Introduction to Neural Networks

ASME
IDETC-CIE 2021

International Design Engineering Technical Conferences & Computers and Information in Engineering Conference

VIRTUAL CONFERENCE AUG 17–19

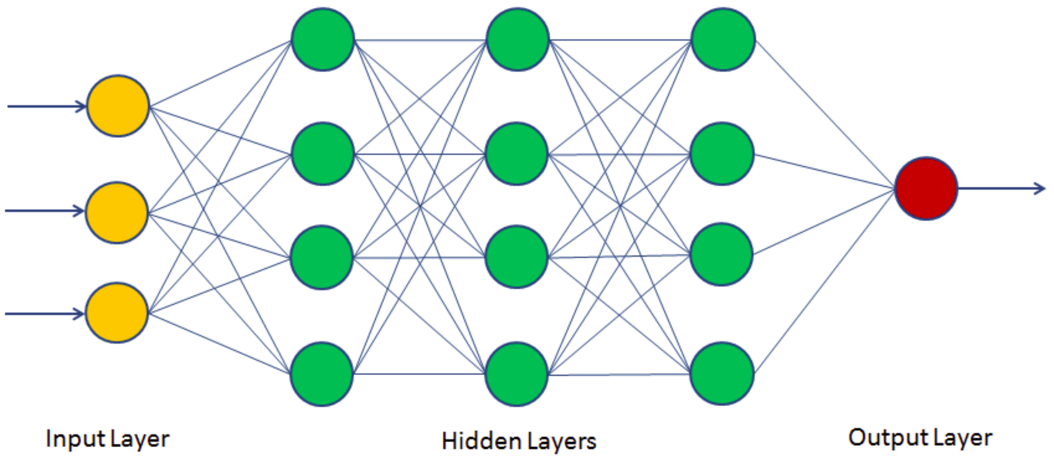
PennState

Binyang Song

Outline

- **Introduction to neural networks (NNs)**
- **Convolution Neural Network (CNN)**
- **Generative Adversarial Network (GAN)**

Introduction to NNs

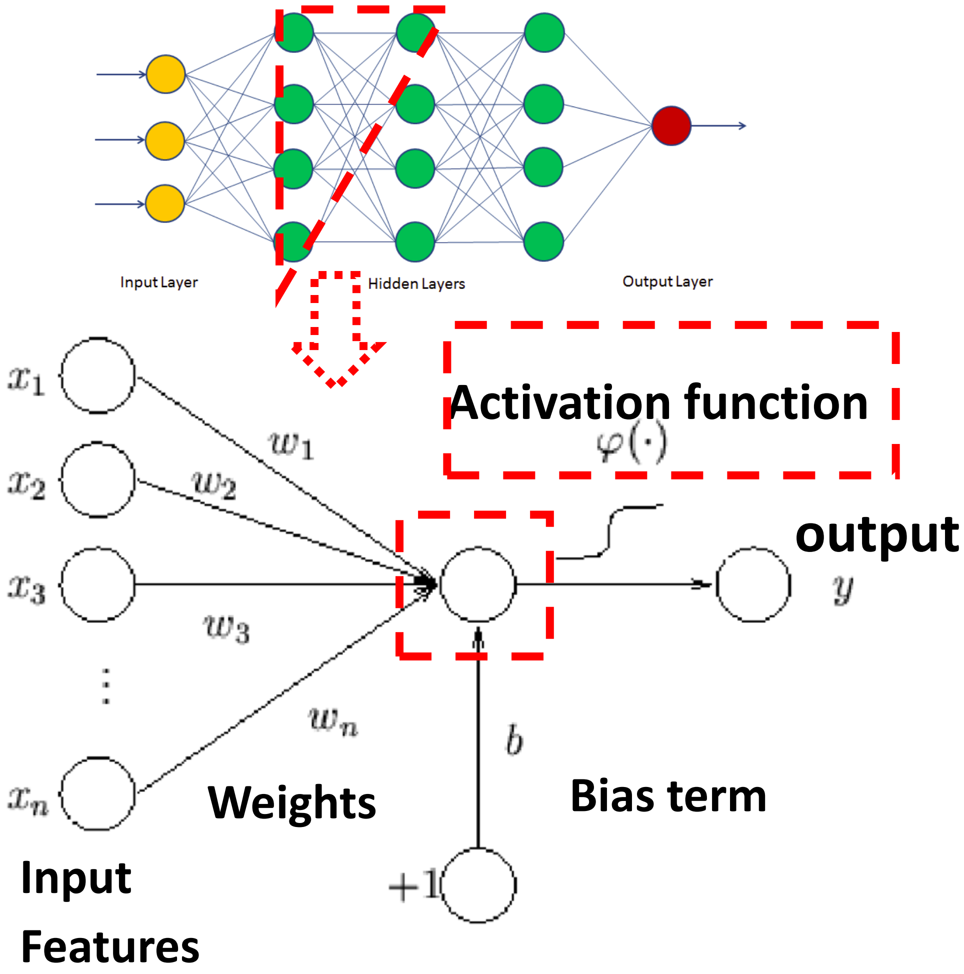


NNs are algorithms that are inspired by the biological neuron system to perform a particular task or function.

