Building Software Systems

Lecture 5.4

Applications of Deep Learning

SAURABH SRIVASTAVA

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT (ISM) DHANBAD

What is Deep Learning?

Deep Learning (DL) refers to the Machine Learning (ML) techniques using “deep” Neural Networks

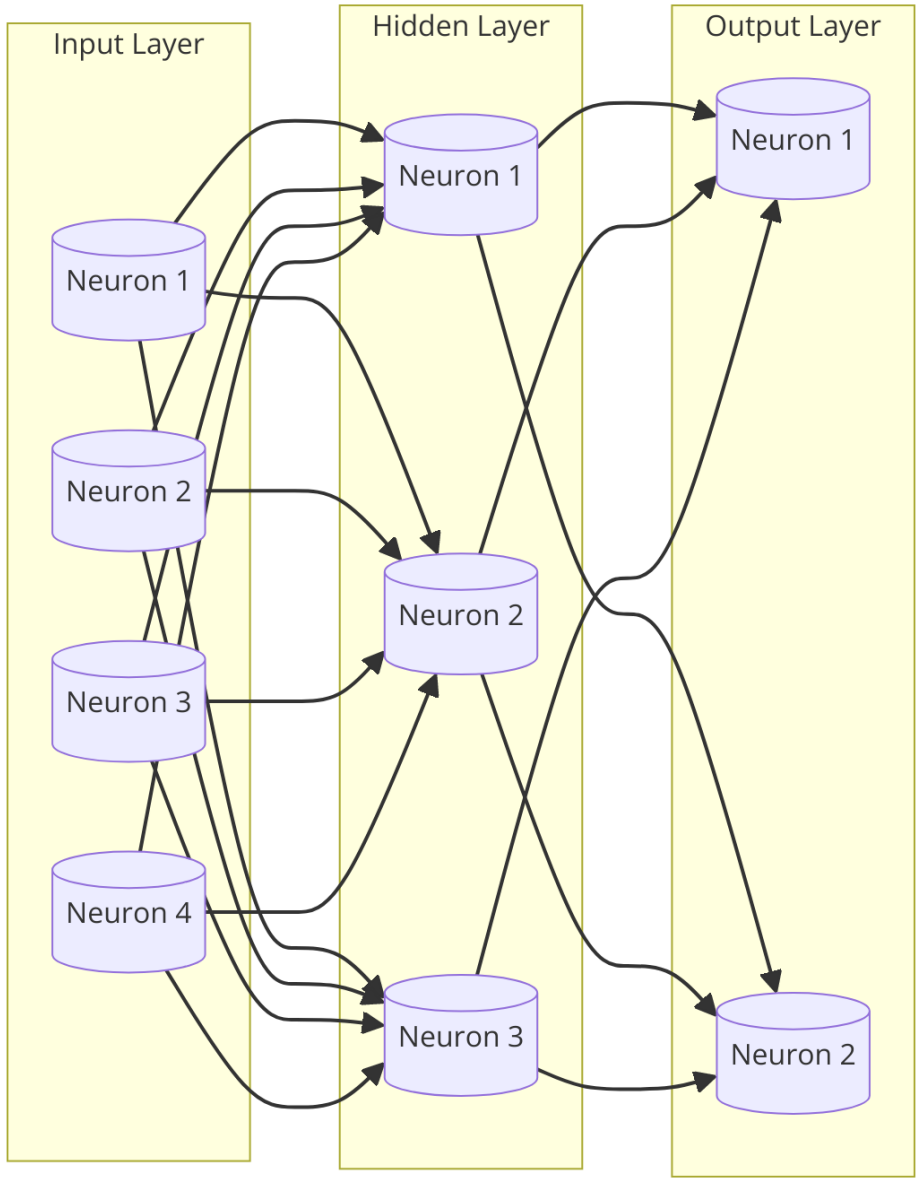
- A Neural Network is an imitation of the human brain, where multiple “neurons” try to collaboratively learn
- There are layers to absorb the input, layers to produce the output, and some hidden (intermediate) layers
- The term “deep” is somewhat subjective, but usually the network must have at least one hidden layer
- Usually, the number of layers are far more than that !!

What does this “depth” offer us in terms of learning?

