Deep Learning

Prof. Douglas Kunda, PhD

Vice Chancellor – ZCAS University

vc@zcasu.edu.zm

Contents

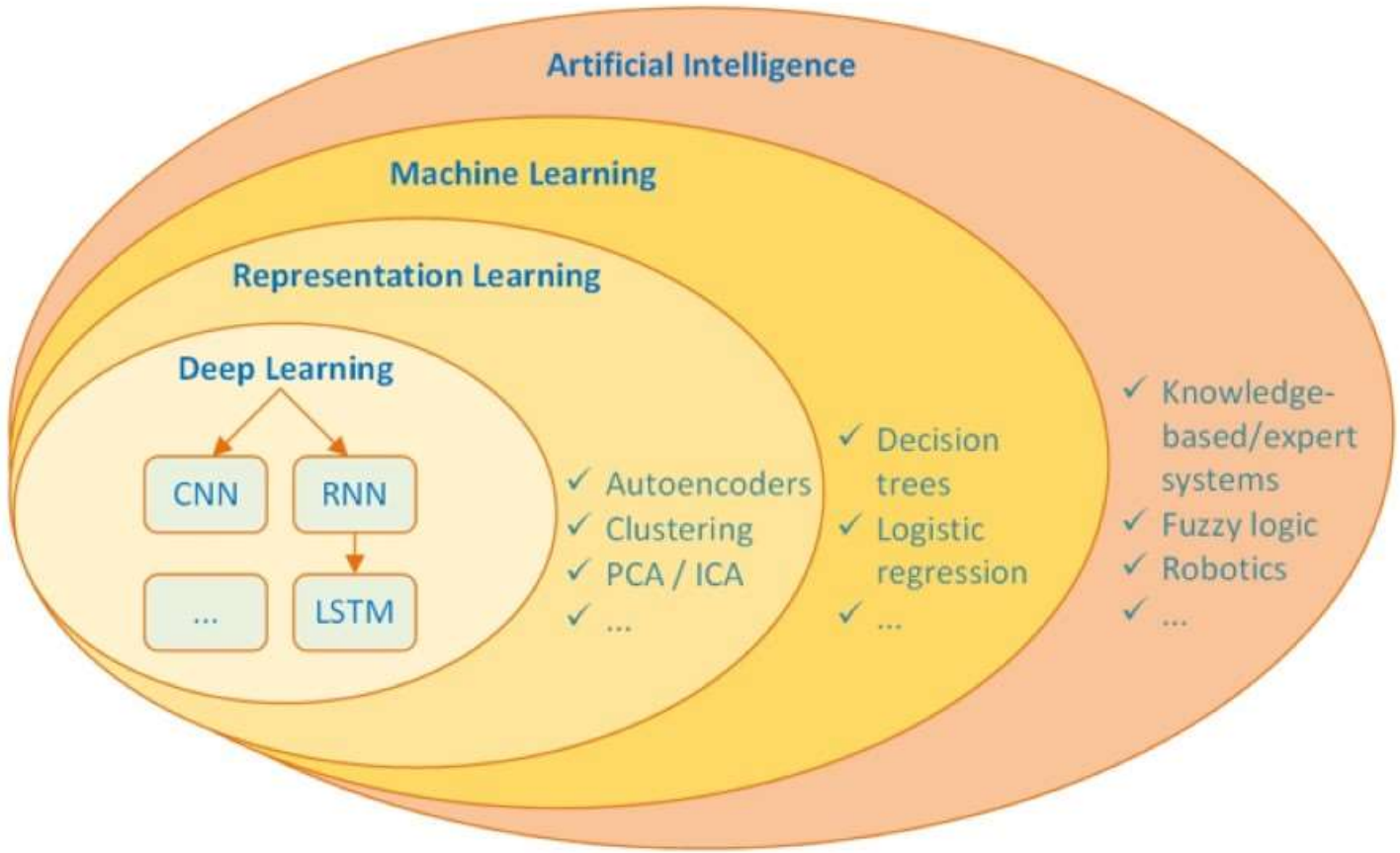
- Introduction to deep learning
  - Deep Learning ≅ Deep Neural Networks
- Foundational concepts of “shallow” neural networks
- Elements and learning mechanisms of neural networks
- Learn about the structure of deep neural networks
- CNN, RNN, and LSTM type deep neural network architectures
- Computing frameworks used for deep neural network implementations
- Cognitive computing and cognitive search
- Generalization and transfer learning

Introduction to Deep Learning

- Imaginative things in the Sci-Fi movies are becoming realities-thanks to AI and ML
- Siri, Google assistant, Alexa, Google home, …
- Self driving cars/drones, autonomous robots, …
- Deep learning is the newest member of the AI/ML family of advanced computations
- They learn better than ever before
- The reason for Deep Learning superiority
  - Depth of knowledge capture and representation
  - Automatic feature extraction

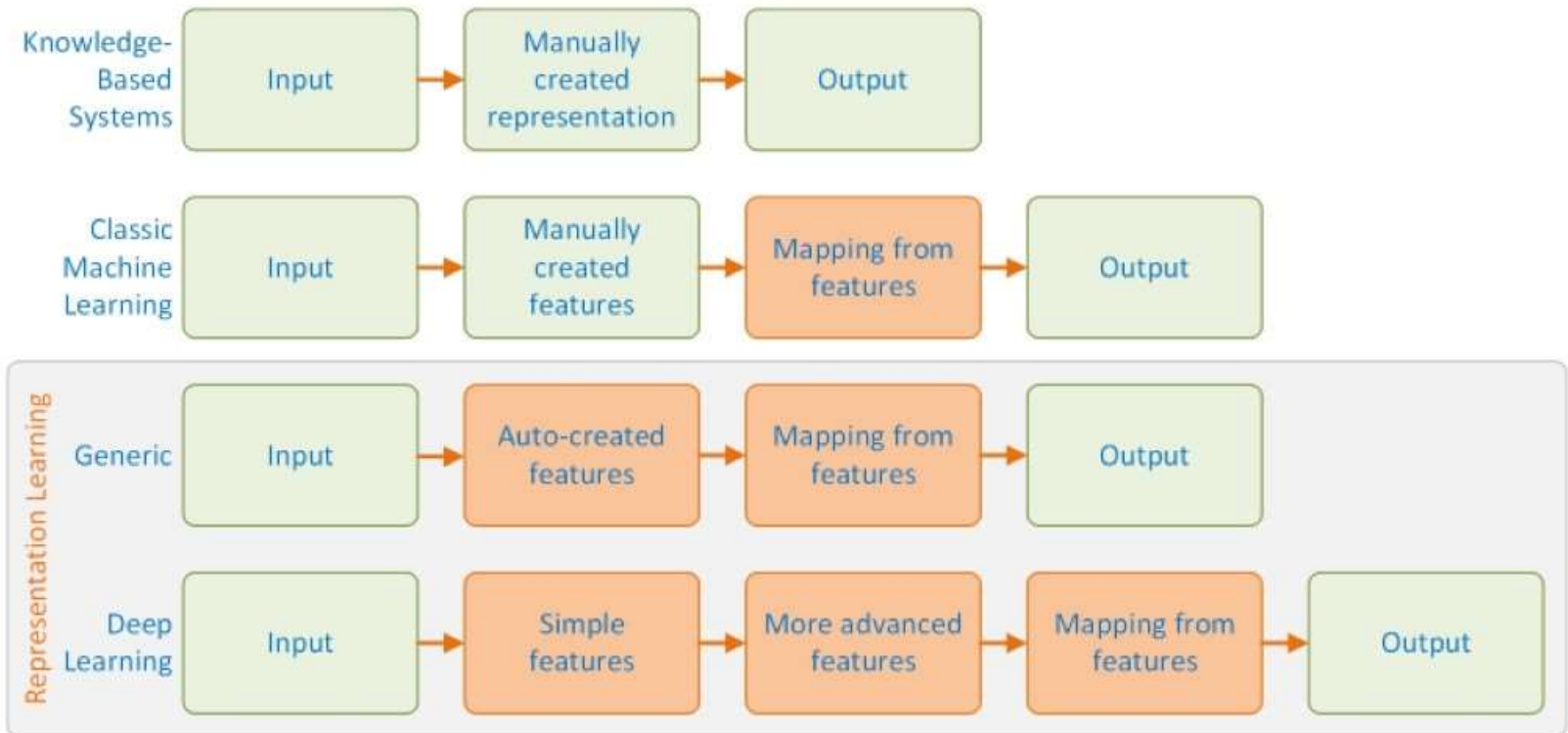
Introduction to Deep Learning

- The placement of Deep Learning within the overarching AI-based learning methods



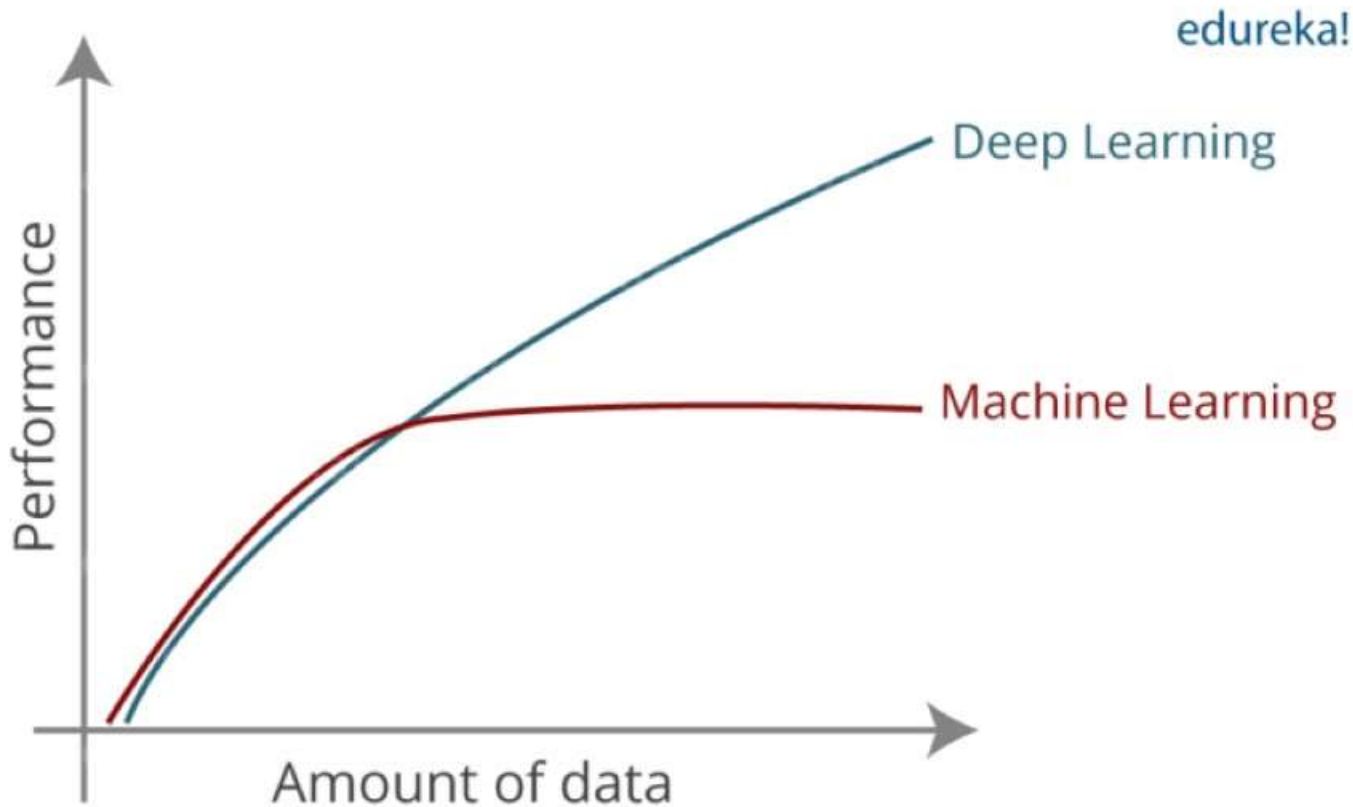
Introduction to Deep Learning

- Differences between Classic Machine-Learning Methods and Representation Learning/Deep Learning



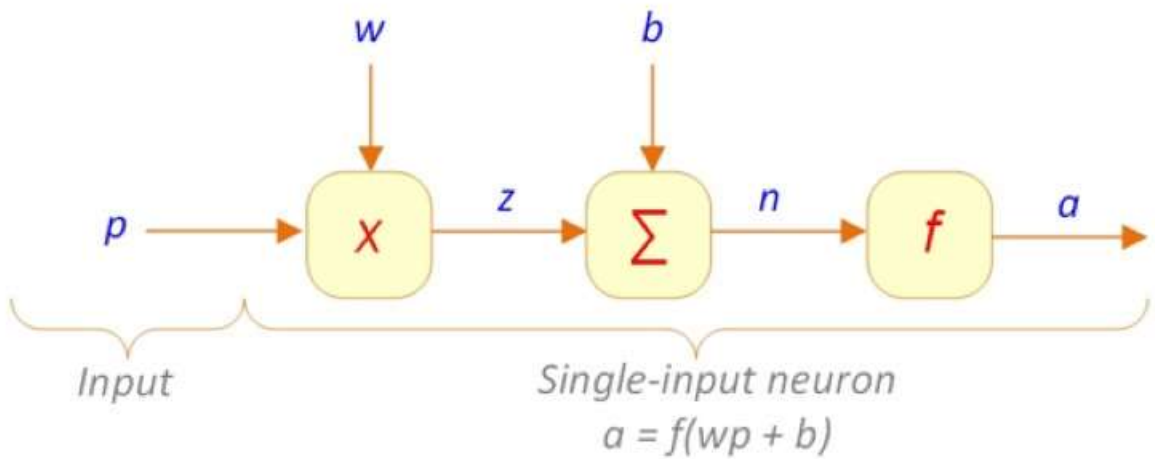
Introduction to Deep Learning

- Machine Learning Algorithms cannot handle High-Dimensional Data – where we have a large number of inputs and outputs: round thousands of dimensions. Handling and processing such type of data becomes very complex and resource exhaustive and it is called Curse of Dimensionality



