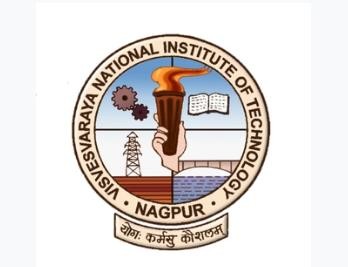
# Visvesvaraya National Institute of Technology, Nagpur



Project Report On

**Densely Connected Convolutional Networks**

**Presented by**

**M SAI KIRAN(MT22MCS011)**

**Under the guidance of**

**Dr. M. M. Dhabu**

# Signature

**Project coordinator (Dr. M. M. Dhabu)**

# Definition

## Project Overview

Classifying the images is a crucial task in image processing today. For example, imagine the ability to automatically classify an image with great

precision using images of the different classes.

In this project, I developed a model that utilizes the publicly available CIFAR10 dataset to train a Densely Connected Convolutional Neural Network model capable of accurately classifying the classes of the images.

## Problem Statement

The objective of this project is to create an application that can detect class of images using only the image pixels. To achieve this goal, the following steps can be taken:

1.Download and extract the complete CIFAR10 onto the local system.

2.Utilize the dataset to train a Densely Connected CNN.

Test the trained model's ability to classify images from different classes using test data and evaluate its accuracy.

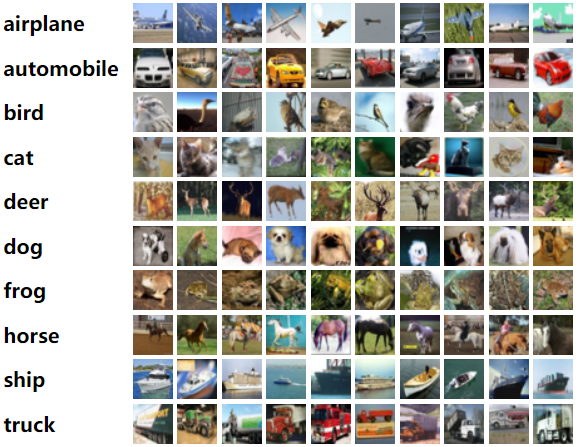
## Metrics

The goal of classifying images from CIFAR10 images is to enable auto classification of images which is required in various fields. The, accuracy is the most appropriate metric to use in this scenario.

# Analysis

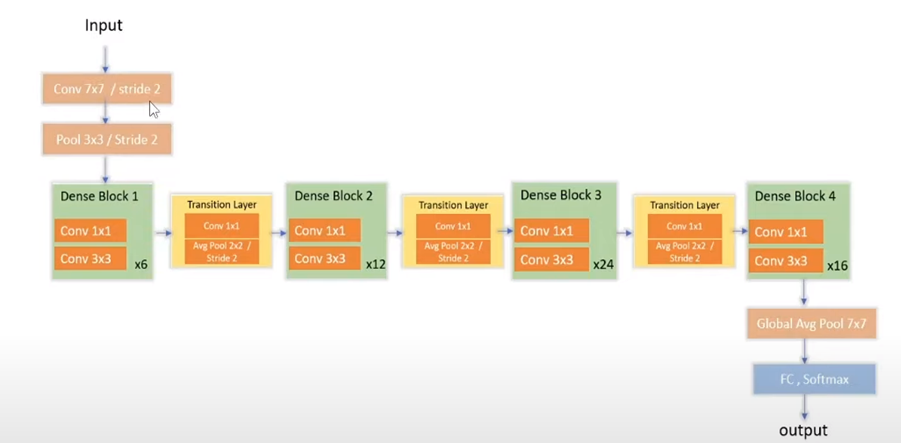
## Data Exploration

The CIFAR10 dataset consist of colored natural images with 32×32 pixels. CIFAR-10 (C10) con sists of images drawn from 10 classes. The training and test sets contain 50,000 and 10,000 images respectively, and we hold out 5,000 training images as a validation set. We adopt a standard data augmentation scheme (mirroring/shifting) that is widely used for the dataset [11, 13, 17, 22, 27, 20, 31, 33]. We denote this data augmentation scheme by a “+” mark at the end of the dataset name (e.g., C10+). For preprocessing, we normalize the data using the channel means and standard deviations. For the final run we use all 50,000 training images and report the final test error at the end of training.