- Globally, input layer, hidden layers, output layer
- Neurons and connections
- Locally, output of one layer is input of next layer

References: https://www.datacamp.com/community/tutorials/neural-network-models-r
https://www.researchgate.net/figure/Signal-flow-graph-of-the-perceptron-A-single-perceptron-is-not-very-useful-because-of-its_fig2_266493320

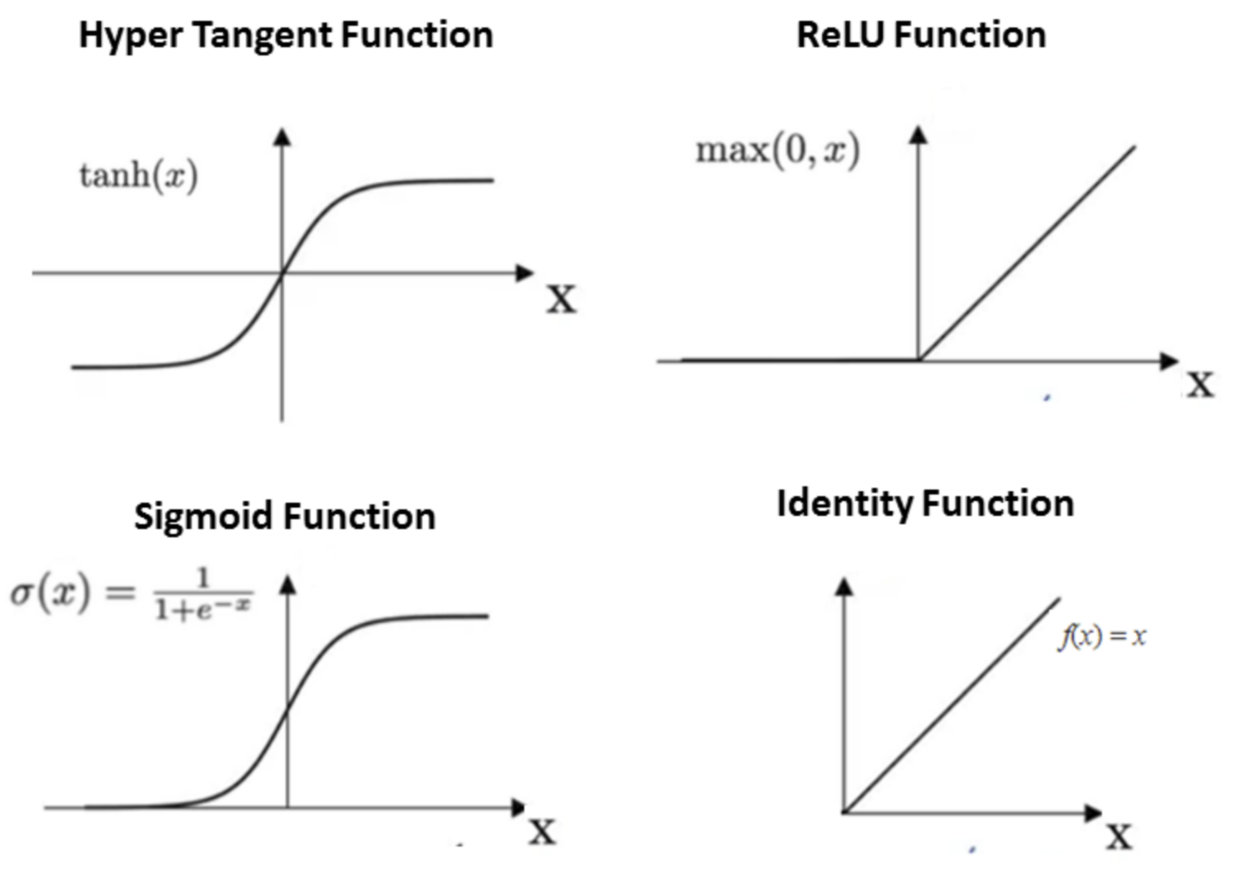
Introduction to NNs: Basics



- Input features, weights, bias term, summation, activation function, output
- Sum: $z = \sum_i x_i w_i + b$
- Output: $y = \varphi(z)$
- Training of NN: get proper weights and bias terms using training data