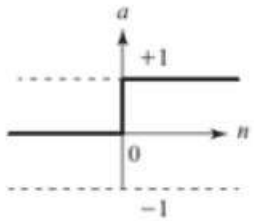
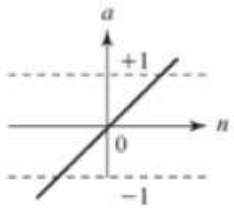
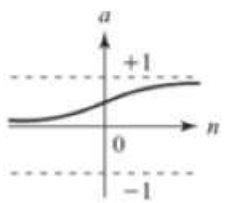
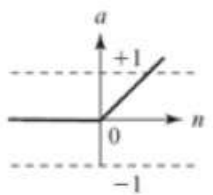
Basics of "Shallow" Learning

- Artificial Neural Networks – abstractions of human brain and its complex biological network of neurons
- Neurons = Processing Elements (PEs)
- Single-input and single-output neuron/PE



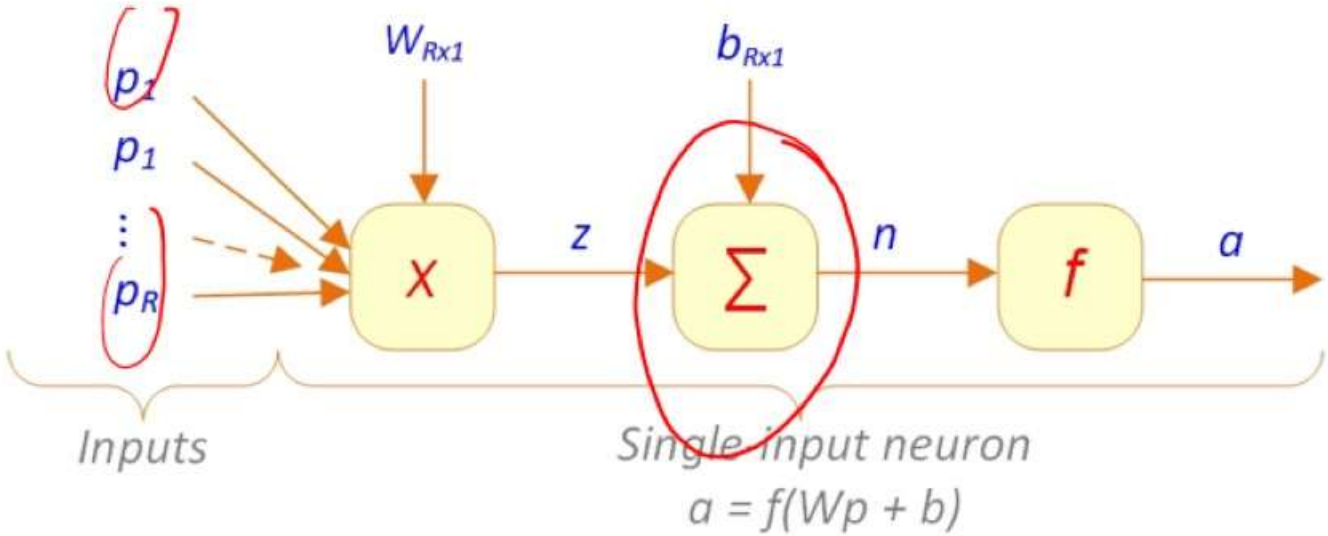
Basics of "Shallow" Learning

- Common transfer (activation) functions

Transfer Function	Form	Operation
Hard limit	 $a = hardlim(n)$	$a = +1$ *if* $n > 0$ $a = 0$ *if* $n < 0$
Linear	 $a = purelin(n)$	$a = n$
Log-Sigmoid	 $a = logsig(n)$	$a = \frac{1}{1 + e^{-n}}$
Positive linear (a.k.a. rectified linear or ReLU)	 $a = poslin(n)$	$a = n$ *if* $n > 0$ $a = 0$ *if* $n < 0$

Basics of "Shallow" Learning

- Typical multiple-input neuron with *R* individual inputs

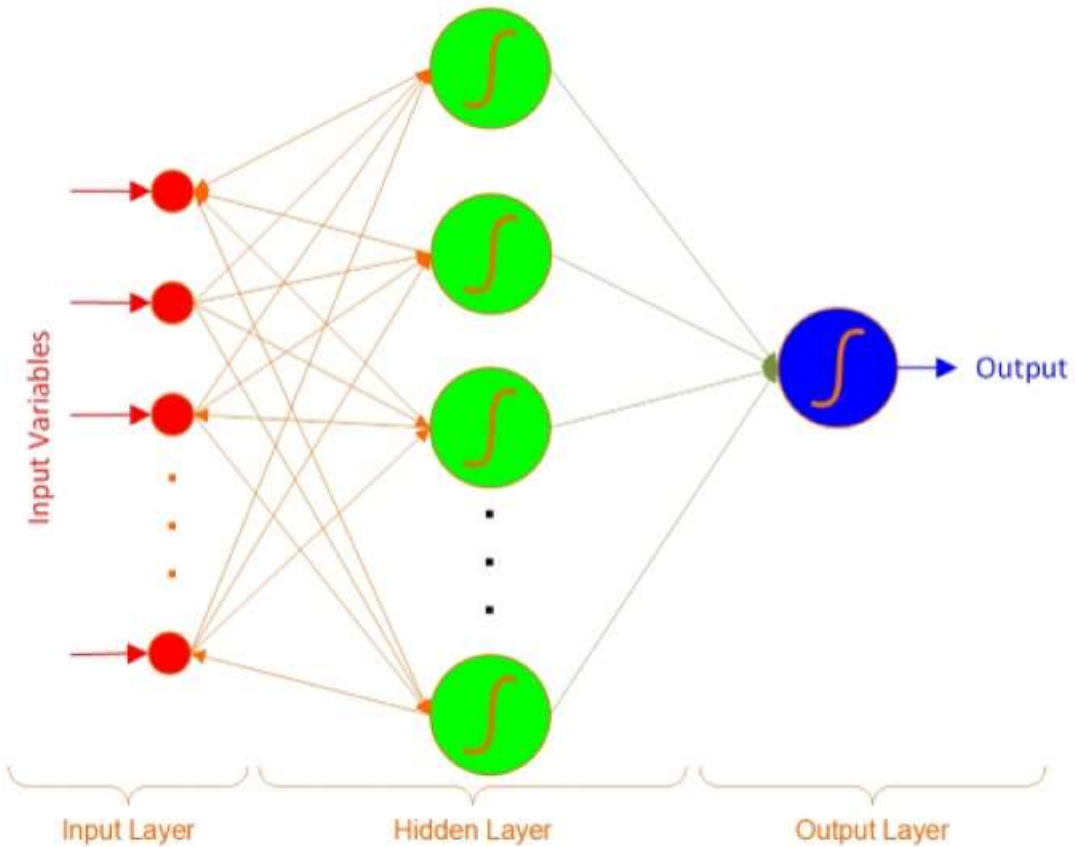


$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,3}p_3 + \ldots + w_{1,R}p_R + b$$

$$n = \mathbf{W}\mathbf{p} + b$$

Basics of "Shallow" Learning

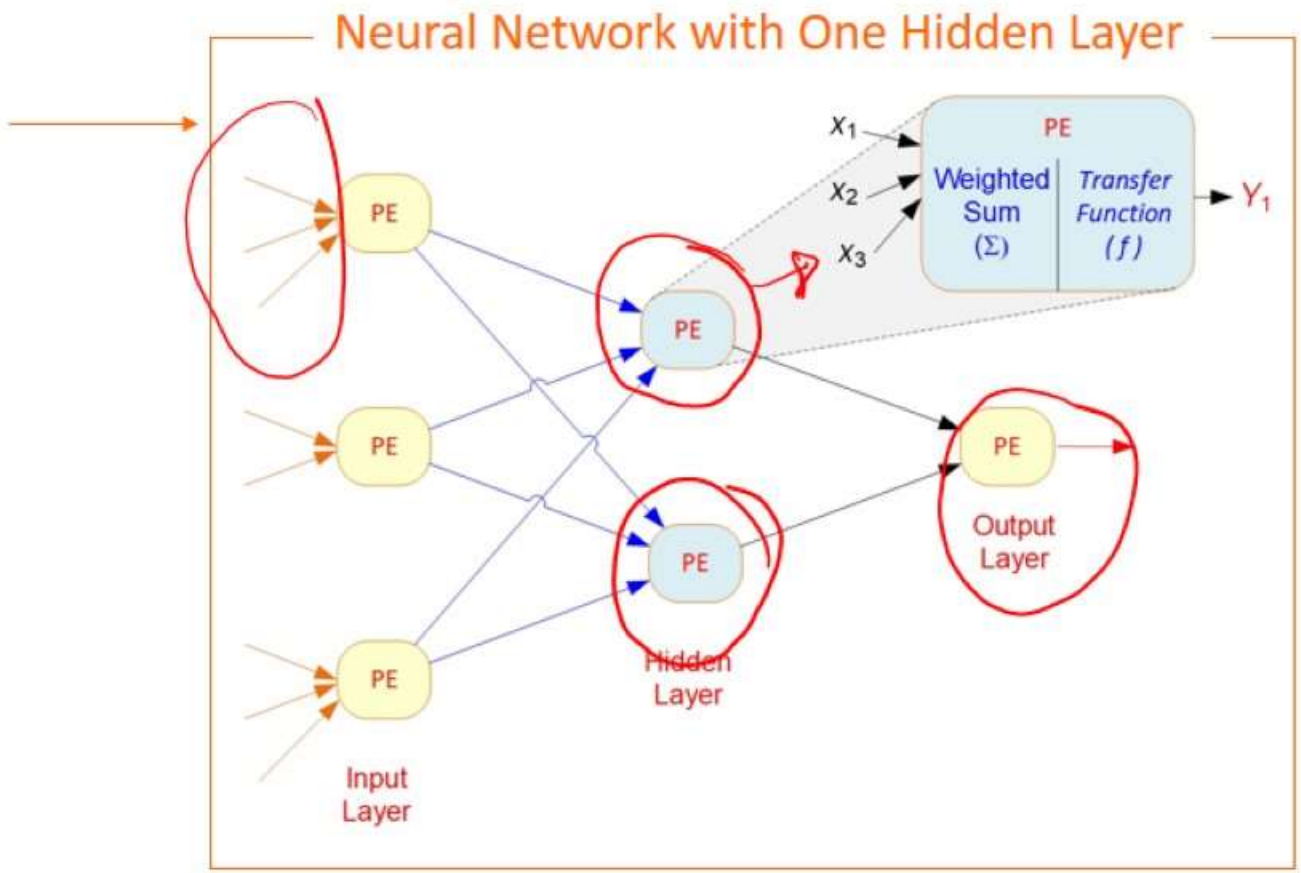
- Typical Neural Network with three layers



Elements of an Artificial Neural Network

Elements of ANN

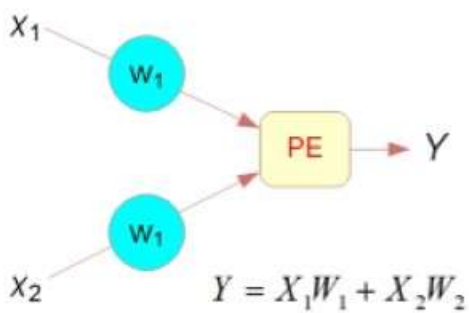
- Processing element (PE)
- Network structure
  - Hidden layer(s)
- Input
- Output
- Connection weights
- Summation function
- Transfer function



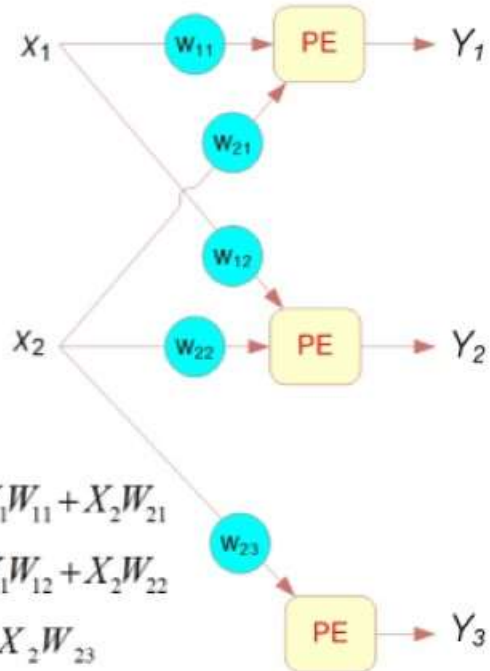
Elements of an Artificial Neural Network

