Deep Learning full notes

Deep Learning (SRM Institute of Science and Technology)

Deeplearning Unit-3

Convolutional Networks

A convolution is the ~~simplest~~ simple application of a filter to an input that results in an activation.

A convolutional neural network (CNN) is a Deep learning algorithm which can take in an input image, assign importance (learnable weights & biases) to various aspects/objects in the image and be able to differentiate one from the other.

Convolution is one of the main building blocks of a CNN. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information.

In case of CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map.

Variants of the basic Convolution function

→ Full convolution

convolution with a stride

o We may want to skip over some positions in the kernel to reduce computational cost

— At the cost of not extracting fine features.

- We can think of this as down-sampling the output of the full convolution function.

- We refer to s as the stride. It is possible to define a different stride for each direction.

→ Unshared Convolution

- In some cases when we do not want to use convolution but want to use locally connected layers.

- useful when we know that each feature should be a function of a small part of space, but ~~for~~ no reason to think that the same feature should occur across all the space.

- It can be also useful to make versions of convolution or local connected layers in which the connectivity is further restricted.

→ Tiled Convolution

- learns a set of kernel that we rotate through as we move through space.

Immediately neighbouring locations will have different filters, but the memory requirement for storing the parameters will increase by a factor of the size of this set of kernels.

LeNet

→ LeNet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition.

→ They used this architecture for recognizing the handwritten and machine-printed characters.

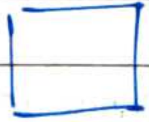
→ The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

Architecture of LeNet-5

The network has 5 layers with learnable parameters and hence named LeNet-5.

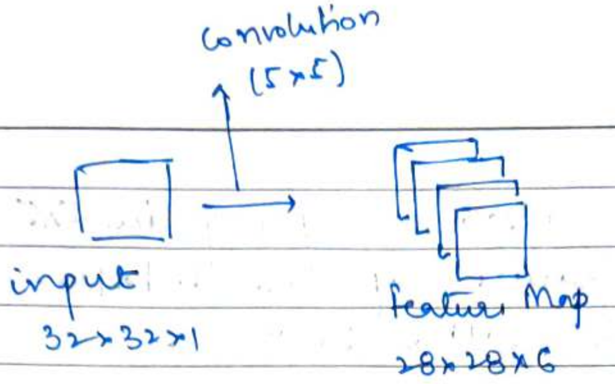
→ It has 3 sets of convolution layers with a combination of average pooling. After the convolution and the average pooling layers, we have two fully connected layers.

→ At last, a softmax classifier which classifies the image into respective class.



input
32 x 32 x 1

→ The input to this model is a 32 x 32 grayscale image hence the number of channels is one.



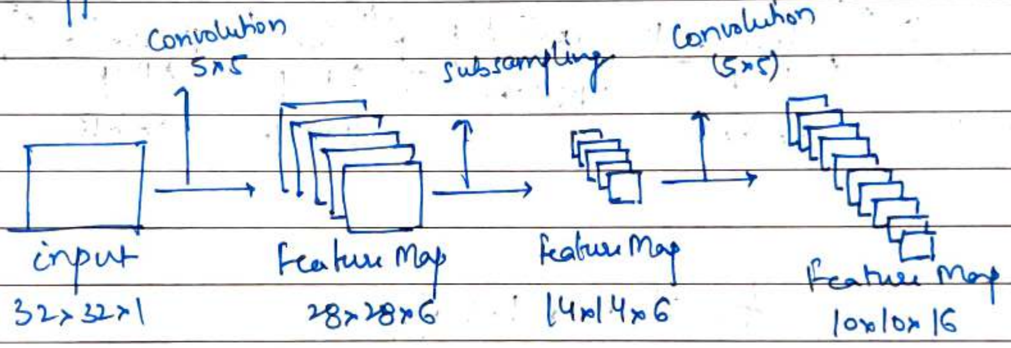
Output shape

((32-5+1) x (32-5+1) x 6)

