Unit – IV

2 MARKS

Category – 1 (Easy)

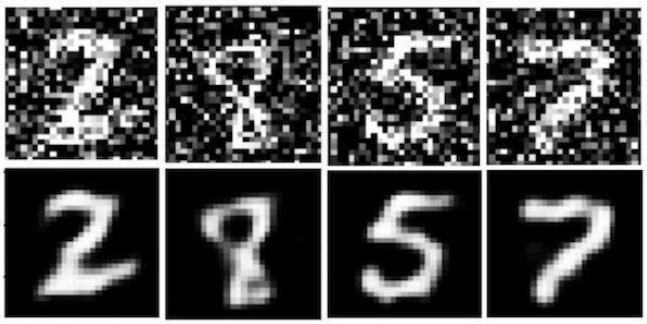
1) What are the different phases of an autoencoder?

Same as qn1 in 8 marks below
Referred: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

2) What are the use cases of autoencoders.

They have 3 common use cases though:

Data denoising: we have seen an example of this on images.



Dimensionality reduction: visualizing high-dimensional data is challenging. t-SNE is the most commonly used method but struggles with large number of dimensions (typically above 32). So autoencoders are used as a preprocessing step to reduce the dimensionality, and this compressed representation is used by t-SNE to visualize the data in 2D space. For great articles on t-SNE refer here and here.

Variational Autoencoders (VAE): this is a more modern and complex use-case of autoencoders. VAE learns the parameters of the probability distribution modelling the input data, instead of learning an arbitrary function in the case of vanilla autoencoders. By sampling points from this distribution we can also use the VAE as a generative model.

More examples- **Image Compression**.; **Image Generation**.; **Feature Extraction**.

Reference: https://www.kaggle.com/shivamb/how-autoencoders-work-intro-and-usecases#2.2-UseCase-2---Image-Denoising

3) Name the popular open source frameworks for deep learning? [choose any 4]

- **TensorFlow** - to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.
- **Keras** - Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **PyTorch** - used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR).
- **Theano** - that allows us to evaluate mathematical operations including multi-dimensional arrays so efficiently. It is mostly used in building Deep Learning Projects. It works a way more faster on Graphics Processing Unit (GPU) rather than on CPU

- **DL4J** - Eclipse Deeplearning4j is the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. Integrated with Hadoop and Apache Spark, DL4J brings AI to business environments for use on distributed GPUs and CPUs.
- **Caffe** - Caffe is being used in academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Yahoo! has also integrated caffe with Apache Spark to create CaffeOnSpark, a distributed deep learning framework.
- **Chainer** - Chainer is a Python-based deep learning framework aiming at flexibility. It provides automatic differentiation APIs based on the define-by-run approach (a.k.a. dynamic computational graphs) as well as object-oriented high-level APIs to build and train neural networks.
- **Microsoft CNTK** - The Microsoft Cognitive Toolkit (CNTK) is a powerful, open source library that can be used to create machine learning prediction models. In particular, CNTK can create deep neural networks, which are at the forefront of artificial intelligence efforts such as Cortana and self-driving automobiles.

4) What are the different hyperparameters of a neural network?

Hyperparameters are the variables which determines the network structure(Eg: Number of Hidden Units) and the variables which determine how the network is trained(Eg: Learning Rate). Hyperparameters are set before training(before optimizing the weights and bias).

Hyperparameters can be roughly divided into 2 categories:

1. Optimizer hyperparameters –

1.1. Learning rate (If the model learning rate is way too smaller than optimal values, it will take a much longer time (hundreds or thousands) of epochs to reach an ideal state. On the other hand, if the learning rate is much larger than optimal value, then it would overshoot the ideal state and the algorithm might not converge. A reasonable starting learning rate = 0.001.)

1.2. Mini-Batch Size (Batch size has an effect on the resource requirements of the training process, speed and number of iterations in a non-trivial way.)

1.3. Number of Epochs (To choose the right number of epochs for the training step, the metric we should pay attention to is the Validation Error.)

2. Model Specific hyperparameters

2.1. Number of hidden units (Number of hidden units is one of the more mysterious hyperparameters. Let's remember that neural networks are universal function approximators, and for them to learn to approximate a function (or a prediction task) , they need to have enough 'capacity ' to learn the function. The number of hidden units is the main measure of model's learning capacity.)

2.2. First hidden layer: Another heuristic involving the first hidden layer is that setting the number of hidden units larger than the number of inputs tends to enable better results in number of tasks, according to empirical observation.

