# PR: PROJ3

## December 30, 2022

**S.M. Erfan Moosavi M.**

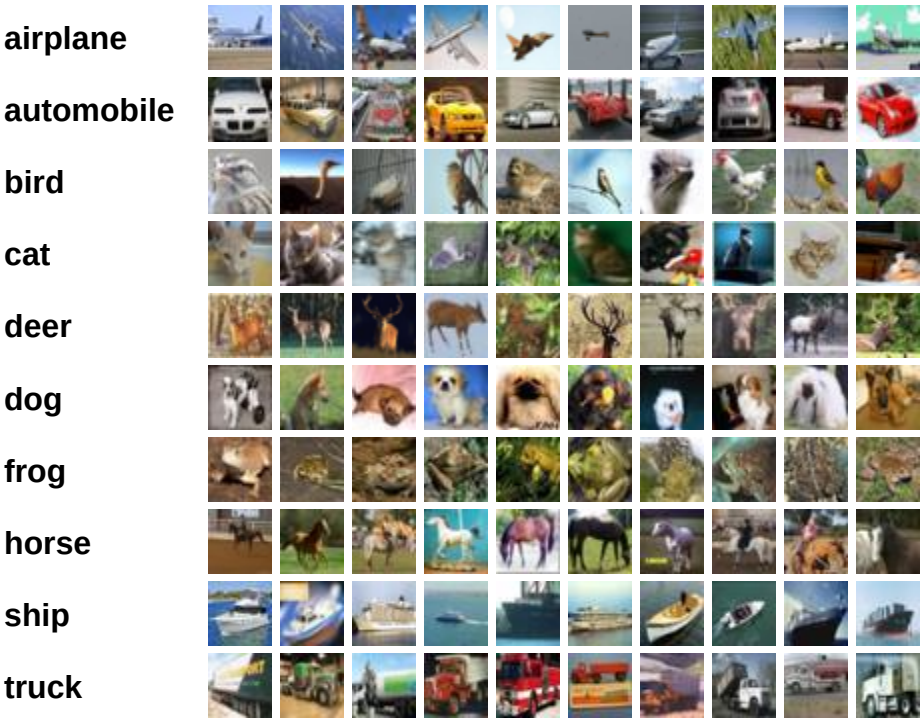
**Special thanks to**
**https://github.com/priyavrat-misra/cifar10**

# TABLE OF CONTENTS

- **Part 1 <pattern.ipynb>**

Done.

- **Part 2 <data-analysis.ipynb>**




I've selected CIFAR-10 dataset from this link:

Datasets — Torchvision main documentation (pytorch.org)

Then with the script <data-analysis.ipynb> I analyzed the data to understand it shape, and some basic qualities.

CIFAR-10 main page:

CIFAR-10 and CIFAR-100 datasets (toronto.edu)

CIFAR-10 torch docs page:

CIFAR10 — Torchvision main documentation (pytorch.org)

Dataset has 50000 training images and 10000 test ones.

Total 60000 samples divided into 10 classes:

'airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck'

Images are 32*32 RGB (3 Channels) and for each class we have 6000 sample images.

Dataset is separated into 5 training batches and 1 test batch, each containing of 10000 images.

Images are not divided between training batches uniformly (which means, one batch may contain more images for one class than the others), But there are exactly 5000 training image and 1000 test image for each class. (This means test images are uniformly divided)

These lines of codes first check the specified path for cached dataset, if it not found any, it downloads the datasets, store them at cache path and gives us the datasets objects.

```
(1)  train_path = 'ds/train/'
(2)  test_path = 'ds/test/'
(3)  dstrain = datasets.CIFAR10(train_path, train=True,
   download=True, transform=transform)
(4)  dstest = datasets.CIFAR10(test_path, train=False,
   download=True, transform=transform)
```

- **Shared Codes**

In each notebook, you can see that the first few cells are the same, so here I'm going to explain those shared codes:

  - *Training:*

First & Second cells:

Importing libs and looking for CUDA. (If CUDA is not available, CPU will be used instead.)

Third cell:

Here we defined the transforms which we're going to apply on datasets before training phase.

MLP: Converting PIL image objects to Tensors and Normalize tensor values.

```
(5)   transform = transforms.Compose([
(6)       transforms.ToTensor(),
(7)       transforms.Normalize((0,), (1,)),
(8)   ])
```

CNN & VGG16:

Augmenting data by applying random flips and rotations. (notice we only augment training data). Applying ToTensor and Normalization as before.

```
(9)   transform_train = transforms.Compose([
(10)      # Augmentations
(11)      transforms.ColorJitter(brightness=0.25,
    saturation=0.1),
(12)      transforms.RandomHorizontalFlip(p=0.5),
(13)      transforms.RandomVerticalFlip(p=0.5),
(14)      transforms.RandomRotation(10), # -10 to +10 degrees
(15)
(16)      transforms.ToTensor(),
```

```
(17)     transforms.Normalize((0.485, 0.456, 0.406), (0.229,
   0.224, 0.225)),
(18) ])
(19)
(20) transform_test = transforms.Compose([
(21)     transforms.ToTensor(),
(22)     transforms.Normalize((0.485, 0.456, 0.406), (0.229,
   0.224, 0.225)),
(23) ])
```

Forth & Fifth & Sixth & Seventh cells: Now we're going to download the dataset (CIFAR10) and sperate it into train and test sets. If a previously downloaded data is available, we're going to use it. We also run previously defined transforms on our datasets.

```
(24) train_path = 'ds/train/'
(25) test_path = 'ds/test/'
(26) dstrain = datasets.CIFAR10(train_path, train=True,
   download=True, transform=transform)
(27) dstest = datasets.CIFAR10(test_path, train=False,
   download=True, transform=transform)
```

Here we defined a loader which is a wrapper around data that can help us divide data into batches and only load those batches that we need. It's saving lots of memory. Also, we defined our batch size to be 64.

```
(28) ldtrain = torch.utils.data.DataLoader(dstrain,
   batch_size=64, shuffle=True)
(29) ldval = torch.utils.data.DataLoader(dstest,
   batch_size=64, shuffle=True)
```

8th & 9th cells: In these cells, we plotted a batch of images.

10th cell: This cell contains the model architecture. Each model along with its architecture and performance are described in the next chapters.

## 5

11th cell: Assigning model to runtime device (CUDA or CPU)

12th cell: Defining training hyper parameters. (Learning rate, epochs, validation frequency, optimizer and loss function)

13th cell: A helper function to calculate accuracy.

14th cell: Training main loop. First, we assign training data (one batch) to the runtime device, then calculate model's output for that batch and use it to compute loss. After that we backward propagate the loss through model layers. After that we take the optimizer one step forward and calculate batch loss and accuracy. After going through all batches, we calculate epoch loss and accuracy, report it back to user (print) and going for the next iteration.

Depending on validation frequency, we calculate the validation loss and accuracy but this time the loss does not propagate through network, we only keep it in order to report back to user.

15th cell: This cell used to plot loss and accuracy against epoch number.

- *Result Calculations*

First and Second and Third cells:

Importing libs and checking for CUDA. Also, we defined some parameters in order to use same notebook for evaluating all models. These paths point to model path, folders to save some visualizations.

Forth cell: Defining transforms and data loaders.

Fifth cell: Defining model architecture.

Sixth cell: In this cell we are going through each batch and feed it to model, get the result and store the results. We do this once for training and once for testing.

Seventh cell: Here we calculated test and train accuracy.

### 8th cell:

This cell used for stacking predictions and real values on each other. The stacked tensor corresponds each real value to predicted one.

### 9th cell:

Here we plotted some sample images with their real and predicted labels. (Labels in parenthesis are real).

### 10th & 11th & 12th cells:

In these cells, we calculated both train and test confusion matrix using the stacked tensors and the visualized those matrices.

### 13th & 14th cells:

These cells used to calculate class-wise accuracies.

### Last but not least cell:

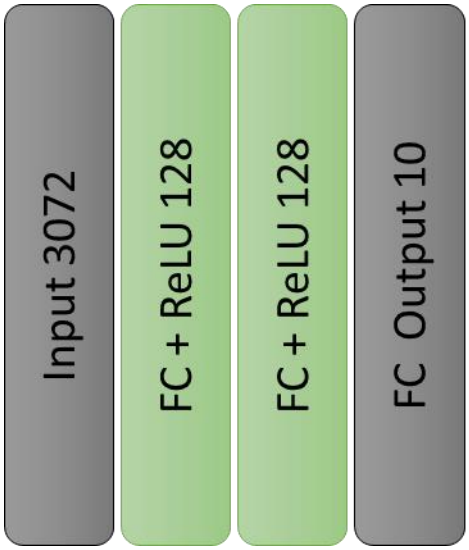
Here we calculated class-wise and total precision, recall, f1 score for both training and test data.