- The traditional ML techniques usually require significant preprocessing (e.g., feature selection) to be effective
- DL methods, on the other hand, have been shown to process huge amounts of relatively unprocessed data
- This has allowed application of ML to problem statements where learning seemed rather intractable
- These include problem statements related to complex domains such as Vision and Natural Languages

We will not discuss DL methods mathematically in this lecture

- Instead, we will look at their usefulness for different problem domains which will be beneficial for you



Example: A Neural Network with One Hidden Layer

The concept of a “DL Architecture” (1/3)

Layers

- A Layer represents a collection of Neurons that perform “the same task” in the network
- The task could be taking inputs, producing outputs or performing a specific intermediate step

Input Layer

- Receives raw data; the initial point of data entry.

Hidden Layers

- Intermediate layers that process inputs from the previous layer using weights, biases, and activation functions

Output Layer

- Produces the final output, such as classification or prediction results

The concept of a “DL Architecture” (2/3)

Neurons or Nodes

- The basic unit of a network where some learning can be performed
- A neuron can be seen as the application of some transformations
- These transformations take “incoming” values (inputs) and produce “outgoing” values (outputs)
- In the process of training the network, some weights are calculated in an iterative fashion …
- … which determined the transformation performed on the neuron

Activation Functions

- Activation functions allow addition of “non-linearity” in the flow
- Non-linearity here means that the output is not a simple, linear function of the inputs
- Instead, it may be dependent on a non-linear function, allowing the learning to learn more complex relations
- The most common Activation Function is ReLU:
  $F(x) = \max(0, x)$

The concept of a “DL Architecture” (3/3)

The core idea of applying DL to a problem is to pick a suitable “DL Architecture”

The DL Architecture essentially answers the following questions

- How many neurons are present in the Input and Output Layers?
- How many Hidden layers does the architecture has?
- How many neurons does each layer has?
- What Activation Function(s) are we going to use? Etc.

In general, you can design any DL architecture of your choice

- In fact, you may be tempted to have a “lot of hidden layers” and/or a “lot of neurons in each layer”
- But having more layers or neurons doesn’t mean that you are on the right path
- It is because you can easily “overfit” the training data (the network does well on training data, but not in general)

Researchers often start with some well-known DL architecture

- You can find in literature that certain types of architectures usually do better for certain types of problems

Convolution Neural Network (CNN)

CNN architectures use a fundamental mathematical operation – Convolution

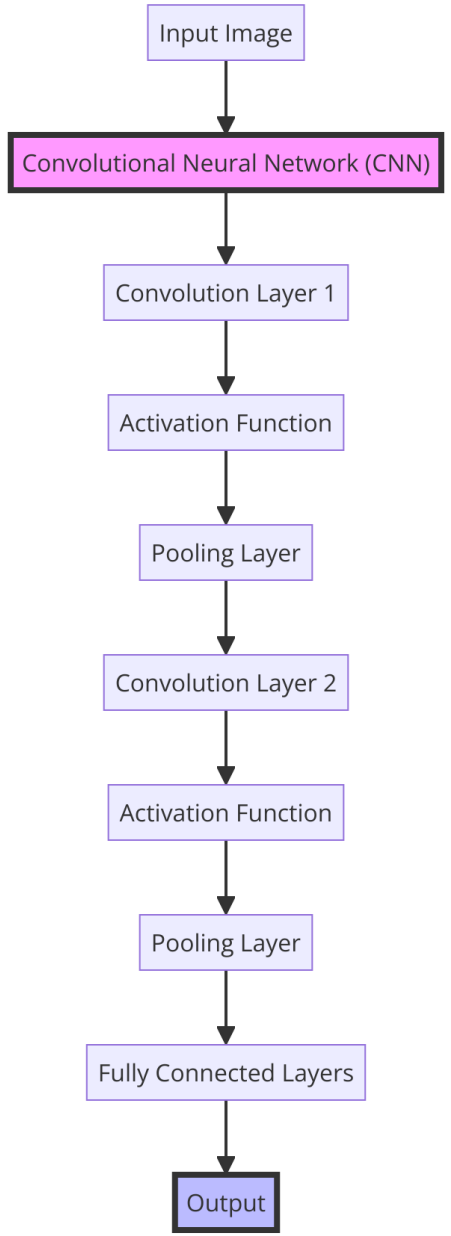
- Convolution involves combining two different functions to provide a third one
- In Image Processing, it usually means applying a “filter” over a “tile” of pixel values (if you are new to CNNs, please check the Further Reading Section)

There are four important aspects of CNN architectures

- Convolution Layers – The neurons here perform convolution operations
- Pooling Layers – The neurons here reduce the size of previous layers (usually by a process called max pooling)
- Use of ReLU Activation functions
- Fully-connected Layers – Next to the Output Layer, performs the actual classification task

CNNs are usually deployed for tasks such as Object Detection and Object Classification

