**Image classification using Convolutional neural network (CNN) with CIFAR 10 Dataset**

**Purpose:**

The primary aim of image classification with the CIFAR-10 dataset using CNNs is to arrange (categorize) the images into one of the ten predefined classes accurately (e.g., airplane, automobile, bird, cat, etc.).

**What are convolutional neural networks?**

Convolutional neural networks (CNNs) are a type of neural network that uses deep learning algorithms to process and analyze visual data such as images, video or graphics.

When CNNs process visual data they recognize, extract and learn meaningful information. Using convolutional layers and filters that detect features from image data, CNNs can recognize and extract patterns. CNNs learns features at different levels of complexity starting from simple, low-level details to more abstract, higher -level details.

**LeNet – 5 Architecture**

Yann LeCun designed LeNet-5 in the late 1990s primarily for digit recognition tasks. The LeNet-5 architecture consists of the input layers, convolutional layers, pooling layers, and fully connected layers. All these layers connects at the end with a SoftMax classifier, which performs the classification. This architecture laid the foundation for modern CNNs, and the significance lies in its innovative use of these layers.

The LeNet-5 architecture systematically breaks down an image for classification into smaller, manageable pieces, which allows the neural network to identify simple features first. These features combines in deeper layers to recognize complex patterns and shapes, which lead to accurate identification of images.

**The CIFAR-10 Dataset**

This dataset commonly and widely find its use for image classification tasks. It consists of 60,000 32 x 32-color images in 10 classes with 60,000 images per class. The images in the CIFAR -10 dataset is has both training and test images in 50,000 and 10,000 images respectively.

During training of the model, it is advisable to split the training sets into validation sets. These validation sets will allow for the evaluation of the model’s performance on unseen data, it will detect overfitting and provide a checkpoint to tune hyper-parameters. They will also adjust the model before testing on the final test sets.

**Convolutional neural network mechanism**

If a convolutional neural network is made to slide on an image with dimension 32 x32 x 3 representing the width, height and depth (RGB channels) in a CIFAR -10 DATASET, the input layer will receive the input image which can be an image or sequence of images.

Pixels are the smallest element in a digitally displayed image. These pixels are composed of sub pixels in the colors of Red, Green and Blue (RGB) and each pixel is an image corresponding to a value of 0 to 255 with zero (0) being black and 255 white.

The convolutional layer extracts the features from the input dataset using a set of learnable filters (kernels) in a convolutional operation. The convolution operation involves sliding the filter through the input image step by step where each step is called a stride. The stride can be one (1), two (2), and three (3) but it is preferable to work with smaller strides.

During the sliding process, a multiplication of a grid from the original input image with the grid from the filter generates a value. Carrying out the average of these values produces the feature map. Smaller filters and smaller strides extract more features from the input image. A filter size of 3x3 and a stride of one (1) is usually preferred.

Two other operations performed in a convolutional neural network architecture include the addition of activation functions and pooling. Activation functions such as ReLu, sigmoid, leaky ReLu brings non-linearity into the mathematical equation of the model making it useful for binary classification task. Using ReLu as the activation function changes all the negative numbers on a grid to zero (0) and when the number is positive and more than zero, it leaves the number as it is.

The application of filters and activation function (ReLu) does not change the output volume of the feature map, pooling makes this possible. The pooling layer also makes the computation faster, reduces memory and prevents overfitting. Max pooling and Average pooling are two most common types of pooling. The max-pooling layer makes use of the maximum number in each stride while average pooling takes the average of the numbers in each stride.

A flattening of the resulting feature maps into a one-dimensional vector after the convolution and pooling layers takes place so that they can be passed into a completely linked layer for categorization.

The fully connected layers take the input from the previous layer and computes the final classification. The output from the fully connected layers is then fed into a logistic function such as Softmax or sigmoid for classification task. This logistic function converts the output of each class into the probability score of each class.

