FADML (CS60020)

Instructor: Aritra Hazra

Scribes By

Adwaith Aravind A (24BM6JP02)

Lecture Date: 18^{th} March 2025 , Tuesday

Part B

Recap of Backpropagation for Layers l and $l-1$

In a neural network, backpropagation helps us figure out how to update the weights and biases so the model gets better at making predictions. Let's focus on how this works between two layers: layer l and the one just before it, layer $l-1$.

1. What happens during the forward pass?

First, during the forward pass, we compute:

- The **pre-activation** of layer l:
$$z^l = W^l a^{l-1} + b^l$$
- The **activation** (i.e., output) of layer l using an activation function σ:
$$a^l = \sigma(z^l)$$

2. Backpropagating the error

Now during backpropagation, we work backwards from the output to compute how much each parameter contributed to the final error (loss).

- First, we define an **error term** for layer l:
$$\delta^l = \frac{\partial \mathcal{L}}{\partial z^l}$$
This tells us how much the output of this layer affected the final loss.
- Next, we calculate the gradients for the parameters in this layer:
  - Gradient with respect to weights:
$$\frac{\partial \mathcal{L}}{\partial W^l} = \delta^l (a^{l-1})^T$$
  - Gradient with respect to biases:
$$\frac{\partial \mathcal{L}}{\partial b^l} = \delta^l$$
- Then, to continue the backpropagation to the previous layer $l-1$, we compute its error term:
$$\delta^{l-1} = (W^l)^T \delta^l \odot \sigma'(z^{l-1})$$
Here, $\odot$ represents element-wise multiplication, and $\sigma'(z^{l-1})$ is the derivative of the activation function used in layer $l-1$.

In short, this process helps us update every weight and bias in the network by figuring out how much each one contributed to the error.

1 Deep Learning

The concept of deep learning is driven by artificial neural networks. In term of neural networks let x_1 and x_2 be input variables.,let bias be w_0 with weight 1. The feed-forward neural network is shown in Fig1.

The neural network classifier categorizes the input by defining decision boundaries. By stacking multiple layers in the neural network, we can create complex boundaries that effectively segregate data points. The boundaries used to classify data points with a neural network are shown in Fig1.

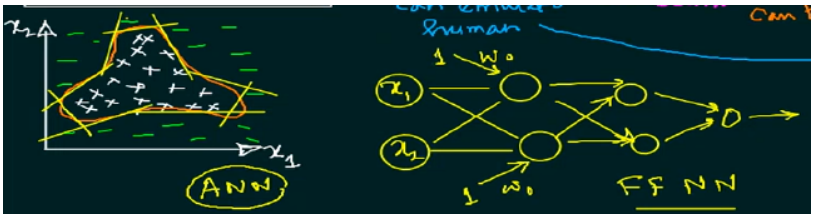


Figure 1: Feed Forward Neural Network

In feed-forward neural networks every layer takes input from every other previous layer. In Fig(2) we can see that layer l_{i-1} feeds input to layer l_i.The two neurons in layer l_i will compute $\theta(\Sigma w_i * x_i)$ and $\theta(\Sigma w_i' * x_i)$. In this model of neural network spatial information is lost. So we extract out features and learn from them using each node.

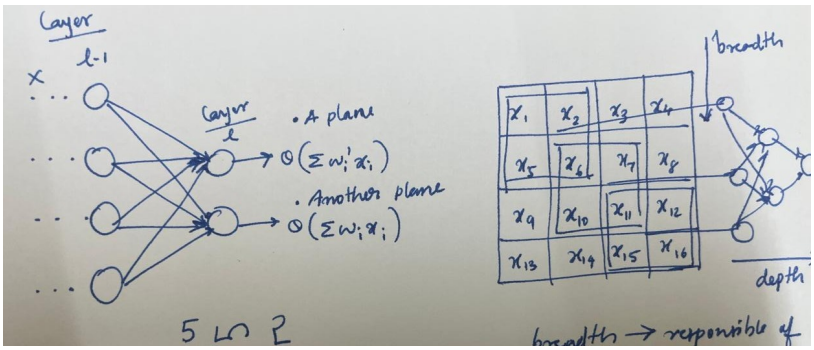


Figure 2: Feature Extraction

1.1 Feature Extraction

In feature extraction, decisions are made based on the composition of extracted features. Consider an image, which is essentially a collection of pixel values at each location.Suppose we have a 4x4 image of pixel value $x_1, \ldots, x_{16}$ as seen in Fig.2. When we feed this image into the network, we focus on the positional relationships between pixels rather than the entire image as a whole. This allows us to extract high-level features that capture essential patterns and structures.