Summation Function

(a) Single neuron



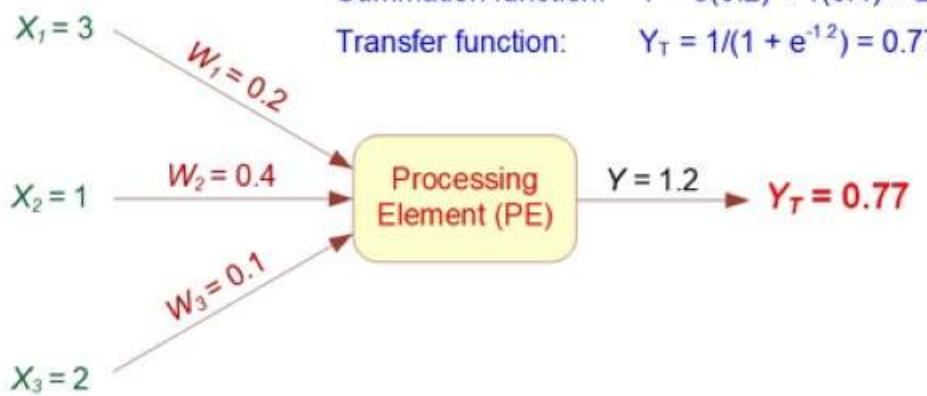
(b) Multiple neurons



Transfer Function

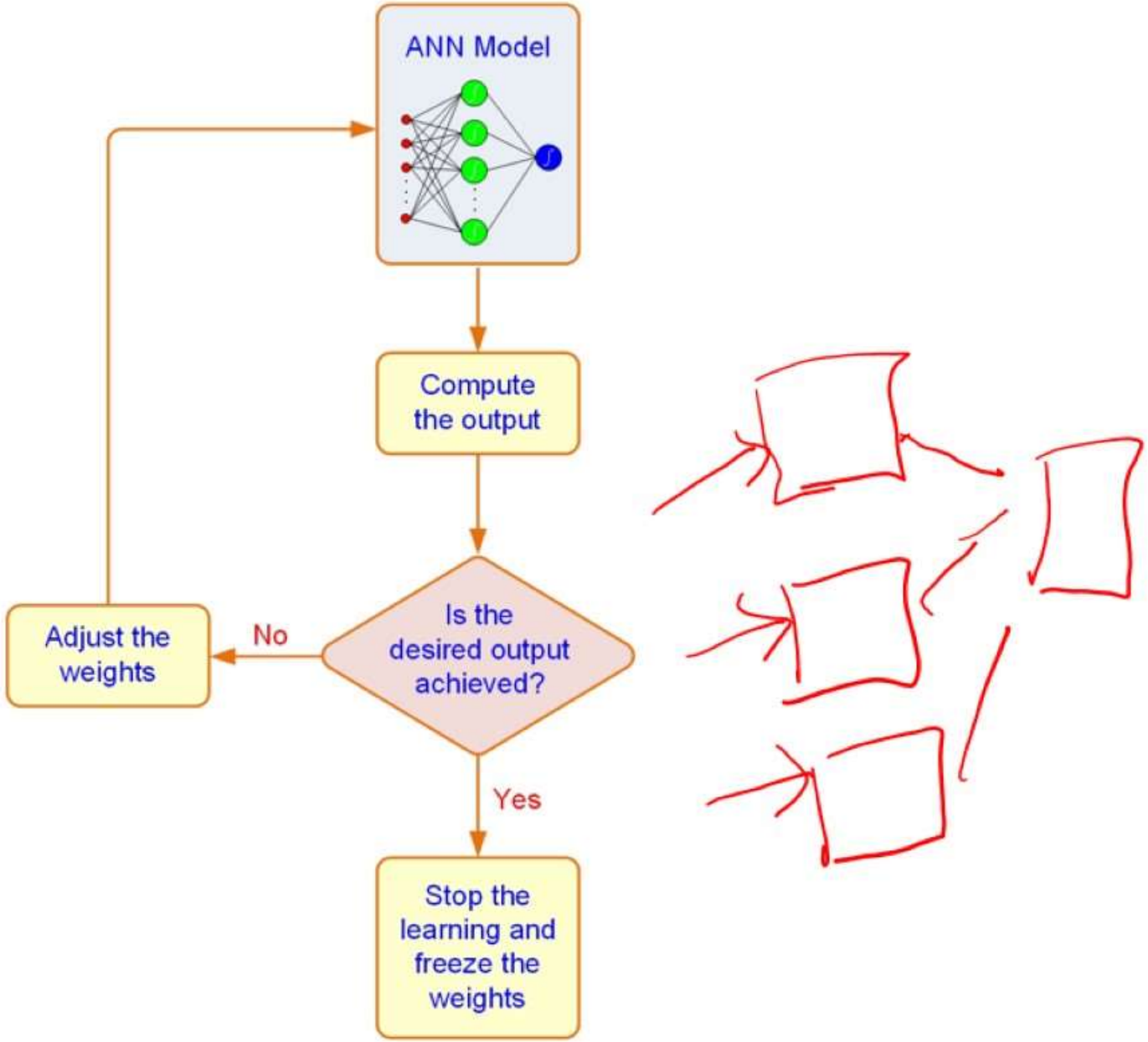
Summation function: Y = 3(0.2) + 1(0.4) + 2(0.1) = 1.2

Transfer function: $Y_T = 1/(1 + e^{-1.2}) = 0.77$



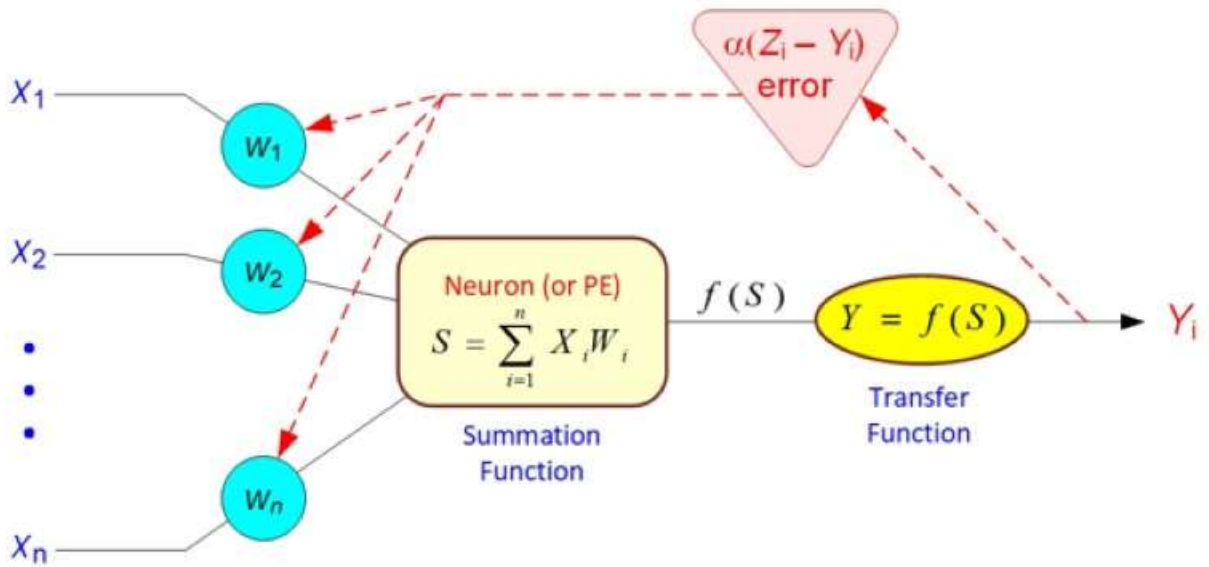
Learning Process in ANN

1. Compute temporary outputs
2. Compare outputs with desired targets
3. Adjust the weights and repeat the process



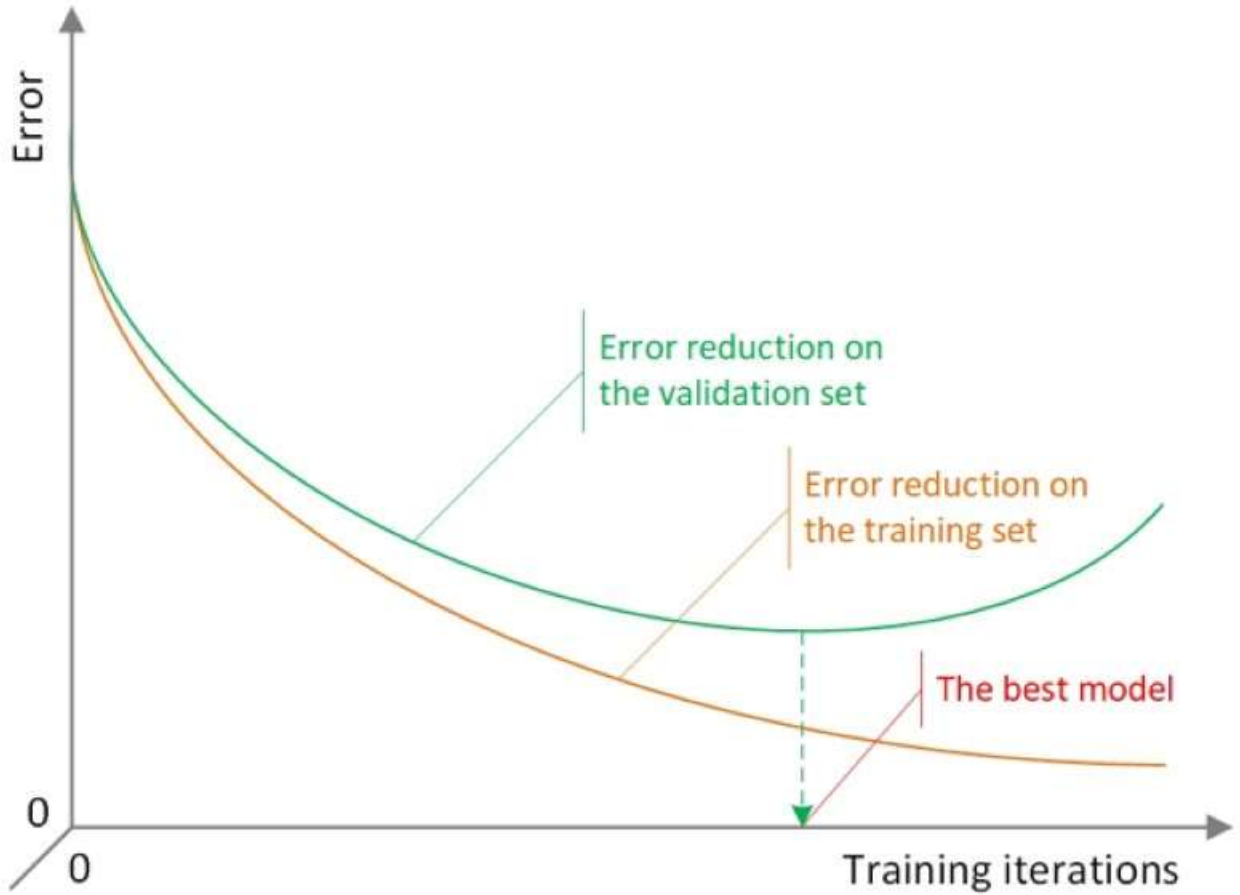
Backpropagation for ANN Training

1. Initialize weights with random values
2. Read in the input vector and the desired output
3. Compute the actual output via the calculations
4. Compute the error.
5. Change the weights by working backward



Backpropagation for ANN Training

- Illustration of the Overfitting in ANN



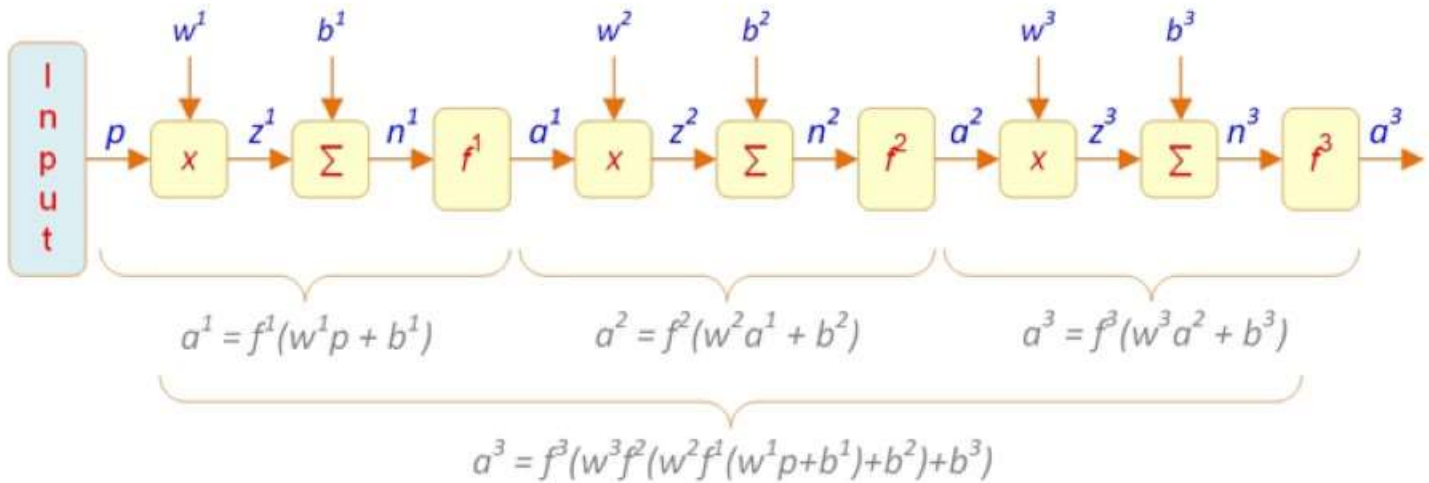
Deep Neural Networks

- Deep: more hidden layers
  - More ability to capture more specific patterns
- In addition to CPU, it also uses GPU
  - With programming languages like CUDA by NVIDIA
- Needs very large datasets for training
- Deep learning uses tensors as inputs
  - Tensor: N-dimensional arrays
  - Image representation with 3-D tensors
- There are different types and capabilities of Deep Neural Networks for different tasks

Deep Neural Networks

Feedforward Multilayer Perceptron (MLP)-Type Deep Networks

- Most common type of deep neural networks
- Vector Representation of the First Three Layers in a Typical MLP Network.