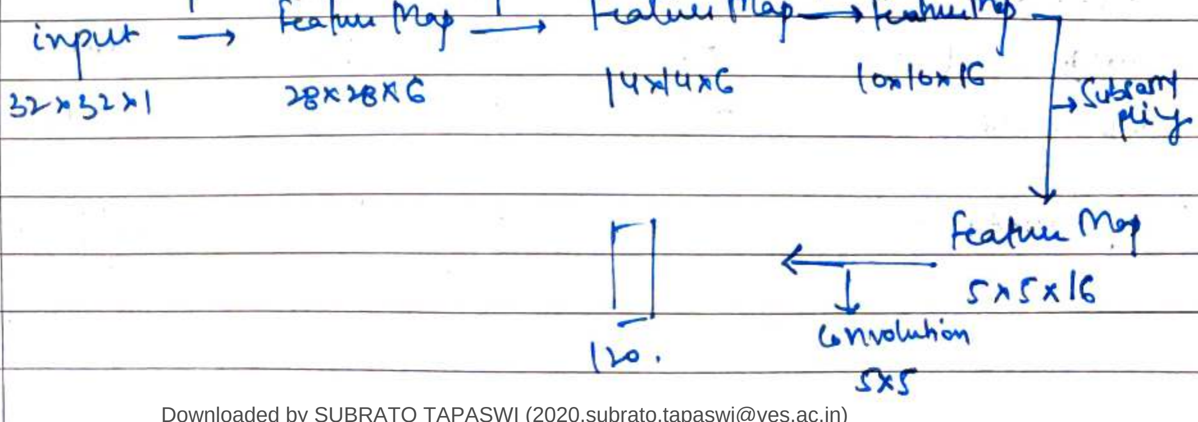
= 28x28x6

→ We then apply the first convolution operation with the filter size 5x5 and we have 6 such filters. As a result, we get a feature map of size 28x28x6.

Here the number of channels is equal to the number of filters applied.



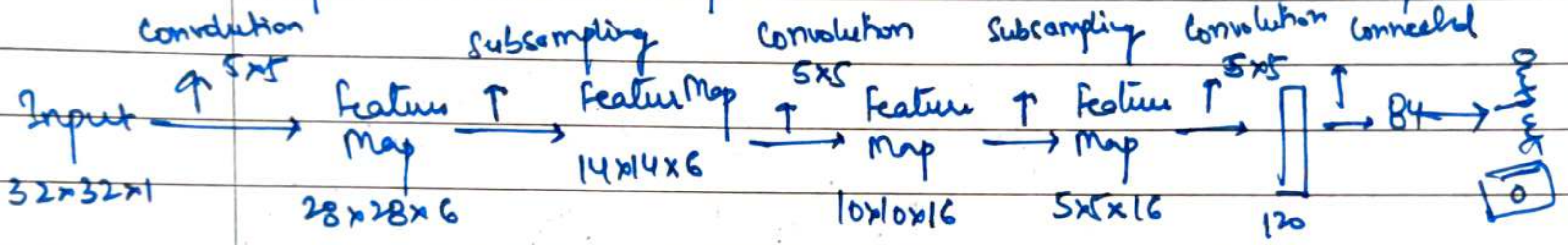
→ ~~Net~~ Next, we have a convolution layer with sixteen filters of size 5x5. Again the feature map changed it is 10x10x16.
The output size is calculated in a similar manner. After this, we again applied an average pooling or subsampling layer, which again reduces the size of the feature map by half ie, 5x5x6.



→ Then we have a final convolution layer of size 5x5 with 120 filters. having the feature map size 1x1x120. After which flatten result is 120 values.

→ After these convolution layers, we have a fully connected layer with eighty-four neurons. At last, we have an output layer with ten neurons since the data have ten classes.

So final architecture of the LeNet-5 model.



Architecture Details

layer	# filters/ neurons	Filter size	Stride	Size of feature map	Activation Function
Input	-	-	-	32x32x1	
Conv1	6	5*5	1	28x28x6	tanh
Avg.pooling1	-	2*2	2	14x14x6	-
Conv2	16	5*5	1	10x10x16	tanh
Avg. pooling2		2*2	2	5x5x16	-
Conv3	120	5*5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

→ 5 layers with learnable parameters.

→ The input to the model is a grayscale image.

→ It has 3 convolution layers, two average pooling layers, and two fully connected layers with a softmax classifier.

→ The number of trainable parameter is 60k.

Pooling layers

The pooling operation involves sliding a 2-D filter over each channel of feature map and summarising the features lying within the region covered by the filter.

Why to use pooling layers?

→ Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

→ The pooling layer summarises the feature present in a region of the feature map generated by a convolution layer.

Types of pooling layers :-

1. Max pooling:

max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.

→ Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

2. Average Pooling:

Average pooling computes the average of the elements present in the region of feature map covered by the filter.

→ Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the averages of the features present in a patch.

3. Global Pooling:

Global pooling reduces each channel in the feature map to a single value.

Global average pooling is a pooling operation designed to replace fully connected layers in CNN's output = (Input - filter size / stride) + 1

Number of filters becomes the channel in the output feature map.

