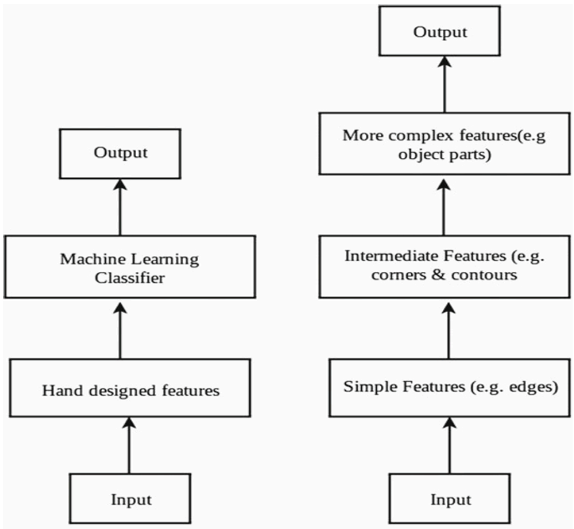
USN | | | | | | | | | |

Internal Assessment Test 1 – September 2025

Sub:	Deep Learning and Reinforcement Learning				Sub Code:	BAI701	Branch:	AIML
Date:	30/09/25	Duration:	90 minutes	Max Marks: 50	Sem/Sec:	VII		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	What is Deep Learning? Explain deep learning approaches using a hierarchy of representations. **Ans:** Deep learning refers to the architectures which contain multiple hidden layers (deep networks) to learn different features with multiple levels of abstraction. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher level learned features defined in terms of lower level features. **(2 marks)** In deep learning, a problem is realized in terms of hierarchy of concepts, with each concept built on the top of the others. The lower layers of the model encode some basic representation of the problem, whereas higher level layers build upon these lower layers to form more complex concepts. Given an image, the pixel intensity values are fed as inputs to the deep learning system. A number of hidden layers then extract features from the input image. These hiddenlayers are built upon each other in hierarchical fashion. At first,the lower level layers of the network detect only edge-like regions. These edge regions are then used to define corners (where edges intersect) and contours (outlines of objects). The layers in the higher level combine corners and contours to lead to more abstract “object parts” in the next layer. The key aspect of deep learning is that these layers of features are not handcrafted and designed by human engineers; rather, they are learnt from data gradually using a general-purpose learning procedure. Finally, the output layer classifies the image and obtains the output class label—the output obtained at the output layer is directly influenced by every other node available in the network. This process can be viewedashierarchical learning as each layer in the network uses the output of previous layers as “building blocks” to construct increasingly more complex concepts at the higher layers. The below figure compares traditional machine learning approaches based on handcrafted features to deep learning approaches based on hierarchical representation learning. Specifically, in deep learning meaningful representations from the input data are learnt by putting emphasis on building complicated mapping using a series of simple mappings. The word “deep” refers to learning successive layers of increasingly meaningful representations of input data. The number of layers used to model the data determines the depth of the model. Current deep learning often involves learning tens or even hundreds of successive layers of representation from the training data automatically. The conventional approaches to machine learning often focus on learning only one or two layers of representations of data; such approaches are often categorized as shallow learning. Deep learning and machine learning are sub fields of Artificial Intelligence (AI).	[10]	1	L2

<table>
<tr><td></td><td></td><td>

(a) Machine Learning (b) Deep Learning

(6 marks)</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>What is a hyperparameter? Briefly outline the important hyperparameters in the convolution layer of the ConvNet with an example.
Ans: Convolutional neural network architecture has many hyperparameters that are used to control the behavior of the model. Some of these hyperparameters control the size of the output while some are used to tune the running time and memory cost of the model. (2 marks)

The four important hyperparameters in the convolution layer of the ConvNet are given below:

a. Filter Size: Filters can be of any size greater than 2×2 and less than the size of the input but the conventional size varies from 11×11 to 3×3.The size of a filter is independent of the size of input.