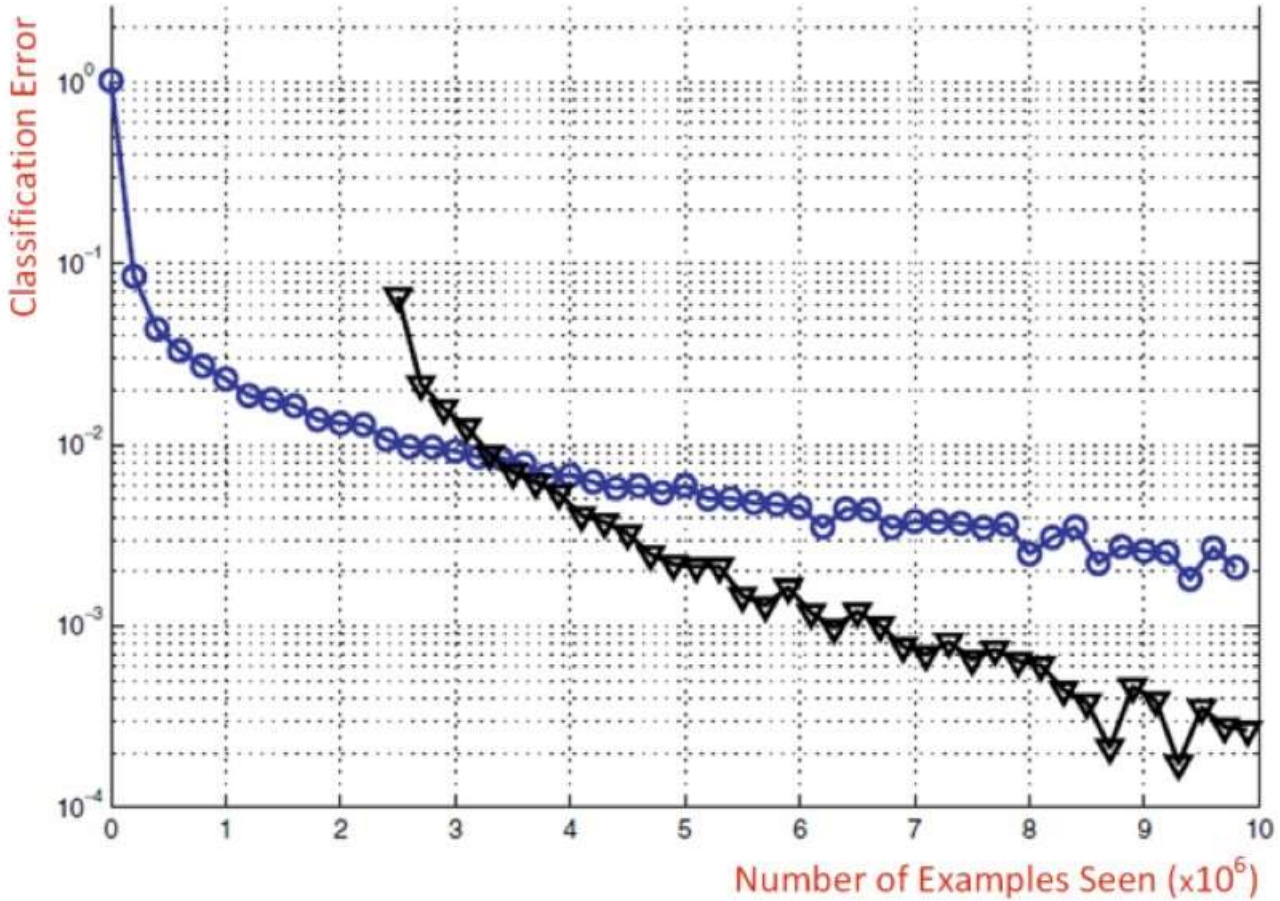


Deep Neural Networks

- Impact of Random Weights in Deep MLP

- The effect of pre-training network parameters on improving results of a classification-type Deep Neural Network

- More hidden layers versus more neurons?

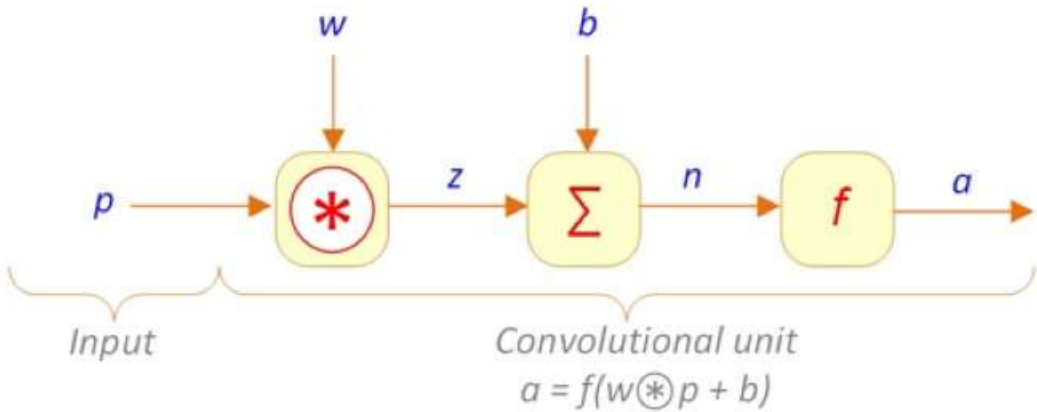


Convolutional "Deep" Neural Networks

- Most popular MLP-base DL method
- Used for image/video processing, text recognition
- Has at least one convolution weight function
  - Convolutional layer
- Convolutional layer → Polling (sub-sampling)
  - Consolidating the large tensors into one with a smaller size-and reducing the number of model parameters while keeping only the important features
  - There can be different types of polling layers

Convolution Function

- Typical Convolutional Network Unit



- Convolution of a 2 x 2 Kernel by a 3 x 6 Input Matrix

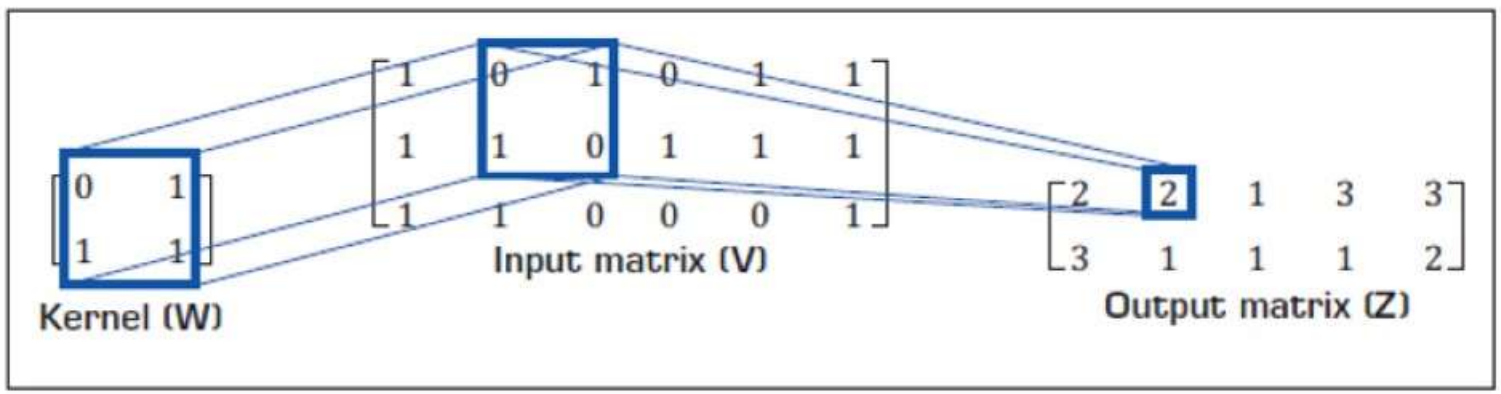


Image Processing Using CNN

- ImageNet (http://www.image-net.org)
- Architecture of AlexNet, a CNN for Image Classification

