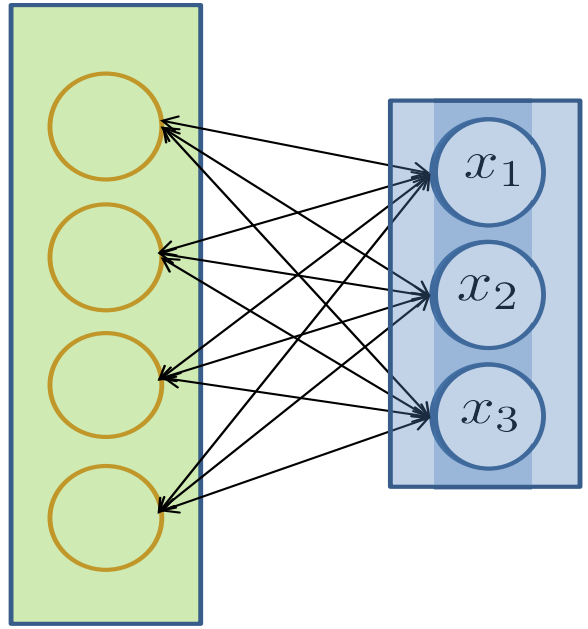
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$RELU(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x >= 0 \end{cases}$$

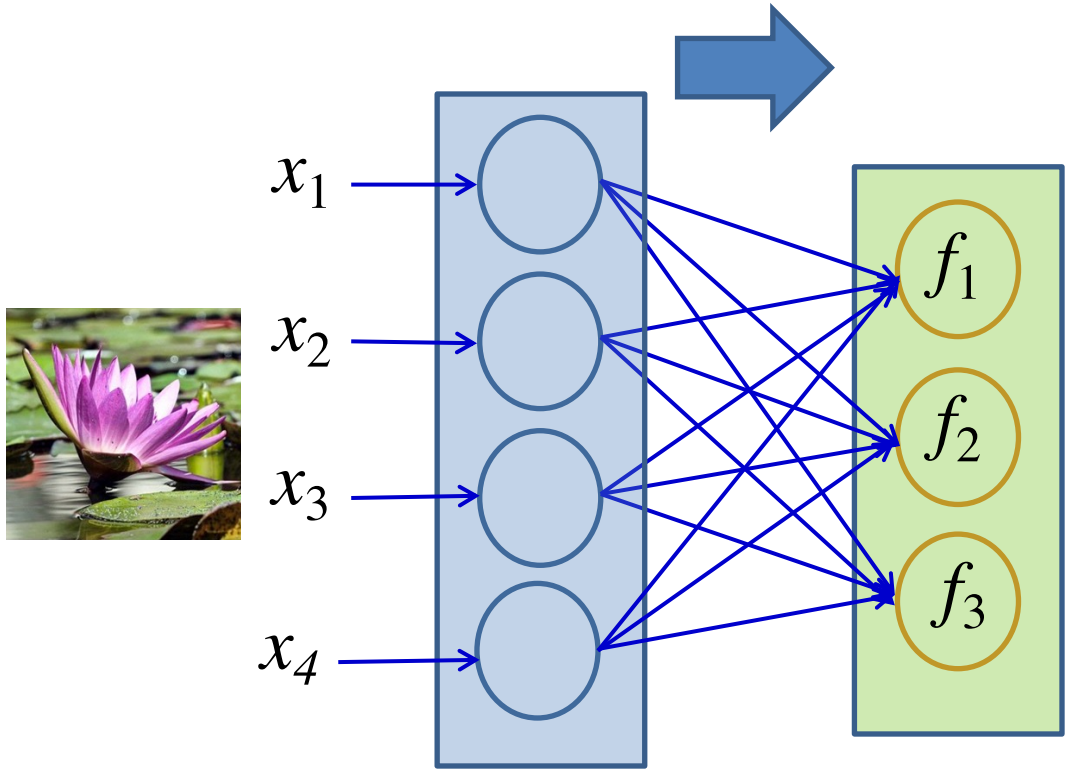
Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN)