- **Part 3 (MLP Model)**

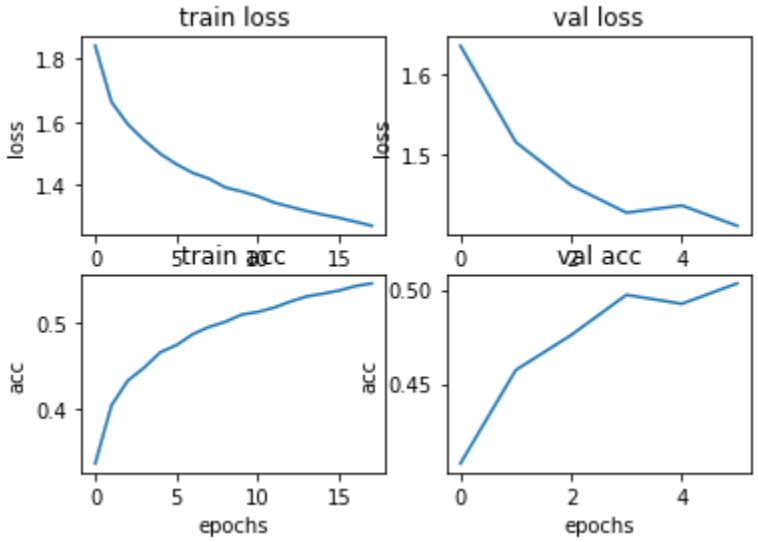
Training Script: <cifar10-mlp.ipynb>

Result Script: <results-mlp.ipynb>

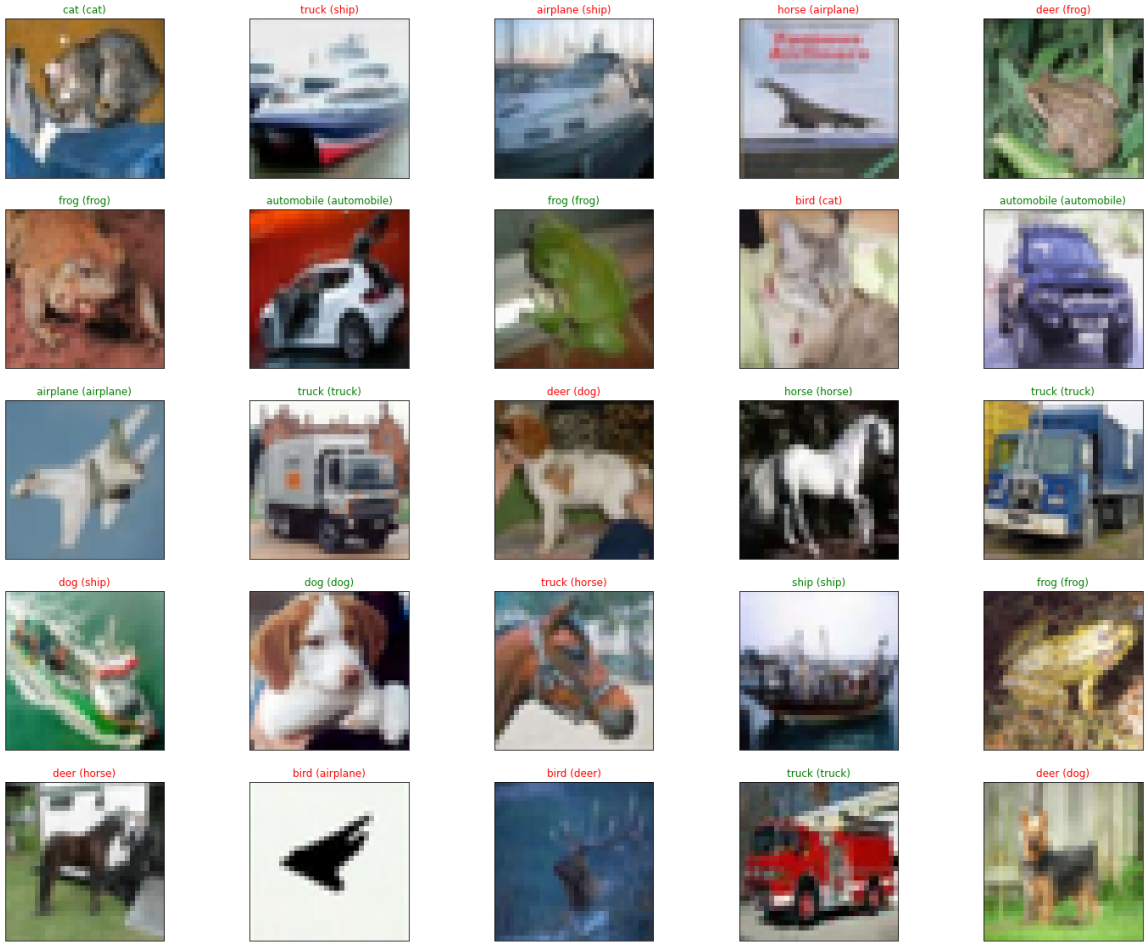
Architecture:



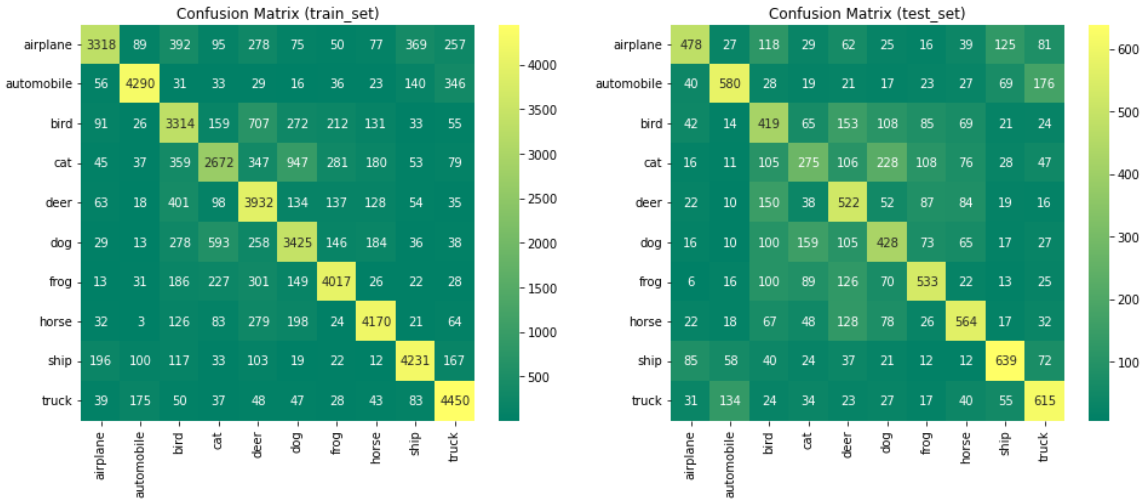

Training loss & accuracy:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(30) Train accuracy of airplane    :    66.36% (3318/5000)
(31) Train accuracy of automobile  :    85.80% (4290/5000)
(32) Train accuracy of bird        :    66.28% (3314/5000)
(33) Train accuracy of cat         :    53.44% (2672/5000)
(34) Train accuracy of deer        :    78.64% (3932/5000)
(35) Train accuracy of dog         :    68.50% (3425/5000)
(36) Train accuracy of frog        :    80.34% (4017/5000)
(37) Train accuracy of horse       :    83.40% (4170/5000)
(38) Train accuracy of ship        :    84.62% (4231/5000)
(39) Train accuracy of truck       :    89.00% (4450/5000)
```

Per Class Test Accuracy:

```
(40) Test accuracy of airplane    :47.80% (478/1000)
(41) Test accuracy of automobile  :58.00% (580/1000)
(42) Test accuracy of bird        :41.90% (419/1000)
(43) Test accuracy of cat         :27.50% (275/1000)
(44) Test accuracy of deer        :52.20% (522/1000)
(45) Test accuracy of dog         :42.80% (428/1000)
(46) Test accuracy of frog        :53.30% (533/1000)
(47) Test accuracy of horse       :56.40% (564/1000)
(48) Test accuracy of ship        :63.90% (639/1000)
(49) Test accuracy of truck       :61.50% (615/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(50) [Train]:
(51)                precision    recall  f1-score   support
(52)
(53)            0       0.85      0.66      0.75      5000
(54)            1       0.90      0.86      0.88      5000
(55)            2       0.63      0.66      0.65      5000
(56)            3       0.66      0.53      0.59      5000
(57)            4       0.63      0.79      0.70      5000
(58)            5       0.65      0.69      0.67      5000
(59)            6       0.81      0.80      0.81      5000
(60)            7       0.84      0.83      0.84      5000
(61)            8       0.84      0.85      0.84      5000
(62)            9       0.81      0.89      0.85      5000
(63)
(64)     accuracy                           0.76     50000
(65)    macro avg       0.76      0.76      0.76     50000
(66) weighted avg       0.76      0.76      0.76     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

```
(67) [Test]:
(68)                precision    recall  f1-score   support
(69)
(70)            0       0.63      0.48      0.54      1000
(71)            1       0.66      0.58      0.62      1000
(72)            2       0.36      0.42      0.39      1000
(73)            3       0.35      0.28      0.31      1000
(74)            4       0.41      0.52      0.46      1000
(75)            5       0.41      0.43      0.42      1000
(76)            6       0.54      0.53      0.54      1000
(77)            7       0.57      0.56      0.56      1000
(78)            8       0.64      0.64      0.64      1000
(79)            9       0.55      0.61      0.58      1000
(80)
(81)     accuracy                           0.51     10000
(82)    macro avg       0.51      0.51      0.51     10000
(83) weighted avg       0.51      0.51      0.51     10000
```

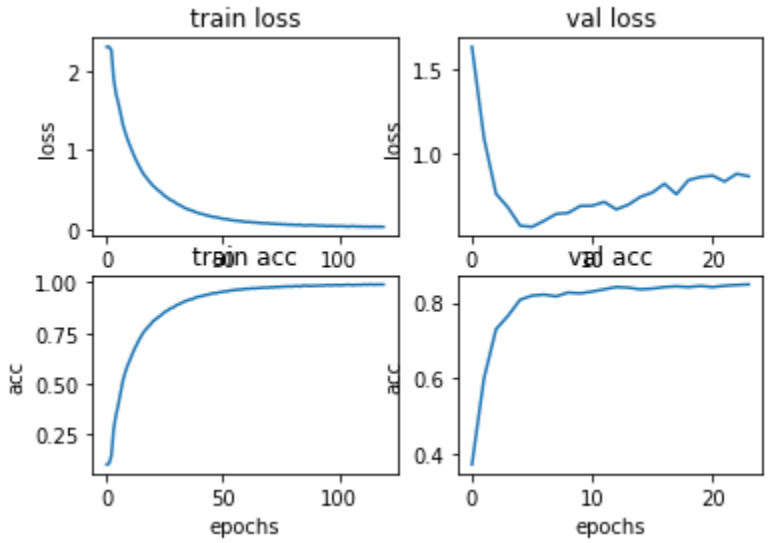
- **Part 4 (CNN without BN & DO)**

Training Script: <cifar10-cnn.ipynb>

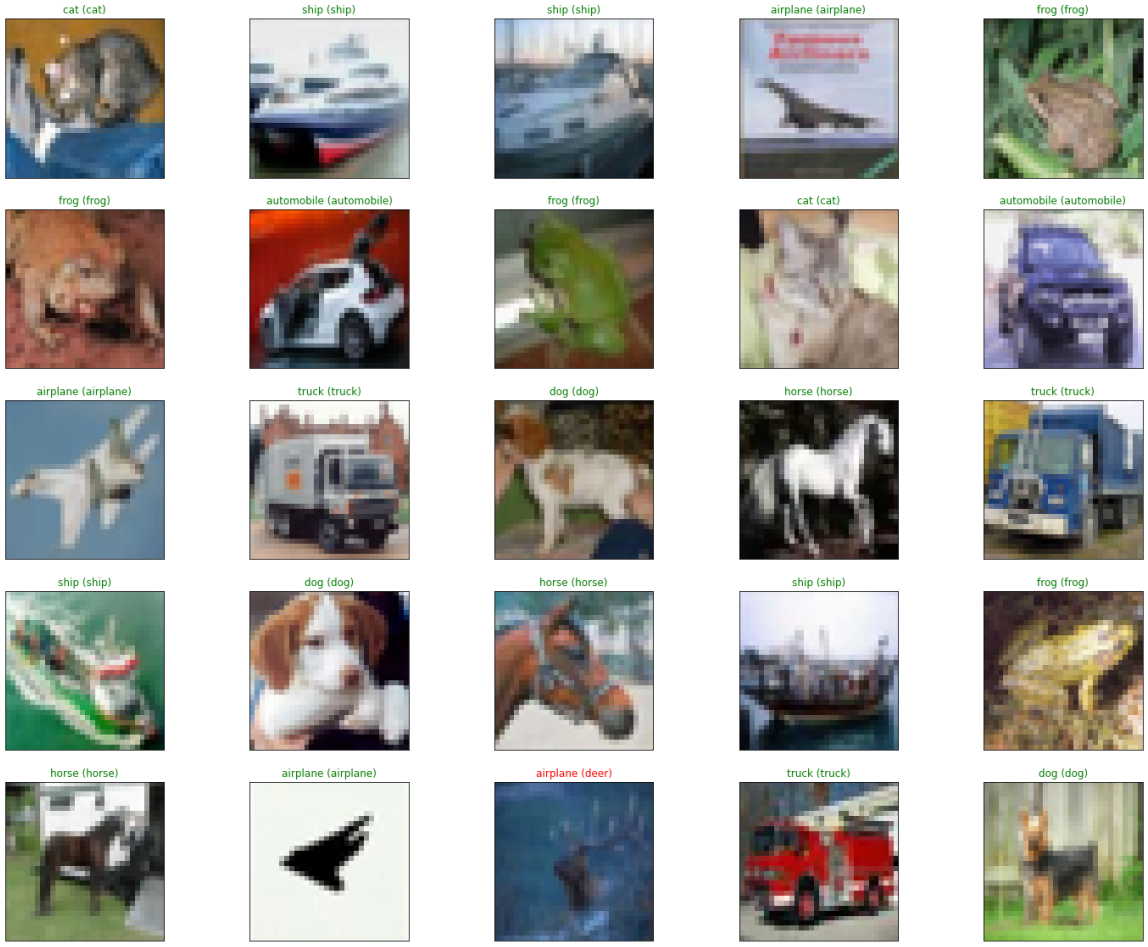
Result Script: <results-cnn.ipynb>

Architecture:

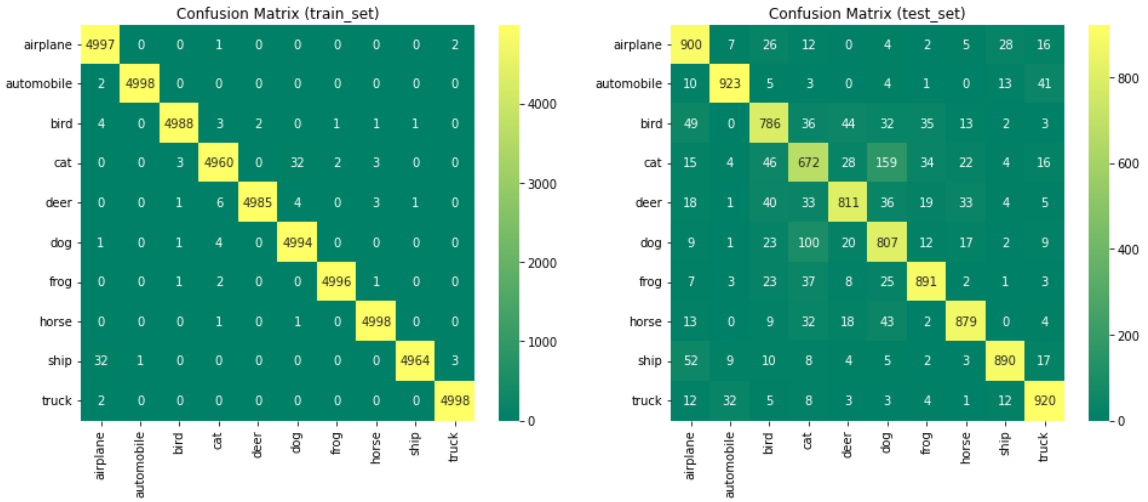

Training loss & accuracy:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(84) Train accuracy of airplane    :    99.94% (4997/5000)
(85) Train accuracy of automobile  :    99.96% (4998/5000)
(86) Train accuracy of bird        :    99.76% (4988/5000)
(87) Train accuracy of cat         :    99.20% (4960/5000)
(88) Train accuracy of deer        :    99.70% (4985/5000)
(89) Train accuracy of dog         :    99.88% (4994/5000)
(90) Train accuracy of frog        :    99.92% (4996/5000)
(91) Train accuracy of horse       :    99.96% (4998/5000)
(92) Train accuracy of ship        :    99.28% (4964/5000)
(93) Train accuracy of truck       :    99.96% (4998/5000)
```

Per Class Test Accuracy:

```
(94) Test accuracy of airplane    :90.00% (900/1000)
(95) Test accuracy of automobile  :92.30% (923/1000)
(96) Test accuracy of bird        :78.60% (786/1000)
(97) Test accuracy of cat         :67.20% (672/1000)
(98) Test accuracy of deer        :81.10% (811/1000)
(99) Test accuracy of dog         :80.70% (807/1000)
(100)Test accuracy of frog        :89.10% (891/1000)
(101)Test accuracy of horse       :87.90% (879/1000)
(102)Test accuracy of ship        :89.00% (890/1000)
(103)Test accuracy of truck       :92.00% (920/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(104)[Train]:
(105)              precision    recall  f1-score   support
(106)
(107)           0       0.99      1.00      1.00      5000
(108)           1       1.00      1.00      1.00      5000
(109)           2       1.00      1.00      1.00      5000
(110)           3       1.00      0.99      0.99      5000
(111)           4       1.00      1.00      1.00      5000
(112)           5       0.99      1.00      1.00      5000
(113)           6       1.00      1.00      1.00      5000
(114)           7       1.00      1.00      1.00      5000
(115)           8       1.00      0.99      1.00      5000
(116)           9       1.00      1.00      1.00      5000
(117)
(118)    accuracy                           1.00     50000
(119)   macro avg       1.00      1.00      1.00     50000
(120)weighted avg       1.00      1.00      1.00     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

```
(121)[Test]:
(122)              precision    recall  f1-score   support
(123)
(124)           0       0.83      0.90      0.86      1000
(125)           1       0.94      0.92      0.93      1000
(126)           2       0.81      0.79      0.80      1000
(127)           3       0.71      0.67      0.69      1000
(128)           4       0.87      0.81      0.84      1000
(129)           5       0.72      0.81      0.76      1000
(130)           6       0.89      0.89      0.89      1000
(131)           7       0.90      0.88      0.89      1000
(132)           8       0.93      0.89      0.91      1000
(133)           9       0.89      0.92      0.90      1000
(134)
(135)    accuracy                           0.85     10000
(136)   macro avg       0.85      0.85      0.85     10000
(137)weighted avg       0.85      0.85      0.85     10000
```

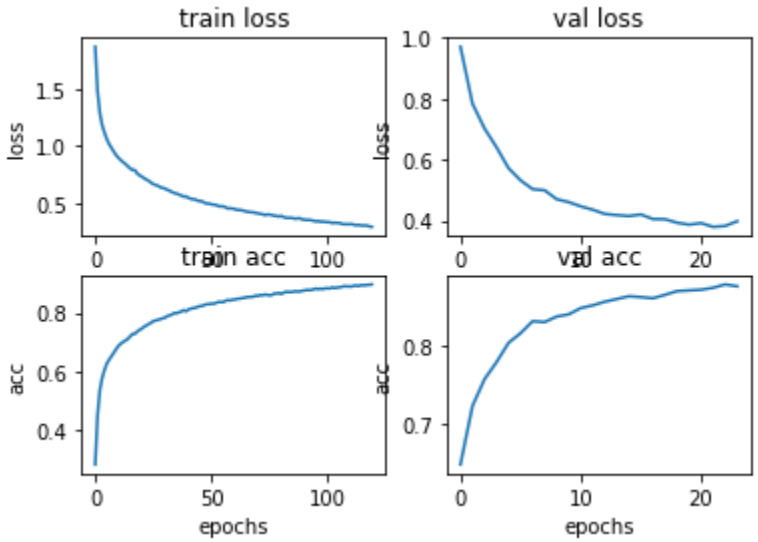
- **Part 5 (CNN with BN & DO)**

Training Script: <cifar10-cnn-bn-do.ipynb>

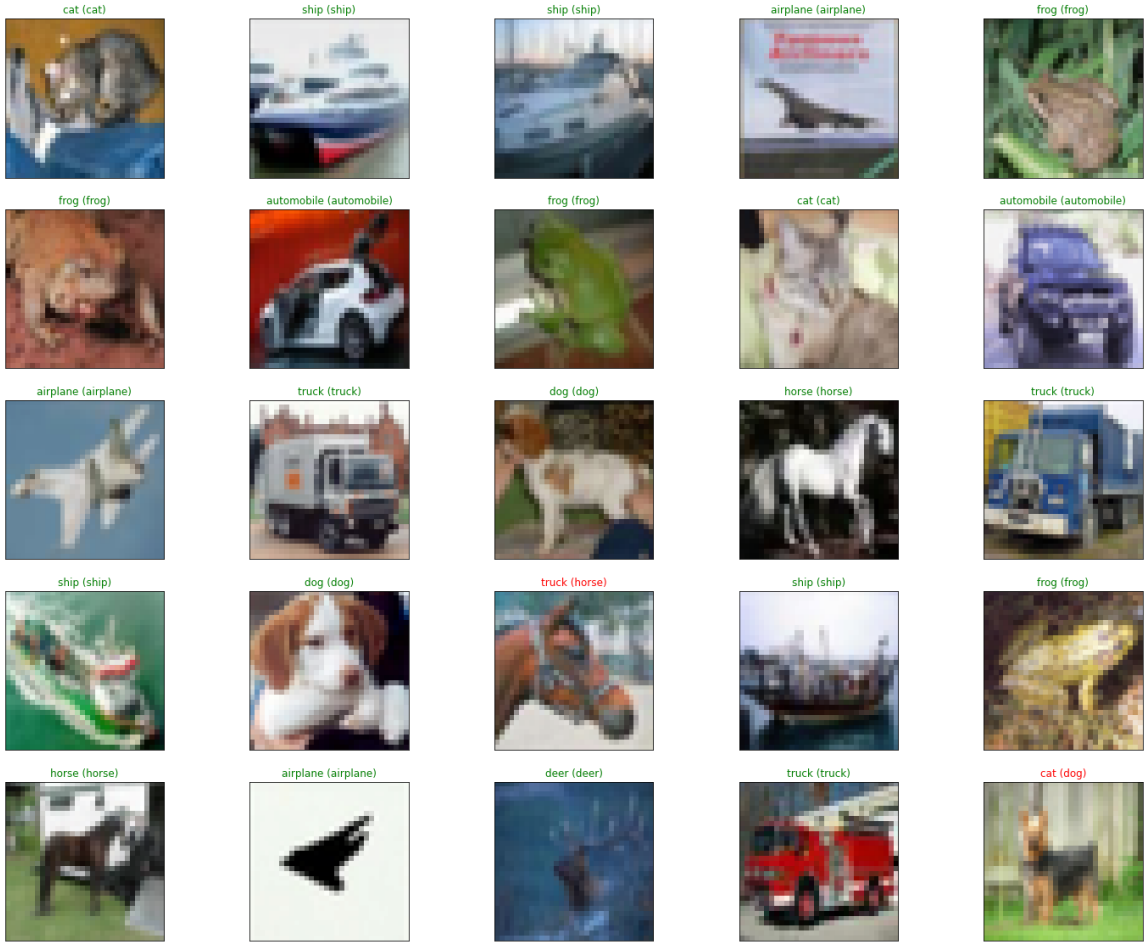
Result Script: <results-cnn-bn-do.ipynb>

Architecture:

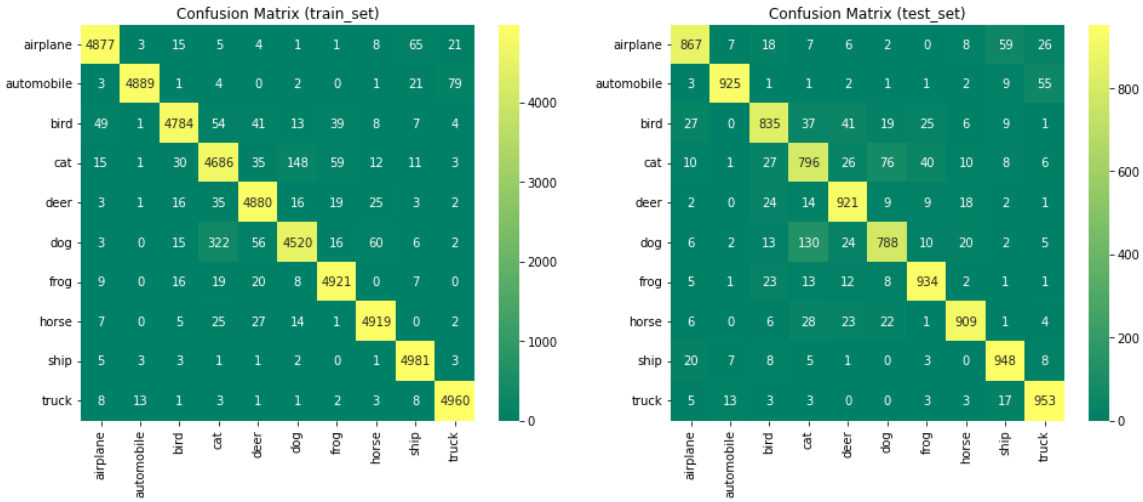

Training loss & accuracy:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(138)Train accuracy of airplane    :    97.54% (4877/5000)
(139)Train accuracy of automobile  :    97.78% (4889/5000)
(140)Train accuracy of bird        :    95.68% (4784/5000)
(141)Train accuracy of cat         :    93.72% (4686/5000)
(142)Train accuracy of deer        :    97.60% (4880/5000)
(143)Train accuracy of dog         :    90.40% (4520/5000)
(144)Train accuracy of frog        :    98.42% (4921/5000)
(145)Train accuracy of horse       :    98.38% (4919/5000)
(146)Train accuracy of ship        :    99.62% (4981/5000)
(147)Train accuracy of truck       :    99.20% (4960/5000)
```

Per Class Test Accuracy:

```
(148)Test accuracy of airplane    :86.70% (867/1000)
(149)Test accuracy of automobile  :92.50% (925/1000)
(150)Test accuracy of bird        :83.50% (835/1000)
(151)Test accuracy of cat         :79.60% (796/1000)
(152)Test accuracy of deer        :92.10% (921/1000)
(153)Test accuracy of dog         :78.80% (788/1000)
(154)Test accuracy of frog        :93.40% (934/1000)
(155)Test accuracy of horse       :90.90% (909/1000)
(156)Test accuracy of ship        :94.80% (948/1000)
(157)Test accuracy of truck       :95.30% (953/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(158)[Train]:
(159)              precision    recall  f1-score   support
(160)
(161)           0       0.98      0.98      0.98      5000
(162)           1       1.00      0.98      0.99      5000
(163)           2       0.98      0.96      0.97      5000
(164)           3       0.91      0.94      0.92      5000
(165)           4       0.96      0.98      0.97      5000
(166)           5       0.96      0.90      0.93      5000
(167)           6       0.97      0.98      0.98      5000
(168)           7       0.98      0.98      0.98      5000
(169)           8       0.97      1.00      0.99      5000
(170)           9       0.98      0.99      0.98      5000
(171)
(172)    accuracy                           0.97     50000
(173)   macro avg       0.97      0.97      0.97     50000
(174)weighted avg       0.97      0.97      0.97     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

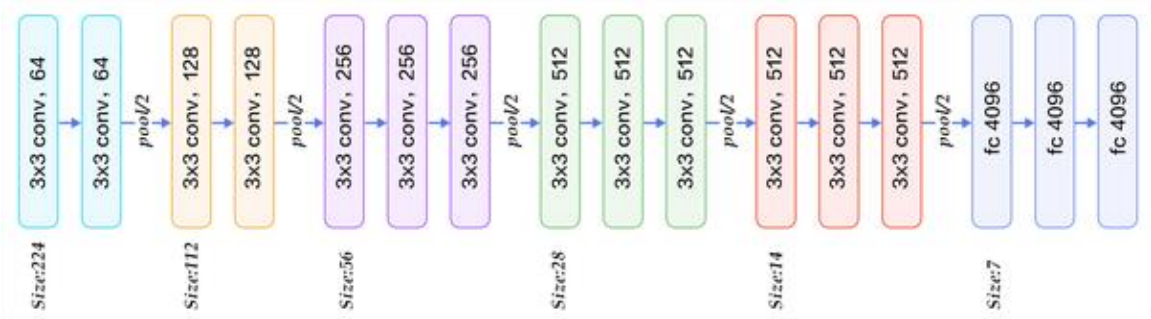
```
(175)[Test]:
(176)              precision    recall  f1-score   support
(177)
(178)           0       0.91      0.87      0.89      1000
(179)           1       0.97      0.93      0.95      1000
(180)           2       0.87      0.83      0.85      1000
(181)           3       0.77      0.80      0.78      1000
(182)           4       0.87      0.92      0.90      1000
(183)           5       0.85      0.79      0.82      1000
(184)           6       0.91      0.93      0.92      1000
(185)           7       0.93      0.91      0.92      1000
(186)           8       0.90      0.95      0.92      1000
(187)           9       0.90      0.95      0.93      1000
(188)
(189)    accuracy                           0.89     10000
(190)   macro avg       0.89      0.89      0.89     10000
(191)weighted avg       0.89      0.89      0.89     10000
```

- **Part 6.1 (VGG16 – Last 2 Layers)**

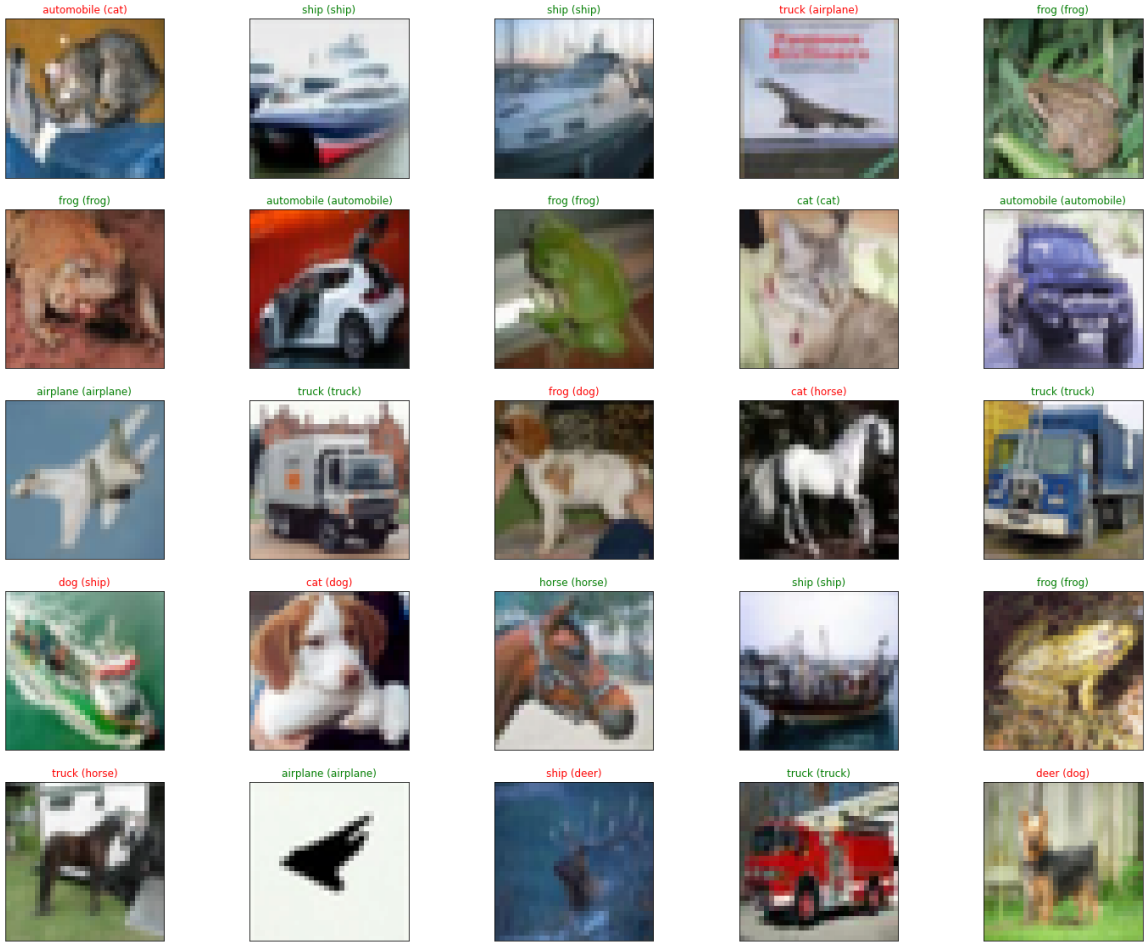
Training Script: <cifar10-vgg16-last2.ipynb>

