SECA4002 – DEEP LEARNING NEURAL NETWORKS

Dr. V. Vedanarayanan B.E., M.E., PhD
Course Co-ordinator
Assistant Professor, Department of ECE,
School of Electrical and Electronics
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

Course Outcomes

SECA4002 – DEEP LEARNING NEURAL NETWORKS

At The end of this Course, Student will be able to

CO1	Select suitable model parameters for different machine learning techniques
CO2	Evaluate the performance of existing deep learning models for various applications
CO3	Realign high dimensional data using reduction techniques
CO4	Analyze the performance of various optimization and generalization techniques in deep learning
CO5	Modify the existing architectures for domain specific applications
CO6	Develop a real time application using deep learning neural networks

Course Objectives

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Course Objectives:

- To present the mathematical, statistical and computational challenges of building neural networks
- To study the concepts of deep learning
- To introduce dimensionality reduction techniques
- To enable the students to know deep learning techniques to support real-time applications
- To examine the case studies of deep learning techniques

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Detailed Syllabus:

UNIT 4: OPTIMIZATION AND GENERALIZATION

Optimization in deep learning– Non-convex optimization for deep networks- Stochastic Optimization Generalization in neural networks- Spatial Transformer Networks- Recurrent networks, LSTM Recurrent Neural Network Language Models- Word-Level RNNs & Deep Reinforcement Learning - Computational & Artificial Neuroscience.

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Recommended Text Books/ Reference Books

- Cosma Rohilla Shalizi, Advanced Data Analysis from an Elementary Point of View, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

Optimization In Deep Learning Network

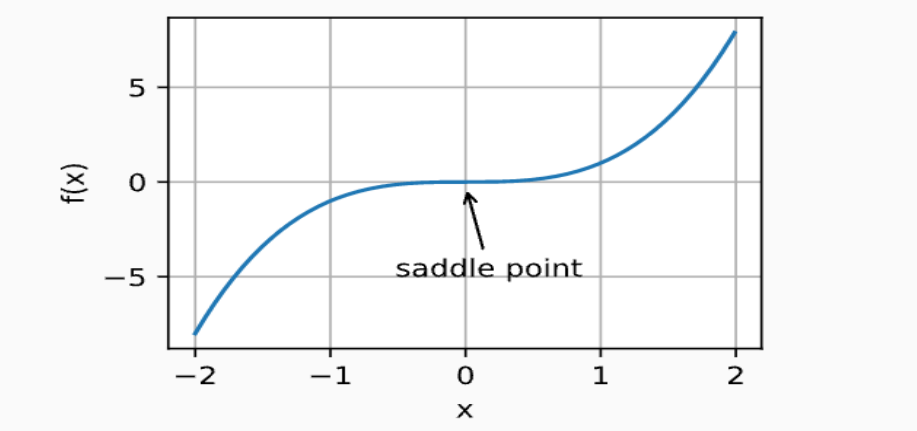
- In Deep Learning, with the help of loss function, the performance of the model is estimated/ evaluated.
- This loss is used to train the network so that it performs better.
- Essentially we try to minimize the Loss function .Lower Loss means the model performs better.
- The process of minimizing (or maximizing) any mathematical expression is called optimization.
- Optimizers are algorithms or methods used to change the features of the neural network such as weights and learning rate so that the loss is reduced.
- Optimizers are used to solve optimization problems by minimizing the function
- The Goal of an Optimizer is to minimize the Objective Function(Loss Function based on the Training Data set)
- Simply Optimization is to minimize the Training Error

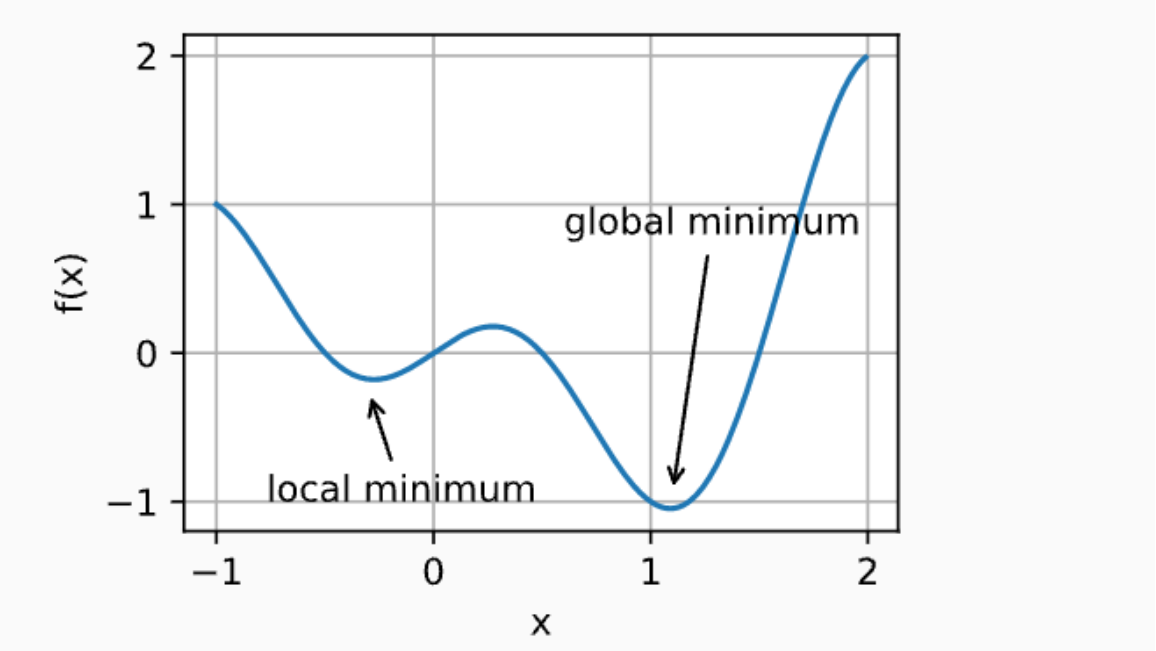
Need for Optimization In Deep Learning Network

Local Minima:

For any Objective function f(x), if the value of f(x) at x is is smaller than f(x) at any other points near x, f(x) is termed as local minimum. If f(x) at x is minimum over the entire region then its called global minimum.

- The objective function of deep learning models usually has many local optima.





SADDLE POINT :

- saddle points are another reason for gradients to vanish. A *saddle point* is any location where all gradients of a function vanish but which is neither a global nor a local minimum

Need for Optimization In Deep Learning Network

Need for optimization:

To reduce Local Minima

To avoid Vanishing Gradients or Exploding Gradients

To select appropriate weight values and other associated model parameters

To minimize the loss value (Training error)

Convex Optimization:

Convex optimization is a kind of optimization which deals with the study of problem of minimizing convex functions .

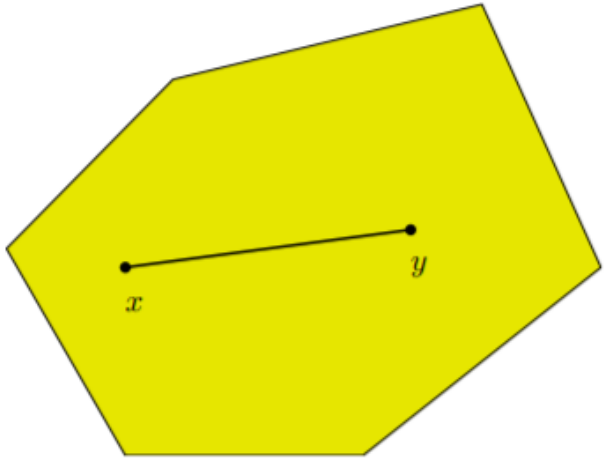
Here the optimization function is Convex function..

All Linear functions are convex, so linear programming problems are convex problems

When we have a convex objective and a convex feasible region, then there can be only one optimal solution, which is globally optimal.