The depth of a neural network is the number of layers in the network.It is responsible for making informed decisions.Initial layers extract low level features, while deeper layers gain information and compose high level features from initial layers. The breadth of each layer refers to the number of neurons within it. This determines the network's ability to distinguish meta-features. In a vision-based system, drawing an analogy to human perception, the breadth represent how many different zones we are looking at and trying to make decisions. As we move deeper into the network, features are progressively composed to form a more abstract understanding of the input. Feature extraction in neural networks is often hierarchical, meaning that simpler patterns are learned first, and these are gradually combined to capture complex structures. At each level,

1. At level 1 we extract the edge of the image.

2. At level 2 we extract boundaries of separating edges.
3. At level 3 we extract shape and separate the object.
4. At level 4 we see association of object.

By feature extraction we can tackle the following properties of data:

1. Orientation
2. Brightness
3. Occulsion
4. Shades

2 Convolutional Neural Network

In a convolutional neural network ,features are extracted from an image to understand objects, and by composing these features, we can recognize them effectively. Suppose X can be written in two ways (X and χ). The images with pixels are shown in Fig3. The pixel which are black(shaded) is $+1$ and pixel which are white are marked as -1. Now we do a feature-wise comparison (enclosed in yellow box in$X$ and χ orange box) in Fig3. To build a neural network we need to decide which extracted unit to take forward.

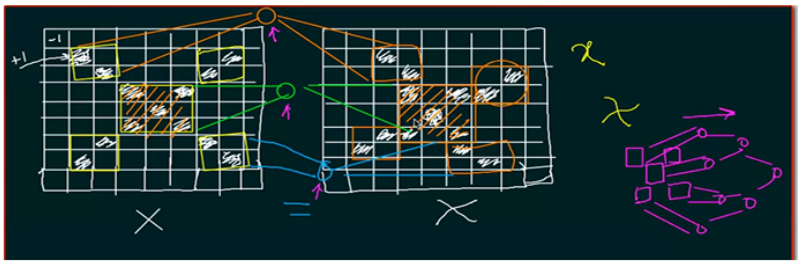


Figure 3: Pixel image of X and χ

The operations required to build a neural network are:

1. **Convolution operation:** Suppose there is 3x3 kernel matrix: $\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$

   If we keep on scanning 3x3 zones overlapping-wise and perform matrix multiplication of each zone with the kernel matrix we get the maximum value as 9. No other zone will get this value. This will give us centre of χ. Hence, on convoluting the zone, the zone which give us the maximum value will be the centre of 'X'.

2. **ReLU (Rectified Linear Unit):** The output of convolution operation i.e., feature extracted will have some values that will be significantly higher and there will be some values that will be significantly lower. We eliminate these insignificant values and replace them with zero.

   $$val = \max(0; val)$$

   The ReLU makes the feature prominent. Suppose we are trying to recognize the car. The CNN to identify the car is shown in Fig4.
   How to select kernel? At each layer of CNN, we train weights. Along with that we need to train the value of kernel. This thing can be applied only if we have a huge amount of training examples. The choice of kernel should dictate whether we want to smoothen or sharpen the image.

