National University
of computer and emerging sciences

Artificial Intelligence

Project Report

Image Classification using CNN

Members:

K17-3730 Eisha Tir Raazia
K17-3658 Maham Tariq

Image Classification using CNN

1. Project Motivation

- To train convolutional neural networks, using Keras, to solve image classification problems.

2. Problem Definition/Structure

- Data set: Cifar-10 image data
    - Input: Images of random classes, out of 10.
    - Output: Class Label
- The shape of Train and Test Data:
    - (50000, 32, 32, 3) (50000, 10)
    - (10000, 32, 32, 3) (10000, 10)

3. Classes to be predicted (Y-column Data):

- No of total classes: 10.
- Classes:
    1. Airplane
    2. Automobile
    3. Bird
    4. Cat
    5. Deer
    6. Dog
    7. Frog
    8. Horse
    9. Ship
    10. Truck

3. Method and Structure:

1. Importing the required libraries and helper functions.
2. Pre-process Data
    2.1. Importing the CIFAR-10 dataset.
    2.2. Creating a subset of the dataset which has just 3 classes instead of 10. This is done for both the training and test set.
    2.3. Randomly shuffling the newly created subset.
3. Visualize Examples
    3.1. Plotting randomly selected examples of a given set.
    3.2. We look at some examples from training and test set along with their labels.

4. Create the Model
    4.1. Creating a Keras Sequential model.
    4.2. Creating a function to add a convolutional block to the model.
    4.3. A look at the model summary.
5. Train the Model
    5.1. Fit the model on the subset.
    5.2. Setting the EarlyStopping callback.
    5.3. Setting the ModelCheckpoint callback.
6. Final Predictions
    6.1. Plotting the training and validation accuracy from the training.
    6.2. Loading the best model.
    6.3. Getting predictions on the test set and displaying the results.
7. Evaluating the model