Definition: A set $C \subseteq R^n$ is convex if for $x, y \in C$ and any $\alpha \in [0, 1]$,

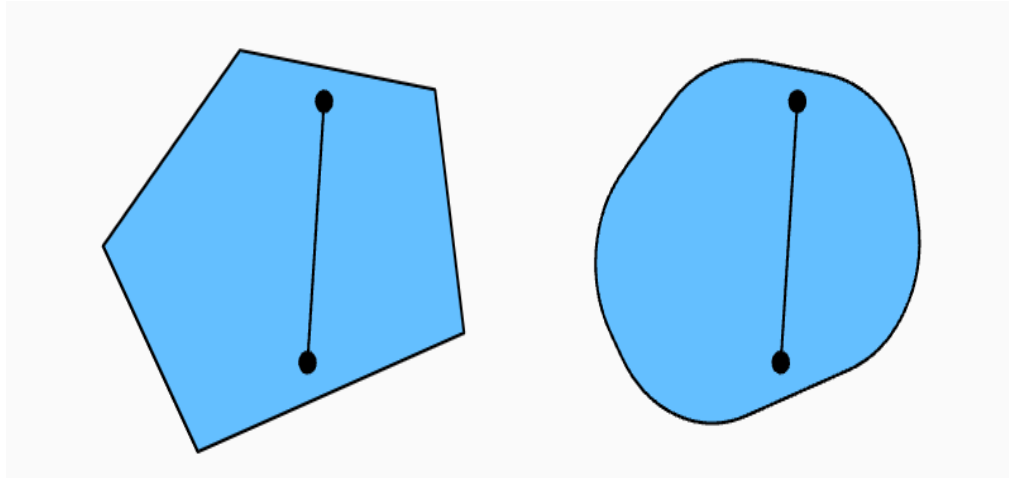
$$\alpha x + (1 - \alpha)y \in C$$



Convexity

Convexity plays a vital role in the design of optimization algorithms. This is largely due to the fact that it is much easier to analyze and test algorithms in such a context.

Select any to points in the region and join them by a straight Line. If the line and the selected points all lie inside the region then we call that region as Convex Region (as Shown in the diagram)



A convex optimization problem is of the form:

$$\min_{x \in D} f(x)$$

subject to

$$g_i(x) \leq 0, \ i = 1, ..., m$$

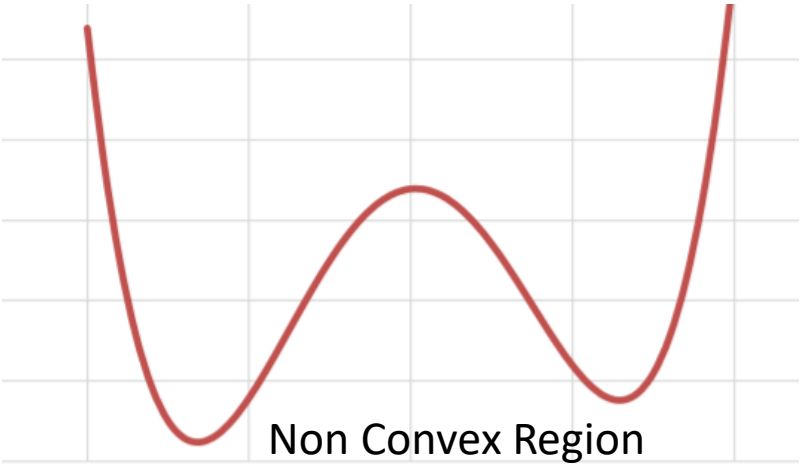
$$h_j(x) = 0, \ j = 1, ..., r$$

where f and g_i are all convex , and h_j are affine. Any local minimizer of a convex optimization problem is a global minimizer.

Non-Convex Optimization

Non Convex Optimization: The objective function is a non convex function.

- It has Multiple feasible regions and multiple locally optimal points.
- There can't be a general algorithm to solve it efficiently in all cases
- Neural networks are universal function approximators, To do this, they need to be able to approximate non-convex functions.



How to solve non-convex problems?

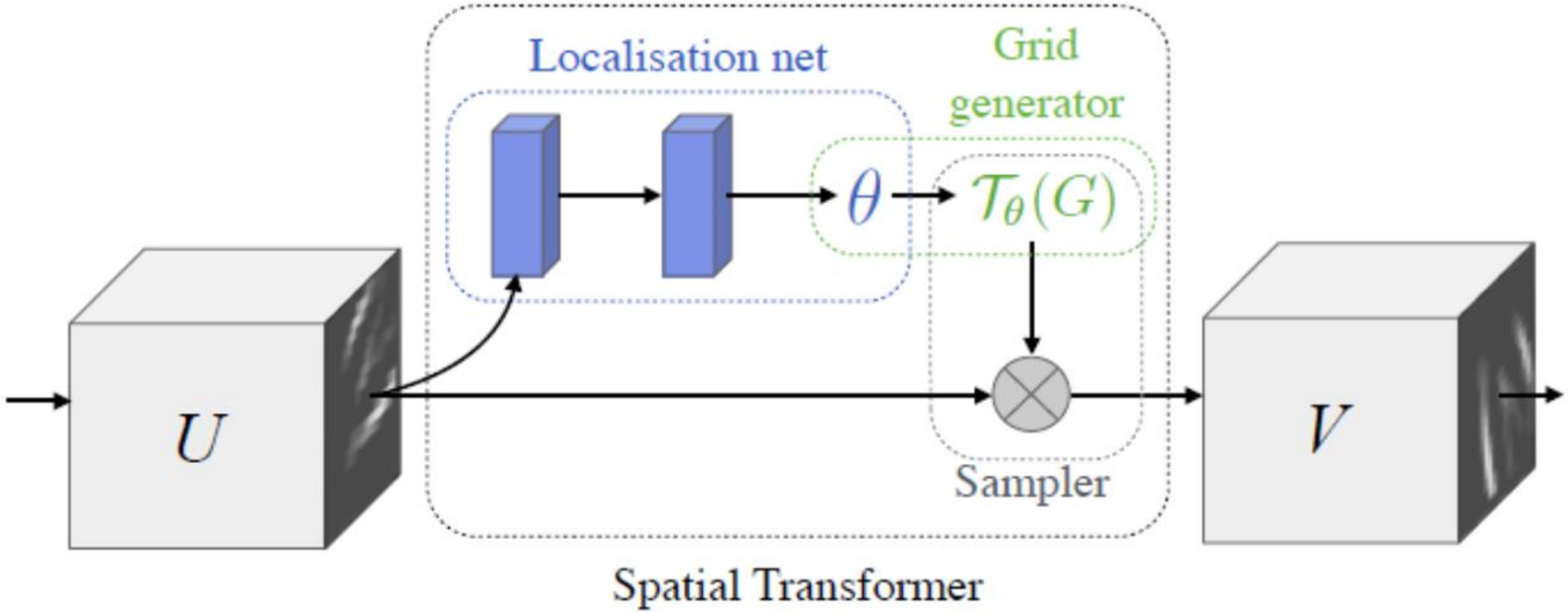
- Stochastic gradient descent
- Mini-batching
- SVRG
- Momentum

Reasons For Non Convexity:

- Presence of many Local Minima
- Prescence of Saddle Points
- Very Flat Regions
- Varying Curvature

Spatial Transformer Networks

Spatial Transformer Network (SSTN) helps to crop out and scale-normalizes the appropriate region, which can simplify the subsequent classification task and lead to better classification performance



Spatial Transformer Networks

STN is composed of **Localization Net**, **Grid Generator** and **Sampler**.

Localisation Net

With **input feature map *U***, with width *W*, height *H* and *C* channels, **outputs are ϑ**, the parameters of transformation $T\vartheta$. It can be learnt as affine transform

Grid Generator

- Suppose we have a regular grid G, this G is a set of points with **target coordinates (xt_i, yt_i)**.
- Then we **apply transformation $T\vartheta$ on G**, i.e. $T\vartheta(G)$.
- After $T\vartheta(G)$, a set of points with **destination coordinates (xt_i, yt_i) is outputted**. These points have been altered based on the transformation parameters. It can be Translation, Scale, Rotation or More Generic Warping depending on how we set ϑ as mentioned above.

Sampler

- **Based on the new set of coordinates (xt_i, yt_i), we generate a transformed output feature map *V***. This *V* is translated, scaled, rotated, warped, projective transformed or affined, whatever.
- It is noted that STN can be applied to not only input image, but also intermediate feature maps.

Spatial Transformer Networks

- **STN is a mechanism that rotates or scales an input image or a feature map in order to focus on the target object and to remove rotational variance .**

- One of the most notable features of STNs is their modularity (the module can be injected into any part of the model) and their ability to be trained with a single backprop algorithm without modification of the initial model.

Advantages:

- Helps in learning explicit spatial transformations like translation, rotation, scaling, cropping, non-rigid deformations, etc. of features.
- Can be used in any networks and at any layer and learnt in an end-to-end trainable manner.
- Provides improvement in the performance of existing models.

Recurrent Neural Networks (RNN)

- RNNs are very powerful, because they combine two properties:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.