2.3. Number of layers: It is often the case that 3-layer Neural Net will outperform a 2-layer one. But going even deeper rarely helps much more. (exception are Convolutional Neural Networks, where the deeper they are, the better they perform).

Reference: https://medium.com/@jorgesleonel/hyperparameters-in-machine-deep-learning-ca69ad10b981

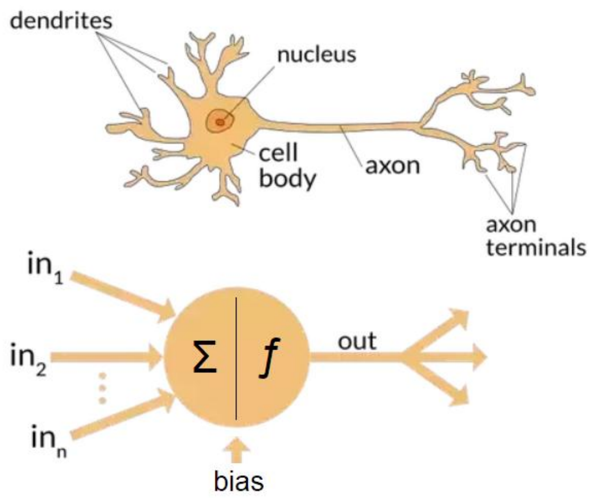
5) What is the function of neuron in a neural network.

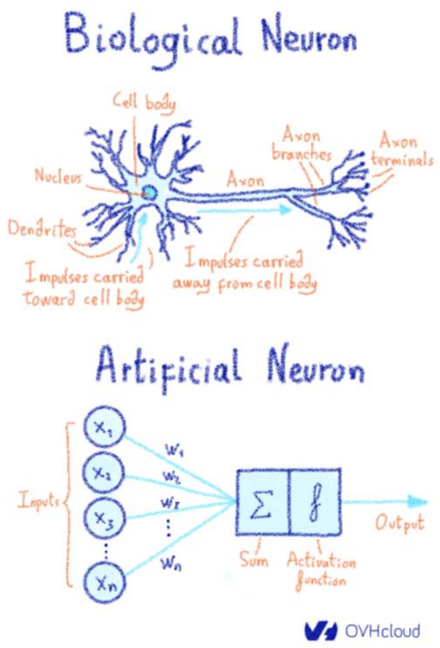
Neural network is a set of neurons organized in layers. Neurons in deep learning models are nodes through which data and computations flow. Neurons work like this: They receive one or more input signals. These input signals can come from either the raw data set or from neurons positioned at a previous layer of the neural net. Each neuron is a mathematical operation that takes it's input, multiplies it by it's weights and then passes the sum through the activation function to the other neurons.

Category – 2 (Moderate)

1) Does artificial neural network function resemble the human brain function ? Justify.

The most obvious similarity between a neural network and the brain is the presence of neurons as the most basic unit of the nervous system. But the manner in which neurons take input in both cases is different. In our understanding of the biological neural network, we know that input is taken in from dendrites and output through the axon. These have significantly different ways of processing input. Research shows that dendrites themselves apply a non-linear function on the input before it is passed to the nucleus. On the other hand, in an artificial neural network, the input is directly passed to a neuron and output is also directly taken from the neuron, both in the same manner.





See this video- https://www.youtube.com/watch?v=1BZm5RAljn0 [10 min]