Restricted Boltzmann Machines (RBM)

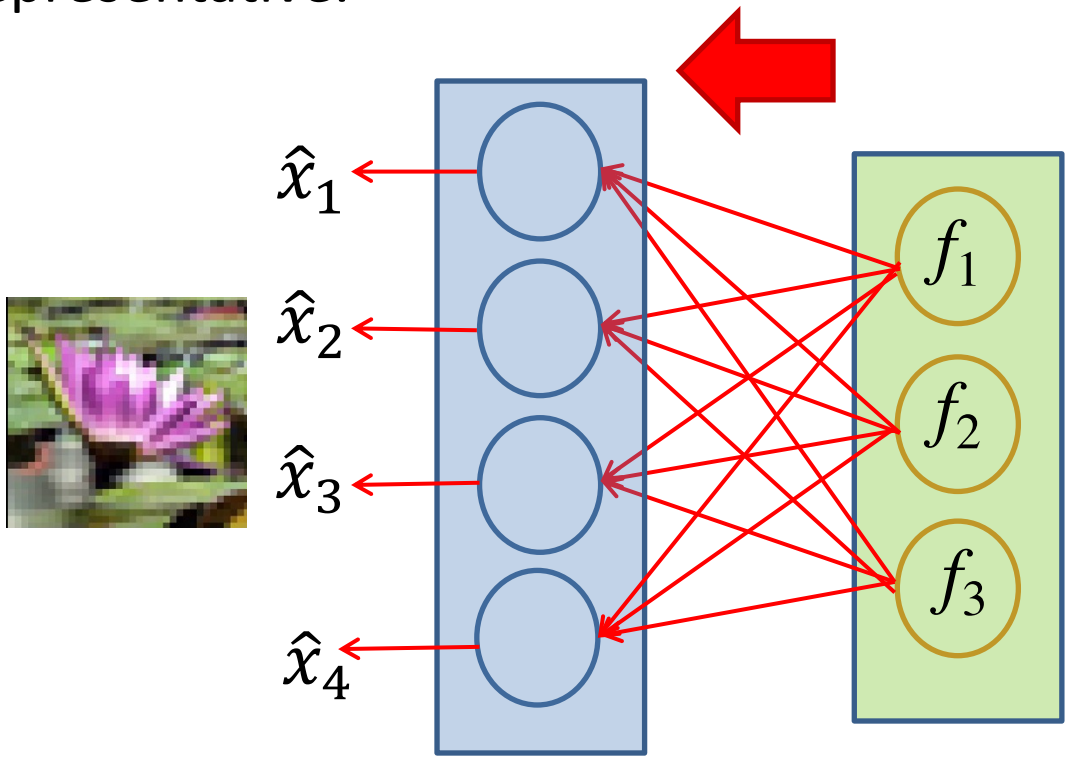
- In 2006, **Geoffrey Hinton** proposed a technique based on **Restricted Boltzmann Machines (RBM)** to improve the problem of slow training of deep neural networks (caused by vanishing gradient)!
- **RBM is a Two-layer BIDIRECTIONAL neural network**. The first layer is called **Visible Layer**, and the second layer is called **Hidden Layer**.
- RBM tries to find the best representative of the input data (the best low level features) in an **Unsupervised** fashion!