RNN were created because there were a few issues in the feed-forward neural network:

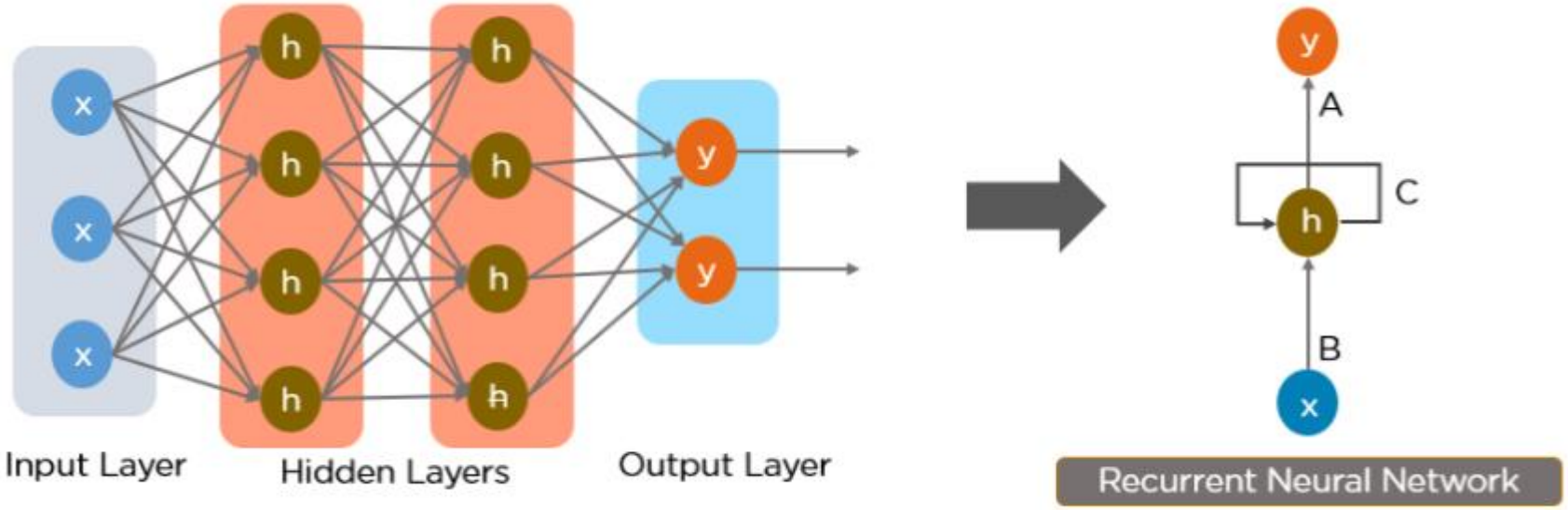
- Cannot handle sequential data
- Considers only the current input
- Cannot memorize previous inputs
- The solution to these issues is the RNN

Recurrent Neural Networks (RNN)

What is a Recurrent Neural Network (RNN)?

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer

We can convert a Feed-Forward Neural Network into a Recurrent Neural Network as below



Recurrent Neural Networks (RNN)

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.

Here, “x” is the input layer, “h” is the hidden layer, and “y” is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.

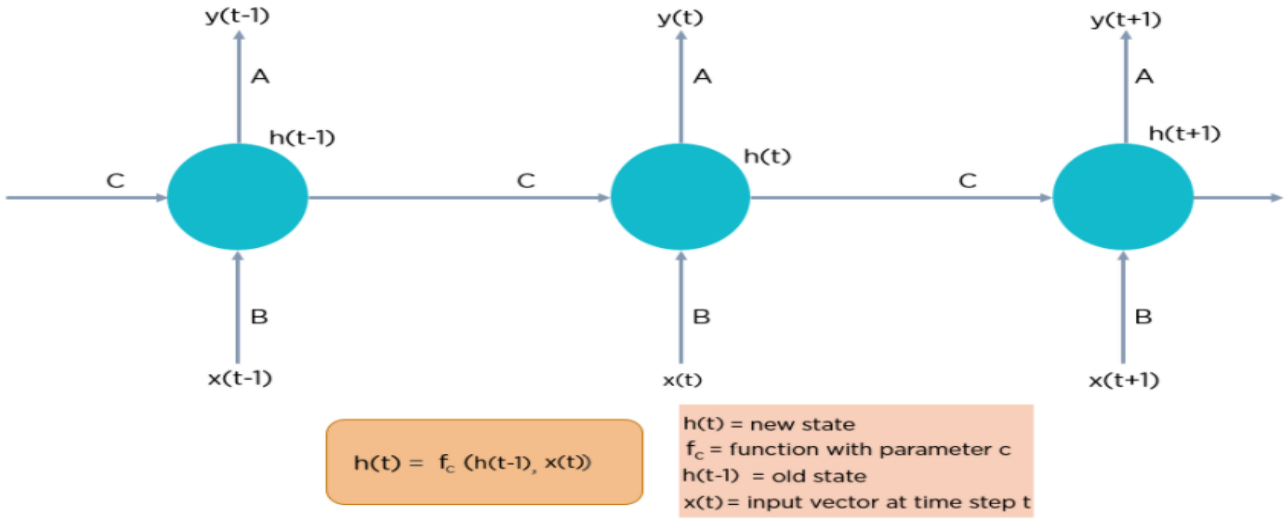


Fig: Fully connected Recurrent Neural Network

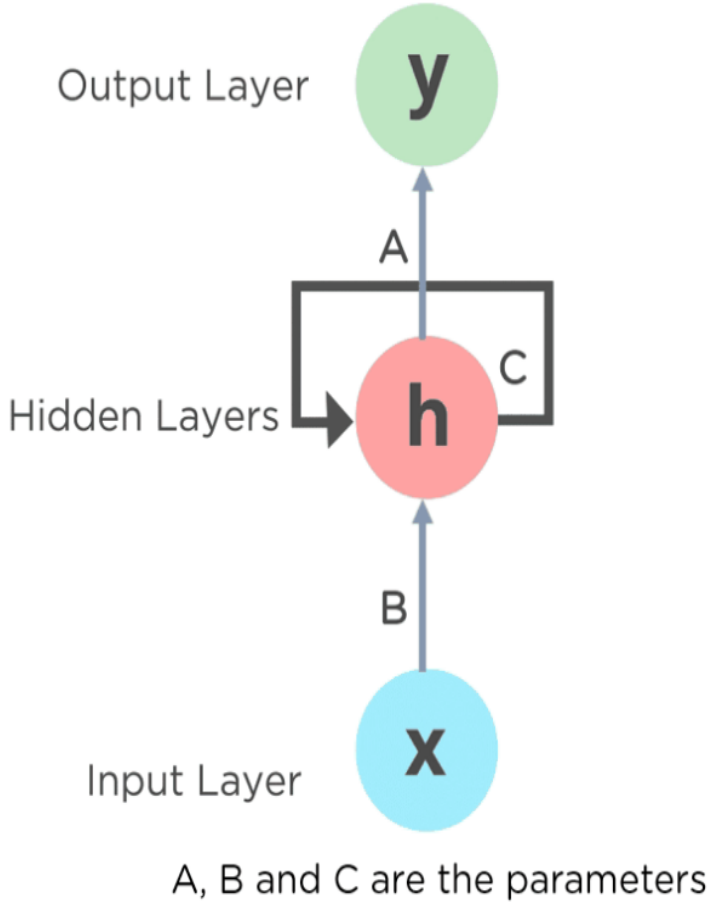


Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Long Short Term Memory (LSTM)

LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.

All RNN are in the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs also have a chain-like structure, but the repeating module is a bit different structure. Instead of having a single neural network layer, four interacting layers are communicating extraordinarily.

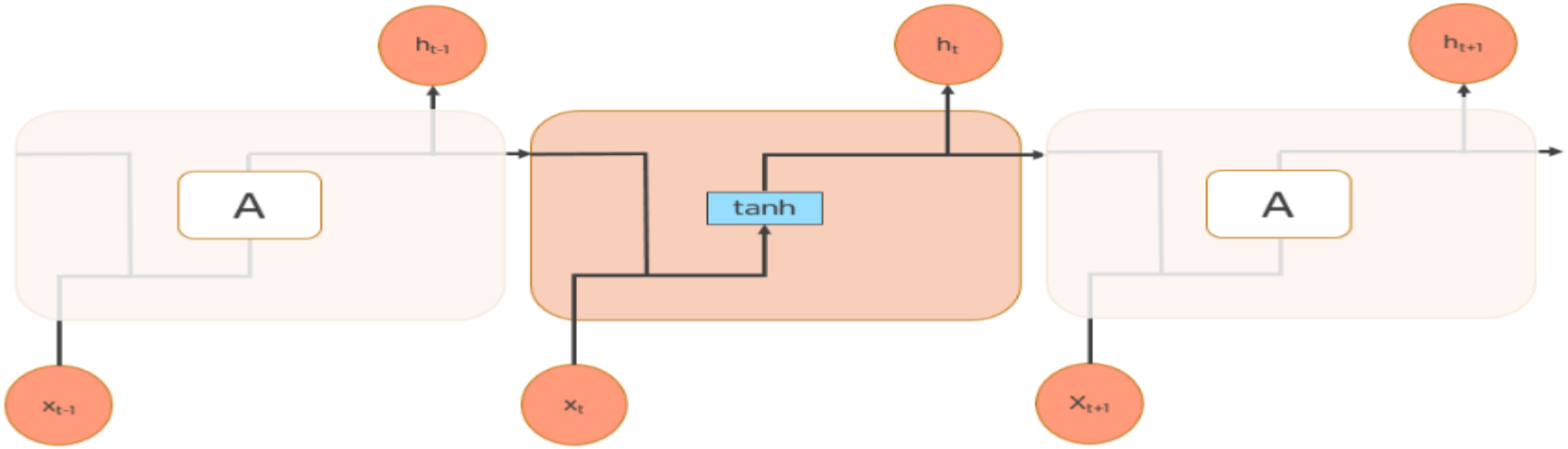


Fig: Long Short Term Memory Networks

Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Long Short Term Memory (LSTM)

LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.