Result Script: <results-vgg16-last2.ipynb>

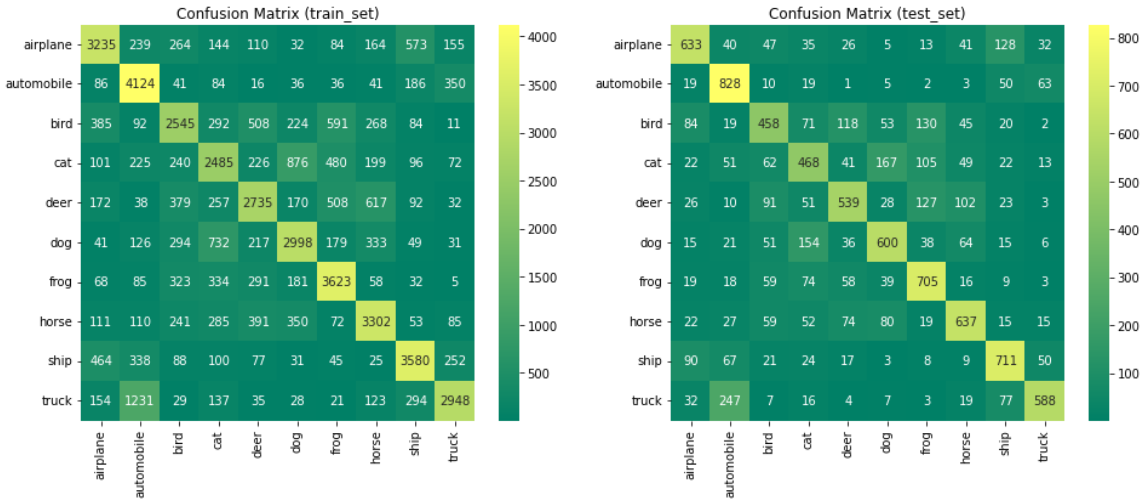
Architecture:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(192)Train accuracy of airplane    :    64.70% (3235/5000)
(193)Train accuracy of automobile  :    82.48% (4124/5000)
(194)Train accuracy of bird        :    50.90% (2545/5000)
(195)Train accuracy of cat         :    49.70% (2485/5000)
(196)Train accuracy of deer        :    54.70% (2735/5000)
(197)Train accuracy of dog         :    59.96% (2998/5000)
(198)Train accuracy of frog        :    72.46% (3623/5000)
(199)Train accuracy of horse       :    66.04% (3302/5000)
(200)Train accuracy of ship        :    71.60% (3580/5000)
(201)Train accuracy of truck       :    58.96% (2948/5000)
```

Per Class Test Accuracy:

```
(202)Test accuracy of airplane    :63.30% (633/1000)
(203)Test accuracy of automobile  :82.80% (828/1000)
(204)Test accuracy of bird        :45.80% (458/1000)
(205)Test accuracy of cat         :46.80% (468/1000)
(206)Test accuracy of deer        :53.90% (539/1000)
(207)Test accuracy of dog         :60.00% (600/1000)
(208)Test accuracy of frog        :70.50% (705/1000)
(209)Test accuracy of horse       :63.70% (637/1000)
(210)Test accuracy of ship        :71.10% (711/1000)
(211)Test accuracy of truck       :58.80% (588/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(212)[Train]:
(213)              precision    recall  f1-score   support
(214)
(215)           0       0.67      0.65      0.66      5000
(216)           1       0.62      0.82      0.71      5000
(217)           2       0.57      0.51      0.54      5000
(218)           3       0.51      0.50      0.50      5000
(219)           4       0.59      0.55      0.57      5000
(220)           5       0.61      0.60      0.60      5000
(221)           6       0.64      0.72      0.68      5000
(222)           7       0.64      0.66      0.65      5000
(223)           8       0.71      0.72      0.71      5000
(224)           9       0.75      0.59      0.66      5000
(225)
(226)    accuracy                           0.63     50000
(227)   macro avg       0.63      0.63      0.63     50000
(228)weighted avg       0.63      0.63      0.63     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

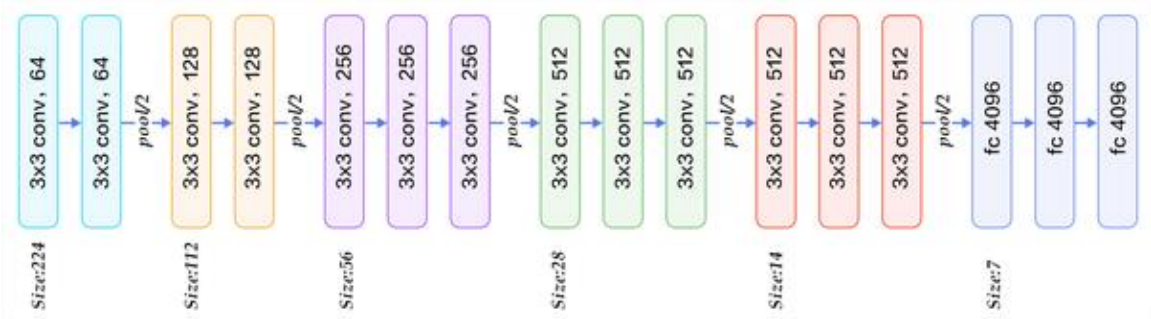
```
(229)[Test]:
(230)              precision    recall  f1-score   support
(231)
(232)           0       0.66      0.63      0.65      1000
(233)           1       0.62      0.83      0.71      1000
(234)           2       0.53      0.46      0.49      1000
(235)           3       0.49      0.47      0.48      1000
(236)           4       0.59      0.54      0.56      1000
(237)           5       0.61      0.60      0.60      1000
(238)           6       0.61      0.70      0.66      1000
(239)           7       0.65      0.64      0.64      1000
(240)           8       0.66      0.71      0.69      1000
(241)           9       0.76      0.59      0.66      1000
(242)
(243)    accuracy                           0.62     10000
(244)   macro avg       0.62      0.62      0.61     10000
(245)weighted avg       0.62      0.62      0.61     10000
```

- **Part 6.2 (VGG16 – Last 3 Layers)**

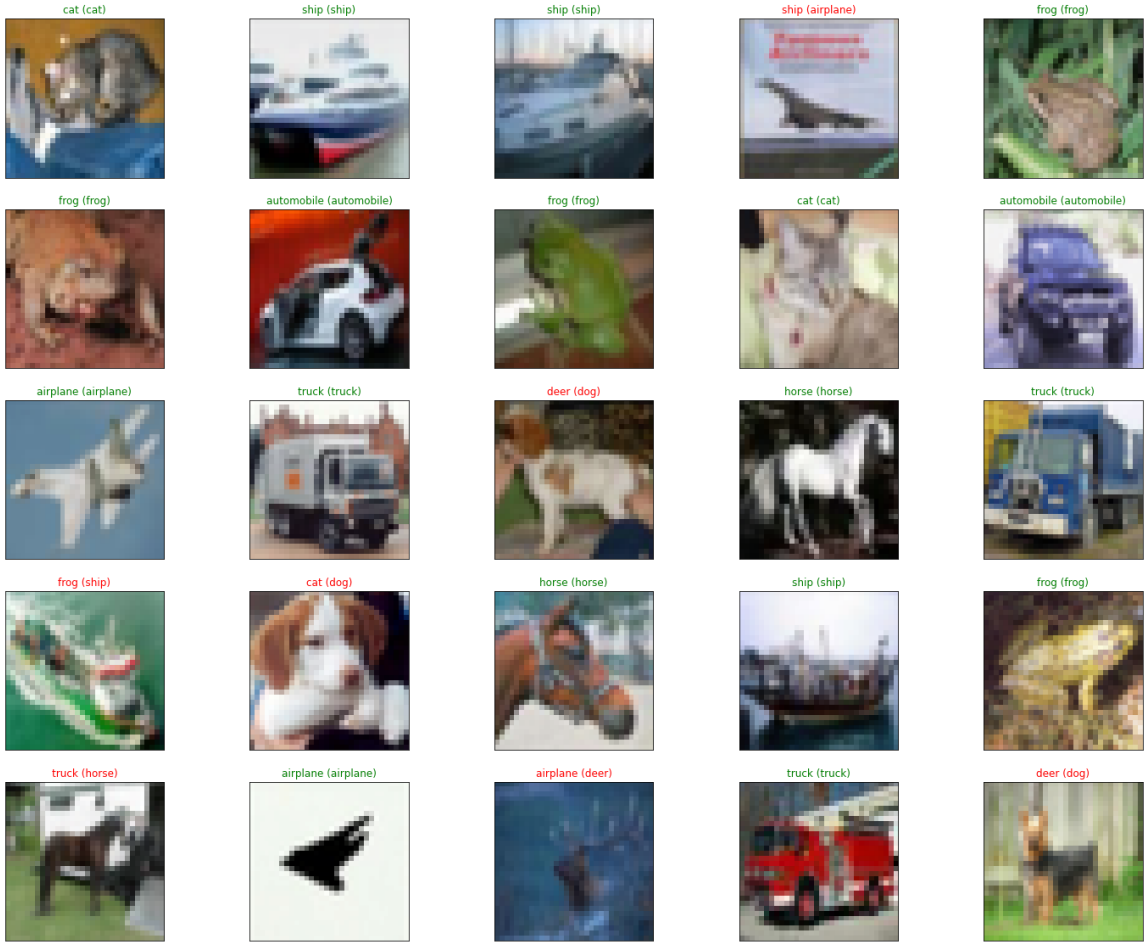
Training Script: <cifar10-vgg16-last3.ipynb>