**Code**

**Importing the required library**

The code in slide 17 of the presentation imports tensorflow, keras, numpy and matplotlib. Tensorflow and keras are for building and training the machine learning models. Numpy is for numerical operation while matplotlib is for plotting and visualizing data.

**Loading the dataset –CIFAR -10**

The dataset consists of training and test sets. This separation of the data into different sets not only prevents overfitting but also ensures the model can recognize data it has not seen before, rather than memorizing the training data. (x\_train, y\_train) are the training set while (x\_test, y\_test) are the test set.

**Splitting the training sets into validation sets**

The splitting the training sets into validation sets makes it possible to evaluate the model’s performance during training. Splitting also provides a checkpoint to tune hyper-parameters and make adjustments before testing on the final test sets.

**Normalizing the pixel of the image**

The normalization of pixel values involves scaling them to a range such as zero (0) to one (1), to ensure that the model treats all pixels uniformly and improve convergence. In addition, normalizing pixel values ensure that the gradients during back propagation remain stable and preventing issues like vanishing gradients.

**Building the machine learning model**

The code shows a sequential model in keras with a flatten layer and multiple dense layer. The model has an input shape of 32 x 32 x3and dense layers, which contain neuron and activation functions.

**Compiling and training the model**

The compile code assemble the keras model with the “Adam” optimizer, sparse categorical cross entropy loss and accuracy as a metric. The “Adam” optimizer adjusts the learning rates, which improves the training efficiency while the sparse categorical cross entropy is suitable for multiclass classification problems with integer labels.

Choosing the right optimizer and loss function is important because they directly affect how the model learns and how well it fits the data. A poor choice of optimizer or loss function can lead to slow convergence, poor accuracy and the inability to learn meaningful patterns.

The training code (model. Fit) trains the model using the training set with 5 epochs meaning that the learning algorithm will work through the entire training dataset five times. The number of epochs balances the training time and model performance.

Too few epochs might lead to under fitting where the model has not learnt enough from the data while too many epochs can cause overfitting where the model learns noise and specific patterns that do not generalize well.

Accuracy is a measure of how well a model correctly predicts the outcome. From the training using CNN, the model achieved a 78% accuracy. This means that out of all the predictions the model will make 78% of them will be correct.

**Evaluating the model**

Evaluating a model on a test set is important because it involves assessing the model’s ability to generalize to new unseen data. This helps to ensure the model is not just memorizing the training data but can perform well in a real-world scenario.

**References:**

A Deep Learning Approach Using CNN and LSTM Network, (*2023). 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*, pp.343-348. (Accessed: 1 September 2025).

Lecun, Bottou, Bengio, Haffner (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. (Accessed: 2 September 2025).

Scherer, Muller, Behnke: (2010) *Evaluation of Pooling Operations in Convolutional Architecture for Object Recognition*. Available at: [http://www.ais.uni-bonn.de](http://www.ais.uni-bonn.de/). (Accessed: 14 September 2025).

*CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more* - <https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5> . (Accessed: 17 September 2025).

Guo, Dong, Gao. Simple convolutional neural network on image classification. In Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 10–12 March 2017;pp. 721–724. (Accessed: 1 September 2025).

Sharma, Jain, Mishra. (2018).An analysis of convolutional neural networks for image classification. *Procedia Comput. Sci.*, *132*, 377–384. (Accessed: 20 September 2025).

Tu, Yin, S.; Ouyang, P.; Tang, S.; Liu, L.; Wei, S. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Trans. Very Large Scale Integr. (Vlsi) Syst.* 2017, *25*, 2220–2233. (Accessed: 21 September 2025).

Ren, Y.; Cheng, X. Review of convolutional neural network optimization and training in image processing. In Proceedings of the Tenth International Symposium on Precision Engineering Measurements and Instrumentation, SPIE, Kunming, China, 8–10 August 2018; Volume 11053, pp. 788–797(Accessed: 27 September 2025).