All RNN are in the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs also have a chain-like structure, but the repeating module is a bit different structure. Instead of having a single neural network layer, four interacting layers are communicating extraordinarily.

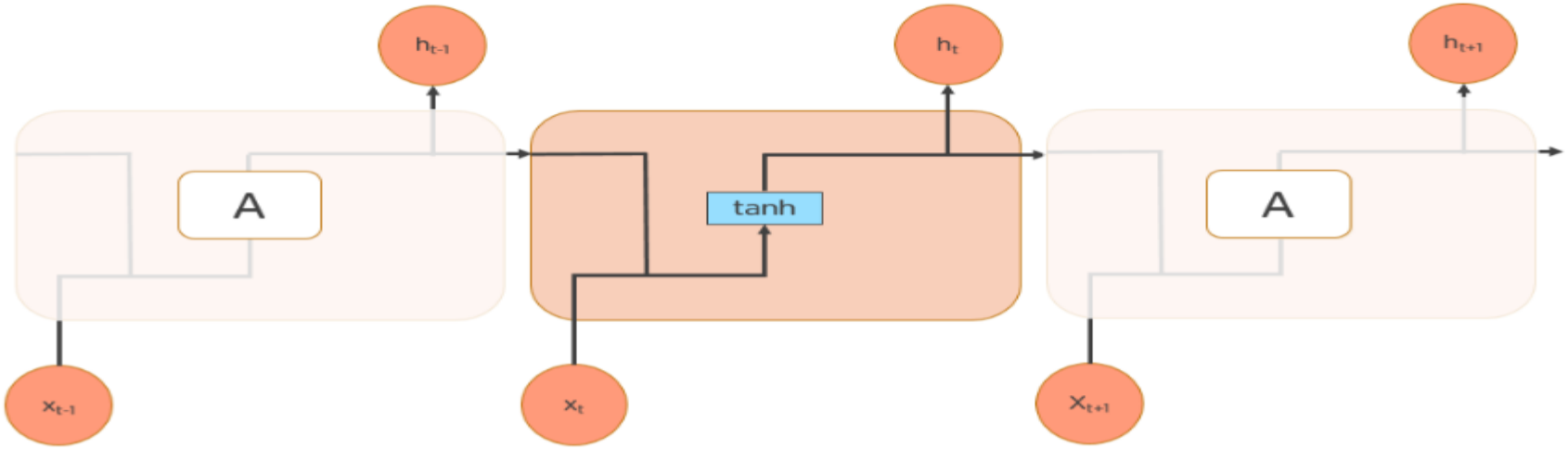


Fig: Long Short Term Memory Networks

Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Long Short Term Memory (LSTM)

LSTMs 3-steps

Step 1: Decide how much past data it should remember

The first step in the LSTM is to decide which information should be omitted from the cell in that particular time step. The sigmoid function determines this. It looks at the previous state (ht-1) along with the current input xt and computes the function.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

f_t = forget gate
Decides which information to delete that is not important from previous time step

Step 2: Decide how much this unit adds to the current state

In the second layer, there are two parts. One is the sigmoid function, and the other is the tanh function. In the *sigmoid* function, it decides which values to let through (0 or 1). *tanh* function gives weightage to the values which are passed, deciding their level of importance (-1 to 1).

Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Long Short Term Memory (LSTM)

Step 2:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t = input gate
Determines which information to let through based on its significance in the current time step

Step 3: Decide what part of the current cell state makes it to the output

The third step is to decide what the output will be. First, we run a sigmoid layer, which decides what parts of the cell state make it to the output. Then, we put the cell state through tanh to push the values to be between -1 and 1 and multiply it by the output of the sigmoid gate.

Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Long Short Term Memory (LSTM)

Step 3: Decide what part of the current cell state makes it to the output

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

o_t = output gate
Allows the passed in information to impact the output in the current time step

Applications of LSTM include:

- Robot control
- Time series prediction
- Speech recognition
- Rhythm learning
- Music composition
- Grammar learning
- Handwriting recognition

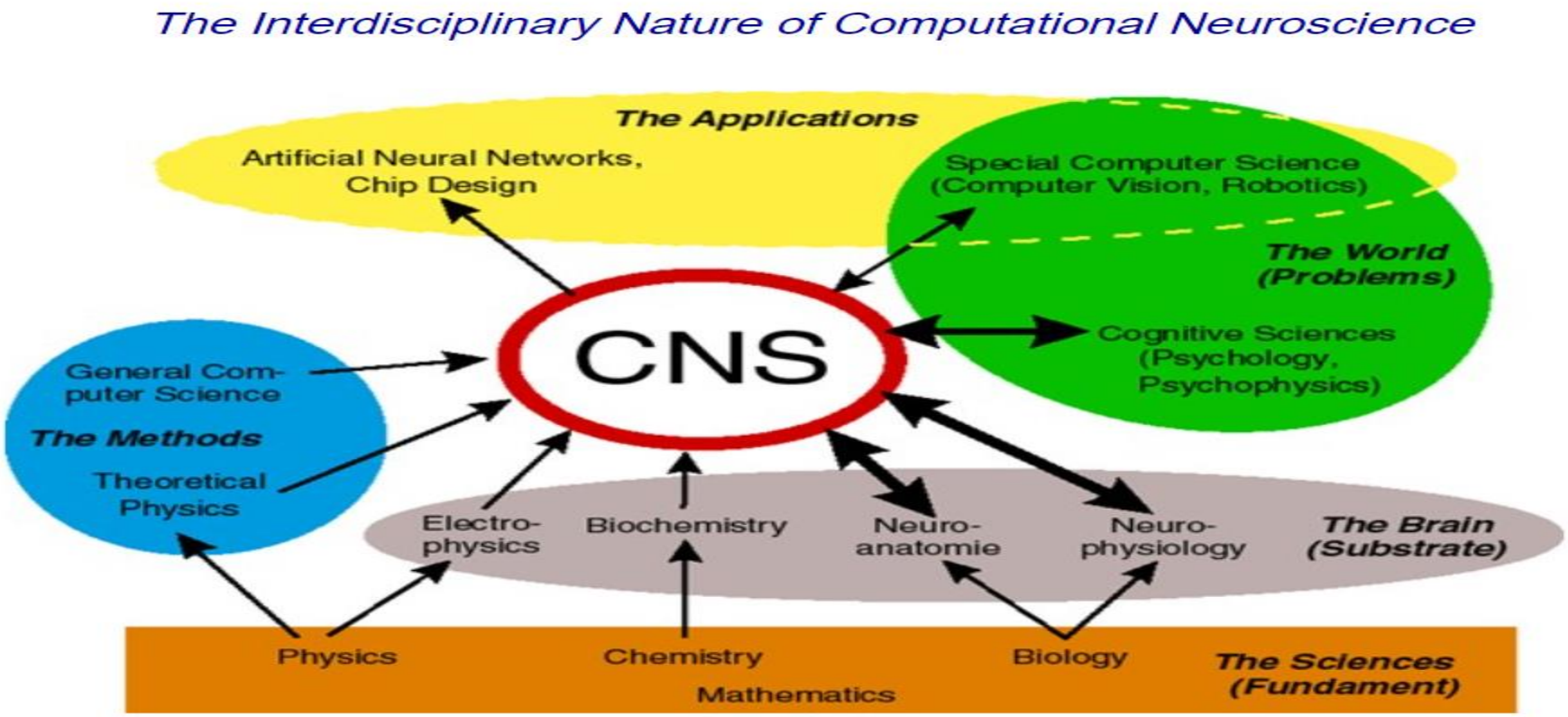
Image & Content: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#why_recurrent_neural_networks

Computational & Artificial Neuro Science

Computational neuroscience is the field of study in which mathematical tools and theories are used to investigate brain function.