- By extension, they can also be used on videos and in the fields of Computer Vision



Example: A Sample CNN Architecture

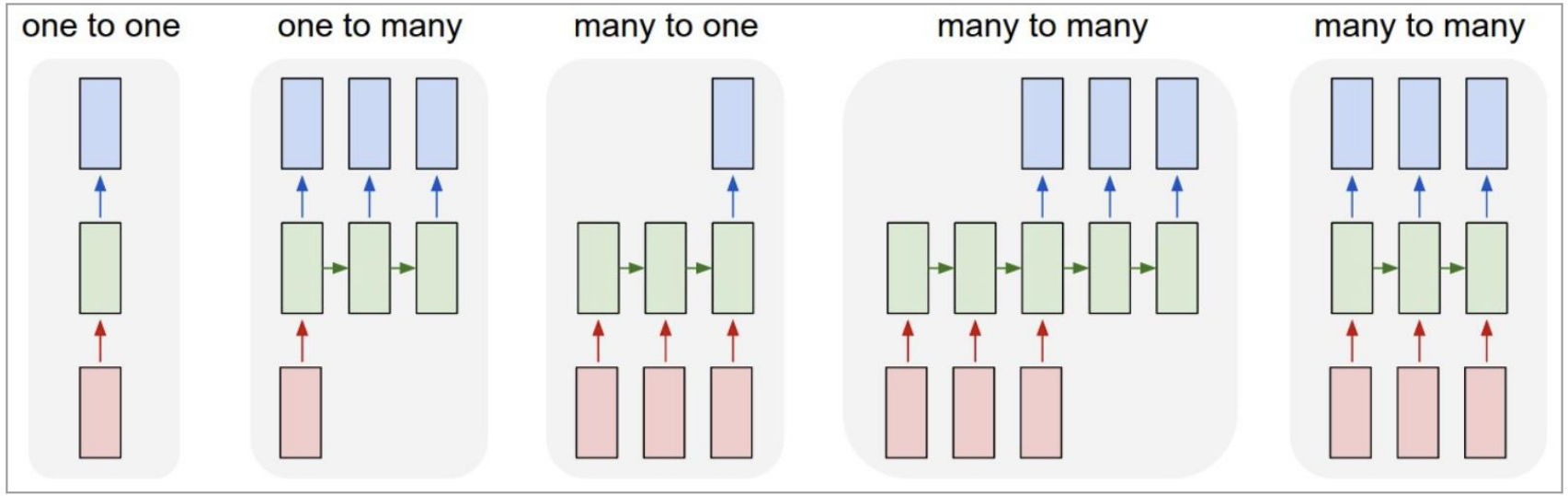
Recurrent Neural Network (RNN)

RNN architectures employ a feedback mechanism within the network

- This means that some or all the outputs from a layer is fed back to the same layer as input
  (if you are new to RNNs, please check the Further Reading Section)
- The other kind of Neural Networks, where this feedback mechanism is absent are called feed-forward networks
- This mechanism essentially provide a kind of “memory” to the networks
- RNNs are significantly more complex than CNNs because of the presence of the memory elements
- They are useful for problems where the data is in the form of time series (i.e., there is an inherent sequence)

Probably the most common RNN architecture is Long Short-Term Memory (LSTM)

- LSTMs are commonly used in Natural Language Processing, especially, Natural Language Generation
  (if you are new to LSTMs, please check the Further Reading Section)
- It is because the next token to “generate” is usually dependent on the previous generated tokens



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

Different Types of Recurrences

Source: The Unreasonable Effectiveness of Recurrent Neural Networks

Transformers

Transformers are probably the most popular DL architectures today

- It is beyond our scope to discuss it in this course – so please have a look at the Further Reading section

OpenAI's ChatGPT, Google's BERT, Microsoft's Copilot are all based on Transformer architecture

- In fact, GPT stands for "Generative Pre-trained Transformer"

Transformers are particularly useful for building what are called Large Language Models (LLMs)

Some LLMs which are based on the Transformer architecture are

- GPT-3.5 and GPT-4 by OpenAI (it may be noted that they are proprietary)
- Llama 2 by Meta (free but not open source)
- Falcon 40B (free and open source)

Further Reading

Go through the Google's Machine Learning Crash Course if you are new to the field

- https://developers.google.com/machine-learning/crash-course/ml-intro

Some resources for you to get a basic idea of the different DL architectures:

- CNN
  https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks
- RNN
  https://karpathy.github.io/2015/05/21/rnn-effectiveness/
- LSTM
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Transformers
  https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0