**ICP-5:**

**Name: Amarnadha Reddy Ankireddypalli**

**ID: 16314889**

**Email: aa9f3@umsystem.edu**

**Description:**

Use the same data (that we used in ICP4 from keras.datasets import cifar10)) and use the model provided in ICP5 to perform image classification. You must change 4 hyper parameters in the source code. Report your findings in detail.

Note: please indicate in your reports which 4 hyperparameters you changed in the source code and why in your opinion these changes are logical.

**Objective:**

* Successfully executing the code and changing 4 hyperparameters in the model
* Validating the model on 5 new images (that are not present in the data set and are not used in training or testing but are taken form internet)
* Providing the logical explanation of the changes that you made to hyper parameters and over all code quality

**Implementation (with screenshots):**

1. Import all the necessary libraries for image classification.

Graphical user interface, text, application

Description automatically generated

1. Split the cifar10 dataset into train and test.

Graphical user interface

Description automatically generated with medium confidence

1. Display the first image of each of the 10 categories present in the cifar10 data.

Graphical user interface, website

Description automatically generated

1. In most circumstances, you'll need to preprocess your data before utilizing it, but since we're using a preloaded dataset, we just need to do a little preprocessing. One of the things we'd like to do is standardize the data. The pixels in the image, which have a value between 0 and 255, represent the input values in this example.
2. So, we can just divide the picture values by 255 to normalize the data. Because the data is currently integer, we must first convert it to a float type. We can accomplish this by first calling astype() and then declaring the data type.

Graphical user interface, text, application

Description automatically generated

1. Defining the Sequential model using various layers:

* Dropout () layer removes unnecessary feature dependencies in the network, making it simpler and reduces overfitting.
* The purpose of max-pooling is to down-sample an input representation. As a result, the model is less sensitive to certain translations.
* Flattening is converting the data into a 1-dimensional array for inputting it to the next layer.

Graphical user interface, text, application

Description automatically generated

1. Fitting the model using 10 epochs with batch size as 32. It took 30min to process the 10 epochs generating train accuracy of 83.78% and test accuracy of 74.67%

Text

Description automatically generated with medium confidence

Chart

Description automatically generated with low confidence

Observations:

The model fitted pretty well with train accuracy of 83.78%, but the predictions it made are awful.

The model predicted airplane as truck. It needs to be hyper tuned for more efficiency in predictions. So, adding 20% dropout after first MaxPooling,30% Dropout after the Dense layer for the next model and changing the optimizer to RMSprop with a learning rate=0.0001, decay=1e-6

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

The model did average with an accuracy of 73.55%.

Tuning furthermore:

* In 1st phase, our network will learn 32 convolutional filters of 3 x 3 size. Activation is relu, which is a straightforward method of creating non-linearity.
* Following that, we have a max-pooling operation with a pool size of 2 x 2, followed by two more 64 convolutional filters, each with a 3 x 3 size and activation and two more 128 convolutional filters of same size and relu activation.
* In each stage, I’m adding BatchNormalization () since it has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.
* In the following stage of the deep pipeline, our model will drop out at 50% and then flatten the layers.
* The deep pipeline concludes with a dense network with 128 units with relu activation, followed by another dense layer with softmax activation layer and 10 classes as output, one for each category.

Text

Description automatically generated with medium confidence

Table

Description automatically generated

Graphical user interface, text

Description automatically generated

Table

Description automatically generated

Chart, histogram

Description automatically generated

1. The model did well with training accuracy of 84.3% and validation accuracy of 85.2%. Time to make predictions out of it.
2. Converting the predictions into label index.

Text

Description automatically generated

1. Evaluating the model against test images to perform predictions.

A picture containing text

Description automatically generated

The model predicted accurately in most of the cases with the testing data. So, checking on the images outside of cifar10.

1. **Testing the model with the images outside the cifar10 dataset.**

Graphical user interface

Description automatically generated

A horse running on grass

Description automatically generated with medium confidence

Graphical user interface, website

Description automatically generated

A bird on a branch

Description automatically generated with medium confidence

A green frog on a leaf

Description automatically generated with low confidence

A picture containing text, screenshot

Description automatically generated

1. Overall, the model predicted the external images accurately (success).

**Difficulties faced:**

* In understanding and proper tuning of Activation and Dense layers for the Sequential model.
* The model execution took a lot of time (3hrs each) for learning after tuning various hyperparameters. Hence, I need to use only 25 epochs for the final model, taking help from BatchNormalization().

**Video Link:** [**https://youtu.be/MXPdV1Yi49U**](https://youtu.be/MXPdV1Yi49U)

**Conclusion:**

From this ICP5, I’ve learnt:

* How to use proper values for dense and activation layers, keeping in mind of their functions.
* How to train a simple deep learning model using keras and understanding the training and validation loss and accuracy, it generates.
* Overall, it’s a good learning having tested, tuning various hyper parameters in evaluating the model’s performance.