GAMAKA
Artificial Intelligence Solutions

DATA SCIENCE INTERVIEW PREPARATION

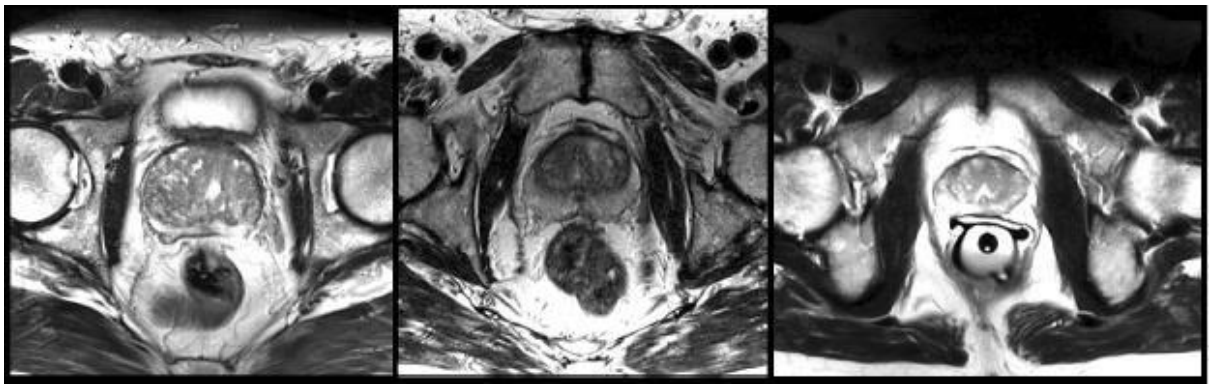
Day22

www.gamakaai.com
+91-7378483656 +91-7378493293

Q1. Explain V-Net (Volumetric Convolution) Architecture with related to Biomedical Image Segmentation?

Answer:

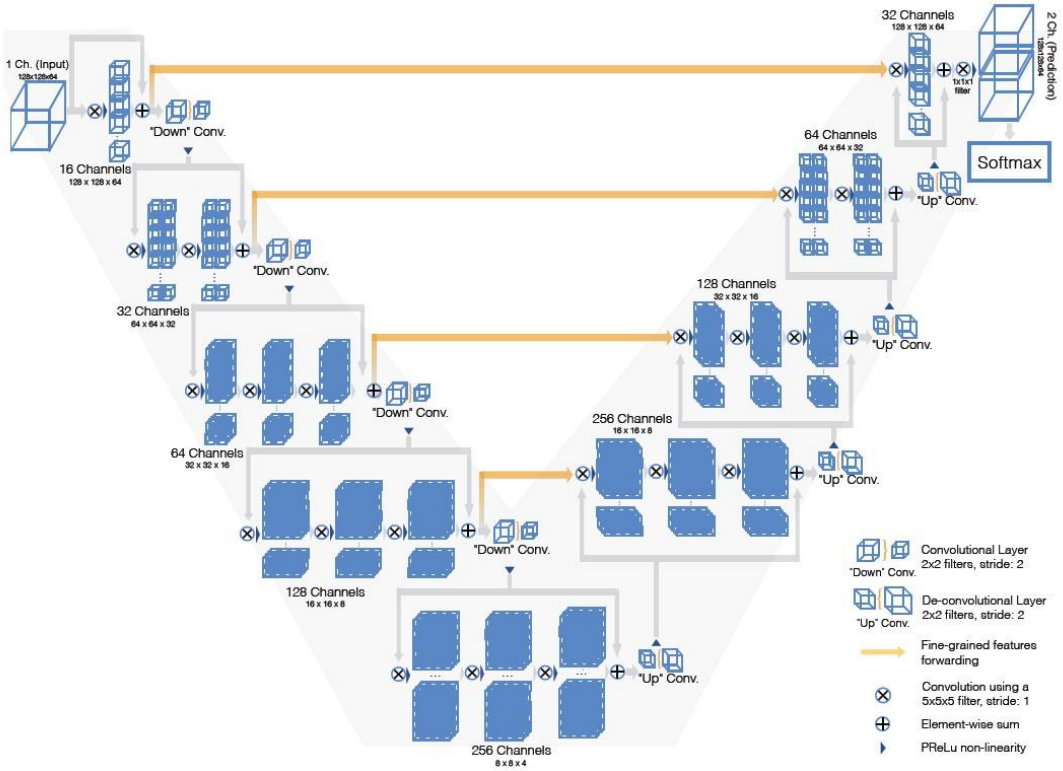
There were several medical data used in clinical practice consists of 3D volumes, such as MRI volumes illustrate prostate, while most approaches are only able to process 2D images. A 3D image segmentation based on a volumetric, fully convolutional neural network is proposed in this work.



Slices from MRI volumes depicting prostate

Prostate segmentation nevertheless is the crucial task having clinical relevance both during diagnosis, where the volume of the prostate needs to be assessed and during treatment planning, where the estimate of the anatomical boundary needs to be accurate. **Architecture**

www.gamakaai.com
+91-7378483656 +91-7378493293



- V-Net, justifies by its name, it is shown as V-shape. The left part of the network consists of a compression path, while on the right part decompresses signal until its original size is reached.
- This is the same as U-Net, but with some difference.

