BT3041

ASSIGNMENT – 2

UDAY TALLAPELLY

ME19b053

PART – 1

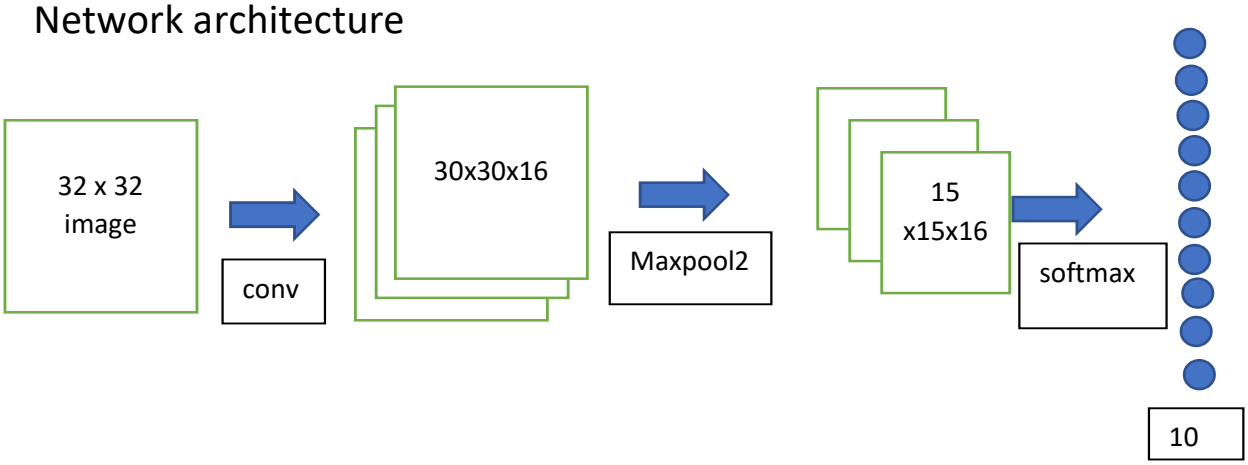
Implement Convolutional Neural Network for classifying the given CIFAR-10 dataset

Functions and classes used in code

1. Unpickle function to load dataset'
2. Rgb2bw function to convert input image with three channels (RGB) to greyscale.
   Input = (32,32,3) NumPy array
   Output = (32,32) NumPy array
3. Conv Class to apply convolution of 2d image (2d NumPy array) and output 3d array of convolution layers. It takes kernel_size, padding, strides as inputs.
    1. iterate_regions: Generates all possible (kernel_size x kernel_size) image regions.
    2. forward: Performs a forward pass of the conv layer using the given input image. Returns a 3d NumPy array with dimensions (h, w, num_filters).
    3. Backprop: Performs a backward pass of the conv layer
4. MaxPool2 class to apply max pooling on 3d array given kernel_size, padding and strides as input. It takes kernel_size, padding, strides as inputs.
    - iterate_regions: Generates non-overlapping (kernel_size x kernel_size) image regions to pool over.
    - Forward: Performs a forward pass of the maxpool layer using the given input. Returns a 3d NumPy array with dimensions (h // 2, w // 2, num_filters)
    - Backprop: Performs a backward pass of the maxpool layer. Returns the loss gradient for this layer's inputs
5. Softmax class takes s takes the output tensor of previous layer as input, and after flattening feeds it to a dense layer, which calculates probabilities for the image to be in a particular class using Softmax function in the forward function. The gradients of loss with respect to dense layer are computed in backprop function as below
    - Forward: Performs a forward pass of the softmax layer using the given input. Returns a 1d NumPy array containing the respective probability values.
    - Backprop: Performs a backward pass of the softmax layer. Returns the loss gradient for this layer's inputs.
6. Forward function completes a forward pass of the CNN and calculates the accuracy and cross-entropy loss.