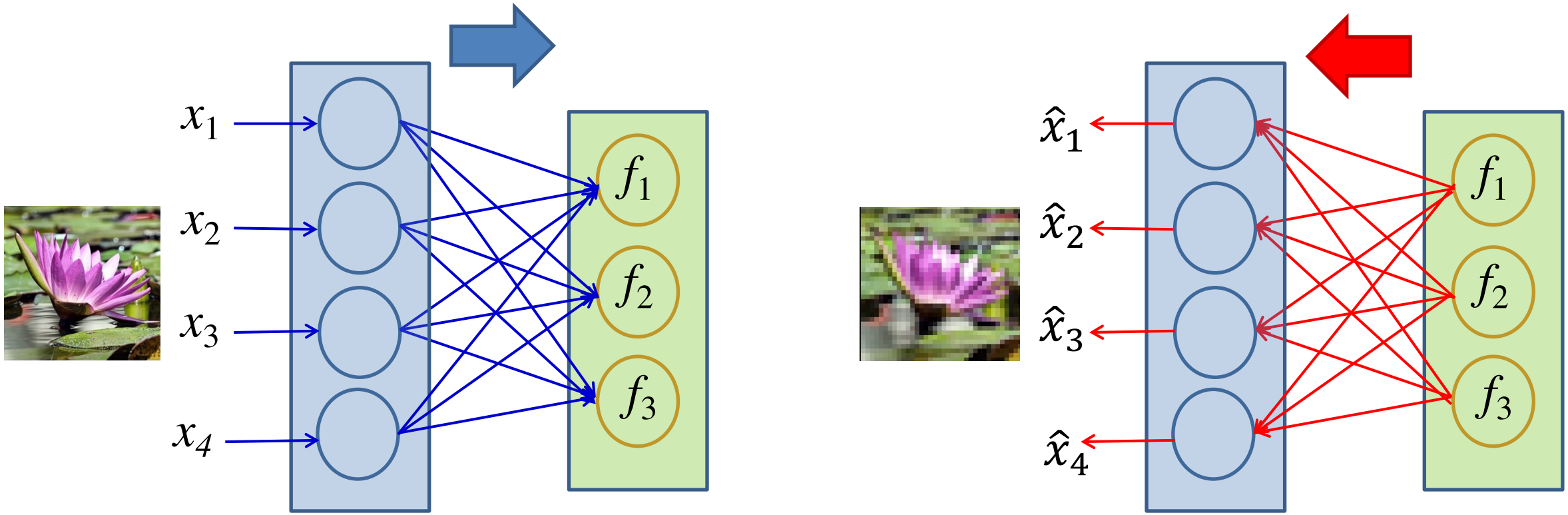
Stage1: Get some inputs data, and Move Forward to generate some random outputs from the inputs.



Stage2: Feed the generated output from stage1 back to the network in reverse direction (Move Backward) to **Reconstruct** the original inputs from the generated representative.