On Left

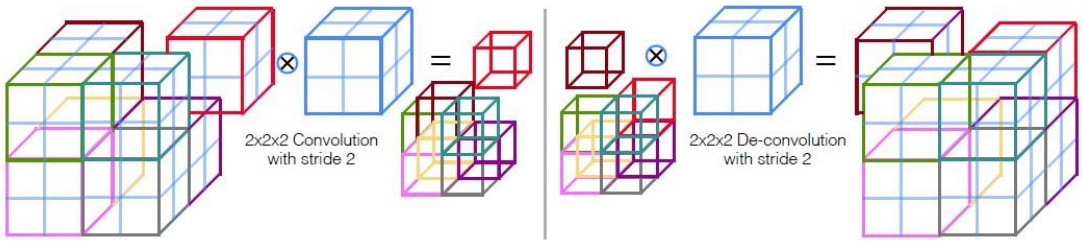
- The left side of the network is divided into different stages that operate at various resolutions. Each stage comprises one to 3 convolutional layers.
- **At each stage, a residual function is learned.** The input of each stage is used in convolutional layers and processed through non-linearities and added to the output of the last convolutional layer of that stage to enable learning a residual function. This V-net architecture ensures convergence compared with non-residual learning networks such as UNet.
- The **convolutions** performed in each stage use **volumetric kernels** having the size of **5×5×5 voxels**. (A voxel represents a value on a regular grid in 3D-space. The term voxel is

commonly used in 3D much 3D space, just like voxelization in a point cloud.)

- Along the compression path, the **resolution is reduced by convolution with 2×2×2 voxels full kernels applied with stride 2**. Thus, the size of the resulting feature maps is halved, with a **similar purpose as pooling layers**. And **number of feature channels doubles at each stage** of the compression path of V-Net.
- Replacing pooling operations with convolutional ones helps to have a smaller memory footprint during training because no switches mapping the output of pooling layers back to their inputs are needed for back-propagation.
- Downsampling helps to increase the receptive field.
- **PReLU** is used as a non-linearity activation function.

On Right Part

- The network extracts features and expands spatial support of the lower resolution feature maps to gather and assemble the necessary information to output a two-channel volumetric segmentation.



- At each stage, a **deconvolution** operation is employed to **increase the size of the inputs followed by one to three convolutional layers,** involving **half the number of 5×5×5 kernels** applied in the previous layer.
- **The residual function** is learned, similar to left part of the network.
- The 2 features maps computed by **a very last convolutional layer**, having **1×1×1 kernel size** and producing **outputs of the same size as input volume.**

www.gamakaai.com
+91-7378483656 +91-7378493293

- These two output feature maps are **the probabilistic segmentation of the foreground and background regions by applying soft-max voxelwise**.

Q2. Highway Networks- Gating Function to highway

Answer:

It is found that difficulties are optimizing a very deep neural network. However, it's still an open problem with why it is difficult to optimize a deep network. (it is due to gradient vanishing problem.) Inspired by LSTM (Long Short-Term Memory), authors thereby **make use of gating function to adaptively bypass or transform the signal so that the network can go deeper.** The deep network with more than 1000 layers can also be optimized.

Plain Network

Before going into Highway Networks, Let us start with plain network which consists of *L* layers where the *l*-th layer (with omitting the symbol for the layer):

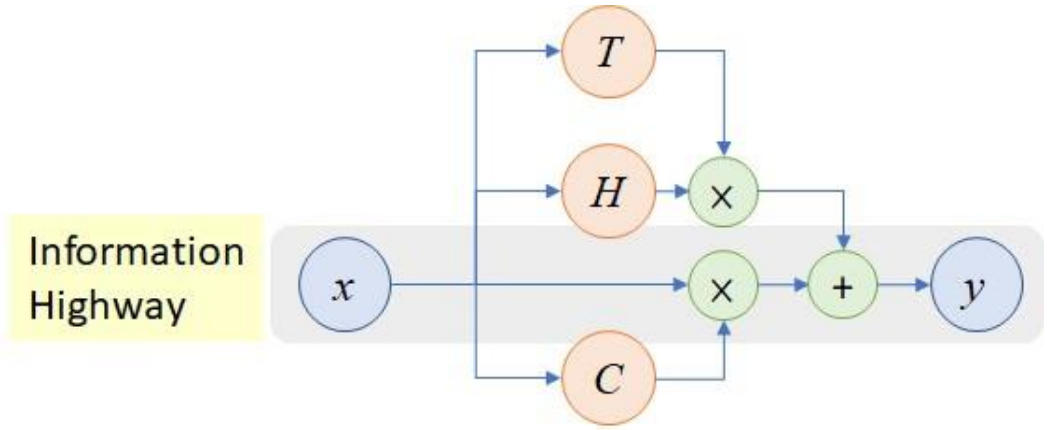
$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}).$$

Where *x* is input, *WH* is the weight, *H* is the transform function followed by an activation function, and *y* is the output. And for *i*-th unit:

$$y_i = H_i(\mathbf{x})$$

We compute the *yi* and pass it to the next layer.

Highway Network



www.gamakaai.com
+91-7378483656 +91-7378493293

In a highway network, 2 non-linear transforms ***T*** and ***C*** are introduced:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C}).$$

where ***T* is Transform Gate**, and **C is the Carry Gate**.