b. Number of Filters: There can be any reasonable number of filters. AlexNet used 96 filters of size 11×11 in the first convolution layer. VGGNet used 96 filters of size 7×7, and another variant of VGGNet used 64 filters of size 11×11 in the first convolution layer.

c. Stride: It is the number of pixels to move at a time to define the local receptive field for a filter. Stride of one means to move across and down a single pixel. The value of stride should not be too small or too large. Too small stride will lead to heavily overlapping receptive fields and too large value will overlap less and the resulting output volume will have smaller dimensions spatially.

d. Zero Padding: This hyperparameter describes the number of pixels to pad the input image with zeros. Zero padding is used to control the spatial size of the output volume. (4 marks)

Each filter in the convolution layer produces a feature map of size $([A-K+2P]/S)+1$ where A is the input volume size, K is the size of the filter, P is the number of padding applied and S is the stride. Suppose the input image has size 128×128, and 5 filters of size 5×5 are applied, with single stride and zero padding, i.e., A 128, F 5,P 0andS 1.The number of feature maps produced will be equal to the number of filters applied, i.e., 5 and the size of each feature map will be $([128-5+0]/1)+1$ 124. Therefore, the output volume will be $124 \times 124 \times 5$.(2 marks)</td><td>[10]</td><td>2</td><td>L2</td></tr>
</table>

Apply a suitable deep learning architecture to the task of character recognition and justify your choice.

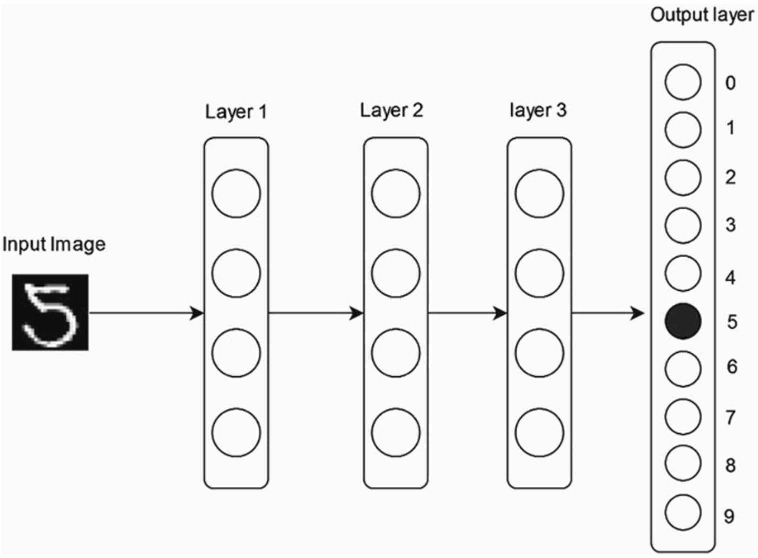


Fig. 1.3 A deep learning network for digit classification

Ans: Some of the aspects that helped in the evolution of deep networks are listed below:

1. Improved computational resources for processing massive amounts of data and training much larger models.
2. Automatic feature extraction.

The term artificial neural networks has a reference to neuroscience but deep learning networks are not models of the brain; however, deep learning models are formulated by only drawing inspiration from the understanding of the biological brain. Not all the components of deep models are inspired by neuroscience; some of them come from empirical exploration, theory, and intuition. The neural activity in our brains is far more complex than might be suggested by simply studying artificial neurons. The learning mechanisms used by deep learning models are in no way comparable to the human brain, but can be described as a mathematical framework for learning representations from data.

Q. No.		Marks	CO	Level
3		[10]	2	L3

Figure 1.3 shows an example of a deep learning architecture that can be used for character recognition. Figure 1.4 shows representations that are learned by the deep learning network. The deep network uses several layers to transform the input image (here a digit) in order to recognize what the digit is. Each layer performs some transformations on the input that it receives from the previous layers.