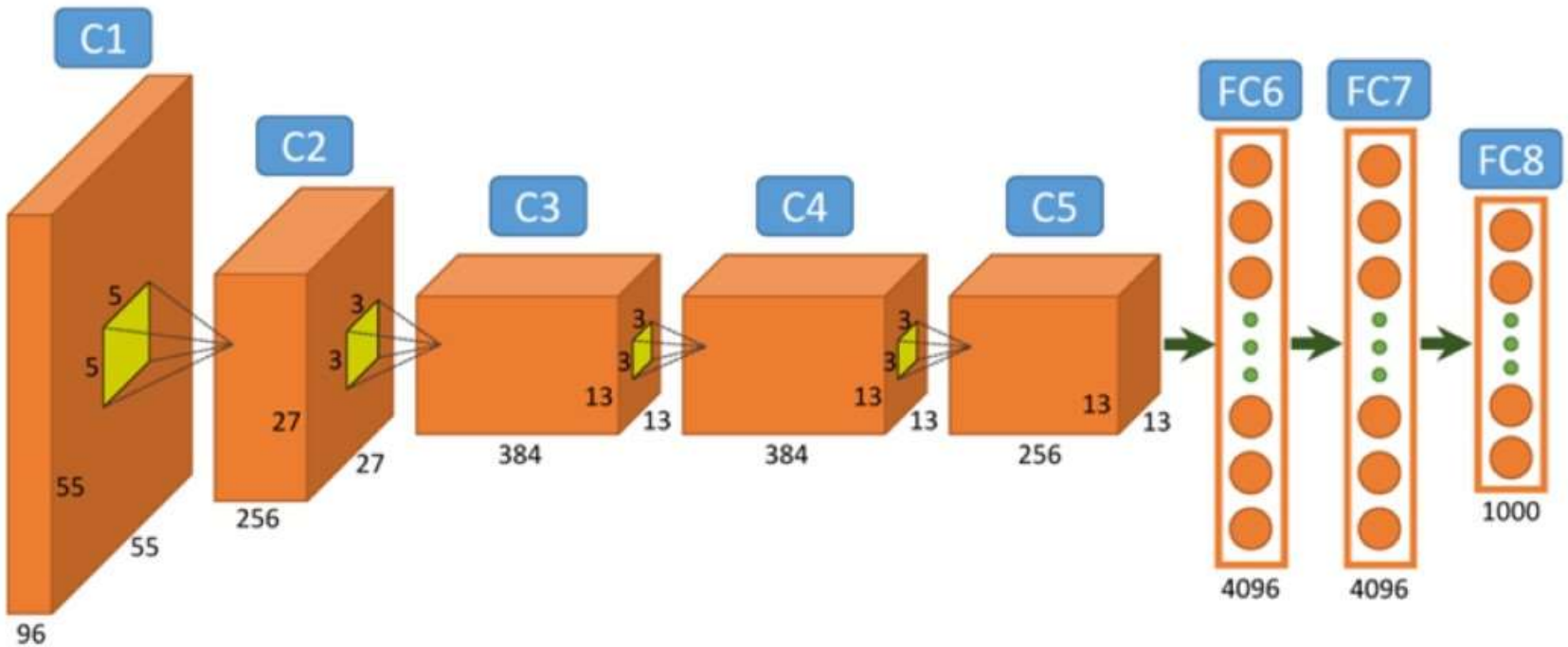


Image Processing Using CNN

- Conceptual Representation of the Inception Feature in GoogLeNet

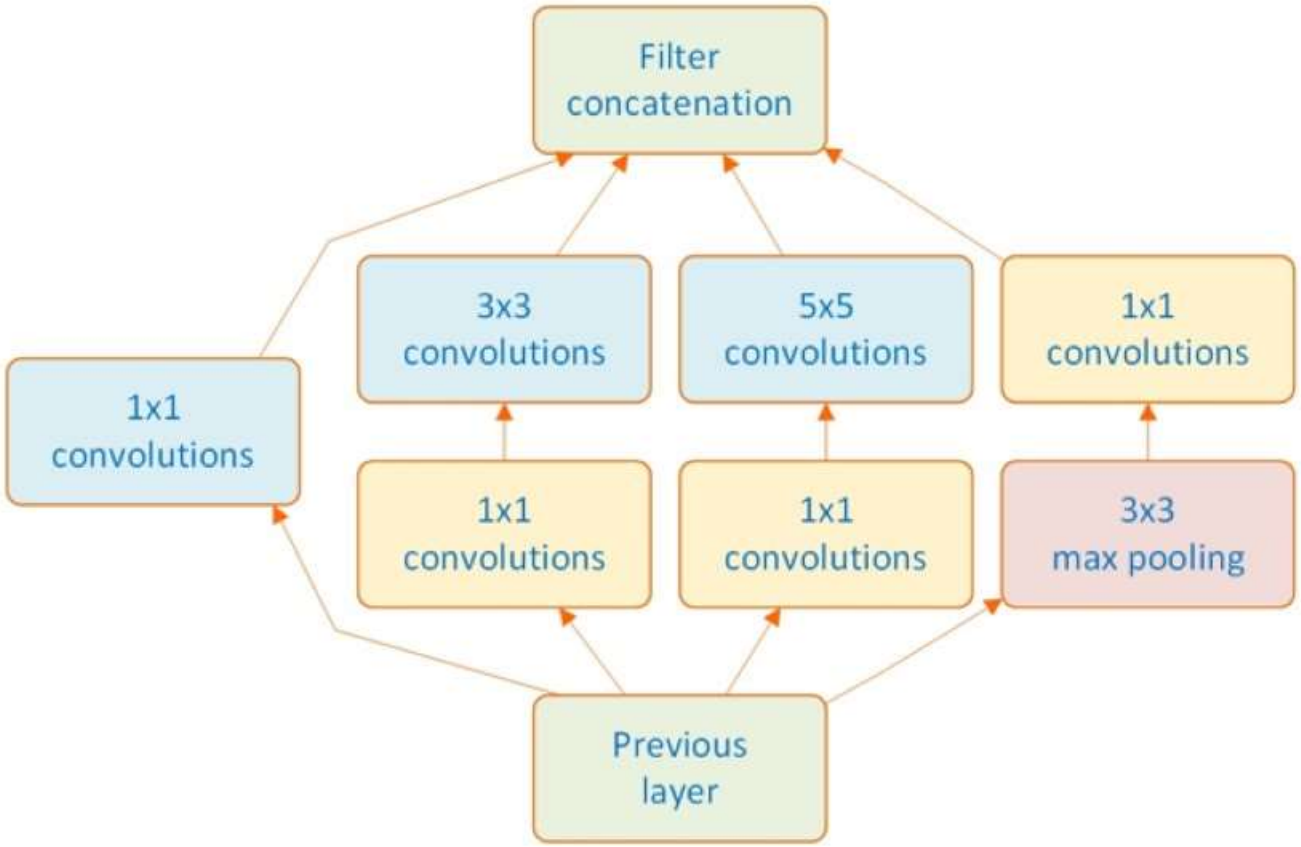
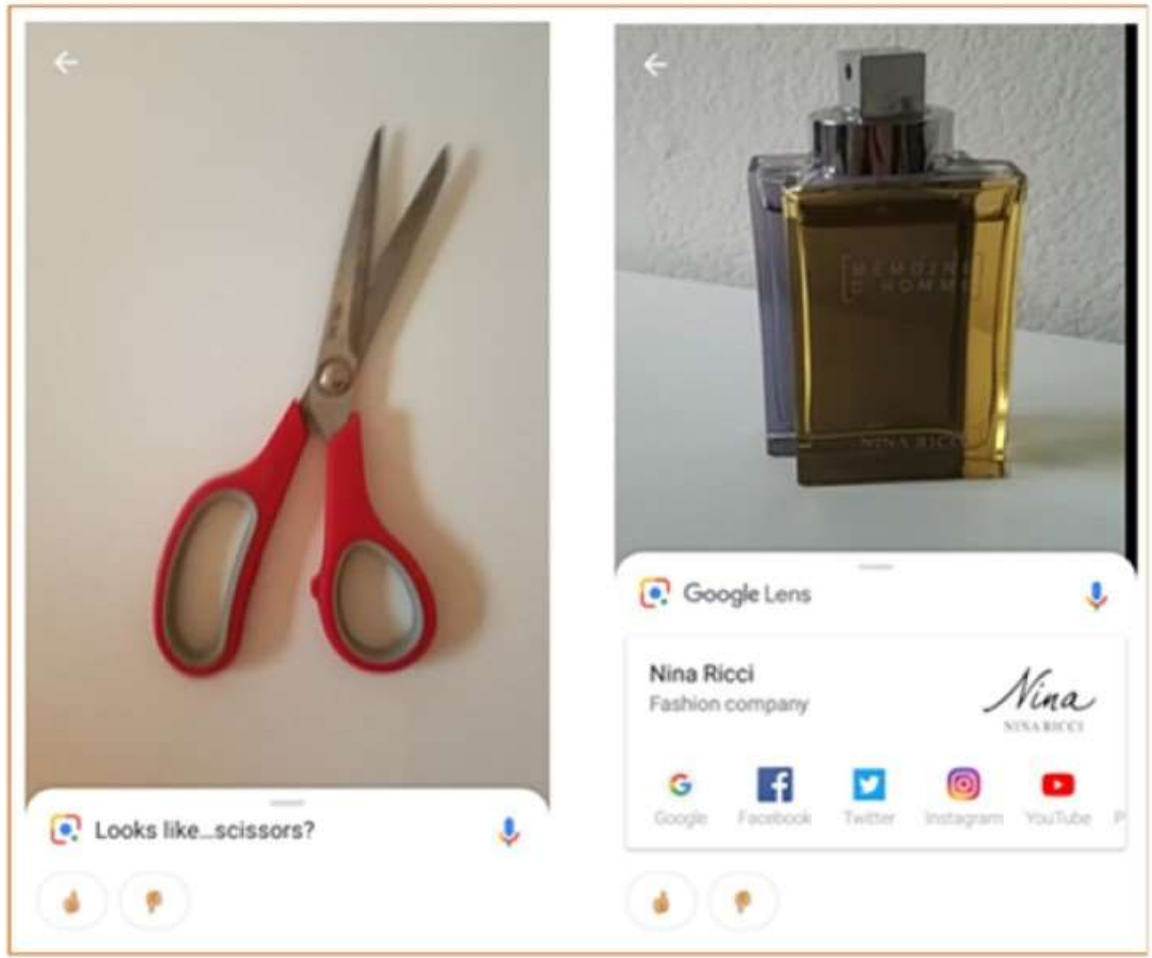


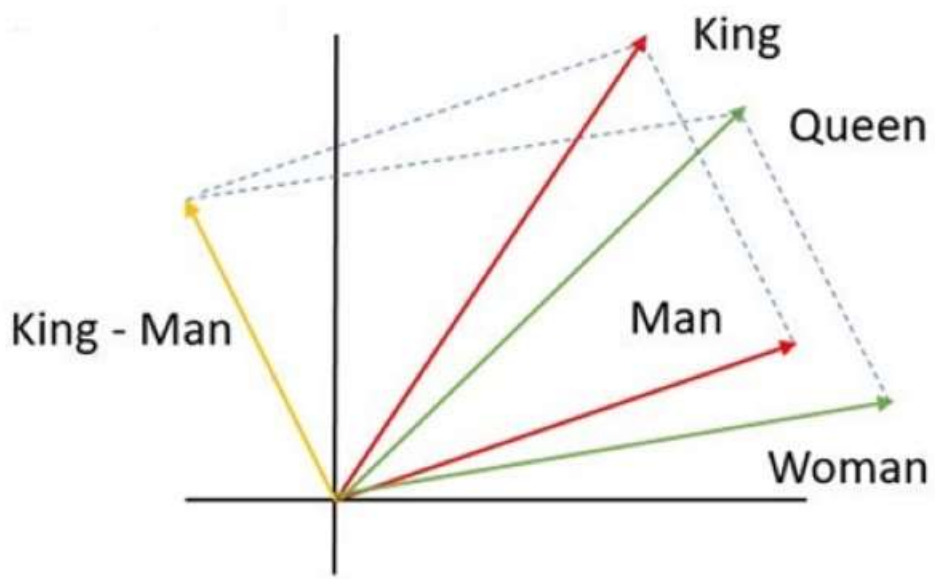
Image Processing Using CNN

- Examples of Using the Google Lens



Text Processing Using CNN

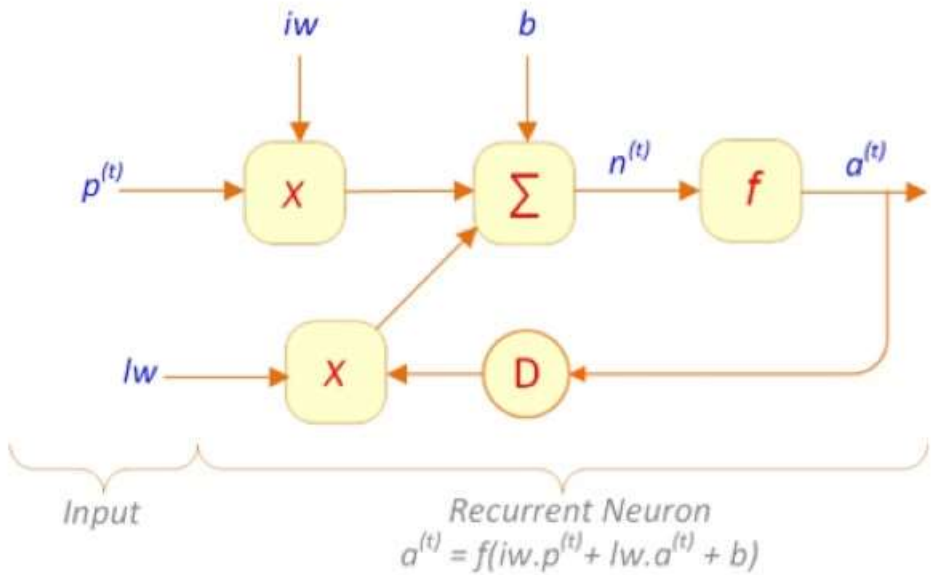
- Google word2vec project
  - Word embeddings
- Typical Vector Representation of Word Embeddings in a Two-Dimensional Space



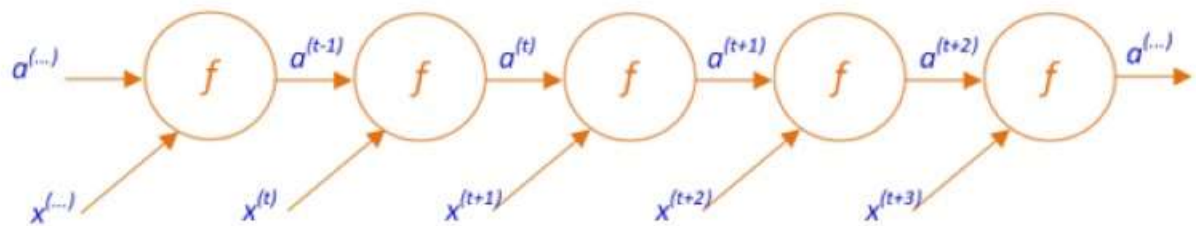
Recurrent Neural Networks (RNN) & Long Short-Term Memory (LSTM)

- RNN designed to process sequential inputs

Typical recurrent unit



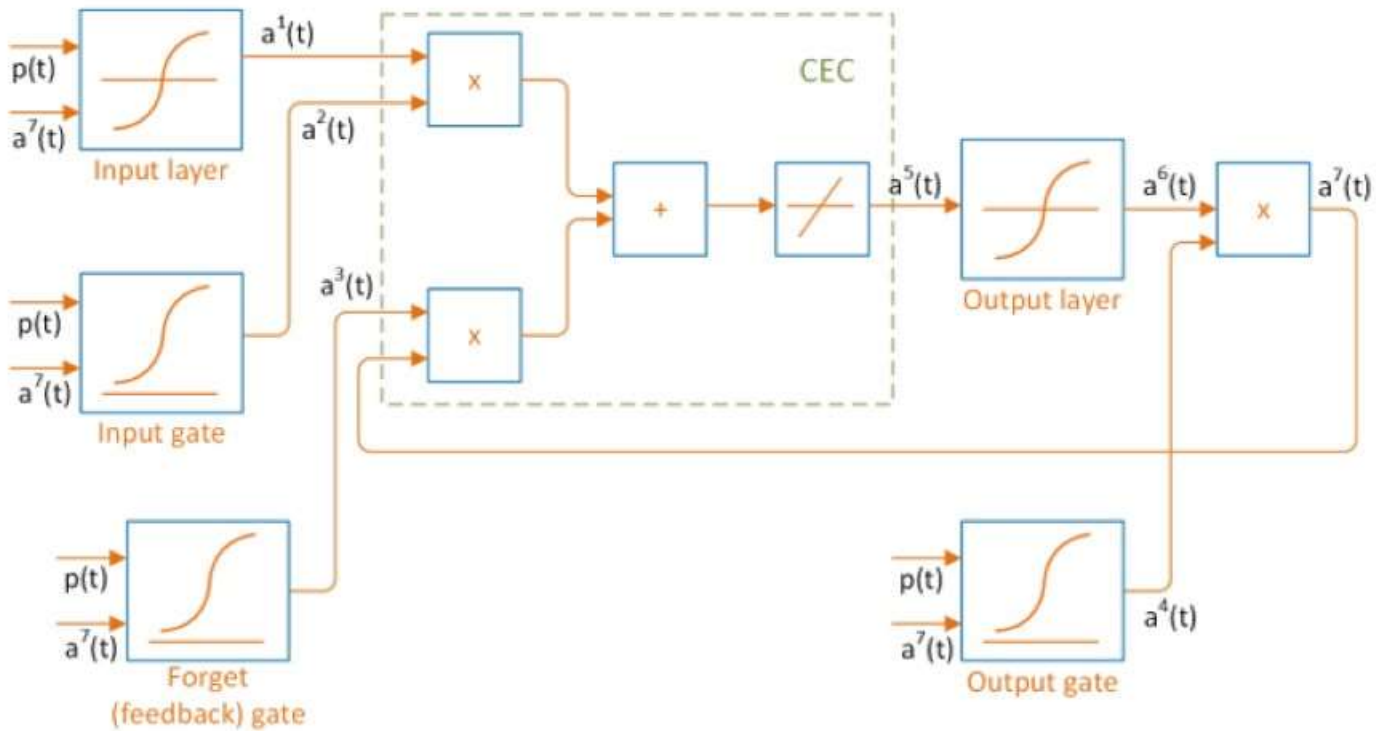
Unfolded view of a typical recurrent unit