In particular, ***C* = 1 - *T***:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W_T})).$$

We can have below conditions for specific *T* values:

$$\mathbf{y} = \begin{cases} \mathbf{x}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{0}, \\ H(\mathbf{x}, \mathbf{W_H}), & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{1}. \end{cases}$$

When *T*=0, we pass input as output directly, which creates an information highway. That's why it is called the Highway Network.

When *T*=1, we use non-linear activated transformed input as output.

Here, in contrast to the *i*-th unit in plain network, the authors introduce the ***block*** concept. For ***i*-th block**, there is a **block state *Hi*(*x*)**, and **transform gate output *Ti*(*x*)**. And the corresponding **block output *yi***:

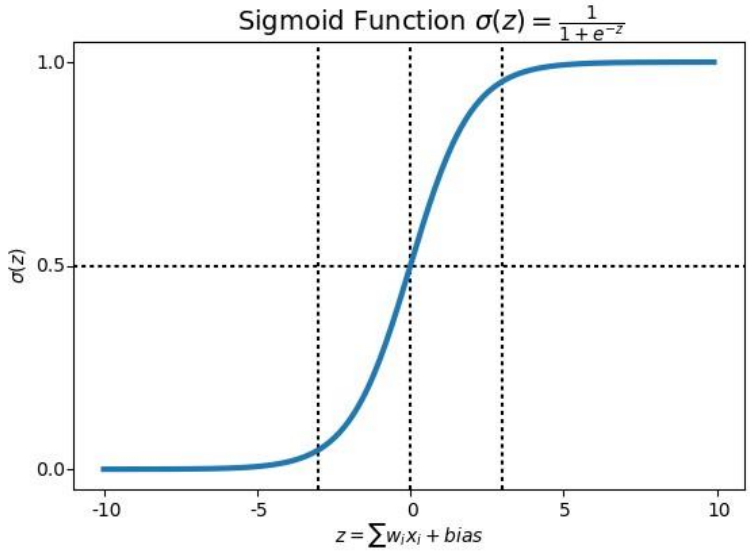
$$y_i = H_i(\mathbf{x}) * T_i(\mathbf{x}) + x_i * (1 - T_i(\mathbf{x}))$$

which is connected to the next layer.

- Formally, **T(*x*) is the sigmoid function**:

$$f_{C+1}(z) = [(f_C \circ f_C)(z)] \oplus [\mathrm{conv}(z)]$$

www.gamakaai.com
+91-7378483656 +91-7378493293



Sigmoid function caps the output between 0 to 1. When the input has a too-small value, it becomes 0. When the input has a too-large amount, it becomes 1. **Therefore, by learning *WT* and *bT*, a network can adaptively pass *H*(*x*) or pass *x* to the next layer.**

And the author claims that this helps to have the simple initialization scheme for *WT* which is independent of nature of *H*.

bT can be initialized with the negative value (e.g., -1, -3, etc.) such that the network is initially biased towards carrying behaviour.

LSTM inspires the above idea as the authors mentioned.

And SGD(**Stochastic Gradient Descent) did not stall for networks with more than 1000 layers.** However, the exact results have not been provided.

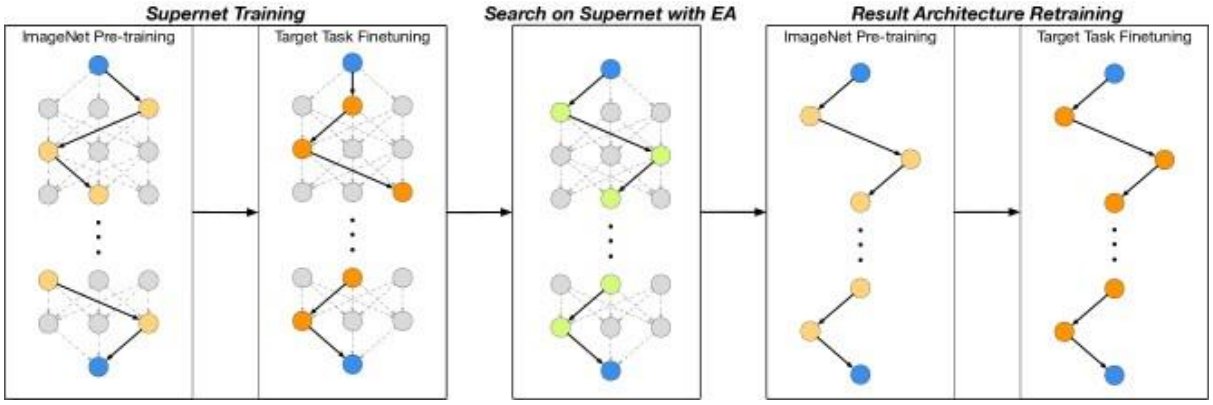
Q3. What is DetNAS: Neural Architecture Search(NAS) on Object Detection?

Answer:

Object detection is one of the most fundamental computer vision(OpenCV) tasks and has been widely used in real-world applications. The performance of object detectors highly relies on features extracted by backbones. However, most works on object detection directly use networks designed for classification as a backbone the feature extractors, e.g., ResNet. The architectures optimized on

image classification can not guarantee performance on object detection. It is known that there is an essential gap between these two different tasks. Image classification basically focuses on ”What” main object of the image is, while object detection aims at finding ”Where” and ”What” each object instance in an image. There have been little works focusing on backbone design for object detector, except the hand-craft network, DetNet.