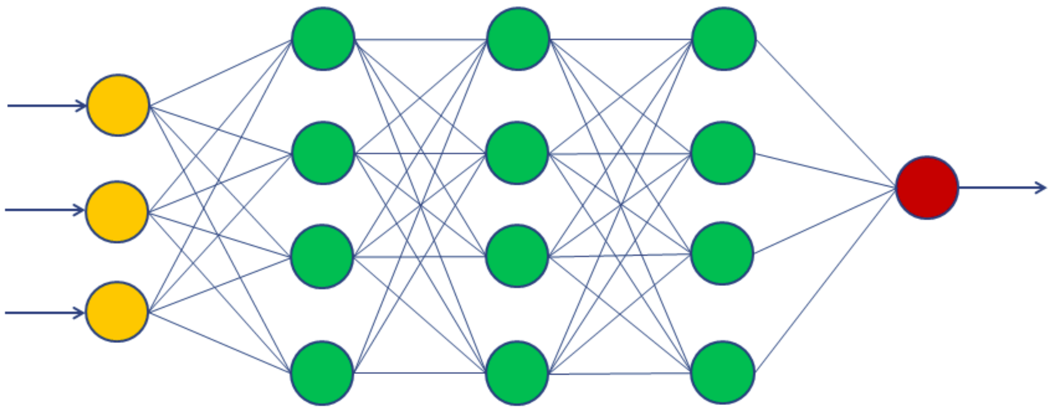
References: https://www.datacamp.com/community/tutorials/neural-network-models-r
https://www.researchgate.net/figure/Signal-flow-graph-of-the-perceptron-A-single-perceptron-is-not-very-useful-because-of-its_fig2_266493320

Introduction to NNs: Activation Function



References: https://www.datacamp.com/community/tutorials/neural-network-models-r
https://www.researchgate.net/figure/Signal-flow-graph-of-the-perceptron-A-single-perceptron-is-not-very-useful-because-of-its_fig2_266493320

Introduction to NNs: Loss Function



Y: True output

Loss = $\mathcal{L}(Y_{pred}, Y)$

Y_{pred} : Predicted output

Choice of loss function is directly related to the activation function used in the output layer.

Regression Problem

- **Output Layer Configuration**: One node with a linear activation unit.
- **Loss Function**: Mean Squared Error (MSE).

Binary Classification Problem

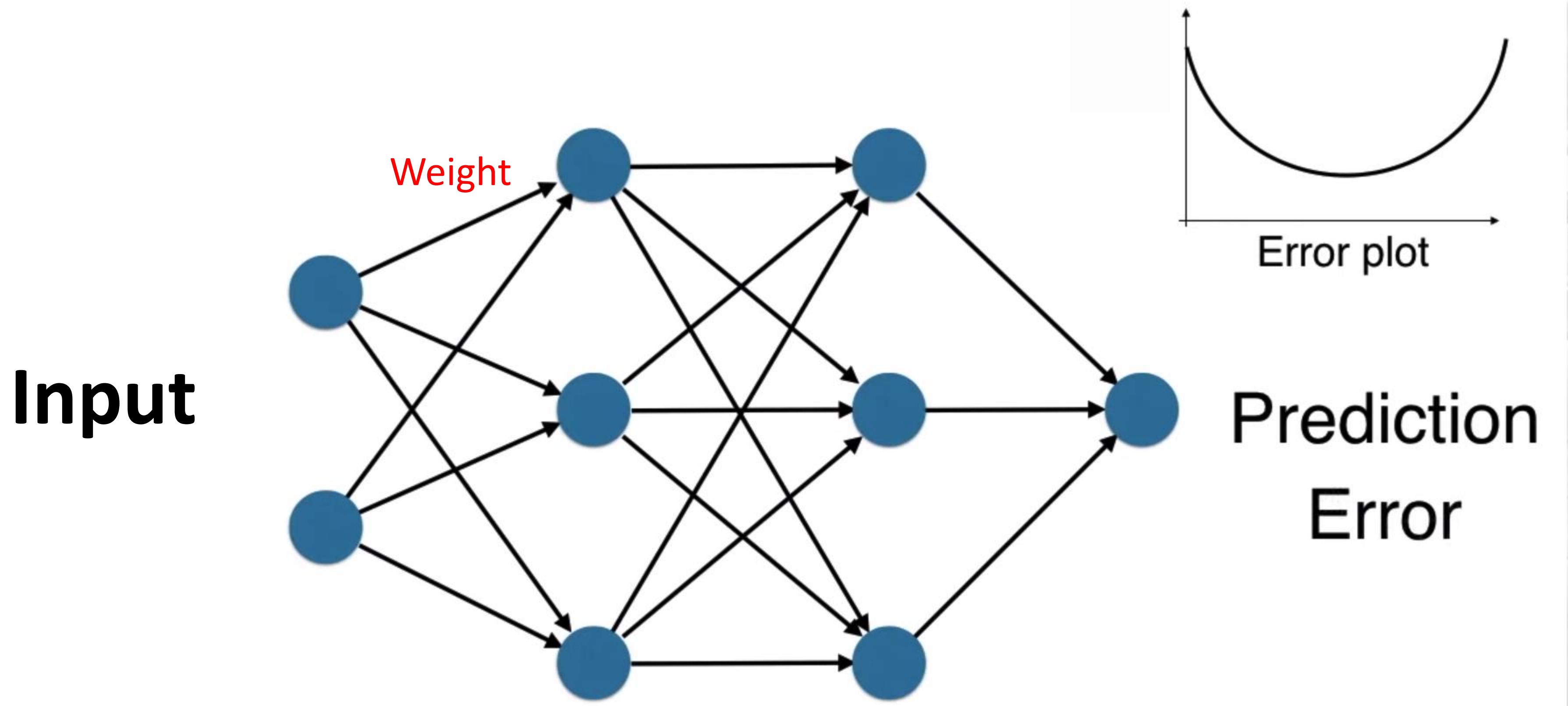
- **Output Layer Configuration**: One node with a sigmoid activation unit.
- **Loss Function**: Cross-Entropy, also referred to as Logarithmic loss.

