CIFAR Project (Deep Learning)

Name: Valentin Gundorov

Id:336490271

1. Problem Statement:

The project goal was defined: image classification into ten classes on the CIFAR-10 dataset.

2. Data Preparation:

Data was loaded and preprocessed.

```
Dataset CIFAR10
    Number of datapoints: 50000
    Root location: C:/Users/gundo/Data/Data
    Split: Train
    StandardTransform
Transform: ToTensor()

Dataset CIFAR10
    Number of datapoints: 10000
    Root location: C:/Users/gundo/Data/Data
    Split: Test
    StandardTransform
Transform: ToTensor()
```

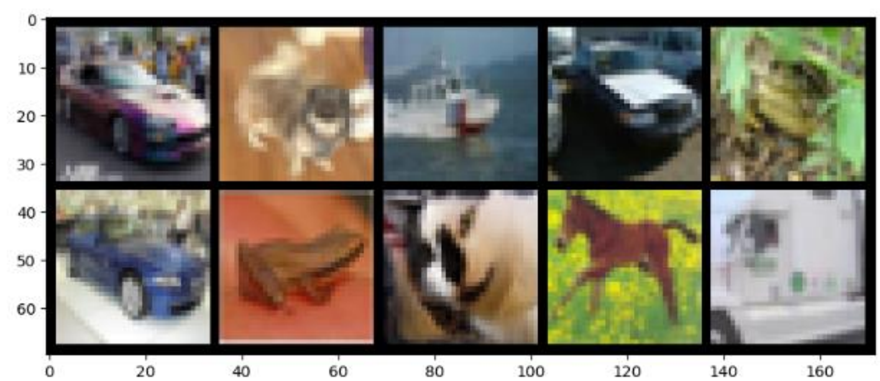
```
torch.manual_seed(101)  # for reproducible results

train_loader = DataLoader(train_data, batch_size=10, shuffle=True)
test_loader = DataLoader(test_data, batch_size=10, shuffle=False)
```

```
class_names = ['plane', '  car', ' bird', '  cat', ' deer', '  dog', ' frog', 'horse', ' ship', 'truck']
```

3. View an example of images from dataset.

```
Label: [    1     5     8     1     6     1     6     3     7     9]
Class:    car   dog  ship   car  frog   car  frog   cat horse truck
```



4. Model Architecture Definition:

A convolutional neural network (CNN) with a specific architecture was chosen and trained.

- Convolutional Layers.

```
self.conv1 = nn.Conv2d(3, 6, 3, 1)
```

  This layer performs convolution with a kernel size of 3x3 on input images. The input has 3 channels (RGB), and the output contains 6 feature maps.

```
self.conv2 = nn.Conv2d(6, 16, 3, 1)
```

  6 filters, 16 random number of filters ,3by3 image kernel

- Fully Connected Layers (Linear Layers).

```
self.fc1 = nn.Linear(576, 120)
```

  *6*6*16=576*

```
self.fc2 = nn.Linear(120,84)
self.fc3 = nn.Linear(84, 10)
```

- Forward Pass.

```
def forward(self, X):
    X = F.relu(self.conv1(X))
    X = F.max_pool2d(X, 2, 2)
    X = F.relu(self.conv2(X))
    X = F.max_pool2d(X, 2, 2)
    X = X.view(-1, 576)
    X = F.relu(self.fc1(X))
    X = F.relu(self.fc2(X))
    X = self.fc3(X)
    return F.log_softmax(X, dim=1)
```

5. Count Parameters:

- *Model Complexity Assessment: The number of parameters gives an idea of the model's complexity.*
- *Controlling Model Size: Knowing the number of parameters helps estimate the model's size and its memory requirements*
- *Task Complexity Assessment: The number of parameters in the model can also provide an idea of the task's complexity in machine learning.*

```
  162
    6
  864
   16
69120
  120
10080
   84
  840
   10
______
81302
```