Neural architecture search (NAS) has achieved significant progress in image classification and semantic segmentation. The networks produced by search have reached or even surpassed the performance of the hand-crafted ones on this task. But object detection has never been supported by NAS before. Some NAS(Neural architecture search) work directly applies architecture searched on CIFAR-10 classification on object detection.



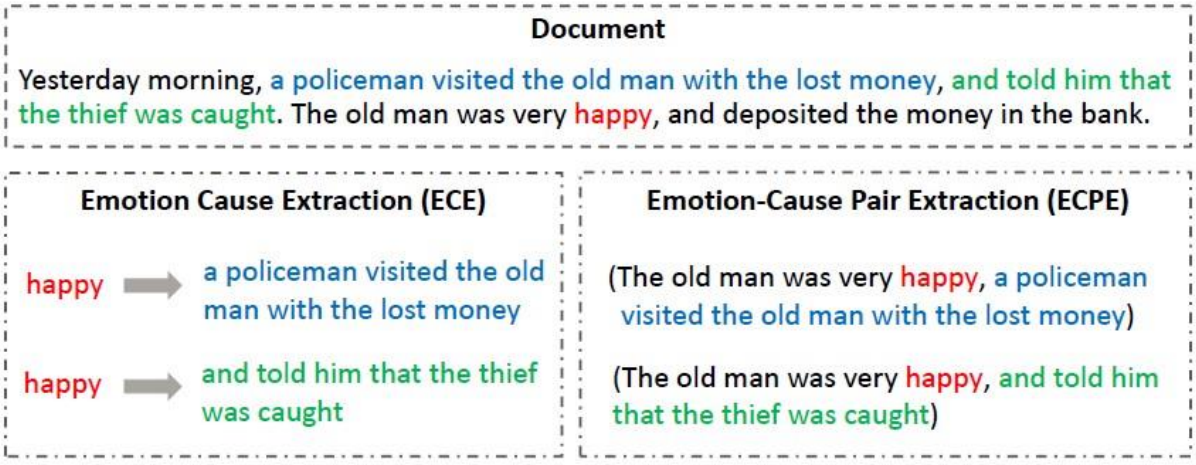
In this work, we present the first effort towards learning a backbone network for object detection tasks. Unlike previous NAS works, our method does not involve any architecture-level transfer. We propose DetNAS to conduct neural architecture search directly on the target tasks. The quests are even performed with precisely the same settings to the target task. Training an objector detector usually needs several days and GPUs, no matter using a pre-train-finetune scheme or training from scratch. Thus, it is not affordable to directly use reinforcement learning (RL) or evolution algorithm (EA) to search the architectures independently. To overcome this obstacle, we formulate this problem into searching the optimal path in the large graph or supernet. In simple terms, DetNAS consists of three steps: (1) training a supernet that includes all sub-networks in search space; (2) searching for the sub-network with the highest performance on the validation set with EA; (3) retraining the resulting network and evaluating it on the test set.

Q4. You have any idea about ECE (Emotion cause extraction).

Answer:

Emotion cause extraction (ECE) aims at extracting potential causes that lead to emotion expressions in the text. The ECE task was first proposed and defined as a word-level sequence labeling problem in Lee et al. To solve the shortcoming of extracting causes at the word level, Gui et al. 2016 released a new corpus which has received much attention in the following study and becomes a benchmark dataset for ECE research.

Below Fig. Displays an example from this corpus, there are five clauses in a document. The emotion "happy" is contained in fourth clause. We denote this clause as an *emotion clause*, which refers to a term that includes emotions. It has two corresponding causes: "a policeman visited the old man with the lost money" in the second clause and, "told him that the thief was caught" in the third clause. We name them as *cause clause*, which refers to a term that contains causes.



In this work, we propose a new task: emotion-cause pair extraction (ECPE), which aims to extract all potential pairs of emotions and corresponding causes in the document. In Above Fig, we show the difference between the traditional ECE task and our new ECPE task. The goal of ECE is to extract the corresponding cause clause of the given emotion. In addition to a document as the input, ECE needs to provide annotated feeling at first before cause extraction.

In contrast, the output of our ECPE task is a pair of emotion-cause, without the need of providing emotion annotation in advance. From Above fig., e.g., given the annotation of feeling: "happy," the goal of ECE is to track the two