Multi-Class Classification Problem

- **Output Layer Configuration**: One node for each class using the softmax activation function.
- **Loss Function**: Cross-Entropy, also referred to as Logarithmic loss.

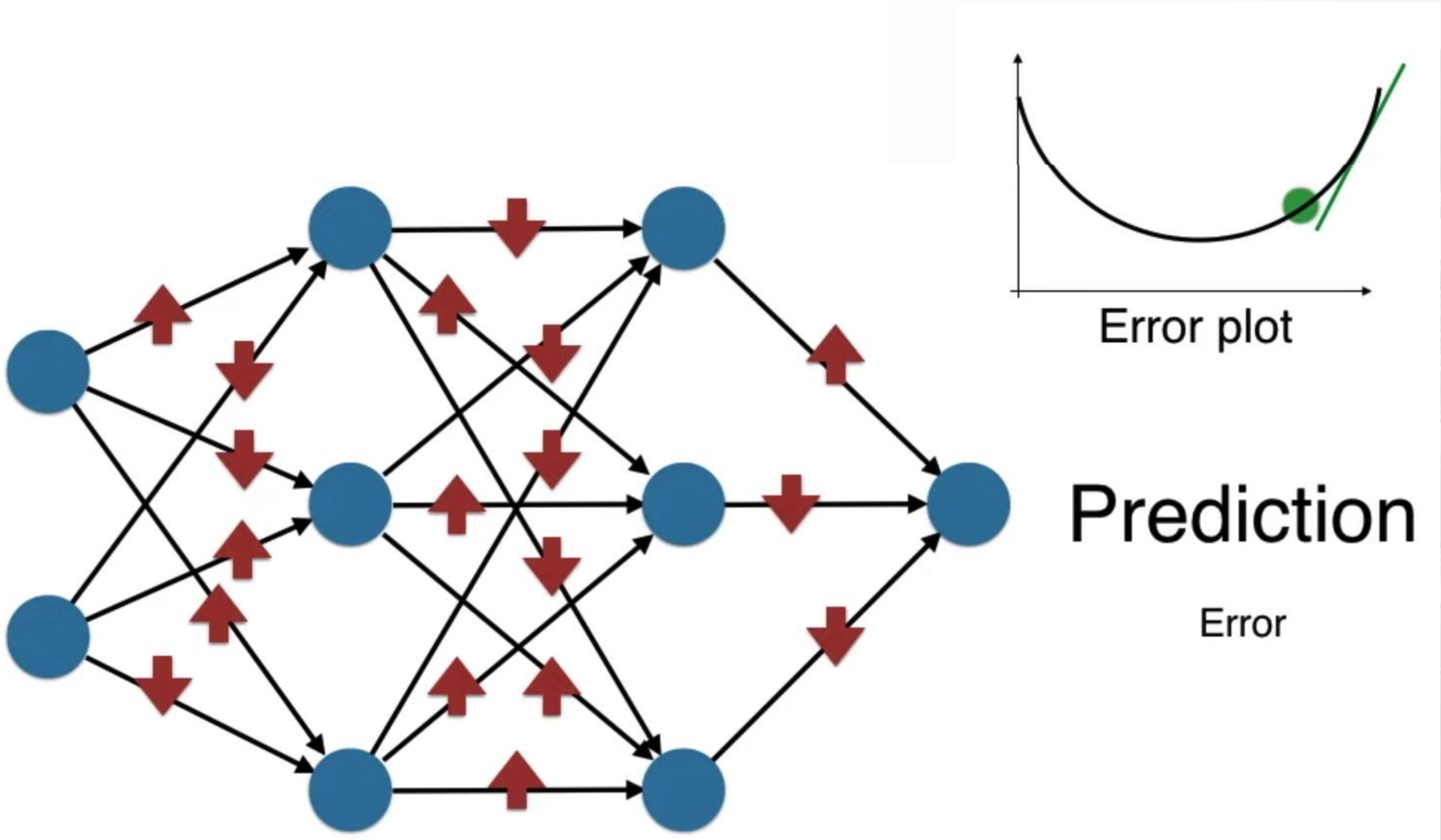
References: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