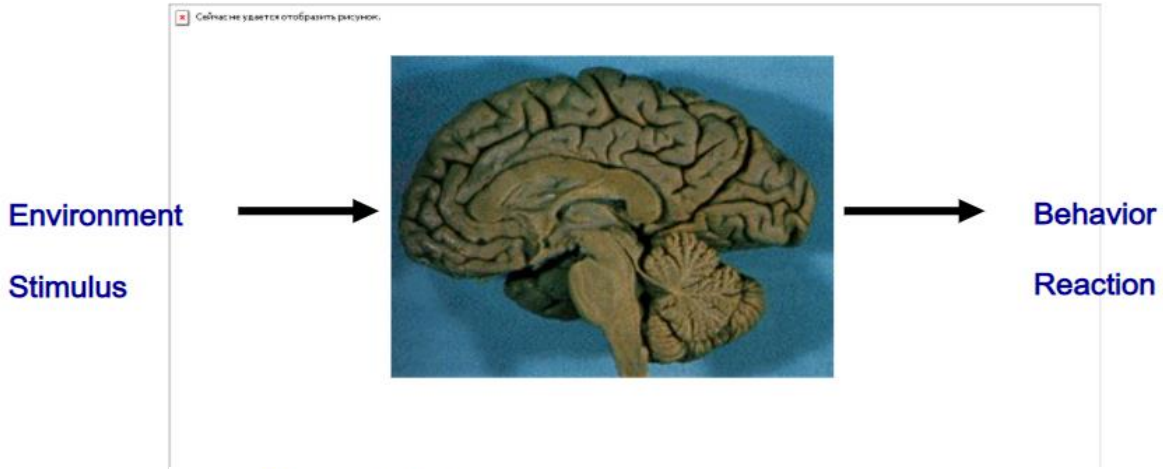
The term “computational neuroscience” has two different definitions:

1. using a computer to study the brain
2. studying the brain as a computer

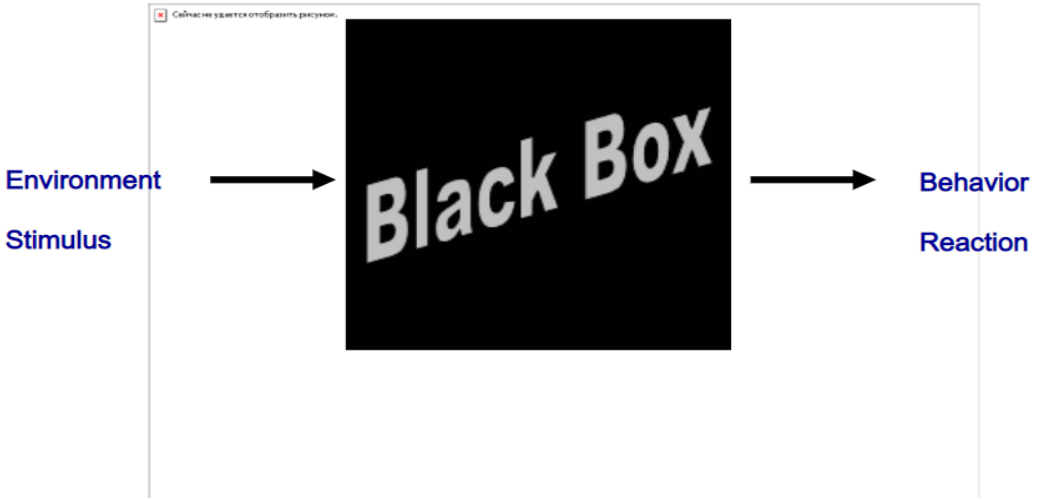


Computational & Artificial Neuro Science

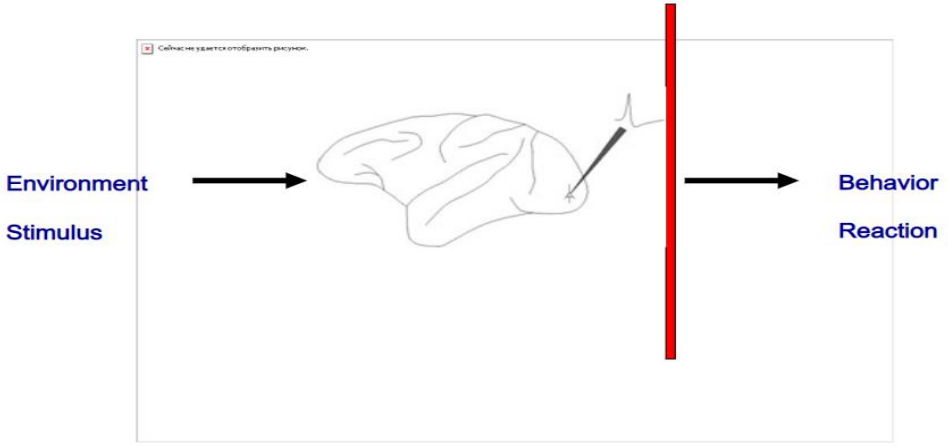
Neuroscience:



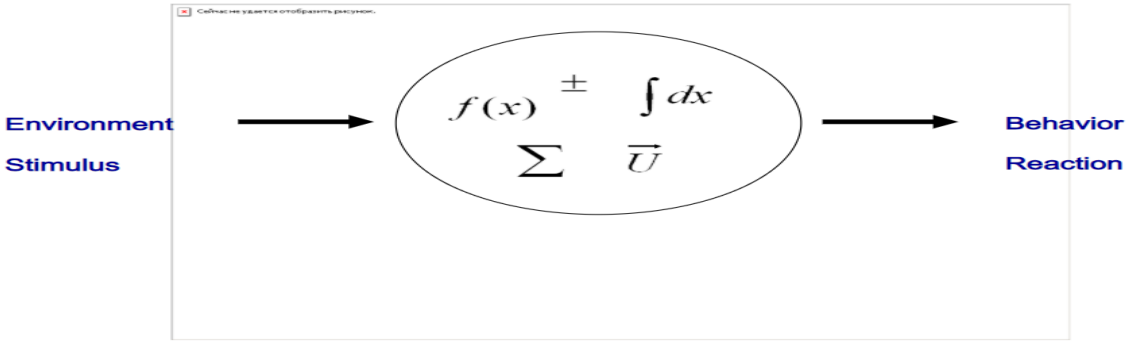
Psychophysics (human behavioral studies):



Neurophysiology:

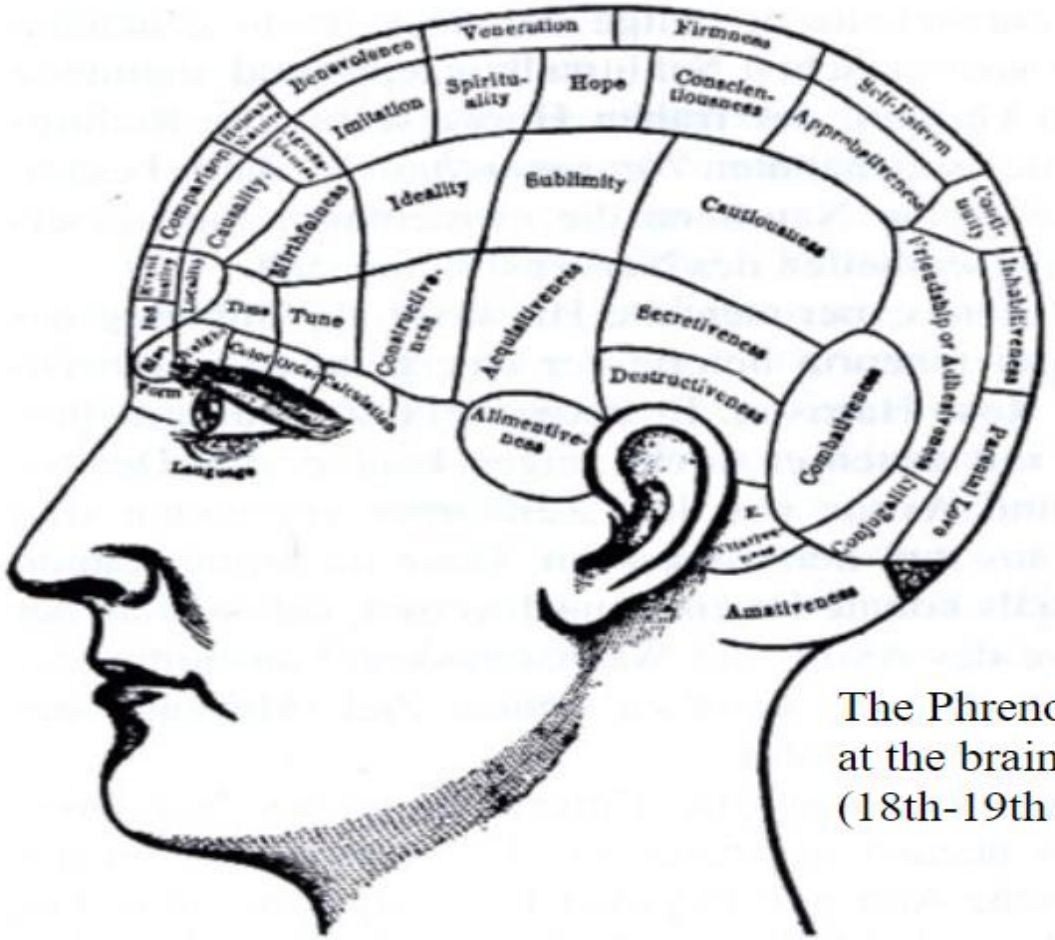


Theoretical/Computational Neuroscience:



Computational & Artificial Neuro Science

Where are things happening in the brain.