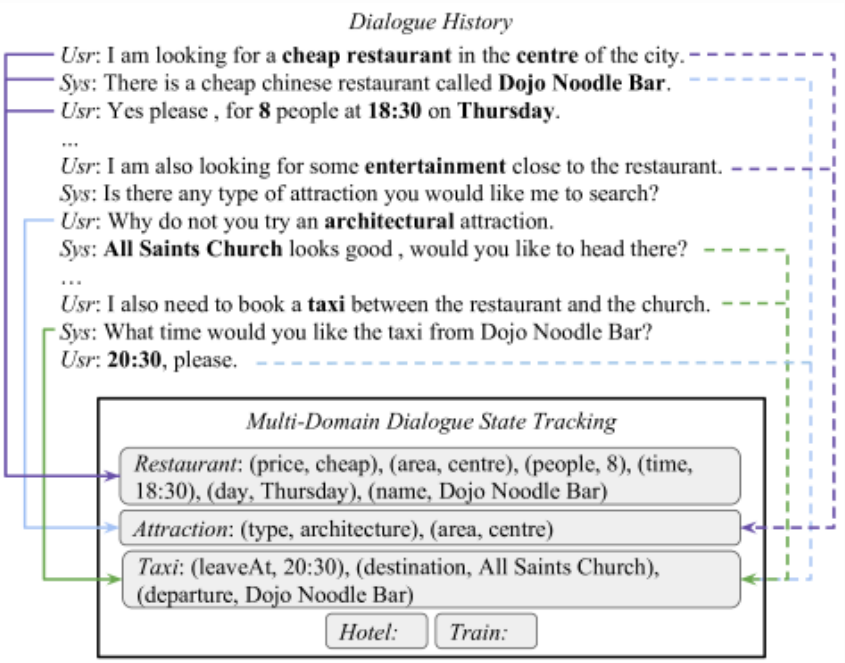
corresponding cause clauses: “a policeman visited the old man with the lost money” and “and told him that the thief was caught.” While in the ECPE task, the goal is to directly extract all pairs of emotion clause and cause clause, including (“The old man was delighted”, “a policeman visited the old man with the lost money”) and (“The old man was pleased”, “and told him that the thief was caught”), without providing the emotion annotation “happy”.

To address this new ECPE task, we propose a two-step framework. Step 1 converts the emotioncause pair extraction task to two individual sub-tasks (emotion extraction and cause extraction respectively) via two kinds of multi-task learning networks, intending to extract a set of emotion clauses and a set of cause clauses. Step 2 performs emotion-cause pairing and filtering. We combine all the elements of the two sets into pairs and finally train a filter to eliminate the couples that do not contain a causal relationship.

Q5.What is DST (Dialogue state tracking)?

Answer:

Dialogue state tracking (DST) is a core component in task-oriented dialogue systems, such as restaurant reservations or ticket bookings. The goal of DST is to extract user goals expressed during conversation and to encode them as a compact set of the dialogue states, i.e., a set of slots and their corresponding values. E.g., as shown in below fig., *(slot, value)* pairs such as *(price, cheap)* and *(area, centre)* are extracted from the conversation. Accurate DST performance is important for appropriate dialogue management, where user intention determines the next system action and the content to query from the databases.



State tracking approaches are based on the assumption that ontology is defined in advance, where all slots and their values are known. Having a predefined ontology can simplify DST into a classification problem and improve performance (Henderson et al., 2014b; Mrkšić et al., 2017; Zhong et al., 2018). However, there are two significant drawbacks to this approach: 1) A full ontology is hard to obtain in advance (Xu and Hu, 2018). In the industry, databases are usually exposed through an external API only, which is owned and maintained by others. It is not feasible to gain access to enumerate all the possible values for each slot. 2) Even if a full ontology exists, the number of possible slot values could be significant and variable. For example, a restaurant name or a train departure time can contain a large number of possible values. Therefore, many of the previous works that are based on neural classification models may not be applicable in real scenarios.

Q6.What is NMT(Neural machine translation)?

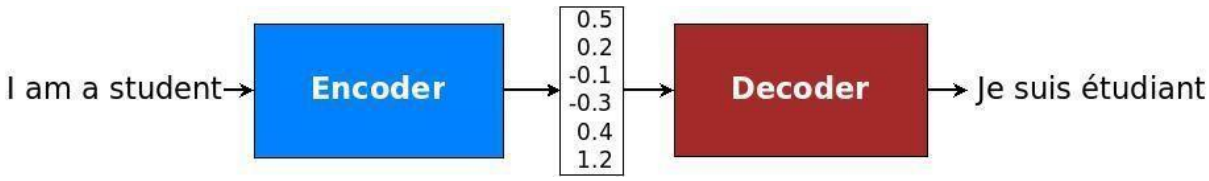
Answer:

NMT stands for Neural machine translation, which is the use of neural network models to learn the statistical model for machine translation.

The key benefit to the approach is that the single system can be trained directly on the source and target text, no longer requiring the pipeline of specialized methods used in statistical (ML)machine learning.

Unlike the traditional phrase-based translation system which consists of many sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

As such, neural machine translation(NMT) systems are said to be end-to-end systems as only one model is required for the translation.



In Encoder

The task of the encoder is to provide the representation of a input sentence.The input sentence is a sequence of words, for which we first consult embedding matrix. Then, as in the primary language model described previously, we process these words with a recurrent neural network(RNN). This results in hidden states that encode each word with its left context, i.e., all the preceding words. To also get the right context, we also build a recurrent neural network(RNN) that runs right-to-left, or, from the end of the sentence to beginning. Having two recurrent neural networks(RNN) running in two directions is known as the bidirectional recurrent neural network(RNN).