The deep network transforms the digit image into representations that tend to capture a higher level of abstraction. Each hidden layer transforms the input image into a representation that is increasingly different from the original image and increasingly informative about the final result. The representations learnt help to distinguish between different concepts which in turn help to find out similarities between it. Deep network can be thought of as a multistage distillation information operation, where layers use multiple filters on the information to obtain an increasingly transformed form of information (i.e., the information useful with regard to some task). In summary, a deep learning network constructs features at multiple levels, with higher features constructed as functions of lower ones. It is a fast-growing field that circumvents the problem of feature extraction which is used as a prelude by conventional machine

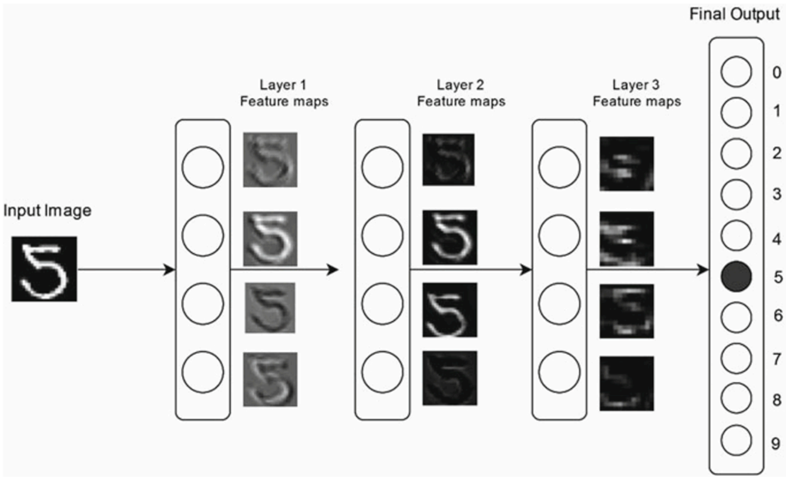


Fig. 1.4 Representations learnt by a deep network for digit classification during the first pass. Network structural changes can be incorporated that result in desired representations at various layers

Q. No.		Question	Marks	CO	Level
4		Explain (i) Dropout	[10]	1	L2

Explain

(i) Dropout

Deep neural networks consist of multiple hidden layers enabling it to learn more complicated features. It is followed by fully connected layers for decision-making. A fully connected layer is connected to all features, and it is prone to overfitting. Overfitting refers to the problem when a model is trained and it works so well on training data that it negatively impacts the performance of the model on new data. In order to overcome the problem of overfitting, a dropout layer can be introduced in the model in which some neurons along with their connections are randomly dropped from the network during training. A reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights. Dropout notably reduces overfitting and improves the generalization of the model.