Reference : https://medium.com/@eraiitk/brain-and-artificial-neural-networks-differences-and-similarities-1d337fe50168

~~~~~~

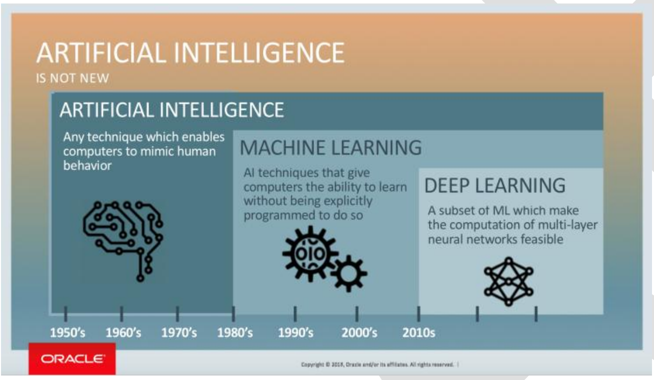
2) When do neural networks perform better compared to standard machine learning algorithms?

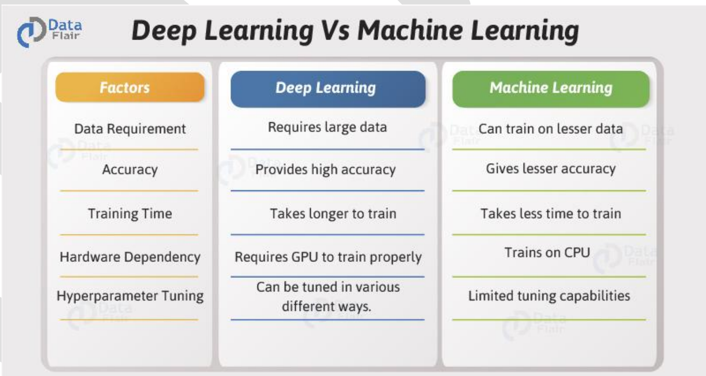
- Machine Learning uses advanced algorithms that **parse data, learns from it**, and use those learnings to discover meaningful patterns of interest. Whereas a Neural Network consists of an **assortment of algorithms** used in Machine Learning for data modelling using graphs of neurons .
~~~~~~

- While a Machine Learning model makes decisions according to what it has learned from the data, a Neural Network arranges algorithms in a fashion that it can make **accurate decisions by itself**. Thus, although Machine Learning models can learn from data, in the initial stages, they may require some human intervention. Neural networks **do not require human intervention** as the nested layers within pass the data through hierarchies of various concepts, which eventually makes them capable of learning through their own errors.
- As we mentioned earlier, Machine learning models can be **categorized** under two types – *supervised and unsupervised learning models*. However, Neural Networks can be **classified** into *feed-forward, recurrent, convolutional, and modular Neural Networks.*
- An ML model works in a simple fashion – it is fed with data and learns from it. **With time**, the ML model **becomes more mature and trained as it continually learns from the data**. On the contrary, the structure of a Neural Network is quite complicated. In it, the data passes through several layers of interconnected nodes, wherein *each node classifies the characteristics and information of the previous layer* **before passing the results** on *to other nodes* in subsequent layers.
- Since Machine Learning models are **adaptive**, they are **continually evolving** by learning through new sample data and experiences. Thus, the models can **identify the patterns in the data**. **Here, data is the only input layer**. However, even in a simple Neural Network model, there are multiple layers. The **first layer is the input layer,** followed by a **hidden layer,** and then finally an **output layer**. Each layer contains one or more *neurons*. By **increasing the number of hidden layers** within a Neural Network model, **you can increase its computational and problem-solving abilities.**

\~~~~~~

3) Differentiate between machine learning and deep learning.





Deep Learning Vs Machine Learning

Factors	Deep Learning	Machine Learning
Data Requirement	Requires large data	Can train on lesser data
Accuracy	Provides high accuracy	Gives lesser accuracy
Training Time	Takes longer to train	Takes less time to train
Hardware Dependency	Requires GPU to train properly	Trains on CPU
Hyperparameter Tuning	Can be tuned in various different ways.	Limited tuning capabilities

\~~~~~~

4) Can we use neural networks for linear regression? How many number of neurons in the output layer and the function performed in the output layer.

Yes, we can

Generally, to do a simple regression problem you can use **a feed-forward network** with **M** input pairs of (X,y) where X is a vector of parameters. y is a scalar, which is approximated by a single node on the output layer.

- *First* you need **to choose your activation functions** for the hidden and output layer(s) (linear, sigmoid etc.).
- Finally, **train** your network on **labelled** x,y pairs,
- **test against your test set to check accuracy**, and
- then **run against unlabeled data to find approximate y values.**

Very similar to a classification problem, **except** your *y is scalar instead of a vector* and values are **not** constrained to be 0<=y<=1 . I would suggest modelling linear regression first, (for example, two inputs one output, no hidden layer).

Reference: https://www.quora.com/How-do-I-use-neural-network-to-do-the-regression-problem-rather-than-classification

---

5) Write the major phases in building a Deep learning model.

Step 1 — Data Pre-processing. ...

Step 2 — Separating Your Training and Testing Datasets. ...

Step 3 — Transforming the Data. ...