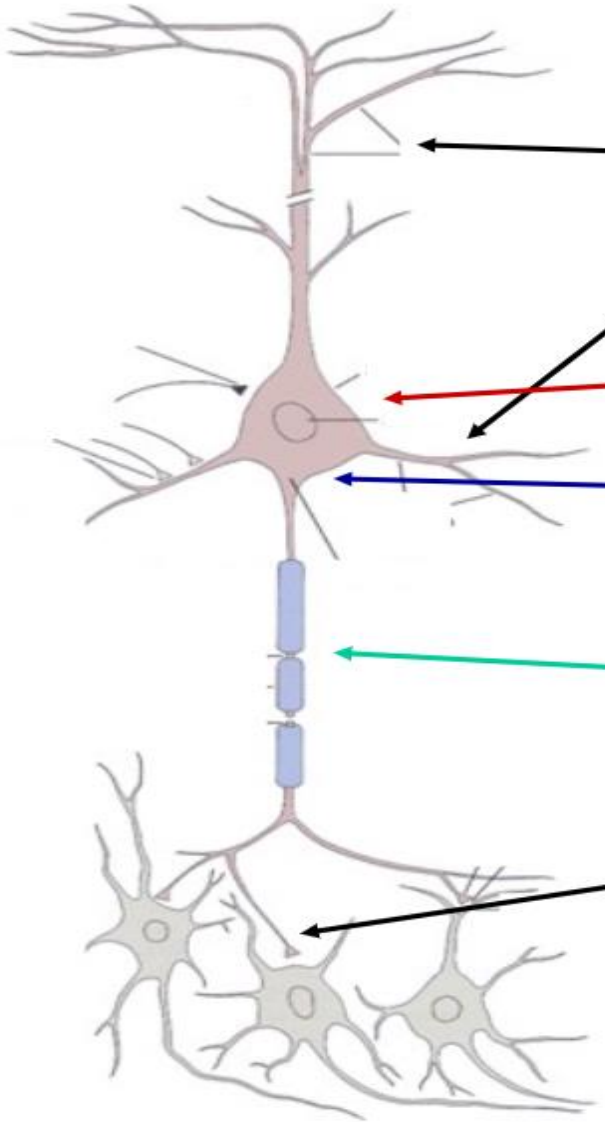
The Phrenologists view at the brain (18th-19th centrury)

Is the information represented locally ?

CNS
Systems
Areas
Local Nets
Neurons
Synapses
Molekules

Computational & Artificial Neuro Science

Structure of a Neuron:



At the dendrite the incoming signals arrive (incoming currents)

At the soma current are finally integrated.

At the axon hillock action potential are generated if the potential crosses the membrane threshold

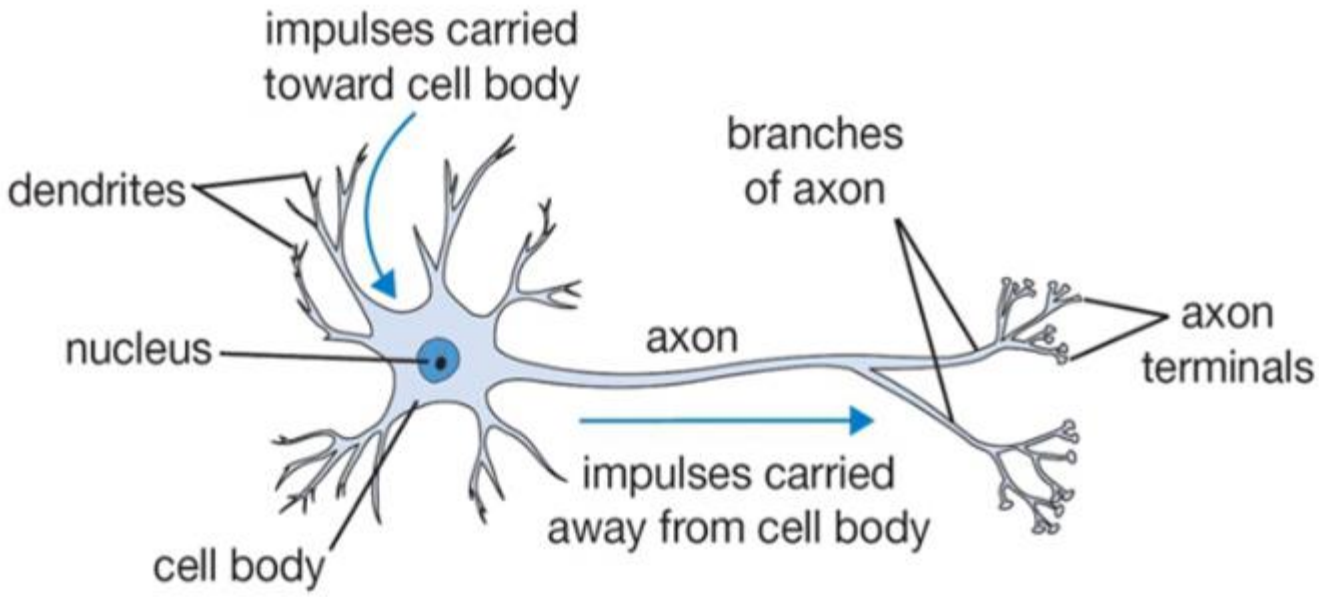
The axon transmits (transports) the action potential to distant sites

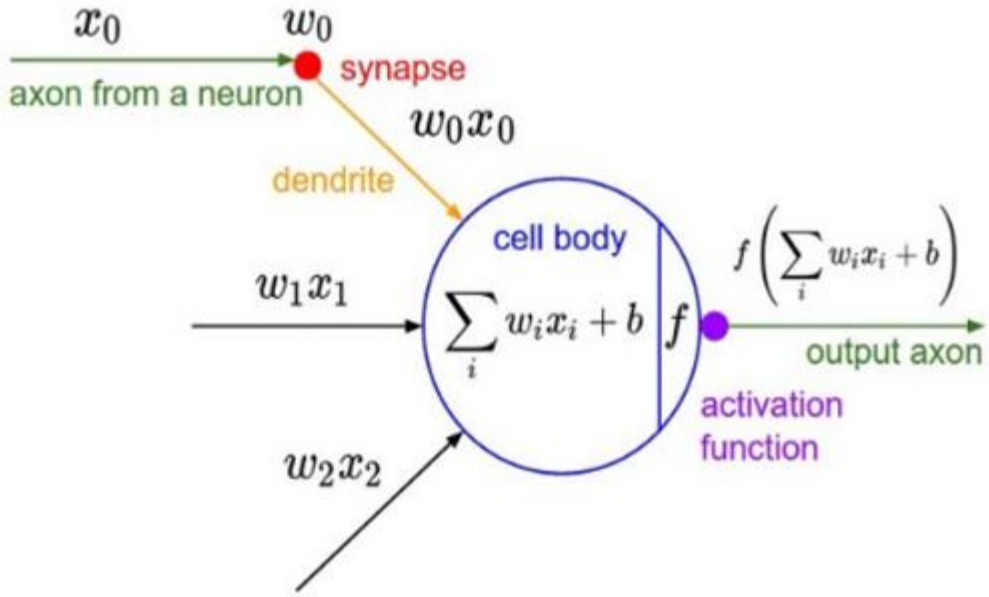
At the synapses are the outgoing signals transmitted onto the dendrites of the target neurons

CNS
Systems
Areas
Local Nets
Neurons
Synapses
Molekules

Computational & Artificial Neuro Science

Analogy Between ANN & BNN





biological neuron (left) and a common mathematical model (right)