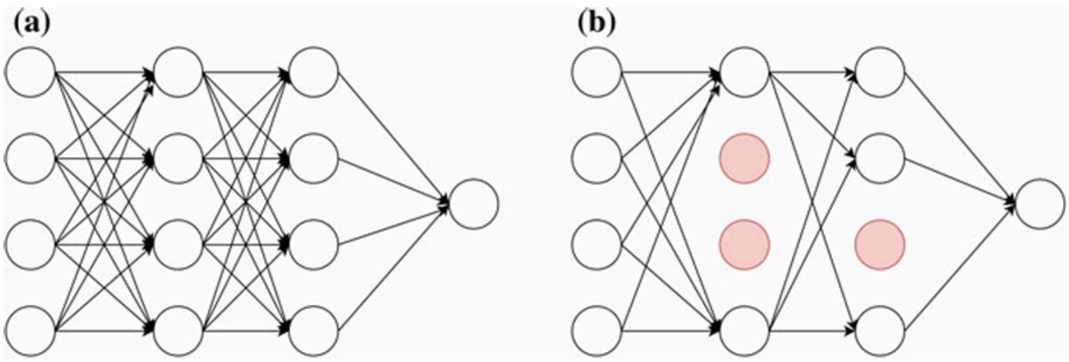


Fig. 2.12 **a** A simple neural network, **b** neural network after dropout

CODE FOR DROPOUT:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(input_dim,)))
# Add a Dropout layer with a dropout rate of 0.5 (50% of neurons dropped)
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
# Add another Dropout layer
```

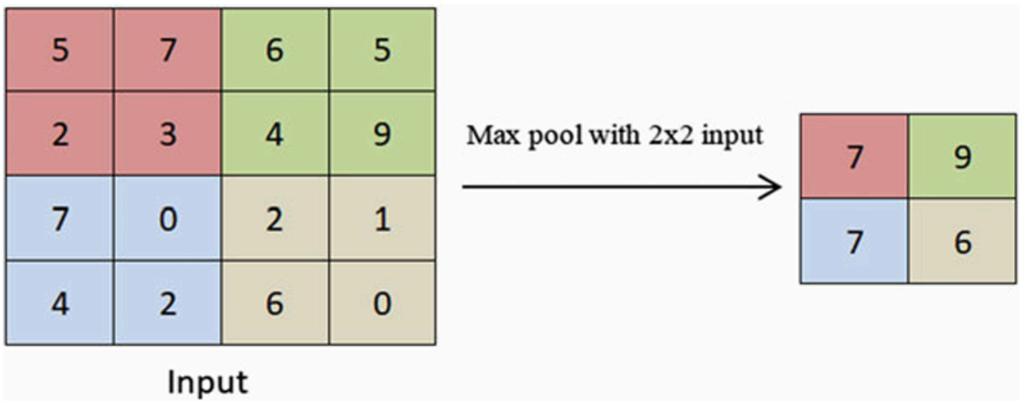
<table>
<tr>
<td></td>
<td></td>
<td>

```
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

(5 marks)

(ii) Max Pooling

In ConvNets, the sequence of convolution layer and activation function layer is followed by an optional pooling or down-sampling layer to reduce the spatial size of the input and thus reducing the number of parameters in the network. A pooling layer takes each feature map output from the convolutional layer and down-samples it, i.e., pooling layer summarizes a region of neurons in the convolution layer. There are few pooling techniques available and the most common pooling technique is max-pooling. Max-pooling simply outputs the maximum value in the input region. The input region is a subset of input (usually 2 × 2). For example, if the input region is of size 2 × 2, the max-pooling unit will output the maximum of the four values as .Other options for pooling layers are average pooling and L2-norm pooling. Pooling layer operation discards less significant data but preserves the detected features in a smaller representation. The intuitive reasoning behind pooling operation is that feature detection is more important than the feature's exact location. This strategy works well for simple and basic problems but it has its own limitations and does not work well for some problems



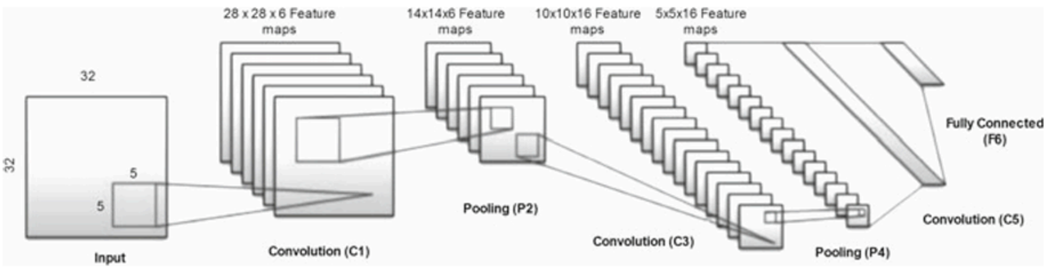
2.10 Max-pooling

</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>

Demonstrate the use of deep learning by explaining how it can be applied in at least four real world applications.