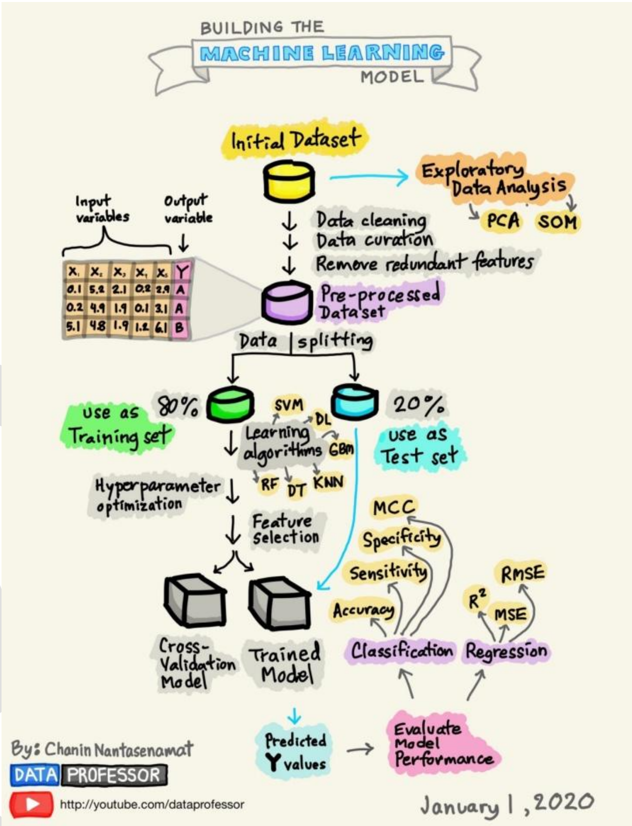
Step 4 — Building the Artificial Neural Network. ...

Step 5 — Running Predictions on the Test Set. ...

Step 6 — Checking the Confusion Matrix. ...

Step 7 — Making a Single Prediction.

Reference:
https://www.digitalocean.com/community/tutorials/how-to-build-a-deep-learning-model-to-predict-employee-retention-using-keras-and-tensorflow



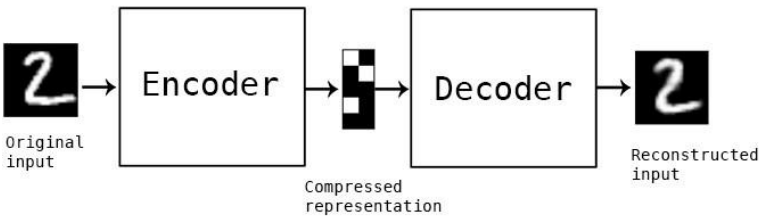
---

8 MARKS

Easy Questions

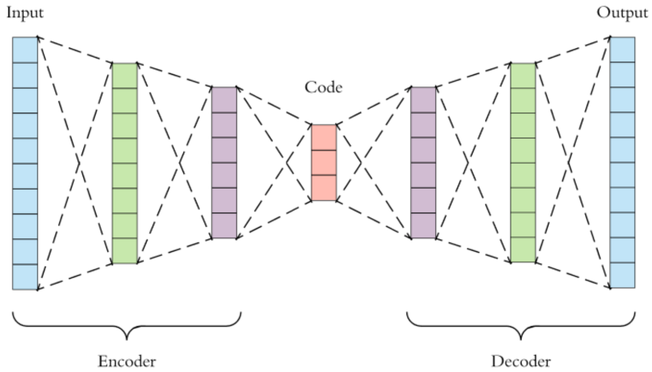
1. Differentiate between Encoding and Decoding phases in autoencoders?

Autoencoder is an **unsupervised** artificial neural network that learns how to **efficiently compress** and **encode** data **then learns** how to **reconstruct the data back from the reduced encoded representation** to a r**epresentation that is as close to the original input** as possible. Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data. **Autoencoder Components**: An autoencoder consists of 3 components: **encoder**, **code** and **decoder**. The *encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code*.



To **build an autoencoder we need 3 things: an encoding method, decoding method, and a loss function to compare the output with the target.**

1- **Encoder**: In which the model learns how to reduce the input dimensions and compress the input data into an encoded representation.
2- **Decoder**: In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.
The training then involves **using back propagation** in order to **minimize the network's reconstruction loss**. Both the encoder and decoder **are fully-connected feed forward neural networks**. **Code** is a **single layer of an ANN** with the dimensionality of our choice. The **number of nodes in the code layer** (code size) is a **hyperparameter** that we set before training the autoencoder.