In Decoder

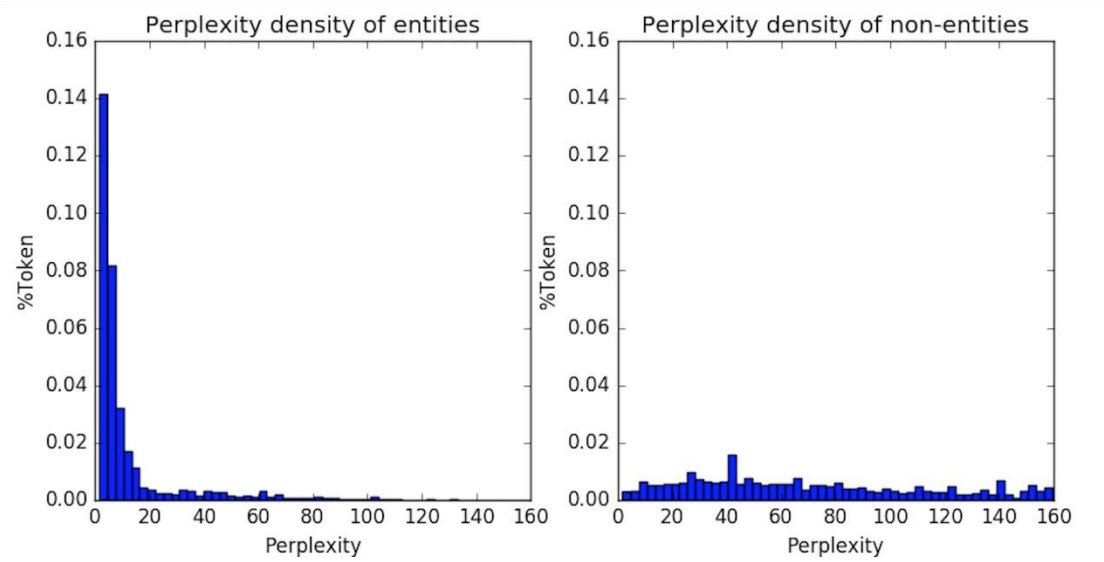
The decoder is the recurrent neural network(RNN). It takes some representation of input context (more on that in the next section on the attention mechanism) and previous hidden state and the output word prediction, and generates a new hidden decoder state and the new output word prediction.

If you use LSTMs for the encoder, then you also use LSTMs for the decoder. From hidden state. You now predict the output word. This prediction takes the form of the probability distribution over entire output vocabulary. If you have a vocabulary of, say, 50,000 words, then the prediction is a 50,000 dimensional vector, each element corresponding to the probability predicted for one word in the vocabulary.

Q7. What is Character-Level models (CLM)?

Answer:

In English, there is strong empirical evidence that the character sequence that create up proper nouns tend to be distinctive. Even divorced of context,human reader can predict that “hoekstenberger” is an entity, but “abstractually” is not. Some NER research explores use of character-level features including capitalization, prefixes and suffixes Cucerzan and Yarowsky; Ratinov and Roth (2009), and character-level models (CLMs) Klein et al. (2003) to improve the performance of NER, but to date there has been no systematic study isolating utility of CLMs in capturing the distinctions between name and non-name tokens in English or across other languages.



We conduct the experimental assessment of the discriminative power of CLMs for a range of languages: English, Arabic, Amharic, Bengali, Farsi, Hindi, Somali, and Tagalog. These languages use the variety of scripts and orthographic conventions (e.g, only three use capitalization), come from different language families, and vary in their morphological complexity. We represent the

effectiveness of CLMs(character-level models) in distinguishing name tokens from non-name tokens, as illustrated by the above Figure, which shows confusion in histograms from a CLM trained on entity tokens. Our models use individual tokens, but perform extremely well in spite of taking no account of the word context.

We then assess the utility of directly adding simple features based on this CLM(character-level model) implementation to an existing NER system, and show that they have the significant positive impact on performance across many of the languages we tried. By adding very simple CLM-based features to the system, our scores approach those of a state-of-the-art(SOTA) NER system Lample et al. (2016) across multiple languages, representing both the unique importance and broad utility of this approach.

Q8.What is LexNLP package?

Answer:

Over the last 2 decades, many high-quality, open-source packages for natural language processing(NLP) and machine learning(ML) have been released. Developers and researchers can quickly write applications in languages such as Python, Java, and R that stand on shoulders of comprehensive, well-tested libraries such as Stanford NLP (Manning et al. (2014)), OpenNLP (ApacheOpenNLP (2018)), NLTK (Bird et al. (2009)), spaCy (Honnibal and Montani (2017), scikitlearn library (Buitinck et al. (2013), Pedregosa et al. (2011)), and Gensim (Řehůřek and Sojka (2010)). Consequently, for most of the domains, rate of research has increased and cost of the application development has decreased.

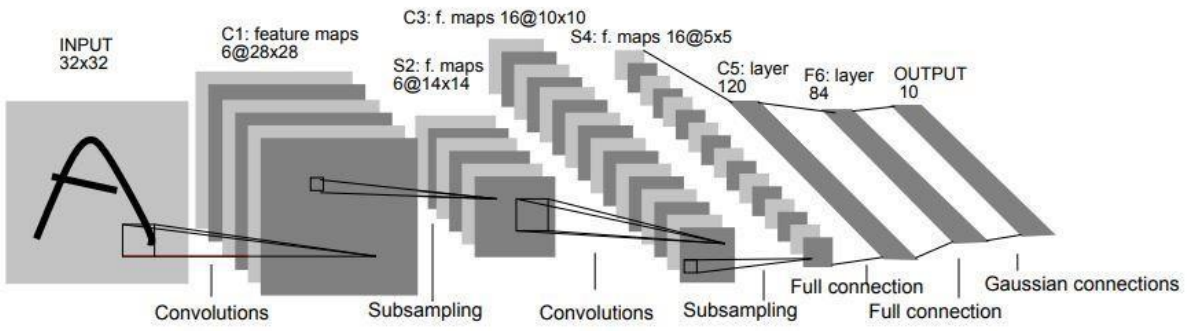
For some specialized areas like marketing and medicines, there are focused libraries and organizations like BioMedICUS (Consortium (2018)), RadLex (Langlotz (2006)), and the Open Health Natural Language Processing(NLP) Consortium. Law, however, has received substantially less attention than others, despite its ubiquity, societal importance, and the specialized form. LexNLP is designed to fill this gap by providing both tools and data for developers and researchers to work with real legal and regulatory text, including statutes, regulations, the court opinions, briefs, contracts, and the other legal work products.