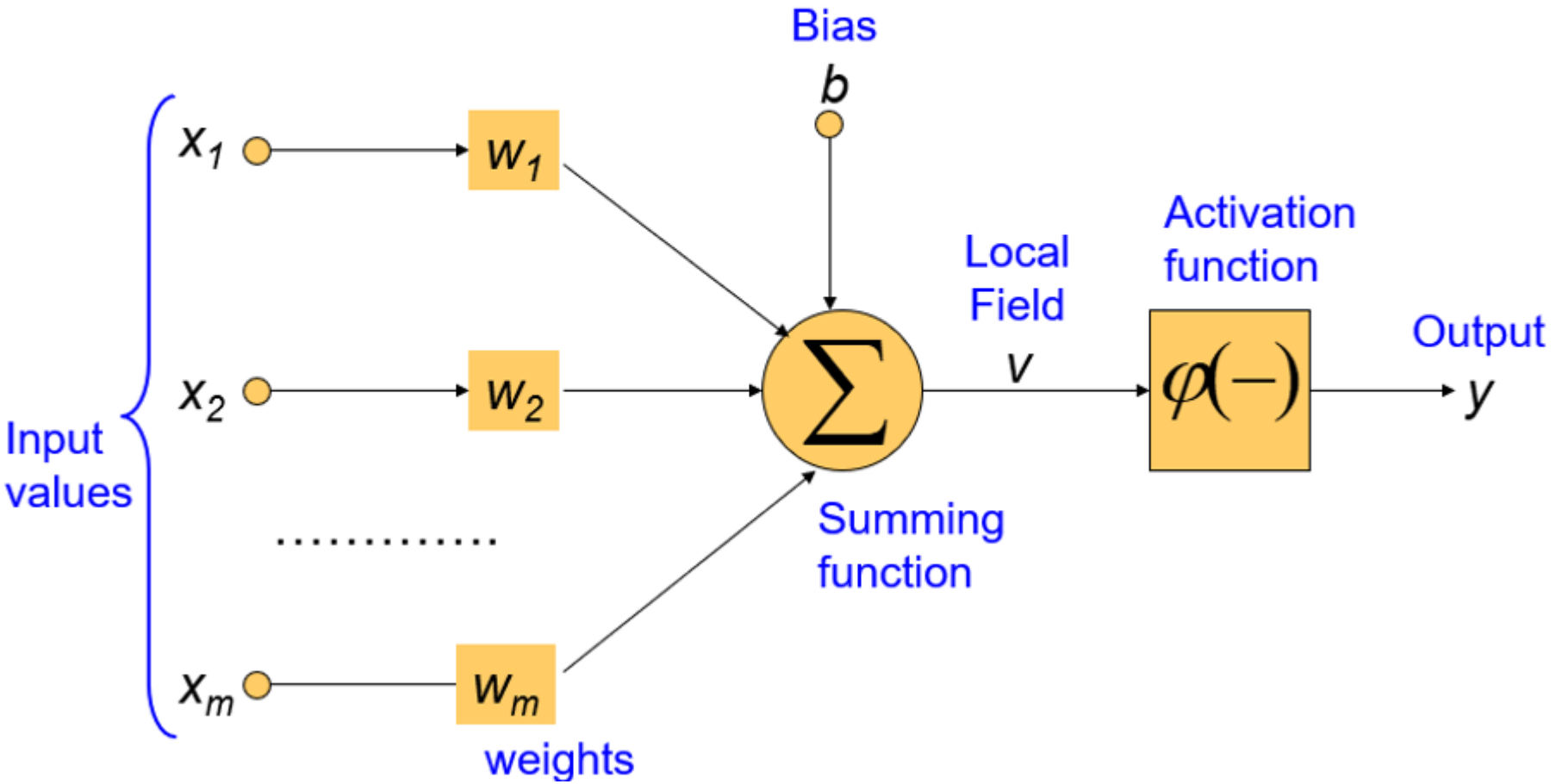
Computational & Artificial Neuro Science

Difference Between ANN & BNN

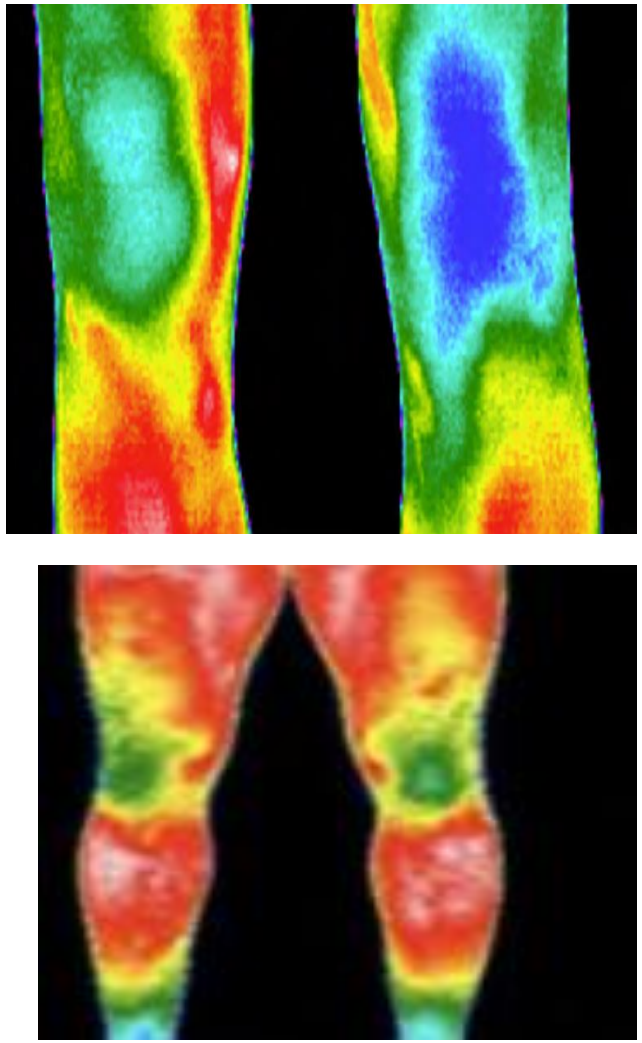
Criteria	BNN	ANN
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes (mainly depends on the type of application and network designer)
Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

Computational & Artificial Neuro Science

Basic ANN Architecture



Application of Deep Learning Neural Network- Joint Detection



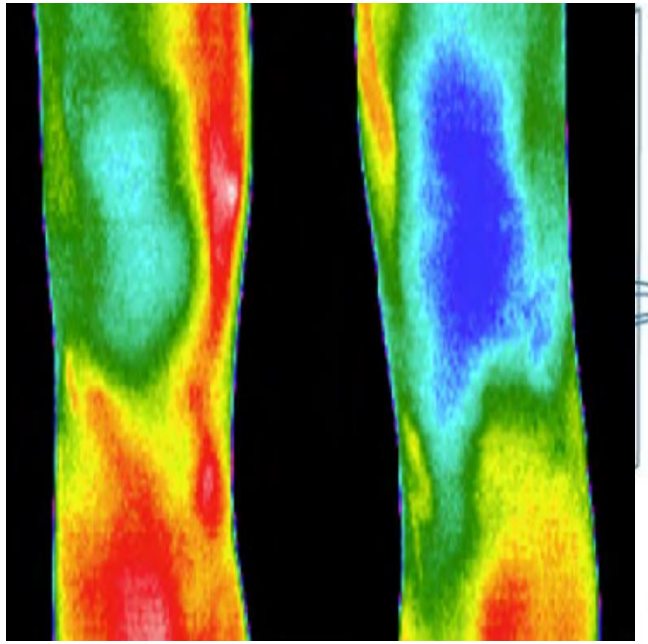
DLNN MODEL

Rheumatoid Arthritis Present

Rheumatoid Arthritis Absent

Image & Content Source: https://www.researchgate.net/figure/Infrared-image-from-the-anterior-view-of-the-knees-TRIATHLON_fig2_221924974