AlexNet

The AlexNet has eight layers with learnable parameters. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and they use ReLU activation in each of these layers except the output layer.

AlexNet Architecture

→ The input to this model is the images of size 227×227×3.

- Convolution and Maxpooling layers

→ Then we apply the first convolution layer with 96 filters of size 11×11 with stride 4.
The activation function used in this layer is relu.
The output layer is 55×55×96.

→ Next we have the first Maxpooling layer, of size 3×3 and stride 2.
Then we get the resulting feature map with the size 27×27×256.

layer	# filters/ neurons	filter size	Stride	Padding	Size of feature maps	Activation function.
Input	-	-	-	-	227×227×3	-
Conv 1	96	11×11	4	-	55×55×96	Relu
Max pool1	-	3×3	2	-	27×27×96	-
Conv 2	256	5×5	1	2	27×27×256	Relu
Max Pool 2	-	3×3	2	-	13×13×256	-
Conv 3	384	3×3	1	1	13×13×384	Relu
Conv 4	384	3×3	1	1	13×13×384	Relu
Conv 5	256	3×3	1	1	13×13×256	Relu
Max Pool3	-	3×3	2	-	6×6×256	-
Dropout1	rate=0.5	-	-	-	6×6×256	-
fully connected1	-	-	-	-	4096	Relu
Dropout2	rate=0.5				4096	-
fully connecte 2					4096	Relu
fully connected3					1000	softmax

→ It has 8 layers with learnable parameters.

→ The input to the Model is RGB images.

→ It has 5 convolution layers with a combination of max pooling layers.

→ Then it has 3 fully connected layers.

→ The activation function used in all layers is Relu.

→ It used two Dropout layers.

→ The activation function used in the output layer is Softmax.

→ The total number of parameters in this architecture is 62.3 million.

Deep learning.

Unit 4

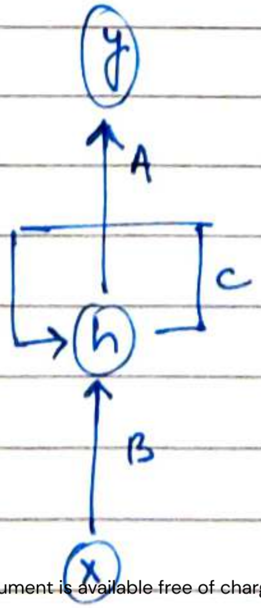
Recurrent Neural Networks.

→ RNN are a type of Neural Network where the output from previous step are fed as input to the current step.

→ These deep learning algorithms are commonly used for ordinal ~~and~~ or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning.

→ They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output.

→ The most important component of RNN is the Hidden state, which rembembers specific information about a sequence.



How does RNN work?

The input layer x receives and processes the neural network's input before passing it on to the middle layer.

Multiple hidden layers can be found in the middle layer h, each with it's own activation functions, weights, and biases.

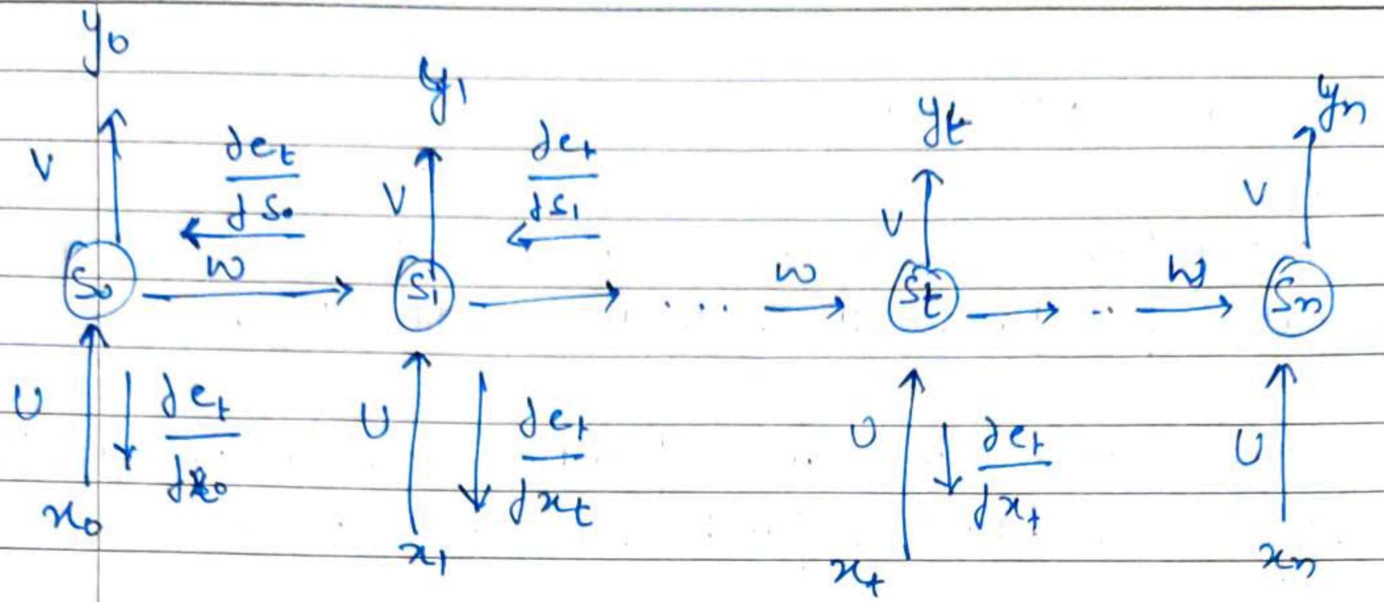
The different activation functions, weights, and biases will be standardized by the Recurrent Neural Network, ensuring that each hidden layer has the same characteristics. Rather than constructing numerous hidden layers, it will create only one and loop over it as many times as necessary.