Introduction to NNs: Backpropagation



Introduction to NNs: Backpropagation

Input

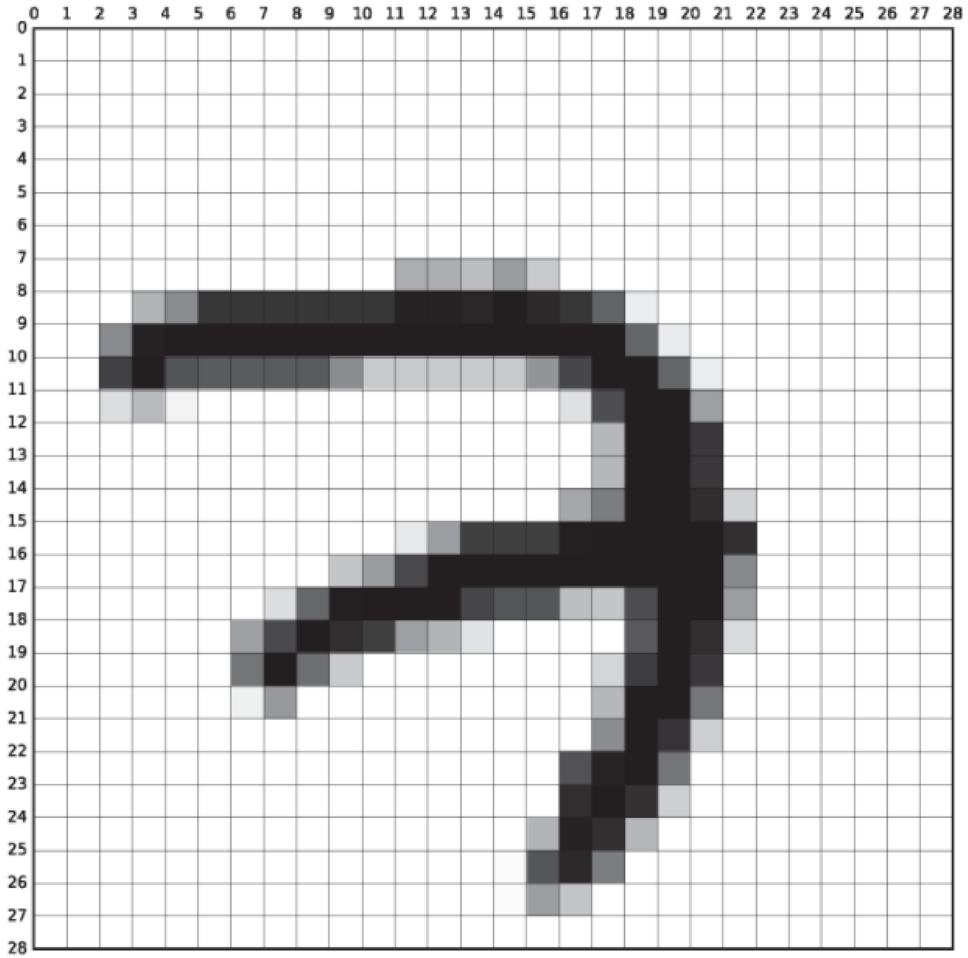


Application of NNs: Image Classification

- **Pattern Recognition**: facial recognition, object detection, etc.
- **Anomaly Detection**: detect the unusual patterns
- **Time Series Prediction**: stock price, weather forecasting, etc.
- **Natural Language Processing**: text classification, Speech Recognition