Application of Deep Learning Neural Network- Joint Detection



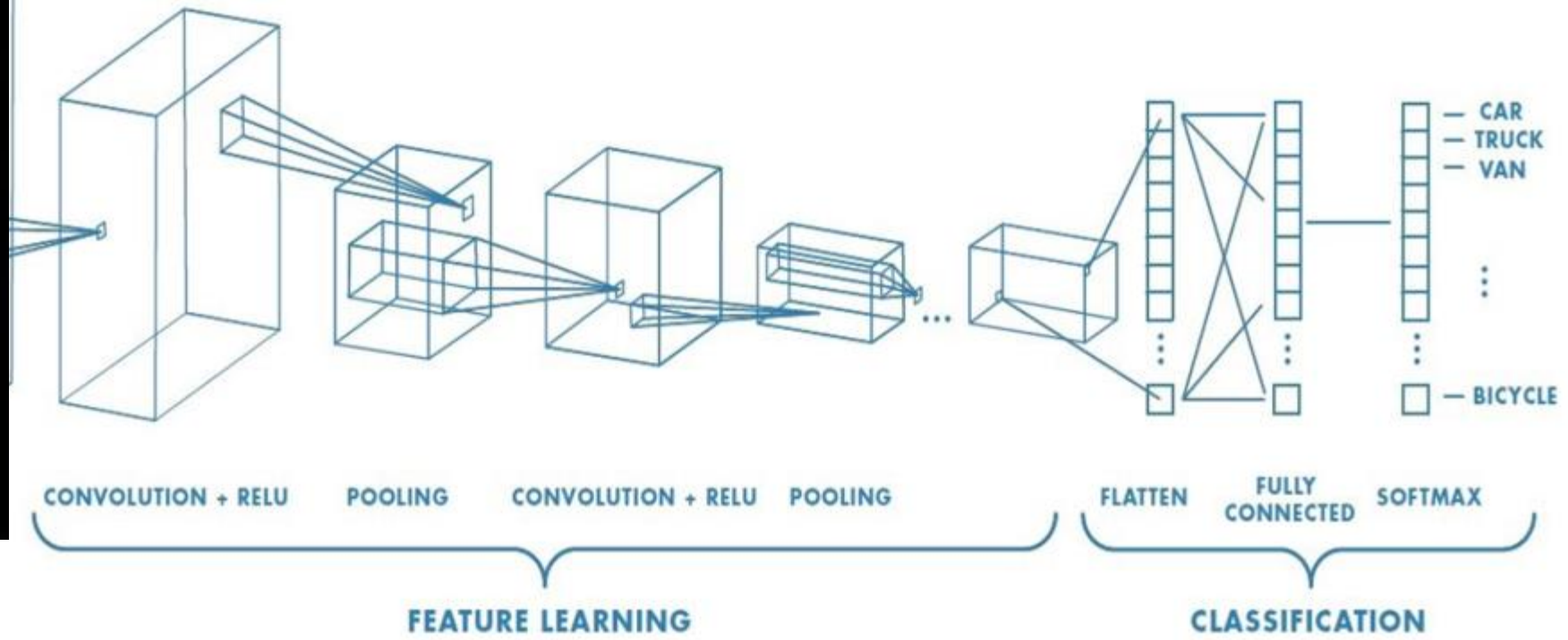


Image & Content Source: https://www.researchgate.net/figure/Infrared-image-from-the-anterior-view-of-the-knees-TRIATHLON_fig2_221924974

Application of Deep Learning Neural Network- Joint Detection

For Detection of Pedestrian

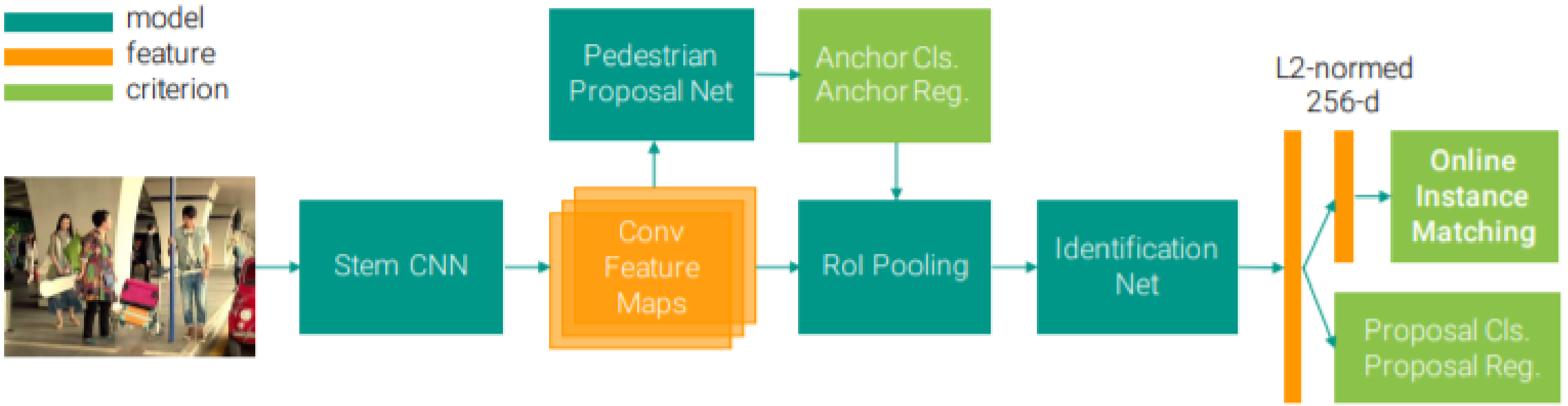


Image & Content Source: https://openaccess.thecvf.com/content_cvpr_2017/papers/Xiao_Joint_Detection_and_CVPR_2017_paper.pdf

Application of Deep Learning Neural Network- Face Recognition

1. **Detect**/identify faces in an image (using a face detection model) — for simplicity, this tutorial will only use images with one face/person in it, not more/less
2. Predict face poses/**landmarks** (for the faces identified in step 1)
3. Using data from step 2 and the actual image, calculate face **encodings** (numbers that describe the face)
4. **Compare** the face encodings of known faces with those from test images to tell who is in the picture

Image & Content Source: https://towardsdatascience.com/facial-recognition-using-deep-learning-a74e9059a150#

Application of Deep Learning Neural Network- Face Recognition

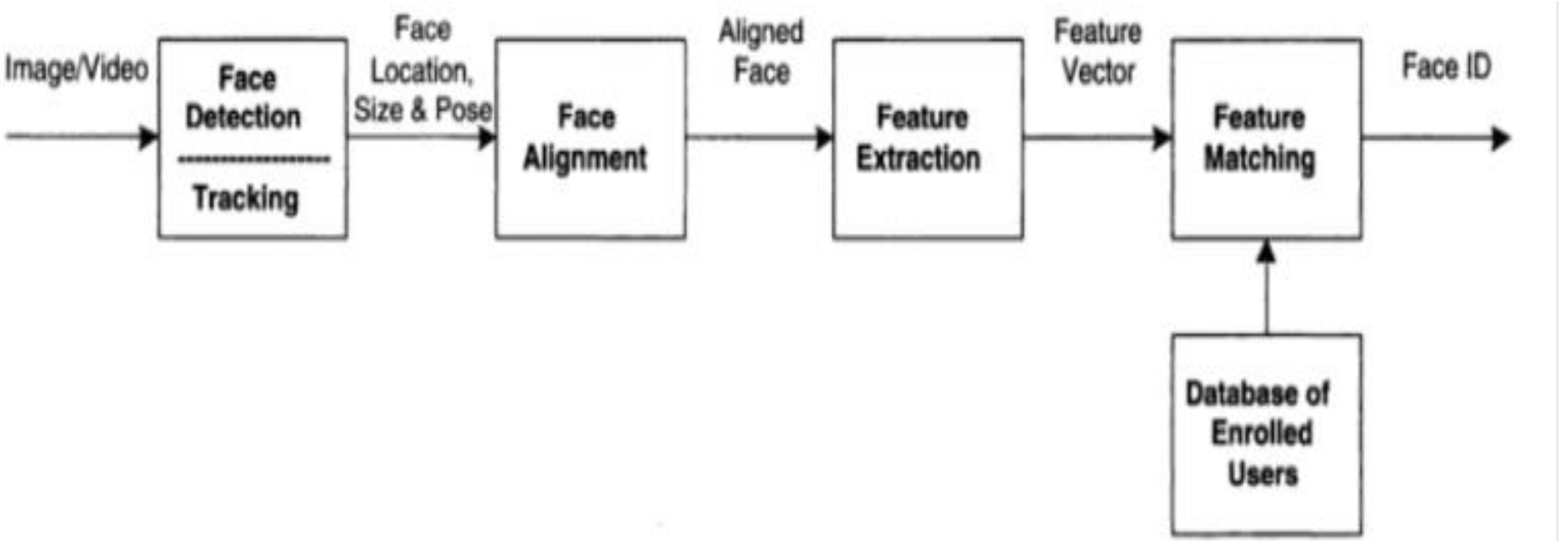


Image & Content Source: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/

Application of Deep Learning Neural Network- Object Recognition

```
Clear;
Load pathToPetFaces.mat;
Petds=image datastore(pathToPetFaces,”IncludeSubFolders”,true);
Load petGT.mat;
Fn = petGT.ImageFilename{};
Im=imread(fn);
Imshow(im);
Bbox= petGT.Madeline{1};
Label=‘Madaline’;
Im=insertObjectAnnotation(im,’rectangle’,bbox,label);
Imshow(im);
Opts=trainingOptions(‘sdgm’,’InitialLearnRate’,0.0001,…
        ‘MaxEpochs’,5,’MiniBatchSize’,1);
Detector=trainFastRCNNObjectDetector(petGT,’alexnet’,opts);
Dogim=imread(‘Ginny43.jpg’)
Imshow(dogim)
[bones,scores,labels]=(detector,dogim)
```

Image & Content Source: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/

Application of Deep Learning Neural Network- Object Recognition

```
For k=1:length(labels)
Label{str}=[char(label),':',num2str(scores)]
Detectedim=InsertObjectAnnotation(dogmim,'rectangles',bboxes,labelstr);
Imshow(detectedim)
```

Image & Content Source: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/

Dr. V. Vedanarayanan B.E., M.E., PhD
Assistant Professor, Department of ECE,
School of Electrical and Electronics
Sathyabama Institute of Science and technology
Vedanarayanan.etc@sathyabama.ac.in