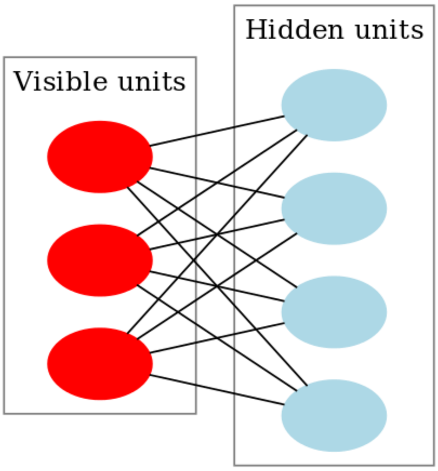


First the input passes through the **encoder**, which is **a fully-connected ANN**, to produce the **code**. The decoder, which has the similar ANN structure, then **produces the output** only using the **code**. The goal is to get an **output identical** with the **input**. Note that the decoder architecture is the mirror image of the encoder. This is *not a requirement* but it's typically the case. The only requirement is the dimensionality of the input and output needs to be the same. Anything in the middle can be played with.
Referred: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
~~~~~~

2. What is a restricted boltzmann machine? What is its function?

RBMs are a **two-layered artificial neural network** with generative capabilities. They have the **ability to learn a probability distribution over its set of input**. RBMs were invented by Geoffrey Hinton and can be used for *dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modelling*.

- RBMs are a special class of Boltzmann Machines and **they are restricted in terms of the connections between the visible and the hidden units**.
- This makes it **easy to implement them** when compared to Boltzmann Machines.
- As stated earlier, they are a two-layered neural network (one being the *visible layer* and the other one being the *hidden layer*) and these two layers are **connected by a fully bipartite graph**. This means that *every node in the visible layer is connected to every node in the hidden layer but no two nodes in the same group are connected to each other.*
- This restriction allows for more efficient training algorithms than what is available for the general class of Boltzmann machines, in particular, the gradient-based contrastive divergence algorithm.
~~~~~~



A Restricted Boltzmann machine is an algorithm useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modelling.

Reference: https://adityashrm21.github.io/Restricted-Boltzmann-Machines/ [if u want to know how it works]

~~~~~~

3. What is the purpose of activation functions in neural networks? Write the popular activation functions.

An activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data. When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron.

1. Binary Step Function - if the input to the activation function is greater than a threshold, then the neuron is activated, else it is deactivated, i.e. its output is not considered for the next hidden layer.

```
def binary_step(x):

    if x<0:

        return 0

    else:

        return 1

binary_step(5), binary_step(-1)
```

Output: `(5,0)`

2.Linear Function

We saw the problem with the step function, the gradient of the function became zero. This is because there is no component of x in the binary step function. Instead of a binary function, we can use a linear function. We can define the function as-
~~~~~~

f(x)=ax

```
def linear_function(x):

    return 4*x

linear_function(4), linear_function(-2)
```

Output: `(16, -8)`

3. Sigmoid

The next activation function that we are going to look at is the Sigmoid function. It is one of the most widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1. Here is the mathematical expression for sigmoid- > f(x) = 1/(1+e^-x)

4. Tanh

The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign. The tanh function is defined as- > tanh(x)=2sigmoid(2x)-1

Referred: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/ [**MANY TYPES AND WITH EXAMPLES ARE THERE**]

~~~~~~

4. What are the different types of sequence data? Write the use cases of sequence models?

Sequences allow you to store multiple values in an organized and efficient fashion. There are several sequence types: strings, Unicode strings, lists, tuples, bytearrays, and range objects. Dictionaries and sets are containers for non-sequential data.

From the official Python Docs –

Strings are immutable sequences of Unicode code points.

Lists are mutable sequences, typically used to store collections of homogeneous items.

Tuples are immutable sequences, typically used to store collections of heterogeneous data (such as the 2-tuples produced by the enumerate() built-in.

Bytearray objects are mutable, they support the mutable sequence operations in addition to the common bytes and bytearray operations

The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.

Reference: https://www.python-course.eu/sequential_data_types.php

~~~~~~

Moderate Questions

1. On what basis are the sequence models categorized? Give examples for each category.

RNN: https://www.analyticsvidhya.com/blog/2019/01/sequence-models-deeplearning/

---

2. Does a neural network perform feature representation learning? How?

Yes, and Multilayer neural networks can be used to perform feature learning, since they learn a representation of their input at the hidden layer(s) which is subsequently used for classification or regression at the output layer. The most popular network architecture of this type is Siamese networks.

https://www.youtube.com/watch?v=8dKD9Ybgmyk [**Feature Representation Learning and it's types**]