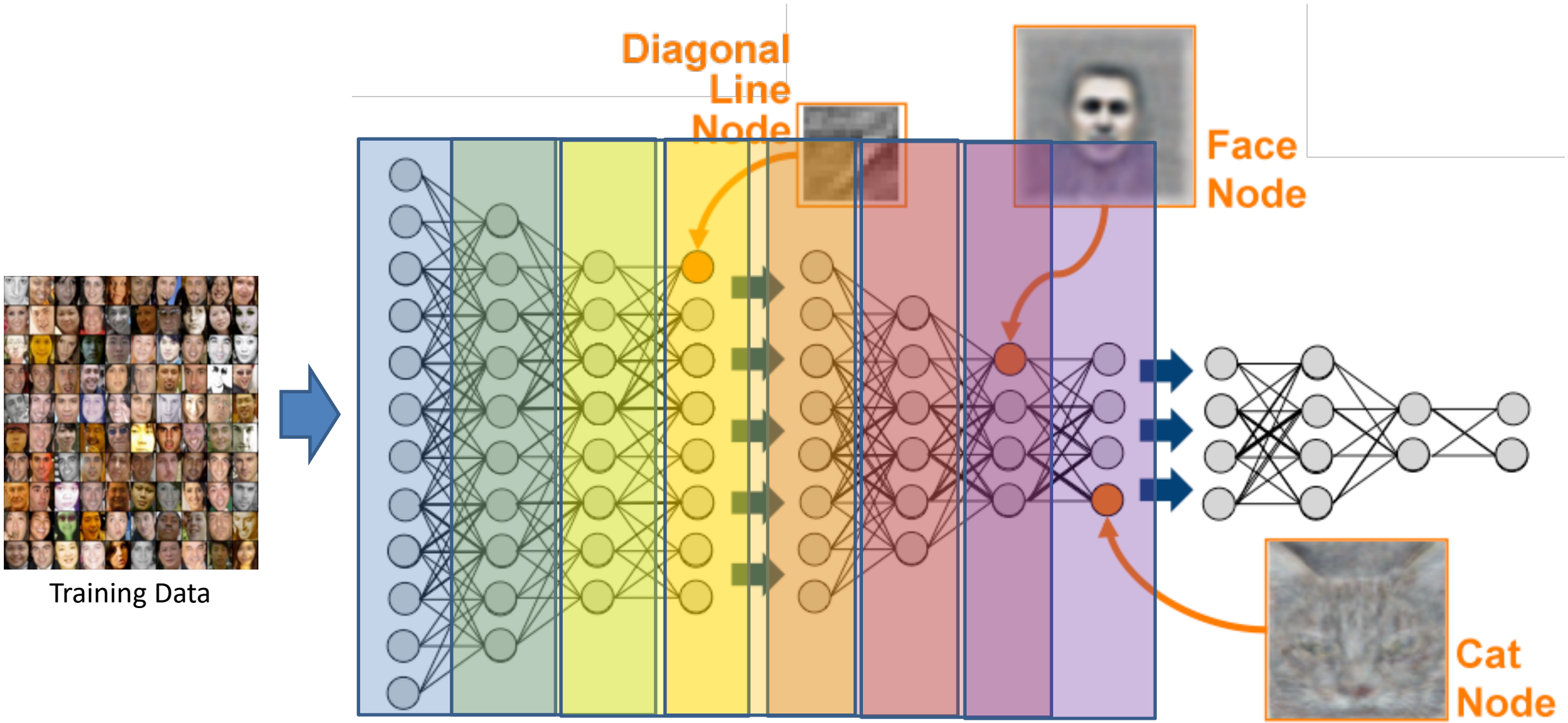


Repeat: We repeat this process on and on! In each back & forth iteration, we **modify the weights** to minimize the difference between the original data (e.g. image) and its reconstruction. After enough iterations, the outputs (f_1, f_2, ...) will be the best representatives (features) that represent the input data!



Deep Belief Networks (DBN)