Recurrent Neural Networks (RNN) & Long Short-Term Memory (LSTM)

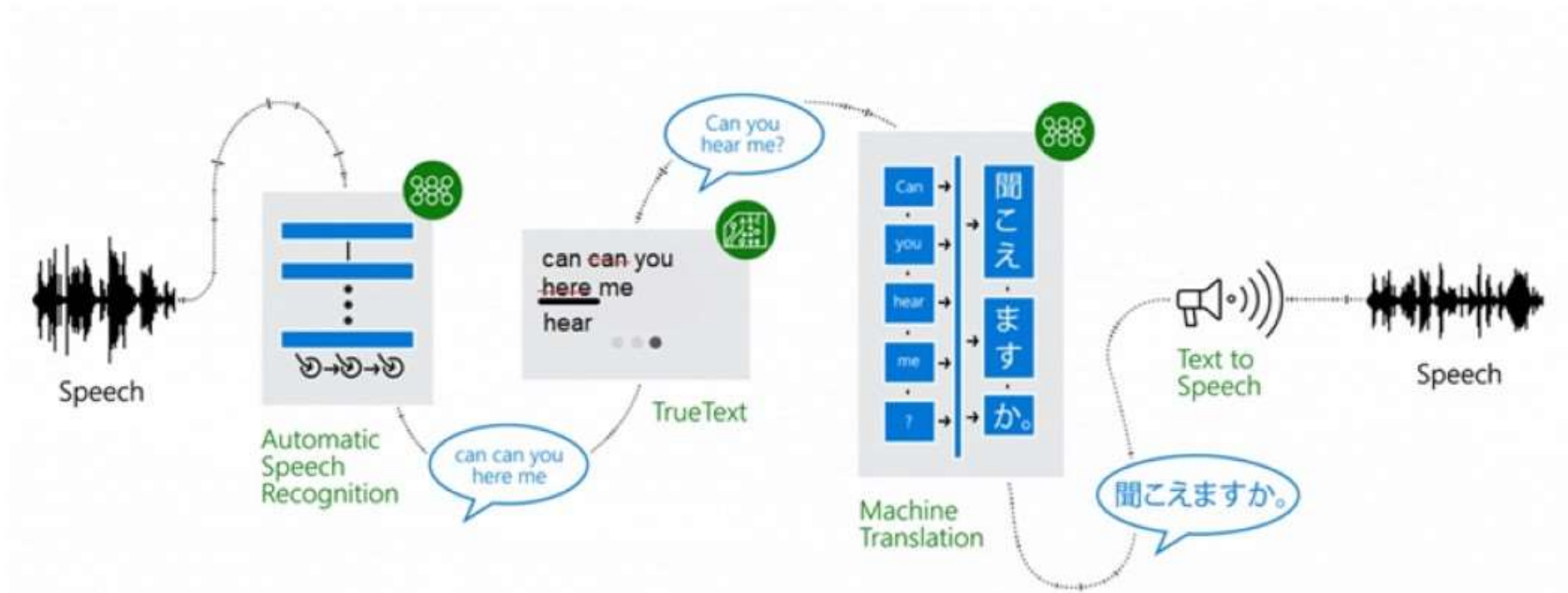
- LSTM is a variant of RNN
  - In a dynamic network, the weights are called the *long-term memory* while the feedbacks role is the *short-term memory*

Typical Long Short-Term Memory (LSTM) Network Architecture



Recurrent Neural Networks (RNN) & Long Short-Term Memory (LSTM)

- Four-Step Process of Translating Speech Using Deep Networks in Microsoft Skype Translator



Computer Frameworks for Implementation of Deep Learning

- Torch (www.torch.ch)
  - ML with GPU
- Caffe (caffe.berkeleyvision.org)
  - Facebook's improved version (www.caffe2.ai)
- TensorFlow (www.tensorflow.org)
  - Google - Tensor Processing Units (TPUs)
- Theano (deeplearning.net/software/theano)
  - Deep Learning Group at the University of Montreal
- Keras (keras.io)
  - Application Programming Interface

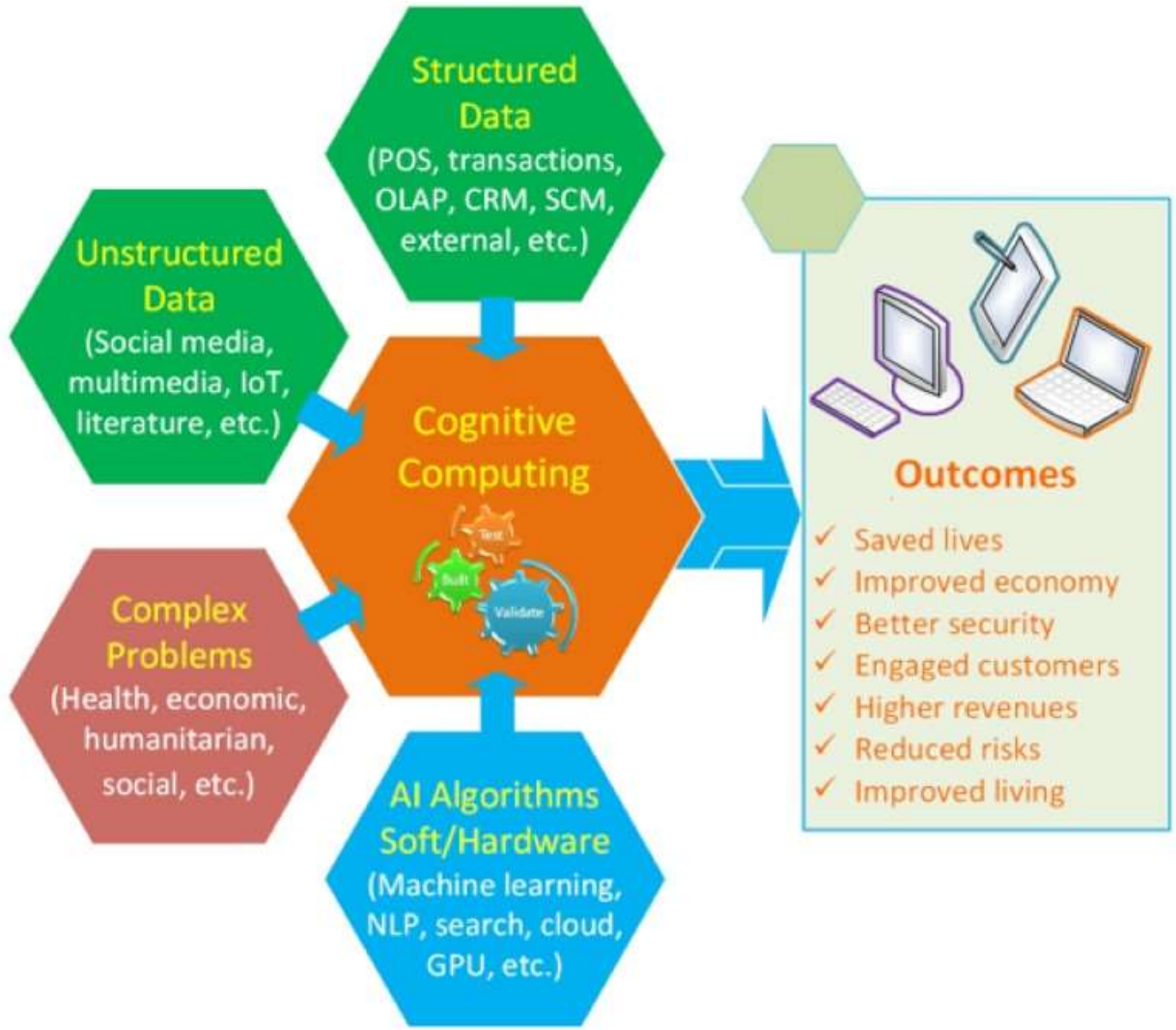
Cognitive Computing

- Systems that use mathematical models to emulate (or partially simulate) the human cognition process to find solutions to complex problems and situations where the potential answers can be imprecise
- IBM Watson on *Jeopardy!*
- How does cognitive computing work?
  - Adaptive
  - Interactive
  - Iterative and stateful
  - Contextual

✓ Data mining,
✓ Pattern recognition,
✓ Deep learning, and
✓ NLP

Mimic the way the human brain works

A Conceptual Framework for Cognitive Computing and Its Promises



Cognitive Computing

- How does cognitive computing differ from AI?

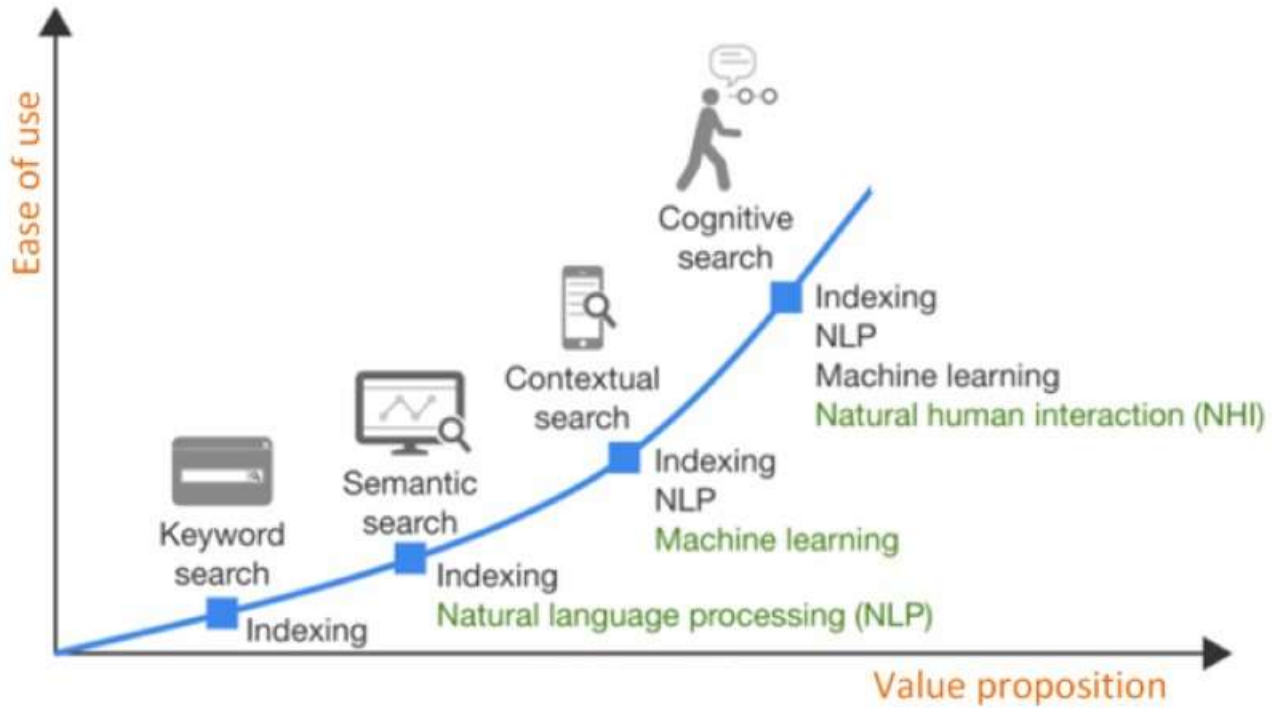
Characteristic	Cognitive Computing	Artificial Intelligence (AI)
Technologies used	• Machine learning • Natural language processing • Neural networks • Deep learning • Text mining • Sentiment analysis	• Machine learning • Natural language processing • Neural networks • Deep learning
Capabilities offered	Simulate human thought processes to assist humans in finding solutions to complex problems	Find hidden patterns in a variety of data sources to identify problems and provide potential solutions
Purpose	Augment human capability	Automate complex processes by acting like a human in certain situations
Industries	Customer service, marketing, healthcare, entertainment, service sector	Manufacturing, finance, healthcare, banking, securities, retail, government

Cognitive Computing

- Cognitive computing use cases
  - Development of smart and adaptive search engines
  - Effective use of natural language processing
  - Speech recognition
  - Language translation
  - Context-based sentiment analysis
  - Face recognition and facial emotion detection
  - Risk assessment and mitigation
  - Fraud detection and mitigation
  - Behavioral assessment and recommendations, …
- Cognitive analytics?...