References:: https://www.datacamp.com/community/tutorials/neural-network-models-r

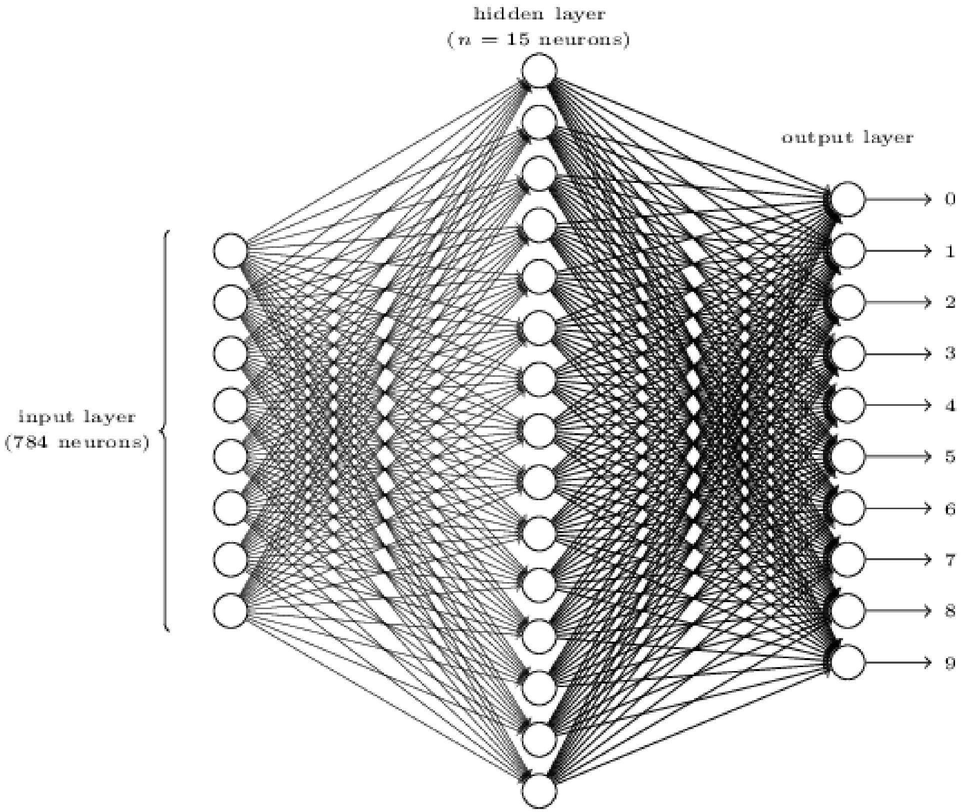
Application of NNs: Image Classification



- Grey scale image (one channel)
- Size 28×28 (pixels)
- Each pixel has a value of 0~255 representing brightness intensity