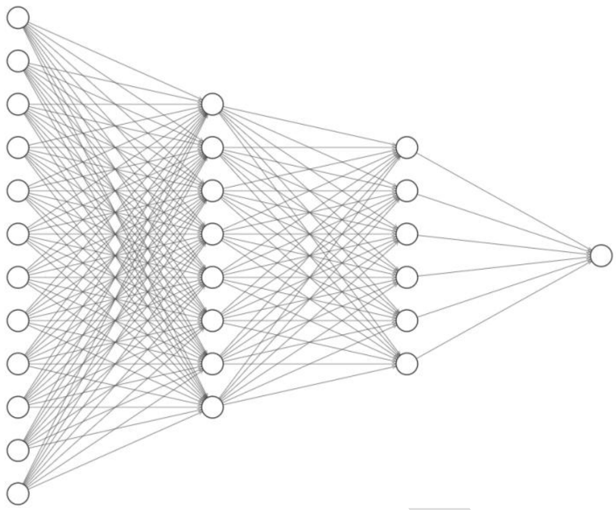
https://www.youtube.com/watch?v=6jfw8MuKwpI [**Siamese Network**]

---

3. Are the neural network layers same for any type of input? Which layers are preferred for different types of input?

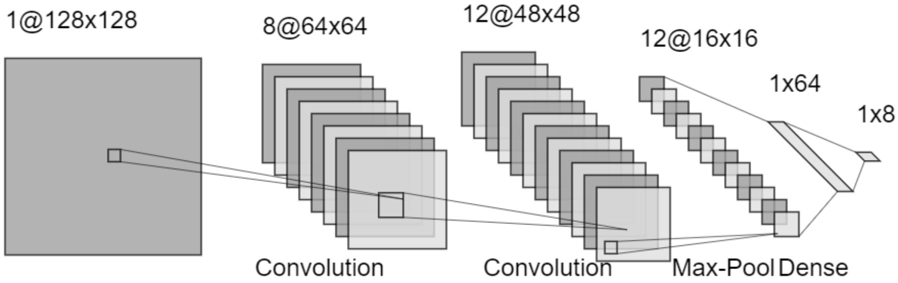
No,

1. Deep Neural Networks (DNN):



These are the fully connected neural networks that are used for classification and regression tasks. These are also sometimes attached to the end of certain more advance architectures

2. Convolution Neural Networks (CNN): These are mostly used to process image data for various computer vision applications such as image detection, image classification, semantic segmentation, etc. Since image data is a multi-dimensional data, it requires different types of processing layers that can detect the most important features of the image. This is what convolutional neural networks do.



3. Recurrent Neural Networks (RNN): These neural networks are used to process sequential data, or data where the current output depends not only on current input, but also on the previous input. It is used for time series forecasting, natural language processing, etc. A unique feature of RNNs is that it contains gates, which allow or omit input from the previous hidden states to be added to the current input, completely configurable by the user. There are three main types of RNN. The most basic one is the "vanilla" RNN, which contains one gate. The other two are the Long Short-term Memory units (LSTM), and the Gated Recurrent Unit (GRU). While the LSTM has 4 gates, the GRU has 3, which makes it computationally faster than LSTM units.

RNN: https://www.analyticsvidhya.com/blog/2019/01/sequence-models-deeplearning/

Reference: https://towardsdatascience.com/ultimate-guide-to-input-shape-and-model-complexity-in-neural-networks-ae665c728f4b

~~~~~~

Standard Questions

1. Draw neural network architecture for images of, two types of animals, where image size is 28 X 28. Use five layers?

~~~~~~

2. Write the applications of deep neural networks. [CHOOSE ANY 8]

Top Applications of Deep Learning across Industries

Self Driving Cars: Deep Learning is the force that is bringing autonomous driving to life. A million sets of data are fed to a system to build a model, to train the machines to learn, and then test the results in a safe environment.

News Aggregation and Fraud News Detection: Extensive use of deep learning in news aggregation is bolstering efforts to customize news as per readers. While this may not seem new, newer levels of sophistication to define reader personas are being met to filter out news as per geographical, social, economical parameters along with the individual preferences of a reader. Fraud news detection, on the other hand, is an important asset in today's world where the internet has become the primary source of all genuine and fake information. It becomes extremely hard to distinguish fake news as bots replicate it across channels automatically.