The choice of features that represent a given dataset has a profound impact on the success of a machine learning system. Better results cannot be achieved without identifying which aspects of the problem need to be included for feature extraction that would be more useful to the machine learning algorithm.This requires a machine learning expert to collaborate with the domain expert in order to obtain a useful feature set. A biological brain can easily determine which aspects of the problem it needs to focus on with comparatively little guidance. This is not the case with the artificial agents, thereby making it difficult to create computer learning systems that can respond to high-dimensional input and perform hard AI tasks.Today, many tech giant companies—Facebook, Baidu, Amazon, Microsoft, and Google—have commercially deployed deep learning applications. These compa nies have vast amount of data and deep learning works well whenever there are vast volumes of data

</td>
<td>[10]</td>
<td>2</td>
<td>L3</td>
</tr>
</table>

		and complex problems to solve. Many companies are using deep learning to develop more helpful and realistic customer service to representatives—Chatbots. In particular, deep learning has made good impact in historically difficult areas of machine learning: • Near-human-level image classification; • Near-human-level speech recognition • Near-human-level handwriting transcription; • Improved self-driving cars; • Digital assistants such as Google Now, Microsoft Cortana, Apple's Siri, and Amazon Alexa; • Improved ad targeting, as used by Google, Baidu, and Bing; • Improved search results on the web; • Ability to answer natural language questions; and • Superhuman Go, Shogi, and Chess playing. **(5 marks)** Explain any 4 real world applications with example**(5 marks)**			
6	a	Illustrate the steps involved in training a Convolutional Neural Network(CNN) with the help of equations. **Ans:** Training supervised deep neural networks is formulated in terms of minimizing a loss function. In this context, training a supervised deep neural network means searching a set of values of parameters (or weights) of the network at which the loss function has minimum value. Gradient descent is an optimization technique which is used to minimize the error by calculating gradients necessary to update the values of the parameters of the network. The most common and successful learning algorithm for deep learning models is gradient descent-based backpropagation in which error is propagated backward from last layer to the first layer. In this learning technique, all the weights of a neural network are either initialized randomly or initialized by using probability distribution. An input is fed through the network to get the output. The obtained output and the desired output are then used to calculate the error using some cost function (error function).  **Fig. 3.1** CNN architecture The CNN model is similar to LeNet but uses ReLU (which has become a standard in deep networks) as activation function, max-pooling, instead of average pooling. The CNN model consists of three convolution layers, two subsamplinglayers,andonefullyconnected layerproducing10outputs.Thefirstconvolutionlayerconvolvesinputofsize32×32 with six 5 ×5filters producing six 28 × 28 feature maps. Next, pooling layer down samples these six 28 × 28 feature maps to size 14 × 14. Second convolution layer convolves the down-sampled feature maps with 5 × 5 filters producing 16 feature maps of size 10 × 10. Second pooling layer down-samples the input producing 16 feature maps of size 5 × 5. The third convolution layer has the input of size 5 × 5 and the filter of size 5 × 5, so no stride happens. We can say that this third layer has 120 filters of size 5 × 5 fully connected to each of the 16 feature maps of size 5 ×5. Last layer in the architecture is a fully connected layer containing 10 units for 10 classes. The output of	[10]	3	L3

the fully connected layer is fed to cost function (Softmax) to calculate the error. Let us describe the CNN architecture in detail including its mathematical operations.

Layer 1 (*C*1) is a convolution layer which takes input of size 32 × 32 and convolves it with six filters of size 5 × 5 producing six feature maps of size 28 × 28.

Mathematically, the above operation can be given as

$$C1_{i,j}^{k} = \left(\sum_{m=0}^{4}\sum_{n=0}^{4} w_{m,n}^{k} * I_{i+m,j+n} + b^{k}\right) \tag{3.1}$$

where $C1^k$ represent six output feature maps of convolution layer $C1$, k represents filter number, (m, n) are the indices of kth filter and (i, j) are the indices of output.