References: http://neuralnetworksanddeeplearning.com/chap1.html

Application of NNs: Image Classification

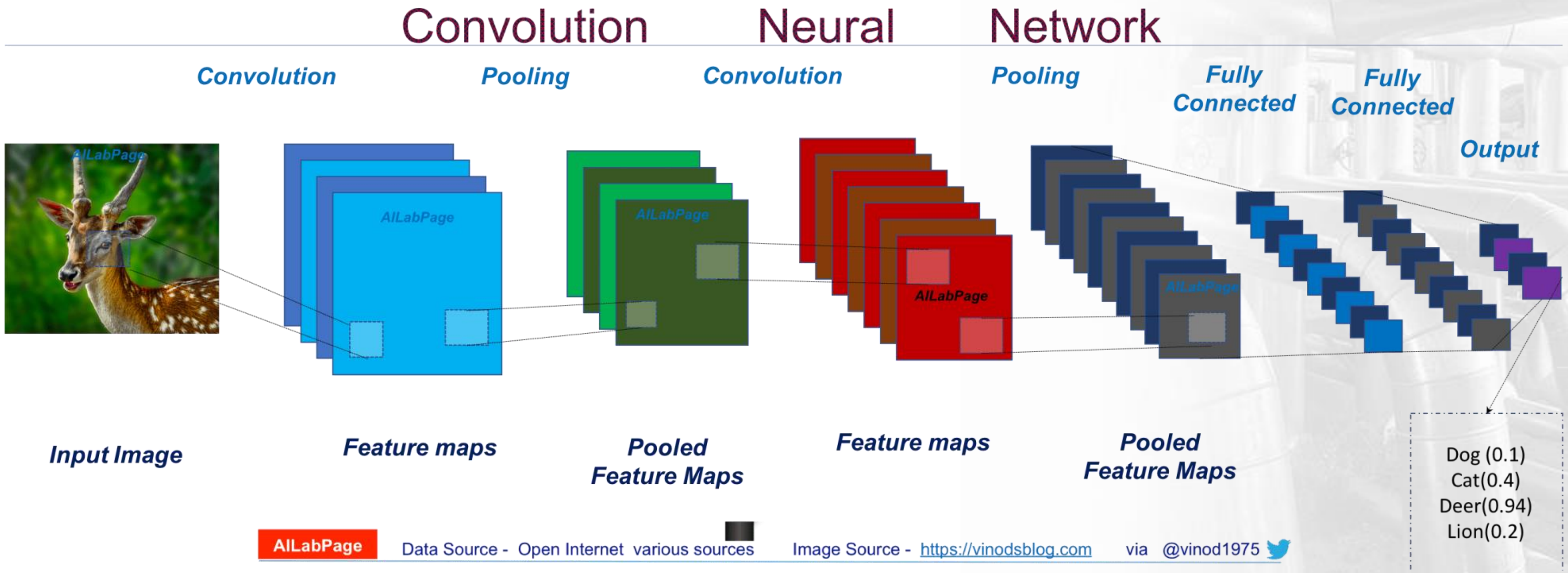


- Flatten to a 784-dim (28×28) vector row by row or column by column as input (features)
- Example: 784-neuron input layer + one hidden layer + output layer of 10 nodes (each for one digit)
- Number of weights: about 12K

References: http://neuralnetworksanddeeplearning.com/chap1.html

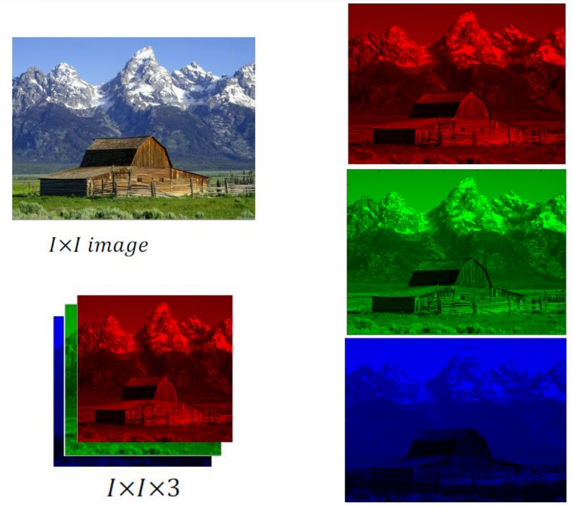
CNN

- CNNs are neural networks with **convolutional layers**
- CNNs are widely used for image classification and others



CNN

- Reasons:
  - Images are big. For example, $(224 \times 224 \times 3) \times 1024 = 150+$ million weights
  - Pixels and their neighbors form small, localized features
  - Positions can change
- CNN can help us mitigate those problems



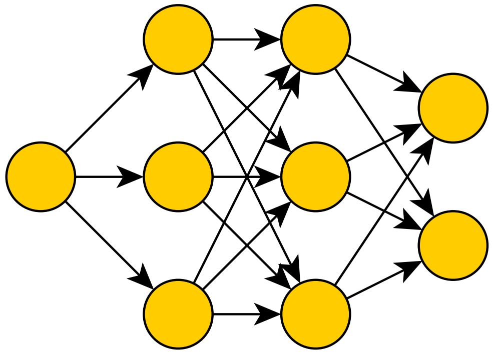
References: https://victorzhou.com/blog/intro-to-cnns-part-1/

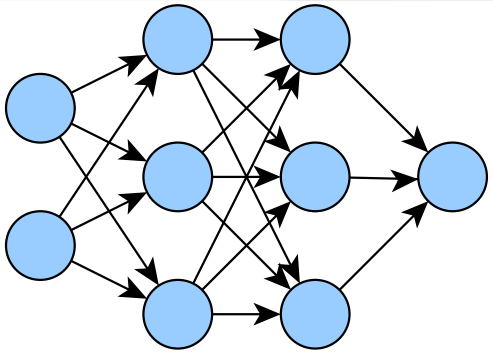
CNN: Self-learning materials

- CNNs, Part 1: An Introduction to Convolutional Neural Networks
- CNNs, Part 2: Training a Convolutional Neural Network
- **Image Classification Using Convolutional Neural Networks: A step by step guide**
- Simple explanation of convolutional neural network