Law is the domain driven by language, logic, and the conceptual relationships, ripe for computation and analysis (Ruhl et al. (2017)). However, in our experience, natural language processing(NLP) and machine learning(ML) have not been applied as fruitfully or widely in legal as one might hope. We believe that the key impediment to academic and commercial application has been lack of tools that allow users to turn the real, unstructured legal document into structured data objects. The Goal of LexNLP is to make this task simple, whether for the analysis of statutes, regulations, court opinions, briefs or the migration of legacy contracts to smart contract or distributed ledger systems.

Q9.Explain The Architecture of LeNet-5.

Answer:

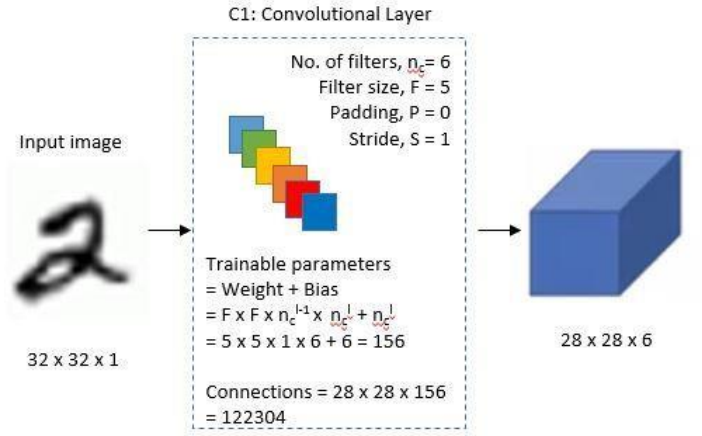
Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner proposed the neural network architecture for the handwritten and machine-printed character recognition in the 1990's which they called them LeNet-5. The architecture is straightforward and too simple to understand that's why it is mostly used as a first step for teaching (CNN)Convolutional Neural Network. **Architecture**



This architecture consists of two sets of convolutional and average pooling layers, followed by the flattening convolutional layer, then 2 fully-connected layers and finally the softmax classifier.

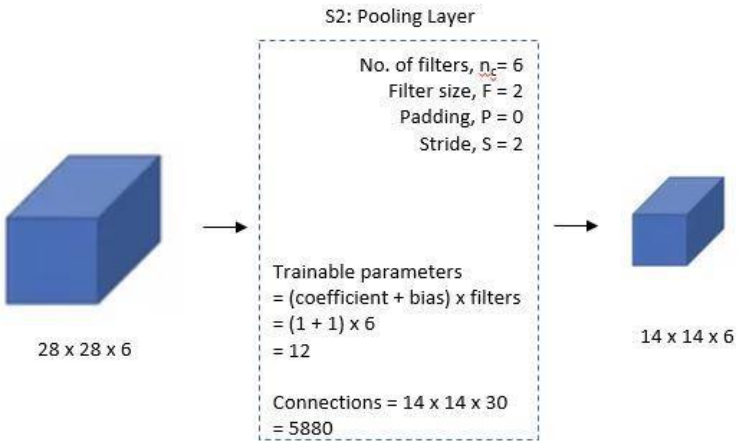
In the First Layer:

The input for LeNet-5 is the 32×32 grayscale image which passes through first convolutional layer with 6 feature maps or filters having size 5×5 and the stride of one. Image dimensions changes from 32x32x1 to 28x28x6.



In Second Layer:

Then it applies average pooling layer or sub-sampling layer with the filter size 2×2 and stride of two. The resulting image dimension will be reduced to 14x14x6.



Third Layer:

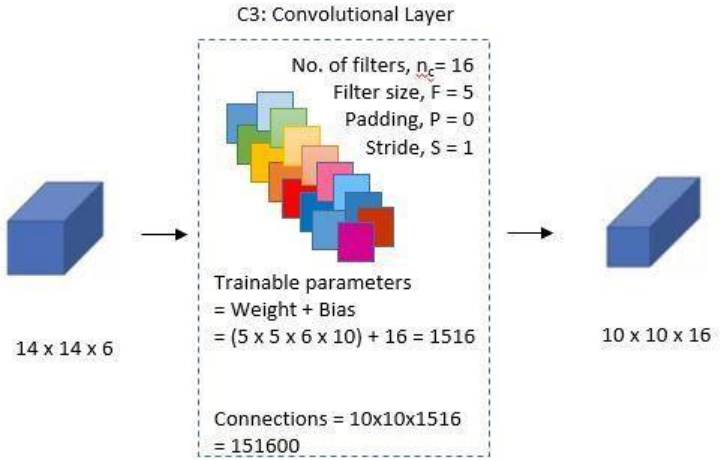
Next, there is the second convolutional layer with 16 feature maps having size 5×5 and the stride of 1. In this layer, only ten out of sixteen feature maps are connected to 6 feature maps of previous layer as shown below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