Equation 3.1 is the mathematical form of convolution operation. The output of convolution operation is fed to an activation function to make the output of linear operation nonlinear. In deep networks, the most successful activation function is ReLU given by

σ(x) max(0,x)

$$C1_{i,j}^{k} = \sigma\left(\sum_{m=0}^{4}\sum_{m=0}^{4} w_{m,n}^{k} * I_{i+m,j+n} + b^{k}\right) \tag{3.3}$$

Equation 3.3 is the output of layer $C1$.

Layer 2(*P*2) is a max-pooling layer. The output of the convolution layer $C1$ is fed to max-pooling layer. The max-pooling layer takes six feature maps from $C1$ and performs max-pooling operation on each of them.

The max-pooling operation on $C1^k$ is given as

$$P2^{k} = \mathrm{MaxPool}\left(C1^{k}\right)$$

For each feature map in $C1^k$, max-pooling performs the following operation:

$$P2_{i,j}^{k} = \max\begin{pmatrix} C1_{(2i,2j)}^{k}, C1_{(2i+1,2j)}^{k} \\ C1_{(2i,2j+1)}^{k}, C1_{(2i+1,2j+1)}^{k} \end{pmatrix} \tag{3.4}$$

where (i, j) are the indices of kth feature map of output, and k is the feature map index.

Layer 3 (*C*3) is the second convolution layer which produces 16 feature maps of size 10 × 10 and is given by

$$C3_{i,j}^{k} = \sigma\left(\sum_{d=0}^{5}\sum_{m=0}^{4}\sum_{n=0}^{4} w_{m,n}^{k,d} * P2_{i+m,j+n}^{d} + b^{k}\right)$$

where $C3^k$ represents the 16 output feature maps of convolution layer $C3$, k is the index of the output feature map, (m, n) are the indices of filter weights, (i, j) are the indices of output, and d is the index of the number of channels in the input.

Layer 4 (*P*4) is a max-pooling layer which produces 16 feature maps $P4^k$ of size 5×5.

The max-pooling operation is given as

$$P4^k = \text{MaxPool}\left(C3^k\right)$$

$$P4^k_{i,j} = \max\begin{pmatrix} C3^k_{(2i,2j)}, C3^k_{(2i+1,2j)} \\ C3^k_{(2i,2j+1)}, C3^k_{(2i+1,2j+1)} \end{pmatrix}$$

where (i, j) are the indices of kth feature map of output, and k is the feature map index.

Layer 5 (*C*5) is third convolution layer that produces 120 output feature maps and is given by

$$C5^k_{i,j} = \sigma\left(\sum_{d=0}^{15}\sum_{m=0}^{4}\sum_{n=0}^{4} w^{k,d}_{m,n} * P4^d_{i+m,j+n} + b^k\right)$$

where $C5^k$ represents the 120 output feature maps of convolution layer $C5$ of size 1×1, k is the index of the output feature map, (m, n) are the indices of filter weights, d is the index of the number of channels in input and (i, j) are the indices of output, since output is only 1×1, the index (i, j) remains (0, 0) for each filter. This formula can be simplified as the filter size is equal to the size of input, so no convolution stride happens

$$C5^k = \sigma\left(\sum_{d=0}^{15}\sum_{m=0}^{4}\sum_{n=0}^{4} w^{k,d}_{m,n} * P4^d_{m,n} + b^k\right)$$

Layer 6 (*F*6) is a fully connected layer. It consists of 10 neurons for 10 classes and is mathematically defined as

$$F6^k = \sum_{i=1}^{120} w^k_i * C5^i \tag{3.5}$$

In last layer, for each neuron, the activation function used is softmax function given as

$$Z^k = \text{softmax}\left(F6^k\right)$$

The softmax function is defined as

$$Z^k = \text{softmax}\left(F6^k\right) = \frac{e^{F6^k}}{\sum_{i=1}^{10} e^{F6^i}} \tag{3.6}$$

The softmax activation function produces the final output of the neurons in the range [0, 1] and all outputs add up to 1. Each of the output represents the probability of the input belonging to a particular class.