Backpropagation Through Time (BPTT)

when we apply a backpropagation algorithm to a RNN with time series data as its input, we call it backpropagation through time.

A single input is sent into the network at a time in a normal RNN, and a single output is obtained

Backpropagation, on the other hand, uses both the current and prior inputs as input. This is reffered to as timestep, and one timestep will consists of multiple time series data points entering the RNN at the same time.



The output of the neural network is used to calculate and collect the errors once it has trained on a time set and given you an output.

The network is then rolled backed up, and weights are recalculated and adjusted to account for the faults.

Issues of standard RNN's

→ Exploding Gradients : Exploding gradients occurs when the algorithm gives the weights an absurdly high priority for no apparent reason.

Fortunately, truncating or squashing the gradients is a simple solution to this problem.

→ Vanishing Gradients: Vanishing gradients occur when the gradient values are too small, causing the model to stop learning or take far too long.

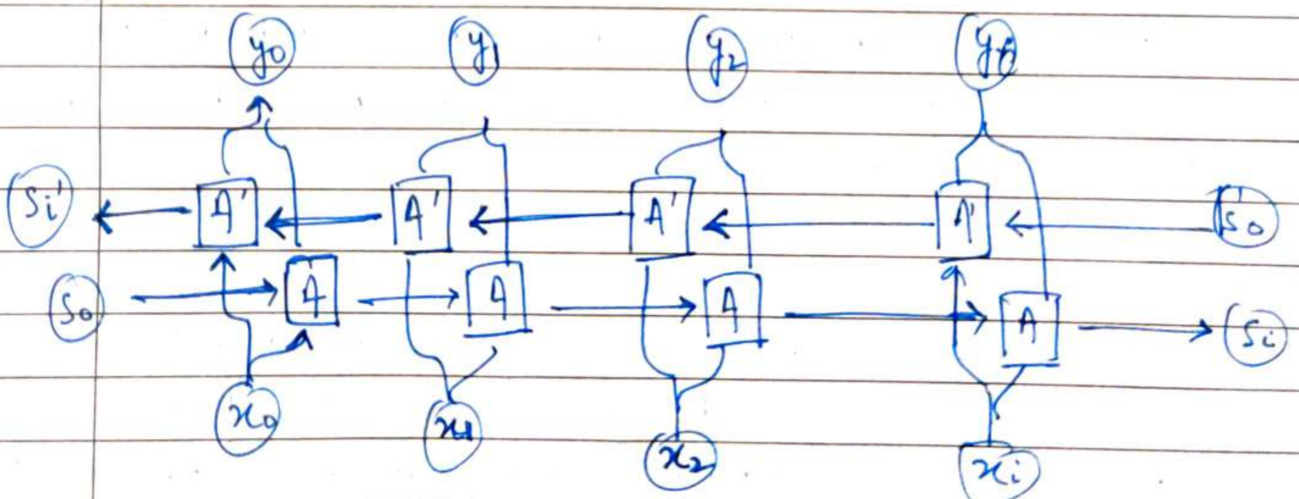
Solution → LSTM's.

Bidirectional RNN's

To enable straight (past) and reverse traversal of input (future), Bidirectional RNNs or BRNN's, are used.

