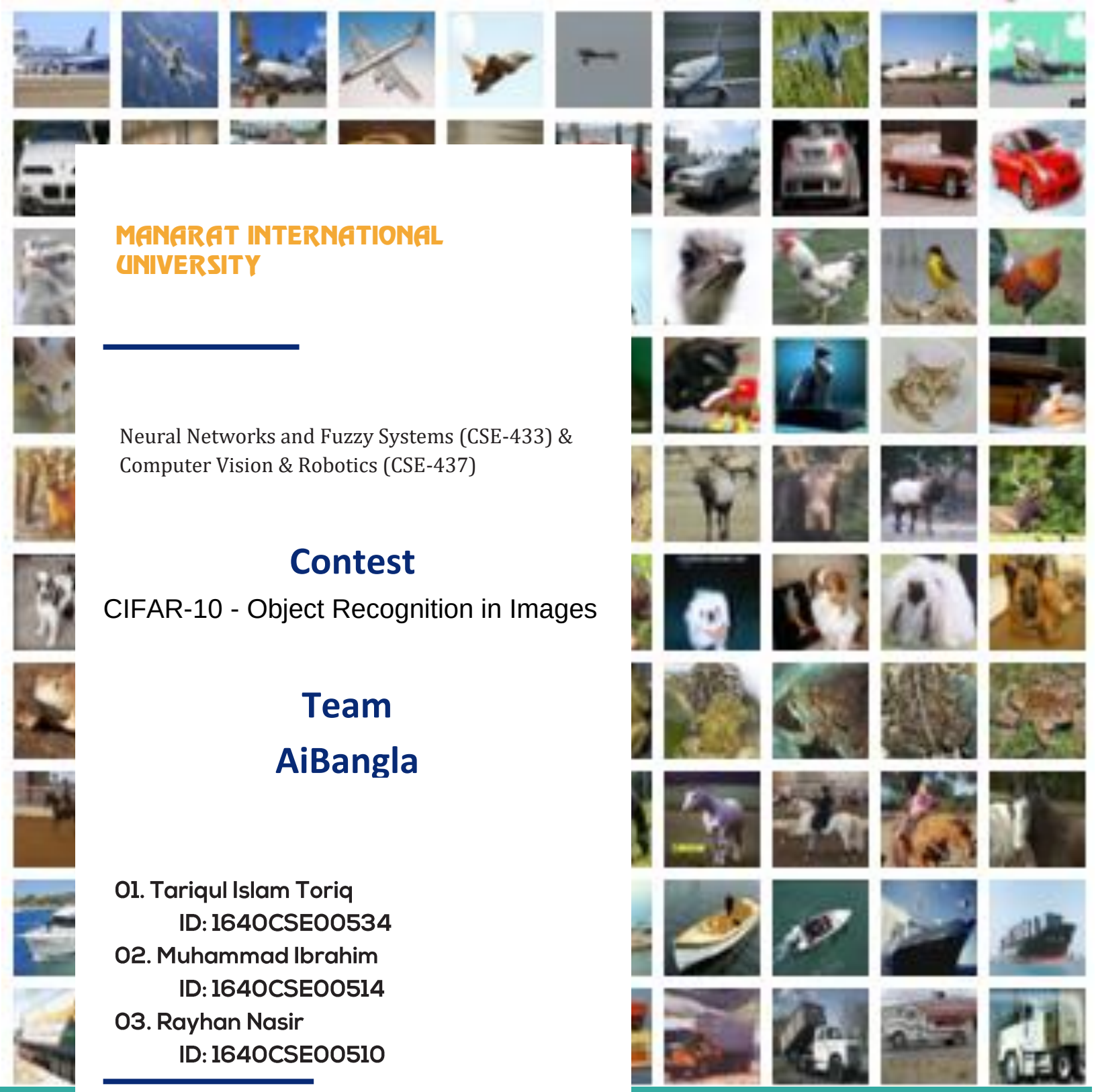


MANARAT INTERNATIONAL UNIVERSITY

Neural Networks and Fuzzy Systems (CSE-433) &
Computer Vision & Robotics (CSE-437)

Contest

CIFAR-10 - Object Recognition in Images

Team

AiBangla

01. Tariqul Islam Toriq
ID: 1640CSE00534
02. Muhammad Ibrahim
ID: 1640CSE00514
03. Rayhan Nasir
ID: 1640CSE00510

Github:
https://github.com/ikardi420/CIFer-10

Introduction

CIFAR-10 is an established computer-vision dataset used for object recognition. The CIFAR-10 data consists of 60,000 (32×32) color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in the official data. The label classes in the dataset are:

- **airplane**
- **automobile**
- **bird**
- **cat**
- **deer**
- **dog**
- **frog**
- **horse**
- **ship**
- **truck**

Data Preprocessing:

Augmentation:

In Keras, We have a ImageDataGenerator class that is used to generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches) indefinitely. The image data is generated by transforming the actual training images by rotation, crop, shifts, shear, zoom, flip, reflection, normalization etc. The below code snippets shows how to initialize the image data generator class.