Here, Z^k is the vector of size 10 containing final output of the network.

Backward Pass

Loss Layer

During training, an input is fed to the network and output is obtained. The obtained output is compared against the actual output to calculate the error or loss. The calculated error is then used to update the weights of the network. The process is repeated until the error is minimized. The function which is used to calculate the error or loss is called cost/loss function, and there are quite a few loss functions available and for softmax layer the two commonly used loss functions are Mean Squared Error (MSE) and cross-entropy loss function. The Mean Squared Error (MSE) of the CNN model can be given as

$$\text{loss} = E(Z, \text{target})$$

The loss function E is defined as

$$E(Z, \text{target}) = \frac{1}{10}\sum_{k=1}^{10}\left(Z^k - \text{target}^k\right)^2 \tag{3.7}$$

Z^k is the kth output (in this case the network will produce 10 outputs representing class probabilities) generated by the CNN and target^k is the ground truth of the input.

$E(Z, \text{target})$ is the error/loss which represents how far the prediction of the network is from the actual target.

The purpose of training is to minimize the loss function and to do so the weights of the network are continuously updated until the minimum value is achieved. After calculating the error, the gradient of the loss function with respect to each weight of the neural network is calculated. The weights of the network are then updated with their respective gradients. This process is continued until the total error is minimized.

To minimize the loss function E, derivative of E is calculated w.r.t weight w^i. Since the loss function E is defined as composition of functions in series as

$$E = \text{loss}(\text{softmax}(F6(C5(P4(C3(P2(C1(input)))))))$$

That is, the loss function composes of the softmax activation function, which is a function of the fully connected layer $F6$ which in turn is a function of previous layer $C5$ and so on.

The loss function E is differentiable and can be differentiated w.r.t any weight at any layer using chain rule. In backpropagation, the weights of the last layer are updated first followed by second last layer and so on. To start with, let us find the derivative of the cost function $w.r.t$ weight w_i^k at last layer $F6$ using chain rule.

$$\frac{\partial E}{\partial w_i^k} = \frac{\partial E}{\partial Z^k} * \frac{\partial Z^k}{\partial F6^k} * \frac{\partial F6^k}{\partial w_i^k} \tag{3.8}$$

where w_i^k is the updated weight and μ is the learning rate.

To solve Eq. 3.8, we first need to find the individual derivatives $\frac{\partial E}{\partial Z^k}$, $\frac{\partial Z^k}{\partial F6^k}$ and $\frac{\partial F6^k}{\partial w_i^k}$, which are given as

(i) $\frac{\partial E}{\partial Z^k} = \frac{\partial \frac{1}{10} \sum_{k=1}^{10} (Z^k - \text{target}^k)^2}{\partial Z^k} = \frac{1}{5}\left(Z^k - \text{target}^k\right)$

(ii) $\frac{\partial Z^k}{\partial F6^k} = Z^k * \left(1 - Z^k\right)$

Here, Z^k is the output of softmax function and the derivative of softmax function $f(x)$ is given as $f(x) * (1 - f(x))$ as shown in Sect. 3.3.3.

(iii) $\frac{\partial F6^k}{\partial w_i^k} = C5^i$

From Eq. 3.5

$$F6^k = \sum_{i=1}^{120} w_i^k * C5^i = \left(w_1^k * C5^1 + \cdots + w_i^k * C5^i + \cdots + w_{120}^k * C5^{120}\right)$$

The derivative of above function is given as

$$\frac{\partial F6^k}{\partial w_i^k} = \left(0 + \cdots + C5^i + \cdots + 0\right) = C5^i$$

Therefore,

$$\frac{\partial E}{\partial w_i^k} = \frac{1}{5}\left(Z^k - \text{target}^k\right) * Z^k * \left(1 - Z^k\right) * C5^i$$