## Algorithms and Techniques

Architecture:



## Dense connectivity: To improve the information flow between layers we use a different connectivity pattern: we introduce direct connections from any layer to all subsequent layers. Consequently, the ith layer receives the feature-maps of all preceding layers, x0,..., x−1, as input: x = H([x0, x1,..., x−1]), (2) where [x0, x1,..., x−1] refers to the concatenation of the feature-maps produced in layers 0,...,−1. Because of its dense connectivity we refer to this network architecture as Dense Convolutional Network (Dense Net). For ease of implementation, we concatenate the multiple inputs of H(·) in eq. (2) into a single tensor.

## Composite function. We define H(·) as a composite function of three consecutive operations: batch normalization (BN), followed by a rectified linear unit (ReLU) and a 3 × 3 convolution (Conv). Pooling layers. The concatenation operation is not viable when the size of feature-maps changes. However, an essential part of convolutional networks is down-sampling layers that change the size of feature-maps. To facilitate down-sampling in our architecture we divide the network into multiple densely connected dense blocks. We refer to layers between blocks as transition layers, which do convolution and pooling. The transition layers used consist of a batch normalization layer and an 1×1 convolutional layer followed by a 2×2 average pooling layer.

## Growth rate. If each function H produces k featuremaps, it follows that the ith layer has k0 +k ×(−1) input feature-maps, where k0 is the number of channels in the input layer. An important difference between DenseNet and existing network architectures is that DenseNet can have very narrow layers, e.g., k = 12. We refer to the hyperparameter k as the growth rate of the network. A relatively small growth rate is sufficient to obtain state-of-the-art results on the dataset. One explanation for this is that each layer has access to all the preceding feature-maps in its block and, therefore, to the network’s “collective knowledge”. One can view the feature-maps as the global state of the network. Each layer adds k feature-maps of its own to this state. The growth rate regulates how much new information each layer contributes to the global state. The global state, once written, can be accessed from everywhere within the network and, unlike in traditional network architectures, there is no need to replicate it from layer to layer.

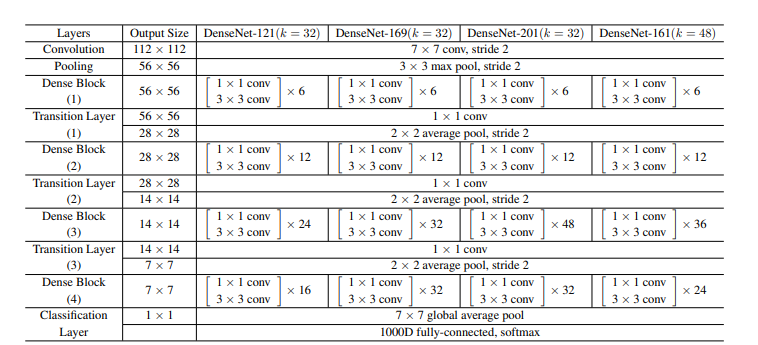
## Bottleneck layers. Although each layer only produces k output feature-maps, it typically has many more inputs. It has been noted that a 1×1 convolution can be introduced as bottleneck layer before each 3×3 convolution to reduce the number of input feature-maps, and thus to improve computational efficiency. This design is especially effective for DenseNet, i.e., to the BN-ReLU-Conv(1× 1)-BN-ReLU-Conv(3×3) version of H, as DenseNet-B.

# Methodology

## Data Pre-processing

The CIFAR10 dataset consist of colored natural images with 32×32 pixels. CIFAR-10 (C10) con sists of images drawn from 10 classes. The training and test sets contain 50,000 and 10,000 images respectively, and we hold out 5,000 training images as a validation set. We adopt a standard data augmentation scheme (mirroring/shifting) that is widely used for the dataset [11, 13, 17, 22, 27, 20, 31, 33]. We denote this data augmentation scheme by a “+” mark at the end of the dataset name (e.g., C10+). For preprocessing, we normalize the data using the channel means and standard deviations. For the final run we use all 50,000 training images and report the final test error at the end of training.

## Implementation



The DenseNet used has four dense blocks that each has an equal number of layers. Before entering the first dense block, a convolution with 16 (or twice the growth rate for DenseNet-BC) output channels is performed on the input images. For convolutional layers with kernel size 3×3, each side of the inputs is zero-padded by one pixel to keep the feature-map size fixed. We use 1×1 convolution followed by 2×2 average pooling as transition layers between two contiguous dense blocks. At the end of the last dense block, a global average pooling is performed and then a softmax classifier is attached. The feature-map sizes in the three dense blocks are 32× 32, 16×16, and 8×8, respectively.