Regularization:

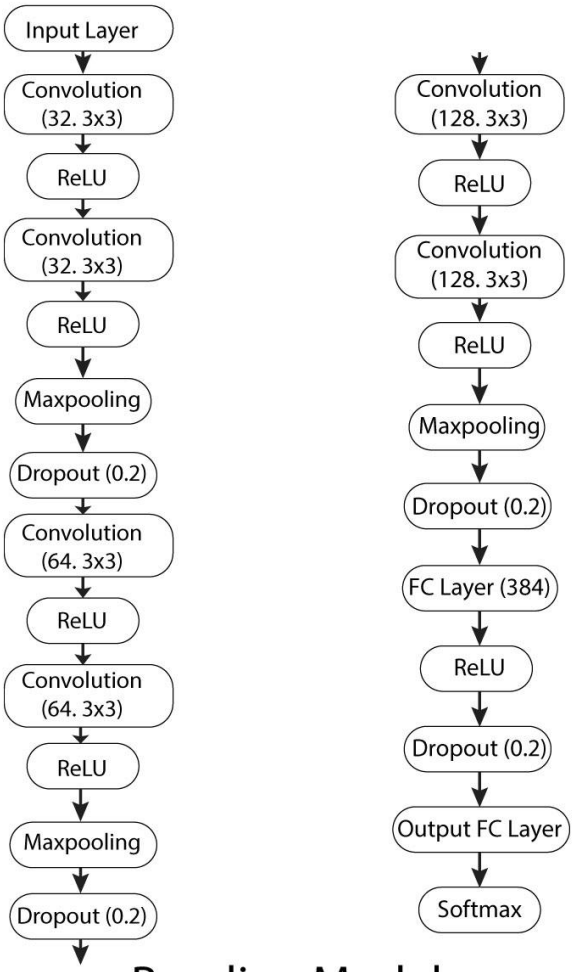
Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Given below are few techniques which were proposed recently and has become a general norm these days in convolutional neural networks.

Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. The reduction in number of parameters in each step of training has effect of regularization. Dropout has shown improvements in the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

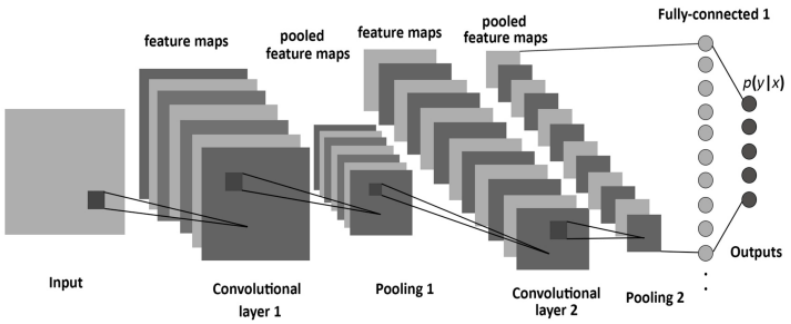
Kernel_regularizer allows to apply penalties on layer parameters during optimization. These penalties are incorporated in the loss function that the network optimizes. This argument in convolutional layer is nothing but L2 regularization of the weights. This penalizes peaky weights and makes sure that all the inputs are considered. During gradient descent parameter update, the above L2 regularization ultimately means that every weight is decayed linearly, that's why called weight decay.

BatchNormalization normalizes the activation of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. It addresses the problem of internal covariate shift. It also acts as a regularizer, in some cases eliminating the need for Dropout. Batch Normalization achieves the same accuracy with fewer training steps thus speeding up the training process

Network Architecture



Baseline Model



6 layered convolution neural network followed by flatten layer

Training Procedure

Preprocess:

Applying batch normalization.

Applying batch normalization.

Data Augmentation: ImageDataGenerator in Keras.

Building Model:

Building a Baseline Model for Training Data (1-4 submission)

Building a deep Convolutional Neural Network. (5-7 submission)

Regularization: Dropout & Kernel regularizers.

Saving & Loading DNN models.

Training:

Loading Training Data

Loading Model

Compile

Checkpoint

Test:

Loading test data

Predicting Result

Store the Result in a CSV file

Result

1st Submission
Score :0.79730 (Baseline Model, Epoch: 60 with early stopping program stop after 37 epoch)

2nd submission:
Score: 0.78360

3rd submission
Score: 0.82210 (Baseline Model Adding Conv-128, Epoch: 60 with early stopping program stop after 52 epoch)

4rth submission:
Score: 0.85040 (6 layered convolution neural network, Epoch: 50 with early stopping.)

5th submission:
Score: .87320 (6 layered convolution neural network, Epoch: 60 without early stopping. With data Augmentation)

6th submission:
Score: 0.87360 (6 layered convolution neural network, Epoch: 120 without early stopping. With data Augmentation)

7th submission:
Score: 0.87390 (6 layered convolution neural network, Epoch: 150 without early stopping. With data Augmentation)

Consolation:

If we had a higher computational power, our team believes that we can achieve better results, especially for the 6 layered convolution neural network with data augmentation. And also, we tried to ensemble our model but we didn't have time and enough computational power because ensemble with Convolution NN and baseline model required 36 hours runtime in GTX 1080 ti.

Overall, we've come to a better understanding of how image classification works and what parameters and elements are crucial to each.