GAN

- Generative
  - A generative model
- Adversarial
  - Trained in an adversarial setting
- Network
  - Use deep neural networks

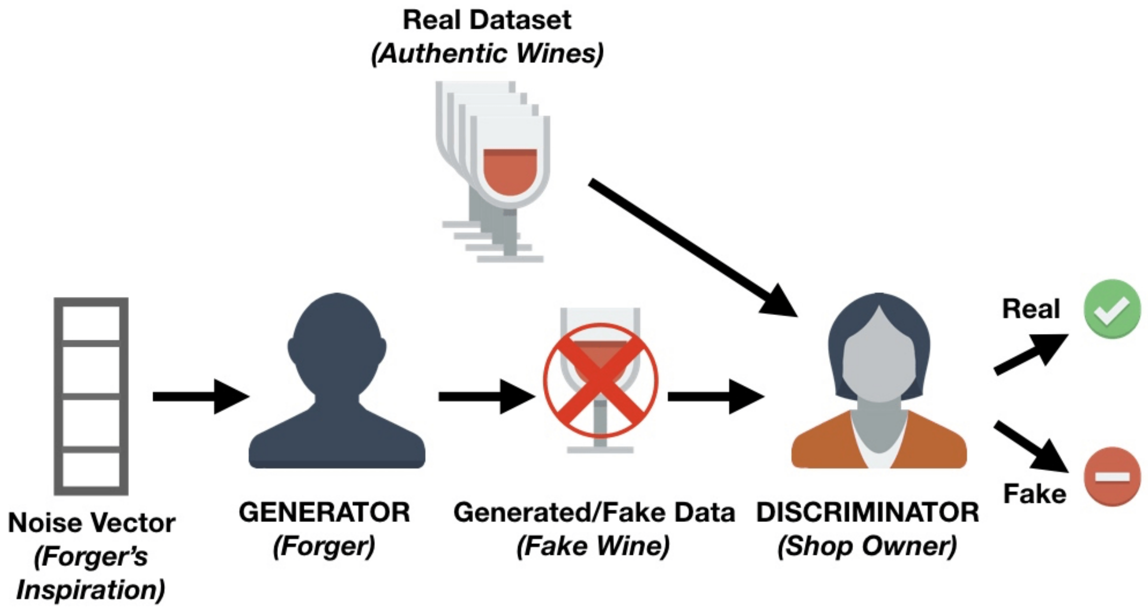




- Generator: generate fake samples, tries to fool the Discriminator
- Discriminator: tries to distinguish between real and fake samples

References: https://www.datacamp.com/community/tutorials/generative-adversarial-networks

GAN



- Generator: generate fake samples, tries to fool the Discriminator
- Discriminator: tries to distinguish between real and fake samples

References: https://www.datacamp.com/community/tutorials/generative-adversarial-networks

GAN

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

References: https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

GAN: Self-learning materials

- A Friendly Introduction to Generative Adversarial Networks (GANs)
- An Introduction to Generative Adversarial Networks (GANs)
- Demystifying Generative Adversarial Nets (GANs)

Prediction Accuracy Metrics

- Mean Error (ME): $ME = \frac{\sum_{i=1}^{n} y_i - \hat{y}_i}{n}$
- Mean Absolute Error (MAE): $MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$
- Mean Squared Error (MSE): $MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$
- Root Mean Squared Error (RMSE): $RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
- Median Absolute Deviation (MAD): $MSE = median(|y_i - \hat{y}_i|)$
- Mean Squared Log Error (MSLE): $MSLE = \frac{\sum_{i=1}^{n} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}{n}$
- Mean Absolute Scaled Error (MASE): $MASE = \frac{\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\frac{1}{T-1}\sum_{t=2}^{T} |y_t - \hat{y}_{t-1}|}$
- Classification Accuracy: $Accuracy = \frac{\#\ of\ Correct\ predictions}{\#\ of\ predictions}$
- Harmonic Mean, F1 Score: $F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$, $Precision = \frac{TruePositive}{TruePositive + FalsePositive}$, $Recall = \frac{TruePositive}{TruePositive + FalseNegative}$
- Logarithmic Loss: Logarithmic Loss $= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$
- Coefficient of Determination, R^2: $R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$

Prediction Accuracy Metrics

Accuracy Evaluation in Regression

ME
MAE
MSE
RMSE
MAD
MSLE
MASE
R^2

Scale independent, applicable to different time series.

Accuracy Evaluation in Classification

Precision
Recall
F1 Score
Classification Accuracy
Confusion Matrix
Logarithmic Loss