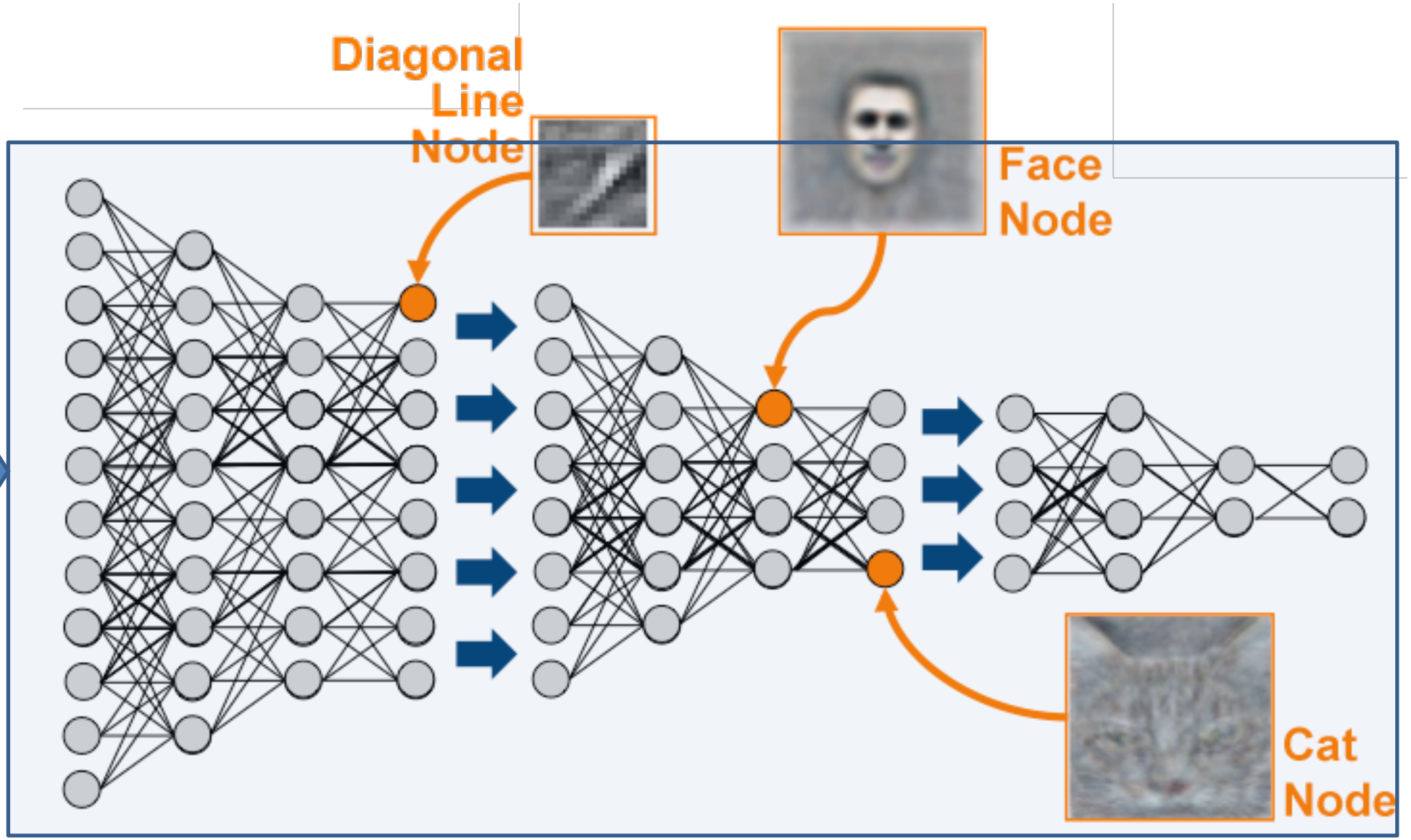
- **Deep Belief Network (DBN)** is a type of deep neural network, composed of **multiple layers of RBM**! In other word, **DBN is a stack of RBMs**!
- **At the very beginning, We use RBM method for each two consecutive layers of the DBN, only to *initialize the weights of DBN!***
- In fact in this method, each layer tries to find the best representative of the previous layer in an **Unsupervised** fashion (by reconstructing the previous layer as we had in RBM).
- After this stage and **initializing the weights**, a DBN can be further trained in a **Supervised** fashion (using some labeled data) to perform classification.
- **Since the weights are already initialized in the best way, the process of updating and finalizing the weights in the supervised stage will be much faster and shorter!**
- **This method can significantly increase the speed of training stage and reduce the effect of Vanishing Gradient Problem!**



- **At the very beginning, We use RBM method for each two consecutive layers of the DBN to *initialize the weights of DBN!***