7. Train function Completes a full training step on the given image and label. Returns the cross-entropy loss and accuracy.

a. Network architecture



Conv() function is fed with 32 X 32 images and it applies 16 convolution filters as outputs a (30,30,16) array for each image. The maxpooling is applied on output of the convolution and this output (15,15,16) NumPy array then this is fed to neural network with 10 neurons and softmax activation function to output (10,) NumPy array with each input as probability of the image belonging to respective class.

b. Activation Function

Relu function was used in activation layer for inducing non-linearity and at the at the output layer softmax is used as activation layer.

c. Loss Function

As this class is a multiclass classification we use cross entropy function as loss function.

d. Learning Algorithm

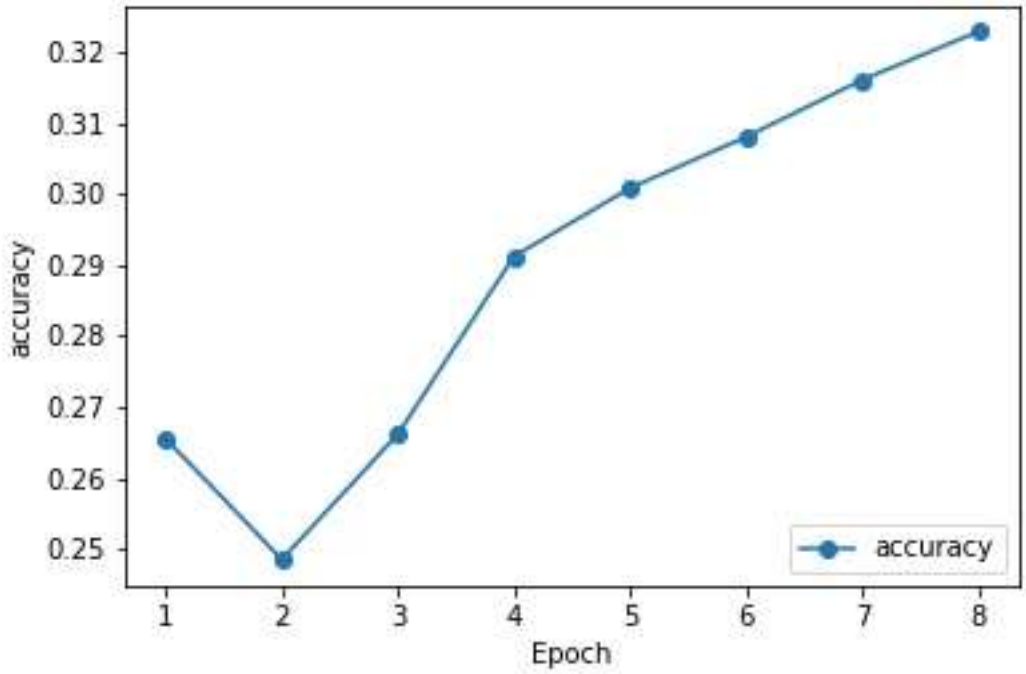
Back propagation of error forms the output layer and updating weights at each layer is used as learning algorithm to improve performance of model.