Cognitive Search

- Can handle a variety of data types
- Can contextualize the search space
- Employ advanced AI technologies
- Enables to build enterprise-specific search applications

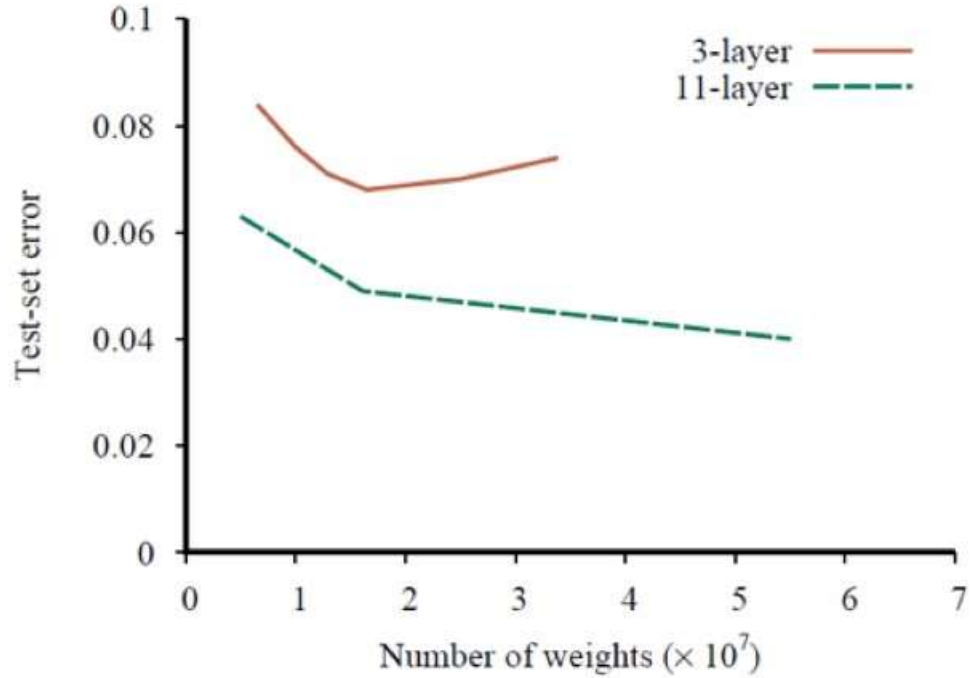


Generalization

Choosing a network architecture

- Some neural network architectures are explicitly designed to generalize well on particular types of data
- When comparing two networks with similar numbers of weights, the deeper network usually gives better generalization performance.
- Deep learning systems perform better than any other pure machine learning approaches for high dimensional inputs (images, video, speech signals, etc)
- Deep learning models lack the compositional and quantificational expressive power
- May also produce unintuitive errors. Tend to produce input–output mappings that are discontinuous

Generalization



Test-set error as a function of layer width (as measured by total number of weights) for three-layer and eleven-layer convolutional networks. The data come from early versions of Google's system for transcribing addresses in photos taken by Street View cars

Generalization

Neural architecture search

- neural architecture search to explore the state space of possible network architectures.
- Evolutionary algorithms: recombination (joining parts of two networks together) and mutation (adding or removing a layer or changing a parameter value)
- Hill climbing with mutation operations
- major challenge is estimating the value of a candidate network
- train one big network, search for subgraphs of the network that perform better
- heuristic evaluation function

Generalization

Weight decay

- weight decay similar to regularization
- adding a penalty to loss function $\lambda \sum_{i,j} W_{i,j}^2$
- λ strength of the penalty, sum over all of the weights in the networ
- beneficial effect of weight decay is that it implements a form of maximum a posteriori (MAP) learning
- usual cross-entropy loss; the second term prefers weights that are likely under a prior distribution

$$\begin{aligned} h_{\text{MAP}} &= \underset{\mathbf{w}}{\text{argmax}}\, P(\mathbf{y}\,|\,\mathbf{X}, \mathbf{W}) P(\mathbf{W}) \\ &= \underset{\mathbf{w}}{\text{argmin}} [-\log P(\mathbf{y}\,|\,\mathbf{X}, \mathbf{W}) - \log P(\mathbf{W})] \,. \end{aligned}$$

Generalization

Dropout

intervene to reduce the test-set error of a network—at the cost of making it harder to fit the training set:

- introducing noise at training time, the model is forced to become robust to noise
- Hidden units trained with dropout useful & compatible with many other possible sets of other hidden units that may or may not be included in the full model.
- dropout approximates the creation of a large ensemble of thinned networks
- applied to later layers in a deep network forces the final decision to be made robustly by paying attention to all of the abstract features of the example

dropout forces the model to learn multiple, robust explanations for each input

Unsupervised Learning and Transfer Learning

Unsupervised learning

Unsupervised learning algorithms take a training set of unlabeled examples **x**

- **To learn new representations**
- **To learn a generative model**

Probabilistic PCA: A simple generative model

- probabilistic principal components analysis (PPCA)
- **z** chosen from a zero-mean, spherical Gaussian, **x** generated from **z** by applying a weight matrix **W** and adding spherical Gaussian noise:

$$P(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$$
$$P_W(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z}, \sigma^2 \mathbf{I})$$

The weights **W** (and optionally the noise parameter σ^2) can be learned by maximizing the likelihood of the data,

$$P_W(\mathbf{x}) = \int P_W(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}) .$$

can be done by gradient methods or by an efficient iterative EM algorithm

Unsupervised Learning and Transfer Learning

Autoencoders

autoencoder is a model containing two parts:

- an encoder that maps from **x** to a representation $\hat{\mathbf{z}}$
- a decoder that maps from a representation $\hat{\mathbf{z}}$ to observed data **x**.

linear autoencoder, simple encoder where both *f* and *g* are linear with a shared weight matrix **W**:

$$\hat{\mathbf{z}} = f(\mathbf{x}) = \mathbf{W}\mathbf{x}$$
$$\mathbf{x} = g(\hat{\mathbf{z}}) = \mathbf{W}^\top \hat{\mathbf{z}}.$$

Variational autoencoder (VAE) more complex can capture more complex kinds of generative models

- use **variational posterior** $Q(\mathbf{z})$, as an approximation
- KL divergence: "as close as possible"

$$D_{KL}(Q(\mathbf{z})\|P(\mathbf{z}|\mathbf{x})) = \int Q(\mathbf{z}) \log \frac{Q(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})} d\mathbf{z},$$

- ***variational*** **lower bound** $\mathcal{L}$ **(evidence lower bound**, or **ELBO)**

$$\mathcal{L}(\mathbf{x}, Q) = \log P(\mathbf{x}) - D_{KL}(Q(\mathbf{z})\|P(\mathbf{z}|\mathbf{x}))$$

Unsupervised Learning and Transfer Learning

Variational autoencoders provide a means of performing variational learning in the deep learning setting.

Variational learning involves maximizing $\mathcal{L}$ with respect to the parameters of both P and Q

The decoder $g(\mathbf{z})$ is interpreted defining log P($\mathbf{x}$ | $\mathbf{z}$).

Autoregressive model (**AR model**):

- each element x_i is predicted based on other elements of the vector $\mathbf{x}$ & has no latent variables
- If $\mathbf{x}$ fixed size, fully observable and possibly fully connected Bayes net.
- Used in analysis of time series data

Deep autoregressive model:

- linear–Gaussian model is replaced by an arbitrary deep network with a suitable output layer depending on whether x_t is discrete or continuous

Unsupervised Learning and Transfer Learning

Generative adversarial networks

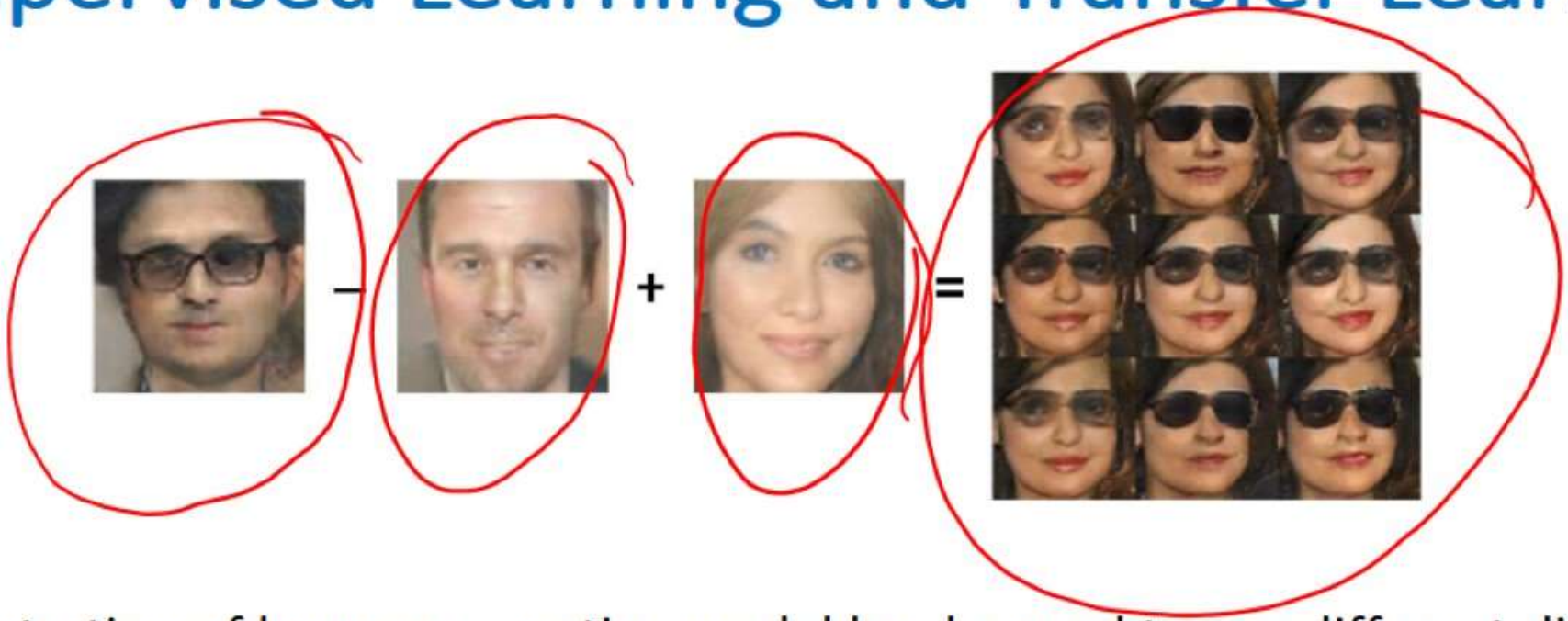
pair of networks that combine to form a generative system

- the **generator**, maps values from **z** to **x**
- the **discriminator**, is a classifier trained to classify inputs **x** as real or fake (generated)

implicit model: samples can be generated but their probabilities are not readily available
generator and the discriminator are trained simultaneously
Generator learning to fool the discriminator and the discriminator learning to accurately separate real from fake data

GANs have worked particularly well for image generation tasks. For example, GANs can create photorealistic, high-resolution images of people who have never existed

Unsupervised Learning and Transfer Learning



A demonstration of how a generative model has learned to use different directions in **z** space to represent different aspects of faces. We can actually perform arithmetic in **z** space. The images here are all generated from the learned model and show what happens when we decode different points in **z** space. We start with the coordinates for the concept of "man with glasses," subtract off the coordinates for "man," add the coordinates for "woman," and obtain the coordinates for "woman with glasses." Images reproduced with permission from (Radford *et al.*, 2015).