Natural Language Processing: Understanding the complexities associated with language whether it is syntax, semantics, tonal nuances, expressions, or even sarcasm, is one of the hardest tasks for humans to learn. Constant training since birth and exposure to different social settings help humans develop appropriate responses and a personalized form of expression to every scenario. Natural Language Processing through Deep Learning is trying to achieve the same thing by training machines to catch linguistic nuances and frame appropriate responses. Document

summarization is widely being used and tested in the Legal sphere making paralegals obsolete. Answering questions, language modelling, classifying text, twitter analysis, or sentiment analysis at a broader level are all subsets of natural language processing where deep learning is gaining momentum

Virtual Assistants: The most popular application of deep learning is virtual assistants ranging from Alexa to Siri to Google Assistant. Each interaction with these assistants provides them with an opportunity to learn more about your voice and accent, thereby providing you a secondary human interaction experience. Virtual assistants use deep learning to know more about their subjects ranging from your dine-out preferences to your most visited spots or your favourite songs. They learn to understand your commands by evaluating natural human language to execute them. Another capability virtual assistants are endowed with is to translate your speech to text, make notes for you, and book appointments. Virtual assistants are literally at your beck-and-call as they can do everything from running errands to auto-responding to your specific calls to coordinating tasks between you and your team members. With deep learning applications such as text generation and document summarizations, virtual assistants can assist you in creating or sending appropriate email copy as well.

Entertainment (VEVO, Netflix, Film Making, Sports Highlights, etc.): RECOMMENDATION SYSTEM Deep Learning, they were able to factor in audience response and match or player popularity to come up with a more accurate model (otherwise it would just have highlights of the most expressive or aggressive players). Netflix and Amazon are enhancing their deep learning capabilities to provide a personalized experience to its viewers by creating their personas factoring in show preferences, time of access, history, etc. to recommend shows that are of liking to a particular viewer.

Visual Recognition: In comes, Deep Learning and now images can be sorted based on locations detected in photographs, faces, a combination of people, or according to events, dates, etc. Searching for a particular photo from a library (let's say a dataset as large as Google's picture library) requires state-of-the-art visual recognition systems consisting of several layers from basic to advanced to recognize elements. Large-scale image Visual recognition through deep neural networks is boosting growth in this segment of digital media management by using convolution neural networks, Tensorflow, and Python extensively.

Fraud Detection: deep Learning is the banking and financial sector that is plagued with the task of fraud detection with money transactions going digital. Autoencoders in Keras and Tensorflow are being developed to detect credit card frauds saving billions of dollars of cost in recovery and insurance for financial institutions. Fraud prevention and detection are done based on identifying patterns in customer transactions and credit scores, identifying anomalous behaviour and outliers. Classification and regression machine learning techniques and neural networks are used for fraud detection.

Healthcare: Helping early, accurate and speedy diagnosis of life-threatening diseases, augmented clinicians addressing the shortage of quality physicians and healthcare providers, pathology results and treatment course standardization, and understanding genetics to predict future risk of diseases and negative health episodes are some of the Deep Learning projects picking up speed in the Healthcare domain.

Personalisations: Deep Learning is empowering efforts of e-commerce giants like Amazon, E-Bay, Alibaba, etc. to provide seamless personalized experiences in the form of product recommendations, personalized packages and discounts, and identifying large revenue opportunities around the festive season. Even recce in newer markets is done by launching products, offerings, or schemes that are more likely to please the human psyche and lead to growth in micro markets

SOME MORE ARE:

Detecting Developmental Delay in Children **Colourisation of Black and White images**

Adding sounds to silent movies

Automatic Machine Translation

Automatic Handwriting Generation

Automatic Game Playing

Language Translations

Pixel Restoration

Photo Descriptions

Demographic and Election Predictions

Deep Dreaming

Reference: https://www.mygreatlearning.com/blog/deep-learning-applications/

---

3. Differentiate between deep boltzmann machine and Deep Belief Networks?

Deep belief network

A Deep Belief Network (DBN) is a powerful generative model that uses a deep architecture of multiple stacks of Restricted Boltzmann machines (RBM).

