Classification of CIFAR-10 database using CNN

1 Introduction

Convolutional neural networks are best when classifying images because of its ability to extract features from images and able to relate those feature within the feature hierarchy. CNNs can be very useful when we have a small database. We'll be doing it in Keras because of its large functional availability and ease of use. Keras also provides data preprocessing and data augmentation facilities.

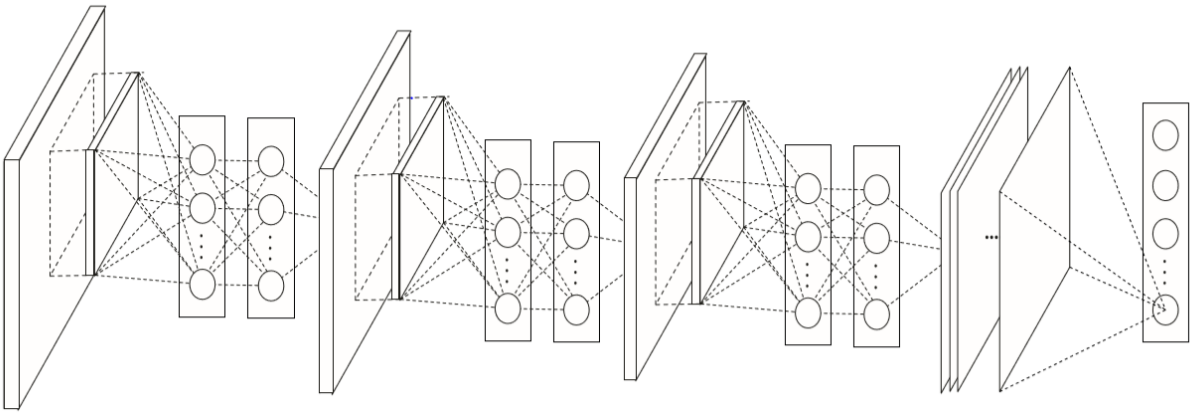
2 Dataset

CIFAR-10

The CIFAR-10 dataset [3] is a labeled subset of the 80 million tiny images dataset collected by Krizhevsky, Nair, and Hinton. It consists of 60,000 32x32 color images in 10 classes. There are 6,000 images per class. The first 50,000 images were used for training, while the remaining 10,000 comprised the validation set. We pre-processed the data in multiple ways to find the best possible configuration.

3 Model

Following is how a general CNN model looks like



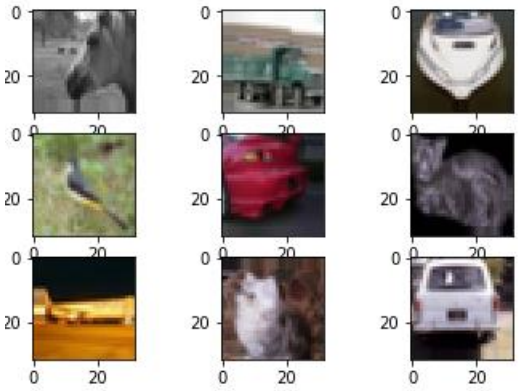
We have used a Sequential CNN model with 4 hidden layers which all use relu function as classifier, with 64 3*3 filters in first two layer the we have a 0.25 Dropout and then we have two layers of 96 3*3 filters with 0.25 Dropout for next layer. At the final layer we need to flatten the data and use a simple layer of 512 neurons activated by relu and before the final SoftMax classification we are having a Dropout of 0.5 to shuffle data more accurately.

The most important thing we explored in this project was Data Augmentation and Data preprocessing. In data preprocessing we shuffle and randomize the 50000 images that we use. But we also have to be careful that we do not have data leakage otherwise it increases chances of overfitting. In Data augmentation we process the existing label images to create its variation which produces more images. This we can inflate a dataset of 60000 images to a lot bigger data base. Using data augmentation also

Link to github : https://github.com/truptigore17/Cognitive-Computing
YouTube link : https://youtu.be/sebgpiOwvQQ

help improve training efficiency and practical performance of the model as model has already been trained in different versions of given classes it easily recognizes the classes and functions even better than humans in some cases.

To make changes easy to adapt and stream line the variation of the model we created a config.csv file from which the data is being read. We can also specify different variations of our model in consecutive rows. Our modified code will iterate through all the variation, print the augmented images and create graphical representation for each variation for comparison. So now, we have tried multiple variations of data augmentation and we have selected the model which process the most efficient network for CIFAR-10 classification. This way we could achieve efficiency of 84%.



Following are the different configs we have used:

batch_size	num_clas	epochs	data_augm	num_pre	model_name	steps_per_	featurewise_c	samplewise_cen	featurewise_std_nc	samplewise_std_norm	zca_whitening	rotation_ra	width_sh	height_shi	horizontal	vertical_flip
32	10	90	TRUE	20	keras_cifar10_trained_model1.h5	1600	FALSE	FALSE	FALSE	FALSE	TRUE	0	0.1	0.1	TRUE	FALSE
32	10	82	TRUE	20	keras_cifar10_trained_model2.h5	1600	FALSE	FALSE	FALSE	TRUE	FALSE	0	0.1	0.1	FALSE	TRUE
32	10	100	TRUE	20	keras_cifar10_trained_model3.h5	1600	TRUE	FALSE	FALSE	FALSE	FALSE	0	0.1	0.1	TRUE	FALSE
32	10	60	TRUE	20	keras_cifar10_trained_model5.h5	16	FALSE	FALSE	TRUE	FALSE	FALSE	0	0.1	0.1	FALSE	TRUE

We have achieved maximum efficiency in the second variation. Also increasing number of epochs increases the efficiency.

Conclusion :

We observe that Convolutional Neural Networks works better with images. And in contrast to the earlier algorithm provides a better accuracy overall.

From our base code, we see that the training accuracy has improvised to 84%, by inducing the following parameters :

old :
epochs = 60
data_augmentation = True
Train loss: 0.6113409750175476
Train accuracy: 0.80632

new :
epochs = 72
data_augmentation = True
Train loss: 0.4780898224115372
Train accuracy: 0.84008

Link to github : https://github.com/truptigore17/Cognitive-Computing
YouTube link : https://youtu.be/sebgpiOwvQQ

To achieve a better accuracy, we have increased two hidden layers, with increased filter count from 64 of size 3x3 to 96 filters of size 3x3. This helped us achieve the highest accuracy of all of our trials. The most noticeable achievement of this experiment is that as our training accuracy improvised to 84 % our training loss reduced, which is a sign of an efficiently trained model.

Link to github : https://github.com/truptigore17/Cognitive-Computing
YouTube link : https://youtu.be/sebgpiOwvQQ