[Figure Source]: Google Brain

Training Data



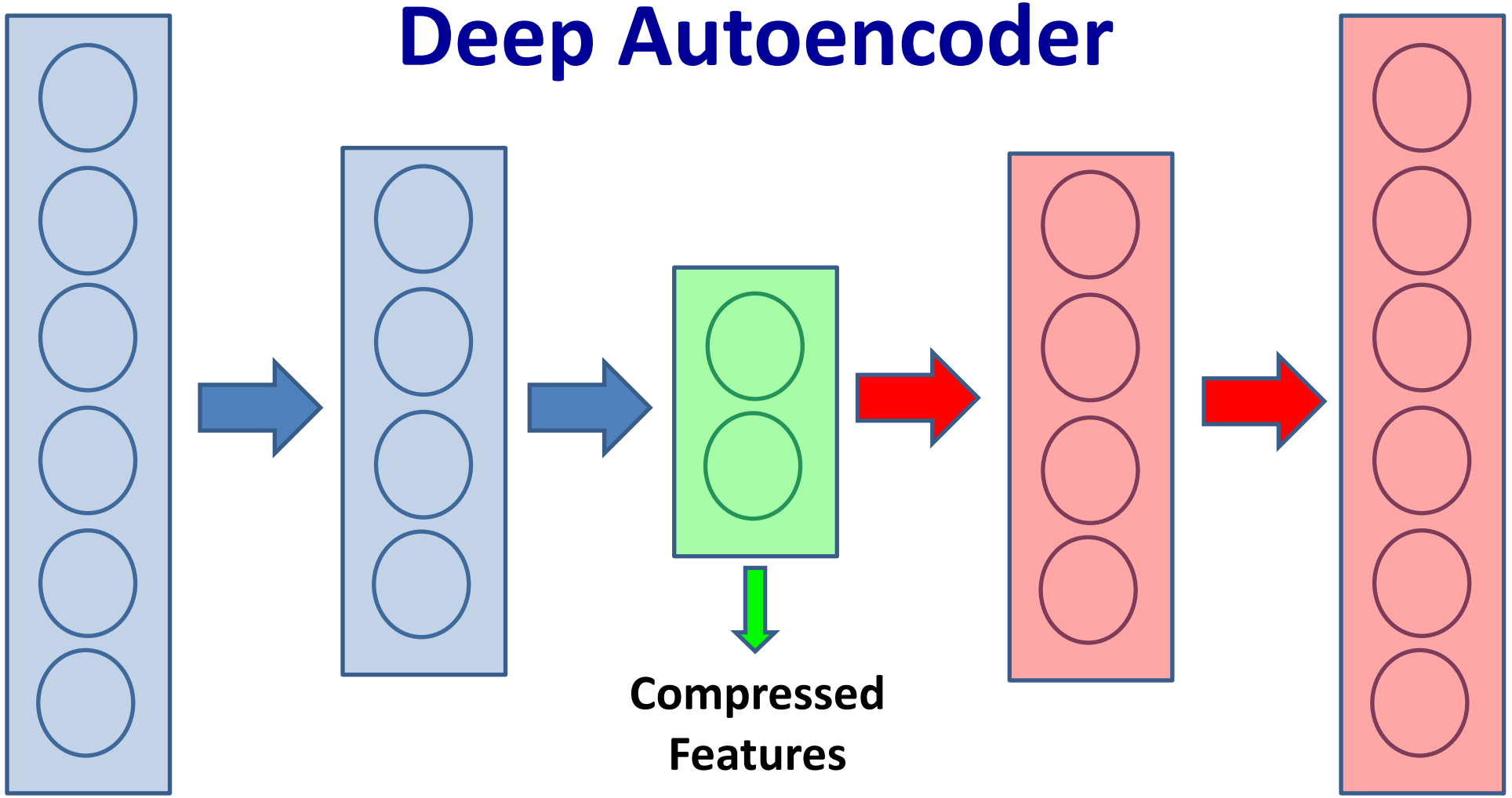
- **After initializing the weights, a DBN can be further trained in a Supervised fashion (using some labeled data) to perform classification.**

Deep AutoEncoder

Deep Autoencoder

- **Deep Autoencoder** is an unsupervised deep neural network. It can be composed of two symmetrical DBN: **Encoding network** and **Decoding network**.
- **Deep Autoencoder** is used to learn data codings to create a representation (encoding) for data in the most efficient data-driven way. It can be one of the best techniques for dimensionality reduction, by learning and extracting the most important part of the data .
- The encoding network learns to compress the input. The decoding network is used to reconstruct the data. By minimizing the difference between the original input data and the reconstruction, autoencoder can extract the best compressed representative (best new features) of the original data!

Deep Autoencoder



Thank You!

Questions?