Here, k is the neuron number in the last layer and i is the index of the weights to that neuron connecting input $C5^i$. For simplification in the backward pass and for propagation of error to previous layers, we calculate the deltas for this layer which is represented by

$$\delta F6^k = \frac{1}{5}\left(Z^k - \text{target}^k\right) * Z^k * \left(1 - Z^k\right)$$

6 b List and explain commonly used loss functions in Neural Network. [10] 3 L2

3.3.1 *Mean Squared Error* (L2) *Loss*

The most commonly used loss function in machine learning is Mean Squared Error (MSE) loss function. The MSE function, also known as $L2$ loss function, calculates the squared average error E of all the individual errors, and is given by

$$E = \frac{1}{n} \sum_{i=1}^{n} e_i^2$$

where e_i represents the individual error of ith output neuron which is given by

$$e_i = \text{target}(i) - \text{output}(i)$$

During training process, a loss function is used at the output layer to calculate the error and its derivative (gradient) is propagated in the backward direction of the network. The weights of the network are then updated with their respective gradients.

3.3.3 *Softmax Classifier*

Softmax classifier is a mathematical function which takes an input vector and produces output vector in range (0–1), where the elements of the output vector add up to 1. That is, the sum of all the outputs of softmax function is 1. Softmax function is given by

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Since softmax function outputs probability distribution, it is useful in the final layer of deep neural networks for multiclass classification.

During backpropagation, the derivative of this loss function is calculated using quotient rule

$$\frac{dS(y_i)}{dy_i} = \frac{\left(e^{y_i} \cdot \sum_{j=1}^{n} e^{y_j}\right) - (e^{y_i} \cdot e^{y_i})}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2}$$

$$\frac{dS(y_i)}{dy_i} = \frac{\left(e^{y_i} \cdot \sum_{i}^{n} e^{y_j}\right)}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2} - \frac{(e^{y_i} \cdot e^{y_i})}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2}$$

$$=> \frac{dS(y_i)}{dy_i} = \frac{(e^{y_i})}{\sum_{j=1}^{n} e^{y_j}} - \frac{(e^{y_i} \cdot e^{y_i})}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2}$$

$$=> \frac{dS(y_i)}{dy_i} = \frac{(e^{y_i})}{\sum_{j=1}^{n} e^{y_j}} - \left(\frac{e^{y_i}}{\left(\sum_{j=1}^{n} e^{y_j}\right)}\right)^2$$

$$=> \frac{dS(y_i)}{dy_i} = S(y_i) - (S(y_i))^2$$

$$=> \frac{dS(y_i)}{dy_i} = S(y_i) \cdot (1 - (S(y_i))$$

$$\text{since } \frac{(e^{y_i})}{\sum_{j=1}^{n} e^{y_j}} = S(y_i)$$

Similarly, derivative of $S(y_i)$ w.r.t y_k is given as

$$\frac{dS(y_i)}{dy_k} = \frac{\left(0 \cdot \sum_{j=1}^{n} e^{y_j}\right) - (e^{y_i} \cdot e^{y_k})}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2}$$

since $\frac{de^{y_i}}{dy_k} = 0$ as e^{y_i} is constant here.

$$\frac{dS(y_i)}{dy_k} = -\frac{(e^{y_i} \cdot e^{y_k})}{\left(\sum_{j=1}^{n} e^{y_j}\right)^2}$$

$$\frac{dS(y_i)}{dy_k} = -\frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \cdot \frac{e^{y_k}}{\sum_{j=1}^{n} e^{y_j}}$$

$$\frac{dS(y_i)}{dy_k} = -S(y_i) \cdot S(y_k)$$

The derivative of $S(y_i)$ is then used in Eq. 3.9 above to update the weights.

The gradient calculations are used to update the weights in such a way that the total error is minimized. The simplest way to minimize the error is to use various gradient-based optimization techniques which are briefly discussed in the next section.

CI **CCI** **HoD**