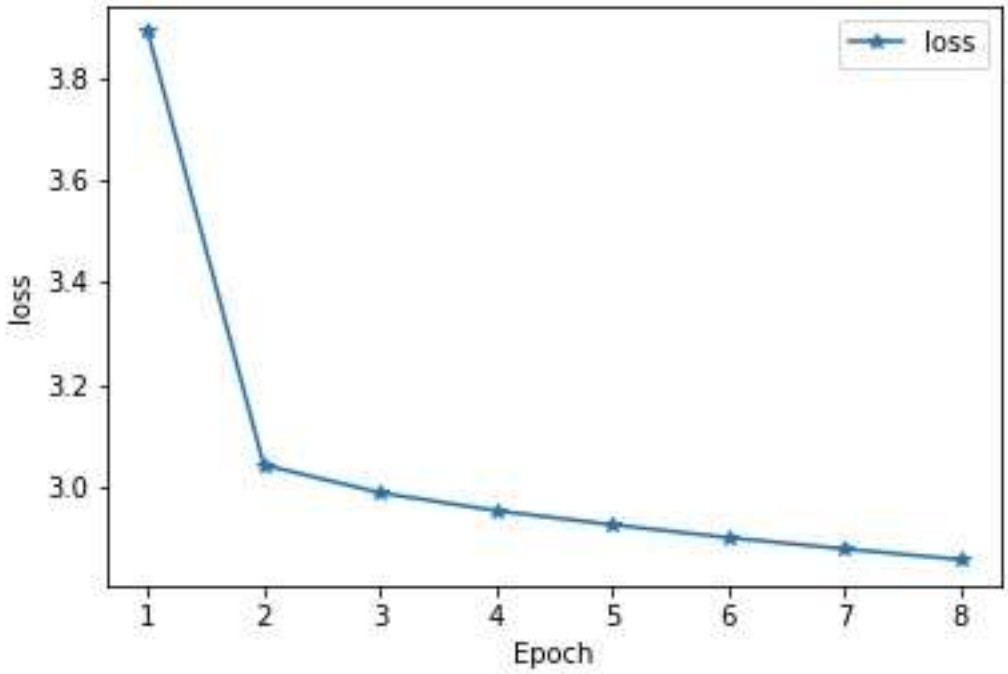
e. Learning Rate

Learning rate is a hyperparameter to be tuned for optimal learning rate. Here in this model I use learning rate of 0.005

```
Train Loss: 2.834
Train Accuracy: 0.338

Test Loss: 2.8873072249543372
Test Accuracy: 0.319
```

Model trained to 5000 images for 8 epochs





3D CONVOLUTION - PART 2:

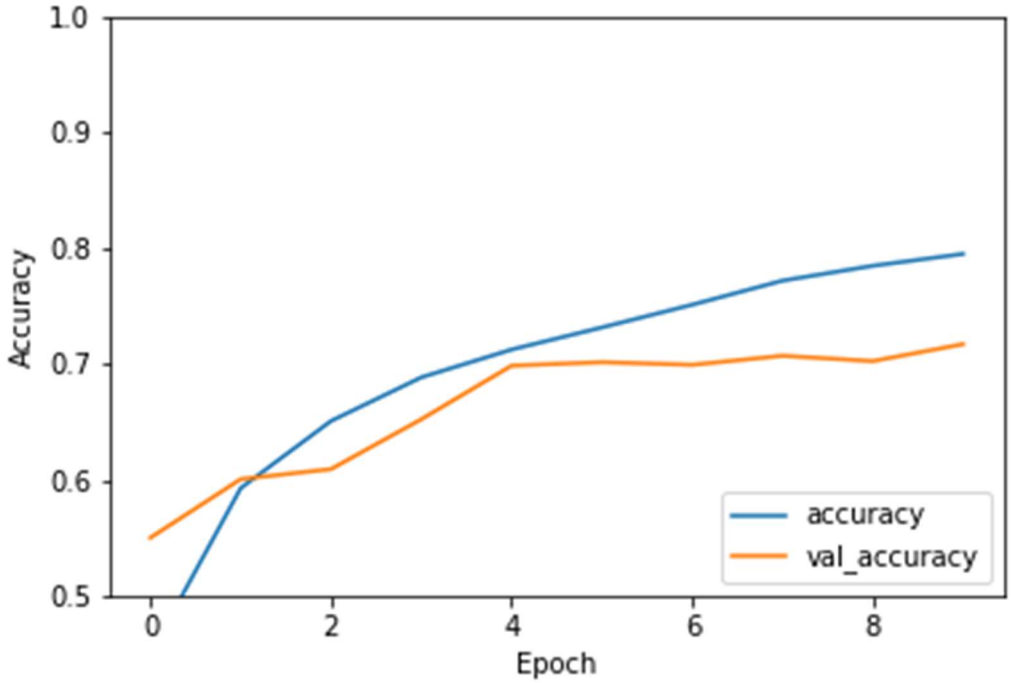
In this part I used TensorFlow library to perform classification on CIFAR10 dataset.

- **Network Architecture**

```
In [8]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

- **Loss Function –** Sparse Categorical Cross entropy
- **Activation Function –** ReLU activation function in each layer
- **Learning Algorithm –** Adam optimizer
- **Learning Rate –** Default learning rate of 0.001 is used




```
Train accuracy = 79.42 percent
Test accuracy = 71.66 percent
```