   Demerits of CNN:

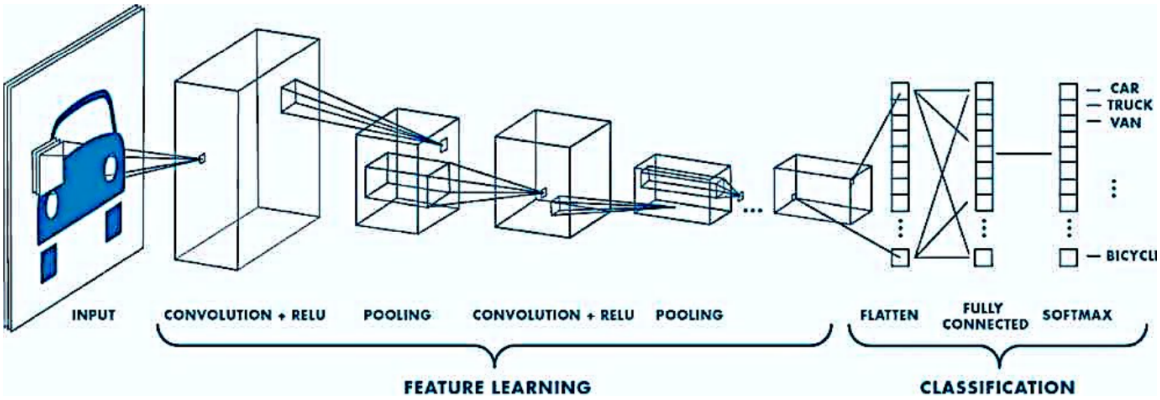


Figure 4: CNN to identify car

- Training time is high.
- Requires high amount of training examples.
- Lost of Explainability aspect.

Applications of CNN: Breast Cancer Detection.

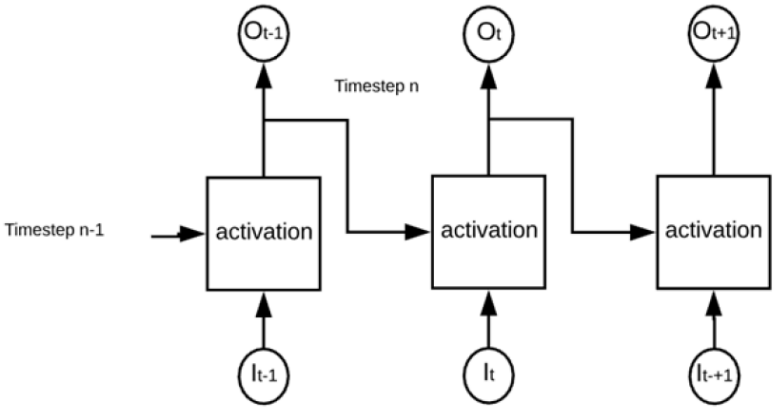
However, in many cases, modeling must be designed to capture information that is not present in a single instance but rather in the relationships across the entire text.This means that context often exhibits temporal dependencies, where understanding one part requires considering its connection to other parts.

To address this, we use a model known as a Recurrent Neural Network (RNN), which is specifically designed to process sequential data while preserving contextual relationships over time

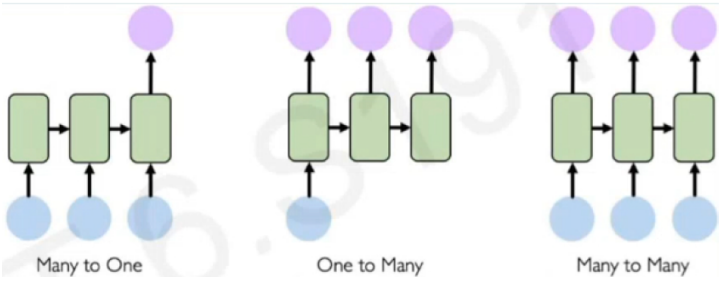
3 Recurrent Neural Network

The key idea of RNNs is to store past information in an internal memory, which is updated after processing each element of the sequence.RNN update learnable parameters through backpropagation w.r.t to time. Moving on to building an RNN we try to unfold the neural network with respect to time to capture the temporal dependency. Mathematically, the output at time t is not only dependent on the input at time t but also associated with the captured information from time $t-1$.

Propagating this, at every timestamp,the network takes in the input at time stamp t along with hypothesis output from time $t-1$, to generate output at time t.