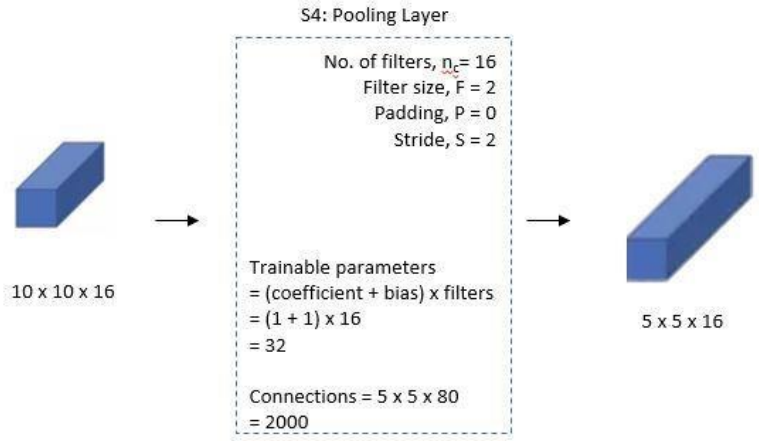
The main reason is to break symmetry in the network and keeps a number of connections within reasonable bounds. That is why the number of training parameters in this layers are 1516 instead of 2400 and similarly, number of connections are 151600 instead of 240000.



Fourth Layer:

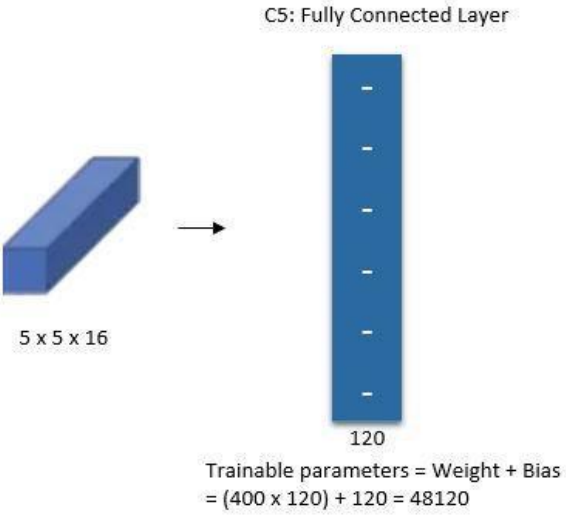
In the fourth layer (S4) is an average pooling layer with filter size 2×2 and stride of 2. This layer is same as second layer (S2) except it has 16 feature maps so output will be reduced to 5x5x16.

www.gamakaai.com
+91-7378483656 +91-7378493293



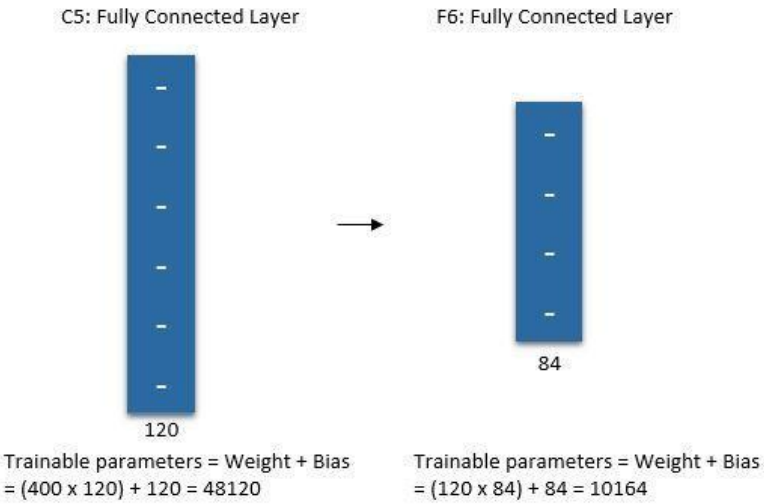
Fifth Layer:

The fifth layer (C5) is the fully connected convolutional layer with 120 feature maps each of the size 1×1. Each of 120 units in C5 is connected to all the 400 nodes (5x5x16) in the fourth layer S4.



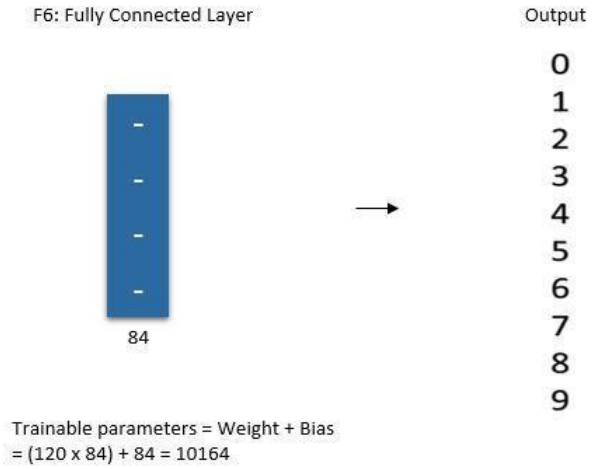
Sixth Layer:

The sixth layer is also fully connected layer (F6) with 84 units.



Output Layer:

Finally, there is fully connected softmax output layer ŷ with 10 possible values corresponding to digits from 0 to 9.



www.gamakaai.com
+91-7378483656 +91-7378493293