→ A BRNN is a combination of two RNN's – one RNN moves forward, beginning from the start of the data sequence, and the other, moves backward, beginning from the end of the data sequence.

→ The network blocks in a BRNN can be either simple RNNs, GRUs, or LSTMs.



A BRNN has an additional hidden layer to accommodate the backward training process.

→ The training of a BRNN is similar to back-Propagation Through Time (BPTT) algorithm.

→ BPTT is the back-propagation algorithm used while training RNNs. A typical BPTT algorithm works as follows:

- Unroll the network and compute errors at every time step.
- Roll up the network and update the weights.

Recursive Neural Network

A recursive neural network is a kind of deep neural networks created by applying the same set of weights recursively over a structured input, to produce a structured prediction over variable-size input structures.

→ Recursive neural networks, sometimes abbreviated as RvNNs, have been successful, for instance, in learning sequence and tree structures in natural language processing.

- Whereas recursive neural networks operate on any hierarchical structure, combining child representations into parent representations, recurrent neural networks operate on the linear progression of time, combining the previous time step of hidden representation into the representation for the current time step.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) aiming to solve the vanishing gradient problem which comes with a standard recurrent neural network.

→ GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

- A GRU instead of having a simple neural network with four nodes as the RNN had previously has a cell containing multiple operations.

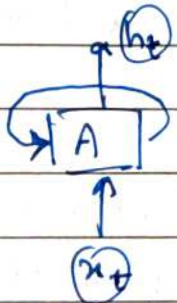
- Now the model that is being repeated every sequence is the green box containing three models where each one of those could be a neural network.

- GRU uses the so called, update gate and reset gate. The sigma notation above represent those gates: which allows a GRU to carry forward information over many time periods in order to influence a future time period.
In other words, the value is stored ~~as~~ in memory for a certain amount of time and at a certain point pulling that value out and using it with the current state to update at a future date.

LSTM

It is a special kind of recurrent neural network that is capable of learning long term dependencies in data. This is because the recurring module of the model has a combination of four layers interacting with each other.

Traditional neural networks can't retain the previous information.
Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



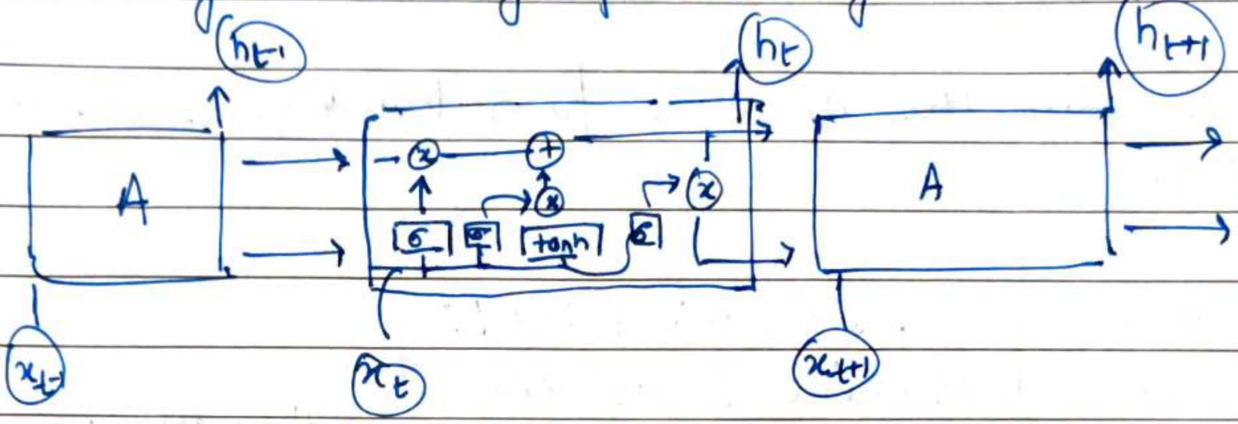
A chunk of neural network, A, looks at some input x_t and outputs a value h_t.

A loop allows information to be passed from one step of the network to the next.

For problems where there is very less need of context ~~of~~ or gap between the relevant information and the place that it's needed is small, RNN's can be used.

But there are also cases when we need more context or gap between the relevant information and the point its needed is very large we use LSTM.

LSTM's have chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



Each line carries an entire vector, from the output of one node to the inputs of others.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy ~~to~~ for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structure called gates.

Gates are a way to optionally let information through. An LSTM has three of these gates, to protect and control the cell state.