Unsupervised Learning and Transfer Learning

Transfer learning and multitask learning

For transfer learning experience with one learning task helps an agent learn better on another task

freeze the first few layers of the pretrained model that serve as feature detectors

modify the parameters of the higher levels only

- problem-specific features and do classification

Common to start with pretrained model such as the ROBERTA model

Followed by fine-tuning the model in two ways:

- giving it examples of the specialized vocabulary used in the desired domain
- training the model on the task it is to perform

Multitask learning is a form of transfer learning in which we simultaneously train a model on multiple objectives

Applications

Vision

- success of the AlexNet deep learning system in the 2012 ImageNet competition that propelled deep learning into the limelight
- supervised learning task with 1,200,000 images in 1,000 different categories
- top-5 error rate has been reduced to less than 2%— below error rate of trained human (5%)

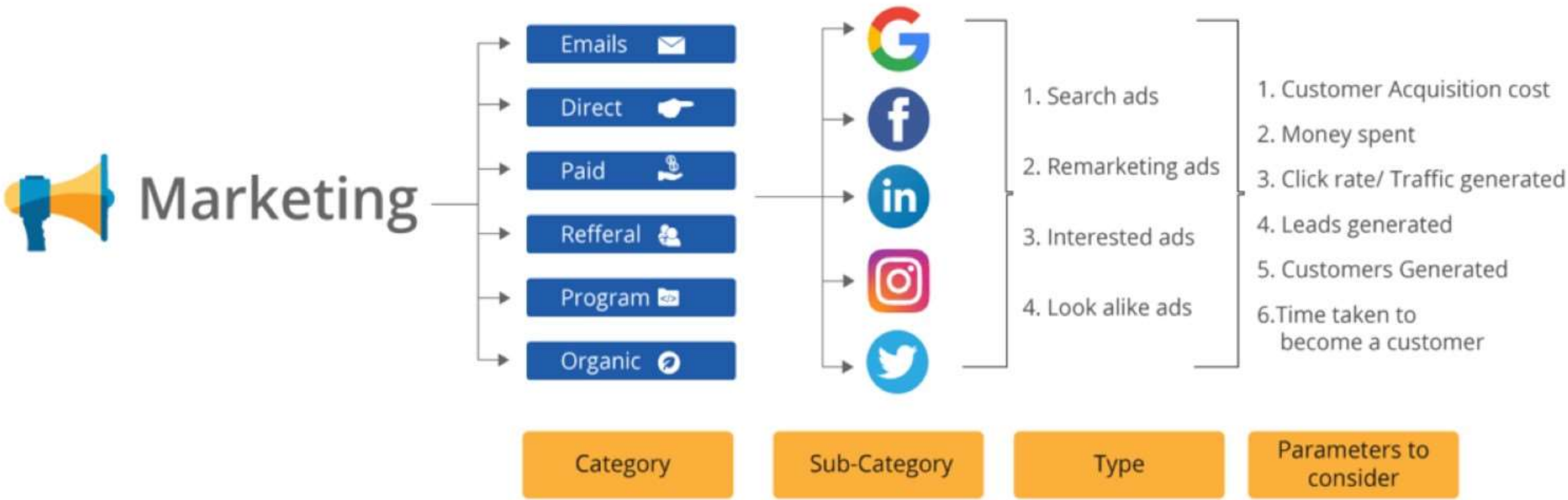
Natural language processing

- machine translation and speech recognition
- end-to-end learning, the automatic generation of internal representations for the meanings of words, and the interchangeability of learned encoders and decoders
- end-to-end learning outperforms classical pipelines
- re-representing individual words as vectors in a high-dimensional space—so-called word embeddings

Reinforcement learning

- decision-making agent learns from a sequence of reward signals that provide some indication of the quality of its behavior
- optimize the sum of future rewards
- learn a value function, a Q-function, a policy, and so on

Application Case: Marketing

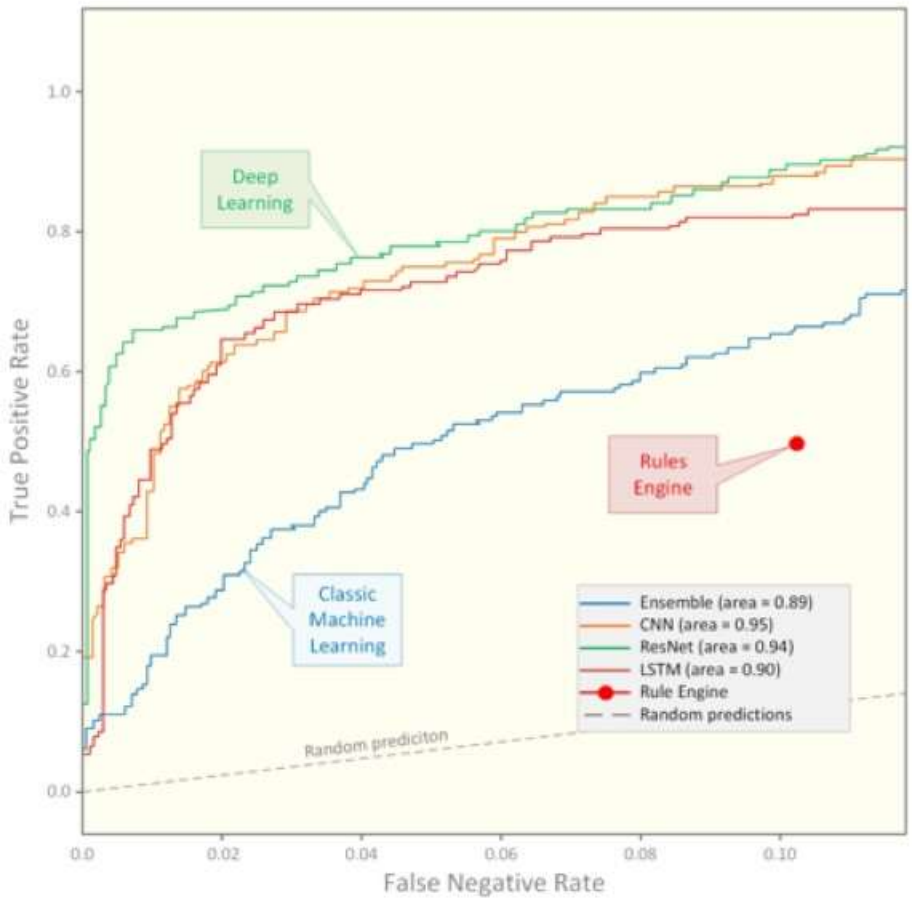


Application Case:
Fighting Fraud with Deep Learning

- Business problem
  - Danske Bank
  - Predictive analytics in banking
    - Fraud detection
- The solution
  - Deep learning
- The results were … impressive

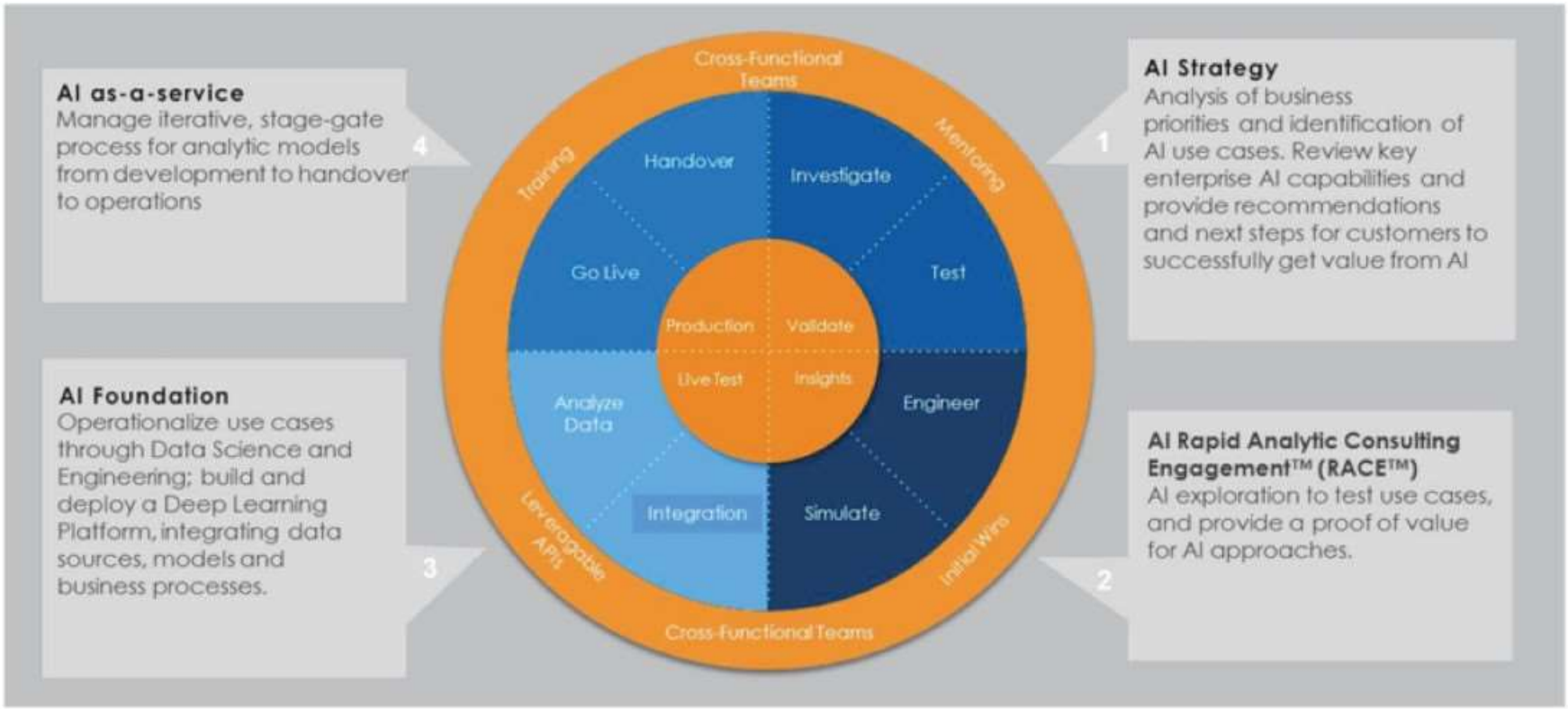
Application Case:
Fighting Fraud with Deep Learning

- The results were impressive
- Accuracy metric
  - ROC curve
- DL vs traditional ML techniques



Application Case:
Fighting Fraud with Deep Learning

- A Generalized Framework for AI and Deep Learning–Based Analytics



Summary

- Neural networks represent complex nonlinear functions with a network of parameterized linear-threshold units.
- The back-propagation algorithm implements a gradient descent in parameter space to minimize the loss function.
- Convolutional networks are particularly well suited for image processing and other tasks where the data have a grid topology.
- Recurrent networks are effective for sequence-processing tasks including language modeling and machine translation.

Neural network with keras python example

```
1.  # first neural network with keras
2.  from numpy import loadtxt
3.  from tensorflow.keras.models import Sequential
4.  from tensorflow.keras.layers import Dense

5.  # load the dataset
6.  dataset = loadtxt('diabetes.csv', delimiter=',')
7.  # split into input (X) and output (y) variables
8.  X = dataset[:,0:8]
9.  y = dataset[:,8]

10. # define the keras model
11. model = Sequential()
12. model.add(Dense(12, input_shape=(8,),
    activation='relu'))
13. model.add(Dense(8, activation='relu'))
14. model.add(Dense(1, activation='sigmoid'))

15. # compile the keras model
16. model.compile(loss='binary_crossentropy',
    optimizer='adam', metrics=['accuracy'])

17. # fit the keras model on the dataset
18. model.fit(X, y, epochs=150, batch_size=10)

19. # evaluate the keras model
20. _, accuracy = model.evaluate(X, y)
21. print('Accuracy: %.2f' % (accuracy*100))

22. # make class predictions with the model
23. predictions = (model.predict(X) > 0.5).astype(int)
```

Neural network with keras python example

```
Epoch 143/150
77/77 [==============================] - 0s 851us/step - loss: 0.5095 - accuracy: 0.7578
Epoch 144/150
77/77 [==============================] - 0s 872us/step - loss: 0.4757 - accuracy: 0.7734
Epoch 145/150
77/77 [==============================] - 0s 1ms/step - loss: 0.4923 - accuracy: 0.7643
Epoch 146/150
77/77 [==============================] - 0s 1ms/step - loss: 0.4974 - accuracy: 0.7682
Epoch 147/150
77/77 [==============================] - 0s 902us/step - loss: 0.4977 - accuracy: 0.7643
Epoch 148/150
77/77 [==============================] - 0s 892us/step - loss: 0.4922 - accuracy: 0.7734
Epoch 149/150
77/77 [==============================] - 0s 1ms/step - loss: 0.4748 - accuracy: 0.7721
Epoch 150/150
77/77 [==============================] - 0s 1ms/step - loss: 0.5341 - accuracy: 0.7344
24/24 [==============================] - 0s 560us/step - loss: 0.4898 - accuracy: 0.7604
Accuracy: 76.04
24/24 [==============================] - 0s 569us/step
[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] => 0 (expected 1)
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] => 0 (expected 0)
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] => 1 (expected 1)
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] => 0 (expected 0)
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] => 1 (expected 1)
```

Exercise 6

1. Develop new ML and AI algorithms,

- N/A

2. Develop new and efficient ML packages and implementation,

- N/A

3. Application of existing ML and AI algorithms to solve new or existing problems,

- Develop a python program that implements deep learning using tensorflow keras mnis datasets. Include printing of data and performance of the algorithm

4. Use of existing ML and AI software to solve new or existing problems,

- Identify 5 software that use deep learning that you can adopt in your organization.

References and further reading

- Stuart Russell (2021), Artificial Intelligence: A Modern Approach, 4th edition. Pearson publishing
- Dursun Delen (2021), Predictive Analytics: Data Mining, Machine Learning and Data Science for Practitioners, 2nd edition, Pearson FT Press
- Paul Deitel and Harvey M. Deitel (2020), Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud, 1st edition, Pearson Publishing
- Mark E. Fenner (2020), Machine Learning with Python for Everyone, Addison-Wesley Publishing
- Pang-Ning Tan, Michael Steinbach, Vipin Kumar (2018), Introduction to Data Mining (2nd Edition), Pearson publishing
- Data mining: Concepts and Techniques, by Jiawei Han and Micheline Kamber, Morgan Kaufmann Publishers, ISBN 1-55860-489-8.
- Machine Learning, by Tom M. Mitchell, McGraw-Hill, ISBN 0-07-042807-7
- Modern Information Retrieval, by Ricardo Baeza-Yates and Berthier Ribeiro-Neto, Addison Wesley, ISBN 0-201-39829-X

References and further reading

- https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
- https://www.edureka.co/blog/deep-learning-with-python/
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-with-python