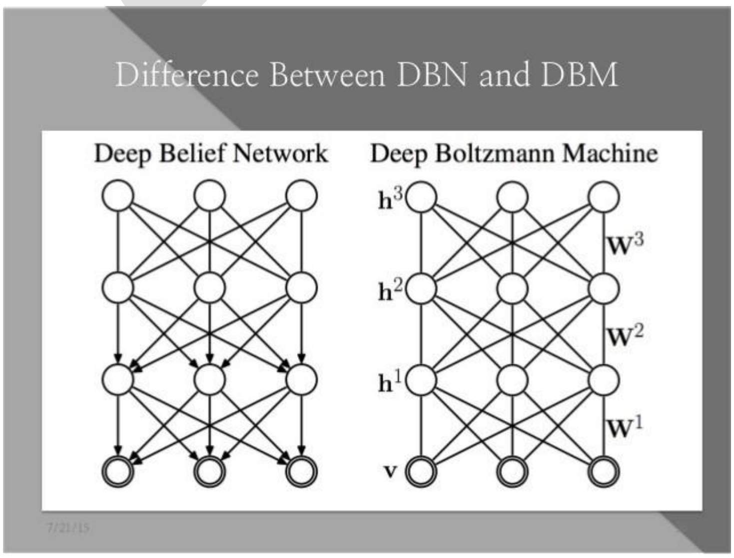
Each RBM model performs a non-linear transformation (much like a vanilla neural network works) on its input vectors and produces as outputs vectors that will serve as input for the next RBM model in the sequence.

This allows lot flexibility to DBNs and makes them easier to expand.

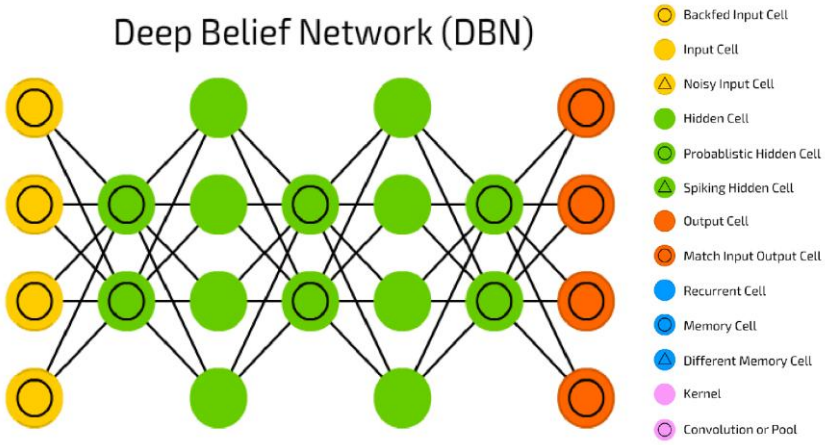
Being a generative model allows DBNs to be used in either an unsupervised or a supervised setting. Meaning, DBNs have the ability to do feature learning/extraction and classification that are used in many applications, more on this in the applications section.

Precisely, in feature learning we do layer-by-layer pre-training in an unsupervised manner on the different RBMs that form a DBN and we use back-propagation technique (i.e. gradient descent) to do classification and other tasks by fine-tuning on a small labelled dataset.

Deep Boltzmann Machine



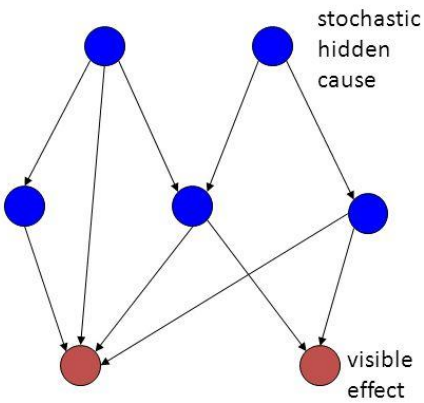
After DBNs another model called Deep Boltzmann Machine (DBM) was created that trains better and achieves a lower loss, although it had some issues like being hard to generate sample from. A DBM is a three-layer generative model. They are similar to a Deep Belief Network, but they while DBNs have bidirectional connections in the bottom layer on the other hand DBM has entirely undirected connections. Now that we are equipped with the theory it is time to dive into the implementation details. A deep belief network consists of a sequence of restricted boltzmann machines which are sequentially connected. Each of the Boltzmann machines layers is trained until convergence, and then frozen; the result of the "output" layer of the machine is then fed as input to the next Boltzmann machine in the sequence, which is then itself trained until convergence, and so forth, until the entire network has been trained. The following much-shared diagram shows the overall architecture:



Notice that besides the output layer, every pair of layers in the hidden plus input layers compose a RBM

Belief Nets

- A belief net is a directed acyclic graph composed of stochastic variables.
- Can observe some of the variables and we would like to solve two problems:
- The inference problem: Infer the states of the unobserved variables.
- The learning problem: Adjust the interactions between variables to make the network more likely to generate the observed data.



Use nets composed of layers of stochastic binary variables with weighted connections. Later, we will generalize to other types of variable.