Result Script: <results-vgg16-last3.ipynb>

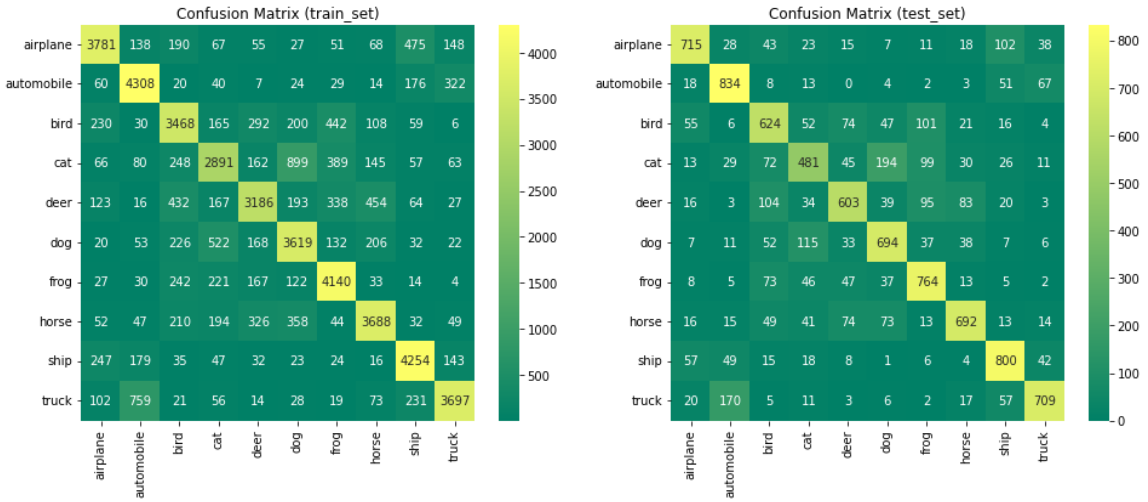
Architecture:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(246)Train accuracy of airplane    :    75.62% (3781/5000)
(247)Train accuracy of automobile  :    86.16% (4308/5000)
(248)Train accuracy of bird        :    69.36% (3468/5000)
(249)Train accuracy of cat         :    57.82% (2891/5000)
(250)Train accuracy of deer        :    63.72% (3186/5000)
(251)Train accuracy of dog         :    72.38% (3619/5000)
(252)Train accuracy of frog        :    82.80% (4140/5000)
(253)Train accuracy of horse       :    73.76% (3688/5000)
(254)Train accuracy of ship        :    85.08% (4254/5000)
(255)Train accuracy of truck       :    73.94% (3697/5000)
```

Per Class Test Accuracy:

```
(256)Test accuracy of airplane    :71.50% (715/1000)
(257)Test accuracy of automobile  :83.40% (834/1000)
(258)Test accuracy of bird        :62.40% (624/1000)
(259)Test accuracy of cat         :48.10% (481/1000)
(260)Test accuracy of deer        :60.30% (603/1000)
(261)Test accuracy of dog         :69.40% (694/1000)
(262)Test accuracy of frog        :76.40% (764/1000)
(263)Test accuracy of horse       :69.20% (692/1000)
(264)Test accuracy of ship        :80.00% (800/1000)
(265)Test accuracy of truck       :70.90% (709/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(266)[Train]:
(267)              precision    recall  f1-score   support
(268)
(269)           0       0.80      0.76      0.78      5000
(270)           1       0.76      0.86      0.81      5000
(271)           2       0.68      0.69      0.69      5000
(272)           3       0.66      0.58      0.62      5000
(273)           4       0.72      0.64      0.68      5000
(274)           5       0.66      0.72      0.69      5000
(275)           6       0.74      0.83      0.78      5000
(276)           7       0.77      0.74      0.75      5000
(277)           8       0.79      0.85      0.82      5000
(278)           9       0.83      0.74      0.78      5000
(279)
(280)    accuracy                           0.74     50000
(281)   macro avg       0.74      0.74      0.74     50000
(282)weighted avg       0.74      0.74      0.74     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

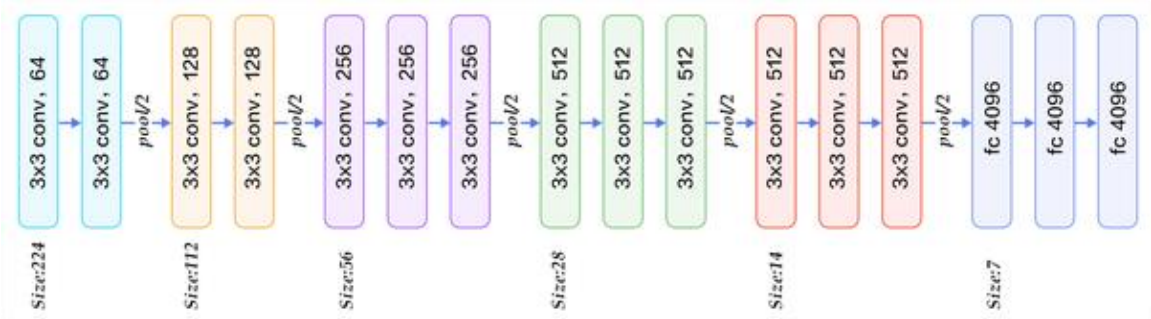
```
(283)[Test]:
(284)              precision    recall  f1-score   support
(285)
(286)           0       0.77      0.71      0.74      1000
(287)           1       0.73      0.83      0.78      1000
(288)           2       0.60      0.62      0.61      1000
(289)           3       0.58      0.48      0.52      1000
(290)           4       0.67      0.60      0.63      1000
(291)           5       0.63      0.69      0.66      1000
(292)           6       0.68      0.76      0.72      1000
(293)           7       0.75      0.69      0.72      1000
(294)           8       0.73      0.80      0.76      1000
(295)           9       0.79      0.71      0.75      1000
(296)
(297)    accuracy                           0.69     10000
(298)   macro avg       0.69      0.69      0.69     10000
(299)weighted avg       0.69      0.69      0.69     10000
```