To target temporal dependencies, models can broadly be classified as :



Many to One:

Dependency present in the input to help model desired output. This model can be co-related to the part of categorization e.g. Sentiment Analysis

One to Many:

Dependencies forming in the output generated to the co-relation with the single input. This type of neural network has a single input and multiple outputs. e.g. Image captioning

Many to Many:

Way of modelling the whole input finding the dependency and converting it into suitable output. e.g. Language Translation

3.1 Issues Arising

- Exploding Gradient Problem: The gradients carry information used in the RNN. When the gradients accumulate and become too large, the gradient explosion occurs with each iteration.
- Vanishing Gradient Problem: When the gradient becomes too small, parameter updates become insignificant, making it difficult to learn long data sequences. In other words, if the gradients are small in each iteration, they keep diminishing. As a result, dependencies over long distances become indistinguishable.

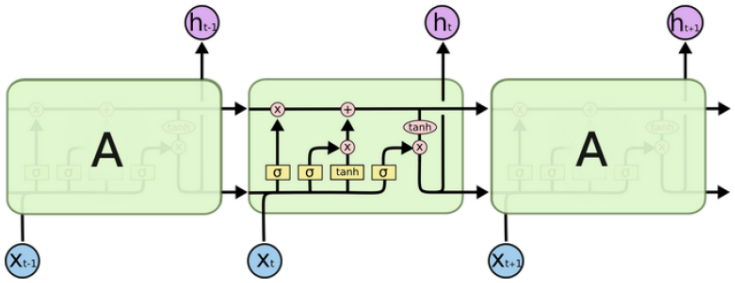
As a consequence of vanishing and exploding gradients, vanilla RNNs are difficult to train and are unable to capture long term dependencies. LSTMs were introduced to address the vanishing/exploding gradient problem in vanilla RNNs and learn long term dependencies.

4 LSTM

LSTMs employ a gating mechanism that effectively controls the flow of information into and out of their internal memory, known as the cell state.

A gate consists of a sigmoid layer followed by element-wise multiplication.

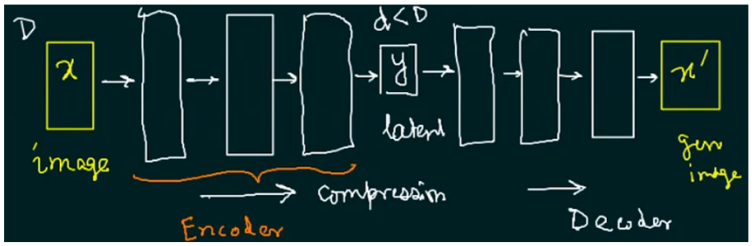
An LSTM comprises three gates: the input gate, the forget gate, and the output gate. It allows information to be added to or removed from the cell state using the input and forget gates



The repeating module in an LSTM contains four interacting layers.

5 Generative Models

5.1 Auto Encoder



The concept of generative modeling is based on the idea that, given an input x, we aim to generate a new input $x^{'}$.The generated $x^{'}$ follows a similar distribution and may represent either an anomalous or a regular case.

The input x is passed through several neural networks (or feature maps), resulting in an internal latent representation (say, y) is provided, which has lower dimension than x ($d < D$, where d is the dimension of y and D is dimension of x). These d dimensions are crucial for expressing the image, functioning similarly to a compression process.

The first set of neural networks in the above figure is called the **encoder**. The last set of neural networks ,which generates new images from the encoded part(y). This is called the **decoder**.

The above discussed is popularly known as **Auto Encoder**.

5.2 Variational Auto Encoder

Since we lose information during compression,how are we able to generate a new image? To generate new images, we introduce stochastic elements such as mean and variance. We define a mean and variance, then apply perturbations to them. This process introduces variation, allowing the generation of something that was not originally present in the input image. This approach is known as the Variational Auto Encoder. We may use squared error to minimize loss.

$$a(x^{'}, x) = \frac{1}{2}|x - x^{'}|^2 \tag{1}$$

For training, we need to minimize the loss from the input to the output. This is achieved by optimizing the loss function and applying techniques like backpropagation.

In a Variational Autoencoder (VAE), since the latent space follows a probabilistic Gaussian distribution with a defined mean and variance, we can introduce slight perturbations to obtain a new Gaussian distribution. This allows us to generate new images with controlled variations.