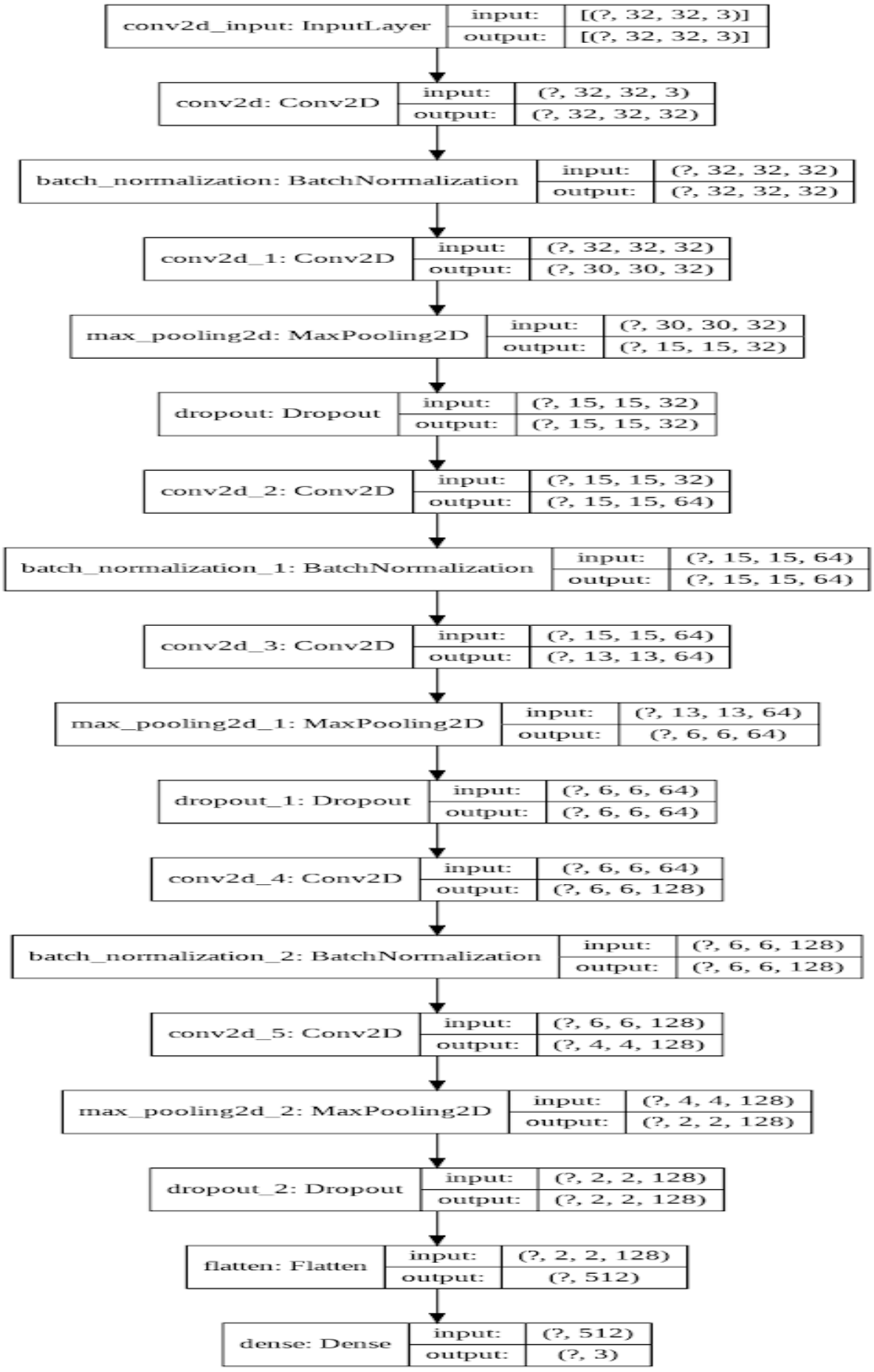
5. Performance Measurement

- Performance of the Neural network has been measured by running epochs on different architectures, we also applied some transfer learning algorithms i.e: resnet-50.
- The model/ layered architecture on which we got maximum accuracy contained the following layers:
    - Conv2D
    - MaxPooling2D
    - Flatten
    - Dropout.

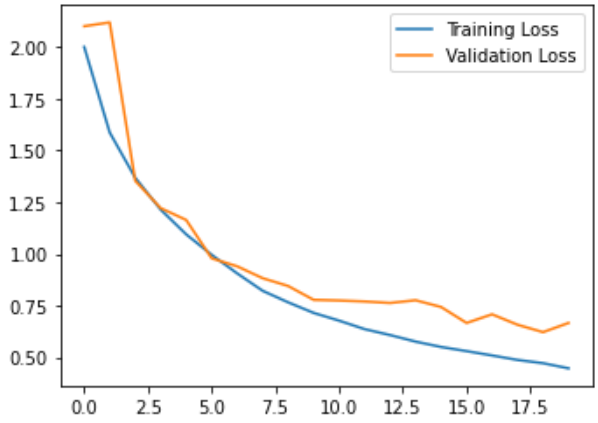
6. Risks and Dependencies

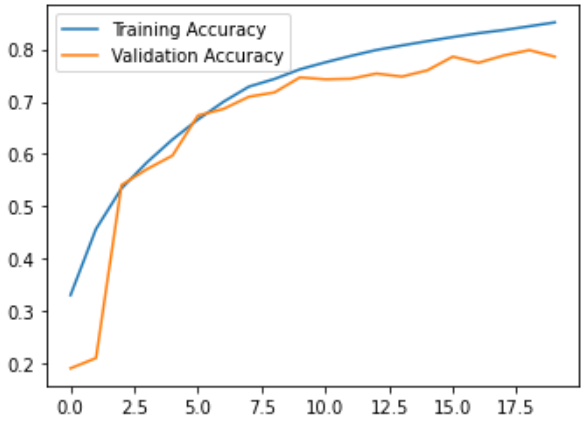
- The images had to be processed with less max-pooling layers.
- Dependent on GPU, for running epochs with better speed.