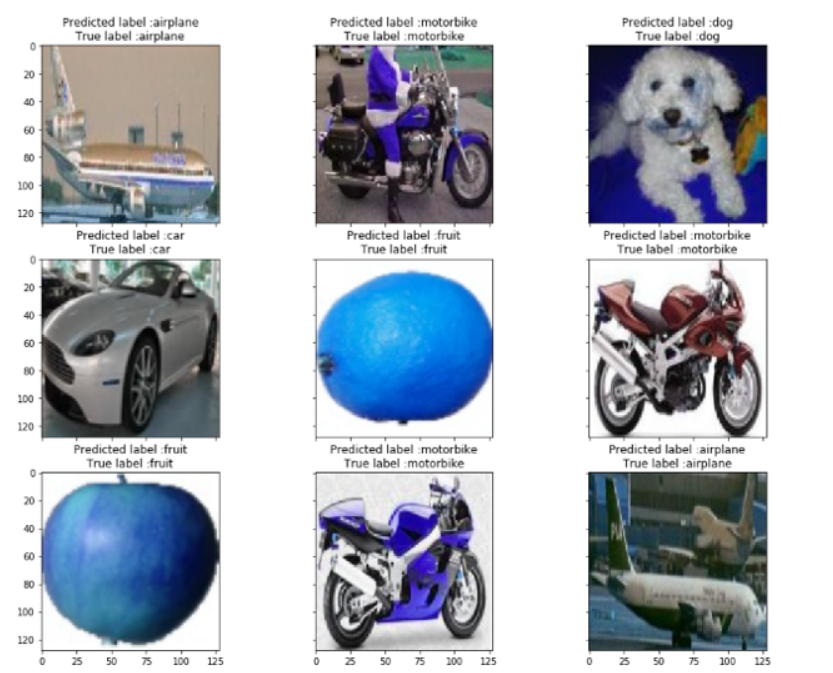
## Training:

All the networks are trained using stochastic gradient descent (SGD). We train using batch size 64 for 40 epochs. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. We add a dropout layer after each convolutional layer (except the first one) and set the dropout rate to 0.2. The test errors were only evaluated once for each task and model setting.

# Results

**Accuracy**. DenseNet with L = 190 and k = 40 outperforms the existing state-of-the-art consistently on the CIFAR dataset. Its error rates of 3.46% on C10+.With dropout, the DenseNet with L = 100 and k = 24 also surpasses the current best result achieved by wide ResNet.

**Capacity**. Without compression or bottleneck layers, there is a general trend that DenseNets perform better as L and k increase. We attribute this primarily to the corresponding growth in model capacity. The error drops from 5.24% to 4.10% and finally to 3.74% as the number of parameters increases from 1.0M, over 7.0M to 27.2M. This suggests that DenseNets can utilize the increased representational power of bigger and deeper models. It also indicates that they do not suffer from overfitting or the optimization difficulties of residual networks.



**Stochastic vs. deterministic connection**. There is an interesting connection between dense convolutional networks and stochastic depth regularization of residual networks. In stochastic depth, layers in residual networks are randomly dropped, which creates direct connections between the surrounding layers. As the pooling layers are never dropped, the network results in a similar connectivity pattern as DenseNet: there is a small probability for any two layers, between the same pooling layers, to be directly connected—if all intermediate layers are randomly dropped. Although the methods are ultimately quite different, the DenseNet interpretation of stochastic depth may provide insights into the success of this regularizer.

# Conclusion

A new convolutional network architecture, which we refer to as Dense Convolutional Network (DenseNet). It introduces direct connections between any two layers with the same feature-map size. DenseNets scale naturally to hundreds of layers, while exhibiting no optimization difficulties. DenseNets tend to yield consistent improvement in accuracy with growing number of parameters, without any signs of performance degradation or overfitting. Under multiple settings, it achieved state-of-the-art results across several highly competitive datasets. Moreover, DenseNets require substantially fewer parameters and less computation to achieve state-of-the-art performances. The further gains in accuracy of DenseNets may be obtained by more detailed tuning of hyperparameters and learning rate schedules. Whilst following a simple connectivity rule, DenseNets naturally integrate the properties of identity mappings, deep supervision, and diversified depth. They allow feature reuse throughout the networks and can consequently learn more compact and, according to our experiments, more accurate models. Because of their compact internal representations and reduced feature redundancy, DenseNets may be good feature extractors for various computer vision tasks that build on convolutional features.

**REFERENCES**

1. G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.
2. <https://arxiv.org/abs/1608.06993>.
3. [**https://www.kaggle.com/datasets/swaroopkml/cifar10-pngs-in-folders**](https://www.kaggle.com/datasets/swaroopkml/cifar10-pngs-in-folders)
4. <https://en.wikipedia.org/wiki/Convolutional_neural_network>