- **Part 6.3 (VGG16 – Last 5 Layers)**

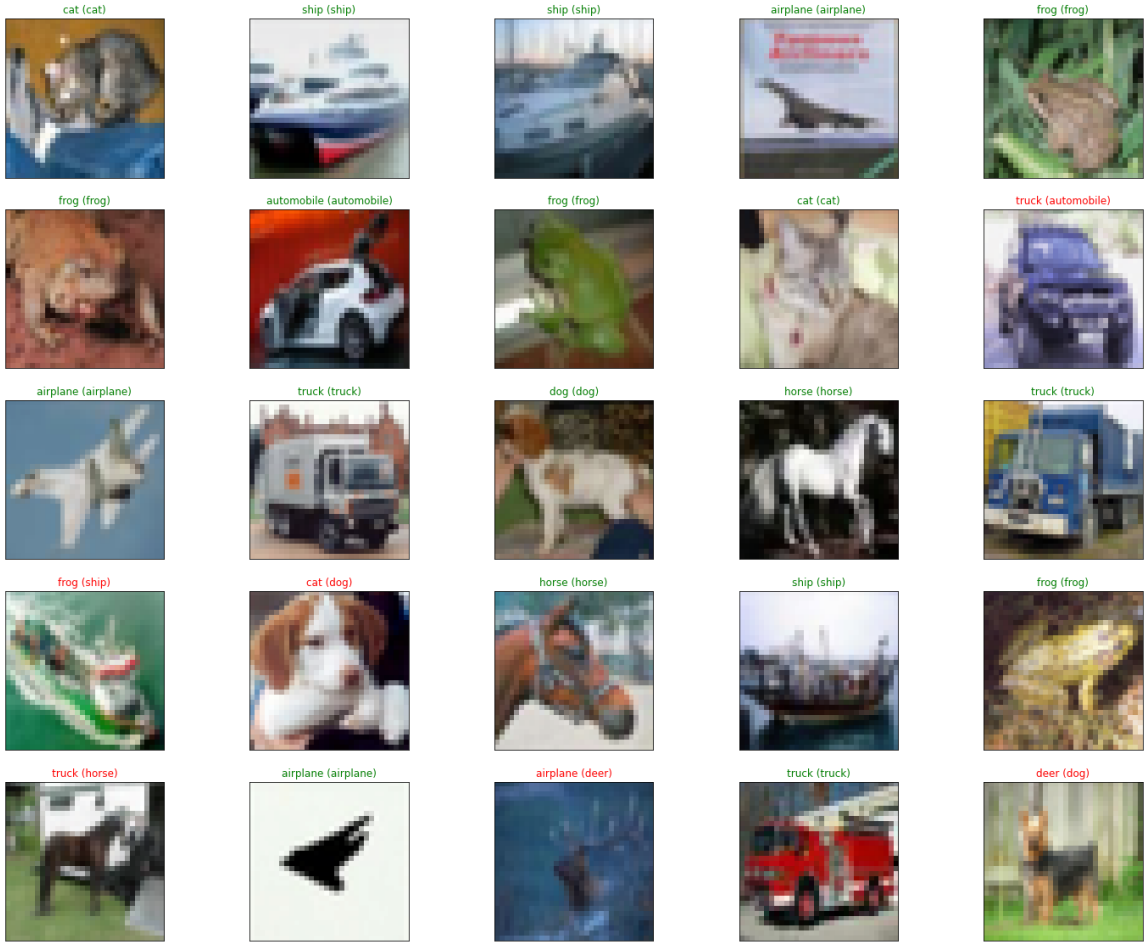
Training Script: <cifar10-vgg16-last5.ipynb>

Result Script: <results-vgg16-last5.ipynb>

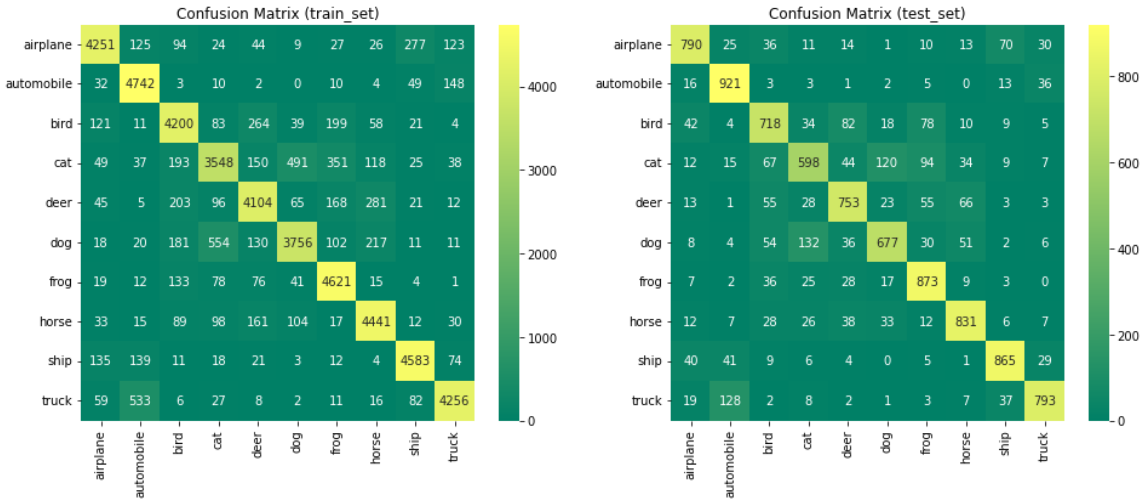
Architecture:



## Some Sample Predictions:




## Confusion Matrix:



Per Class Train Accuracy:

```
(300)Train accuracy of airplane    :    85.02% (4251/5000)
(301)Train accuracy of automobile  :    94.84% (4742/5000)
(302)Train accuracy of bird        :    84.00% (4200/5000)
(303)Train accuracy of cat         :    70.96% (3548/5000)
(304)Train accuracy of deer        :    82.08% (4104/5000)
(305)Train accuracy of dog         :    75.12% (3756/5000)
(306)Train accuracy of frog        :    92.42% (4621/5000)
(307)Train accuracy of horse       :    88.82% (4441/5000)
(308)Train accuracy of ship        :    91.66% (4583/5000)
(309)Train accuracy of truck       :    85.12% (4256/5000)
```

Per Class Test Accuracy:

```
(310)Test accuracy of airplane    :79.00% (790/1000)
(311)Test accuracy of automobile  :92.10% (921/1000)
(312)Test accuracy of bird        :71.80% (718/1000)
(313)Test accuracy of cat         :59.80% (598/1000)
(314)Test accuracy of deer        :75.30% (753/1000)
(315)Test accuracy of dog         :67.70% (677/1000)
(316)Test accuracy of frog        :87.30% (873/1000)
(317)Test accuracy of horse       :83.10% (831/1000)
(318)Test accuracy of ship        :86.50% (865/1000)
(319)Test accuracy of truck       :79.30% (793/1000)
```

Per Class Precision & Recall & F1-Score o Train set:

Overall Precision & Recall & F1-Score & Accuracy over Train set:

```
(320)[Train]:
(321)              precision    recall  f1-score   support
(322)
(323)           0       0.89      0.85      0.87      5000
(324)           1       0.84      0.95      0.89      5000
(325)           2       0.82      0.84      0.83      5000
(326)           3       0.78      0.71      0.74      5000
(327)           4       0.83      0.82      0.82      5000
(328)           5       0.83      0.75      0.79      5000
(329)           6       0.84      0.92      0.88      5000
(330)           7       0.86      0.89      0.87      5000
(331)           8       0.90      0.92      0.91      5000
(332)           9       0.91      0.85      0.88      5000
(333)
(334)    accuracy                           0.85     50000
(335)   macro avg       0.85      0.85      0.85     50000
(336)weighted avg       0.85      0.85      0.85     50000
```

Per Class Precision & Recall & F1-Score over Test set:

Overall Precision & Recall & F1-Score & Accuracy over Test set:

```
(337)[Test]:
(338)              precision    recall  f1-score   support
(339)
(340)           0       0.82      0.79      0.81      1000
(341)           1       0.80      0.92      0.86      1000
(342)           2       0.71      0.72      0.72      1000
(343)           3       0.69      0.60      0.64      1000
(344)           4       0.75      0.75      0.75      1000
(345)           5       0.76      0.68      0.72      1000
(346)           6       0.75      0.87      0.81      1000
(347)           7       0.81      0.83      0.82      1000
(348)           8       0.85      0.86      0.86      1000
(349)           9       0.87      0.79      0.83      1000
(350)
(351)    accuracy                           0.78     10000
(352)   macro avg       0.78      0.78      0.78     10000
(353)weighted avg       0.78      0.78      0.78     10000
```

- **Results & Comparisons**

| Model | Accuracy (Test) | Layers Count | Training Epochs |
|---|---|---|---|
| Random Classifier | 0.1 | 0 | 0 |
| MLP | 0.51 | 3 | 18 |
| CNN | 0.85 | 9 | 120 |
| CNN-BN-DO | **0.89** | 9 | 120 |
| VGG16-Last2 | 0.62 | 16 | 30 |
| VGG16-Last3 | 0.69 | 16 | 30 |
| VGG16-Last5 | 0.78 | 16 | 30 |

  - *Adding more batch normalization and dropout to the same model will result in test accuracy increment. These two layers helped model to reach better generalization over test data.*
  - *Training more VGG16 layers results in accuracy increment. Although increasing the epochs should help. (I had no more resource to train them until 120 epochs).*
  - *MLP network did worse than CNN networks but still works better than random classifier which is going to assign random labels from set of our 10 labels to input samples.*

### Advantages of transfer learning:

You can reach generalization faster by using previously trained embedders or just tweaking the last layers of embedder.

It needs less resource to train than training a model from scratch.

You can use knowledge from similar domains in your work.

### Dataset size and similarity for transfer learning:

| | **very similar dataset** | **very different dataset** |
|---|---|---|
| **very little data** | Finetune linear classifier on top layer | You're in trouble… Try data augmentation / collect more data |
| **quite a lot of data** | Finetune a few layers | Finetune a larger number of layers |