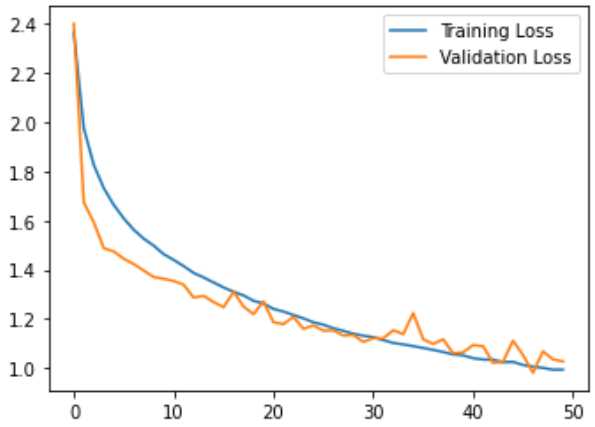
7. Neural Network Architecture

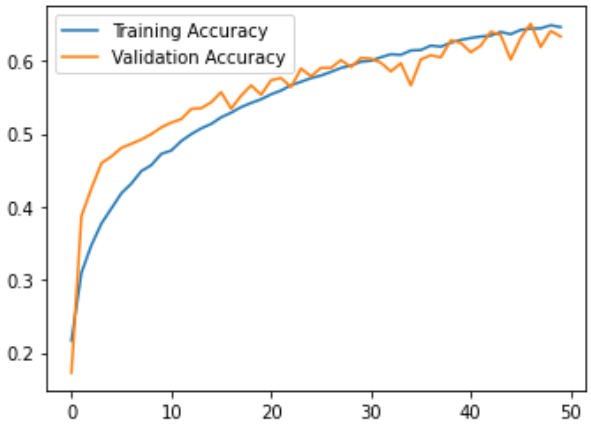

conv2d_input: InputLayer
input: [(?, 32, 32, 3)]
output: [(?, 32, 32, 3)]
conv2d: Conv2D
input: (?, 32, 32, 3)
output: (?, 32, 32, 32)
batch_normalization: BatchNormalization
input: (?, 32, 32, 32)
output: (?, 32, 32, 32)
conv2d_1: Conv2D
input: (?, 32, 32, 32)
output: (?, 30, 30, 32)
max_pooling2d: MaxPooling2D
input: (?, 30, 30, 32)
output: (?, 15, 15, 32)
dropout: Dropout
input: (?, 15, 15, 32)
output: (?, 15, 15, 32)
conv2d_2: Conv2D
input: (?, 15, 15, 32)
output: (?, 15, 15, 64)
batch_normalization_1: BatchNormalization
input: (?, 15, 15, 64)
output: (?, 15, 15, 64)
conv2d_3: Conv2D
input: (?, 15, 15, 64)
output: (?, 13, 13, 64)
max_pooling2d_1: MaxPooling2D
input: (?, 13, 13, 64)
output: (?, 6, 6, 64)
dropout_1: Dropout
input: (?, 6, 6, 64)
output: (?, 6, 6, 64)
conv2d_4: Conv2D
input: (?, 6, 6, 64)
output: (?, 6, 6, 128)
batch_normalization_2: BatchNormalization
input: (?, 6, 6, 128)
output: (?, 6, 6, 128)
conv2d_5: Conv2D
input: (?, 6, 6, 128)
output: (?, 4, 4, 128)
max_pooling2d_2: MaxPooling2D
input: (?, 4, 4, 128)
output: (?, 2, 2, 128)
dropout_2: Dropout
input: (?, 2, 2, 128)
output: (?, 2, 2, 128)
flatten: Flatten
input: (?, 2, 2, 128)
output: (?, 512)
dense: Dense
input: (?, 512)
output: (?, 3)

8. Model Visualization(for 10 classes_Initial Runs)





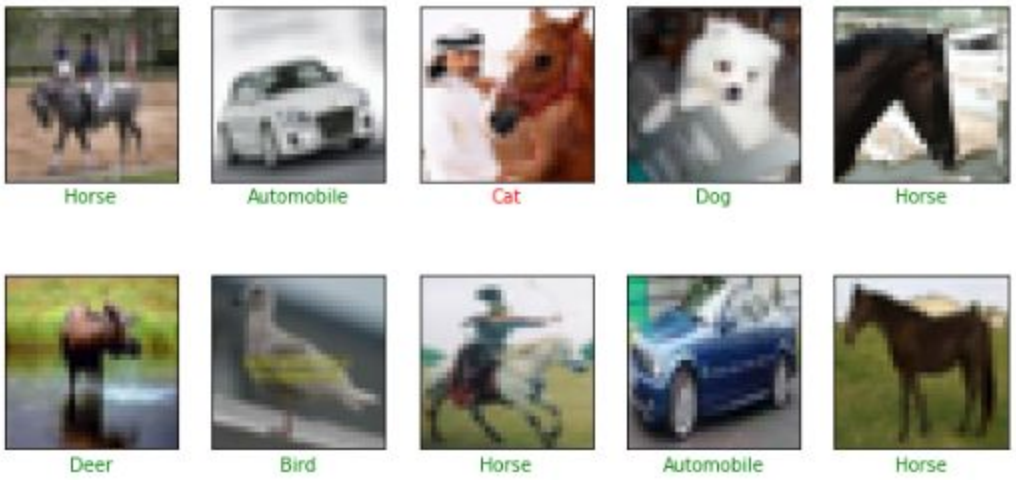




9. Random Test Predictions' Results

```
1 show_random_examples(x_test, y_test, preds)
```


```
1 show_random_examples(x_test, y_test, preds)
```



10. Run performance checks

- No. of total epochs: 30 - 100
- The model with shuffle true gave 80+ accuracy on majority epochs.
- Max accuracy achieved: 91-92% (for 3-5 classes)
- 80+ % accuracy achieved for 10 classes prediction.

11. References

- https://cs231n.github.io/
- https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/ (to see behaviour on different architectures)
- https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/