6. Model Training:

The model was trained on the training set using the selected loss function and optimizer.

```
epoch:  0 loss:1.700 accuracy:  39.822%
epoch:  1 loss:1.392 accuracy:  52.402%
epoch:  2 loss:1.091 accuracy:  57.378%
epoch:  3 loss:1.239 accuracy:  60.048%
epoch:  4 loss:1.010 accuracy:  62.026%
epoch:  5 loss:0.991 accuracy:  63.894%
epoch:  6 loss:0.566 accuracy:  65.184%
epoch:  7 loss:1.105 accuracy:  66.344%
epoch:  8 loss:1.227 accuracy:  67.788%
epoch:  9 loss:0.876 accuracy:  68.754%
epoch: 10 loss:0.716 accuracy:  70.044%
epoch: 11 loss:0.819 accuracy:  70.646%
epoch: 12 loss:0.337 accuracy:  71.710%
epoch: 13 loss:0.782 accuracy:  72.704%
epoch: 14 loss:0.462 accuracy:  73.404%
epoch: 15 loss:0.826 accuracy:  74.006%
epoch: 16 loss:0.635 accuracy:  74.754%
epoch: 17 loss:0.367 accuracy:  75.586%
epoch: 18 loss:0.558 accuracy:  76.054%
epoch: 19 loss:0.855 accuracy:  76.850%
epoch: 20 loss:0.919 accuracy:  77.424%
epoch: 21 loss:0.266 accuracy:  77.786%
epoch: 22 loss:0.659 accuracy:  78.590%
epoch: 23 loss:1.612 accuracy:  78.676%
epoch: 24 loss:0.324 accuracy:  79.272%
epoch: 25 loss:1.147 accuracy:  79.728%
epoch: 26 loss:0.610 accuracy:  80.132%
epoch: 27 loss:0.435 accuracy:  80.568%
epoch: 28 loss:0.365 accuracy:  80.902%
epoch: 29 loss:0.316 accuracy:  81.526%
```

Overall Progress: The model shows improvement in both loss reduction and accuracy increase over time, which is a positive sign of learning.

Loss Instability: There is some instability in the losses, especially at epoch 23, where the losses increase. This could be related to changes in the training process or outliers in the data.

Steady Increase in Accuracy: The accuracy steadily increases, reaching 81.526% at epoch 29, indicating the effectiveness of the model's training.

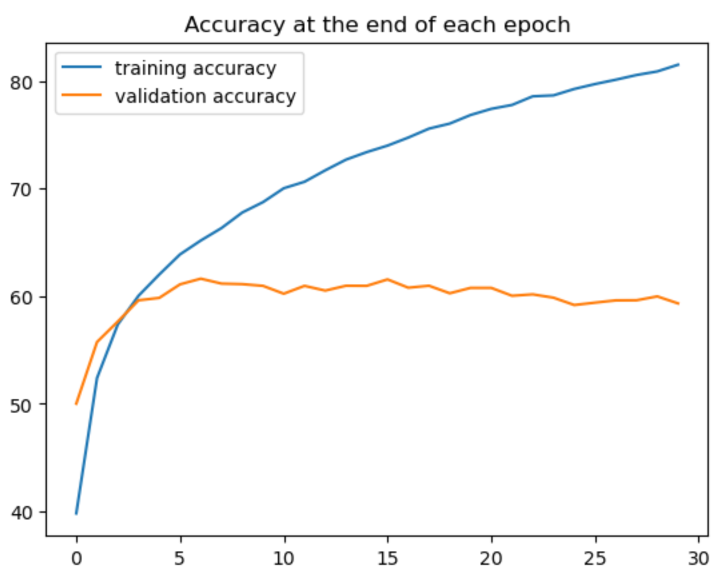
Potential Overfitting. There is some overfitting because of losses on the training and validation datasets.

7. Analysis of results

Reduction of Losses: Both the training set loss (train loss) and the validation set loss (validation loss) decrease over epochs, indicating that the model is learning and adapting to the data.

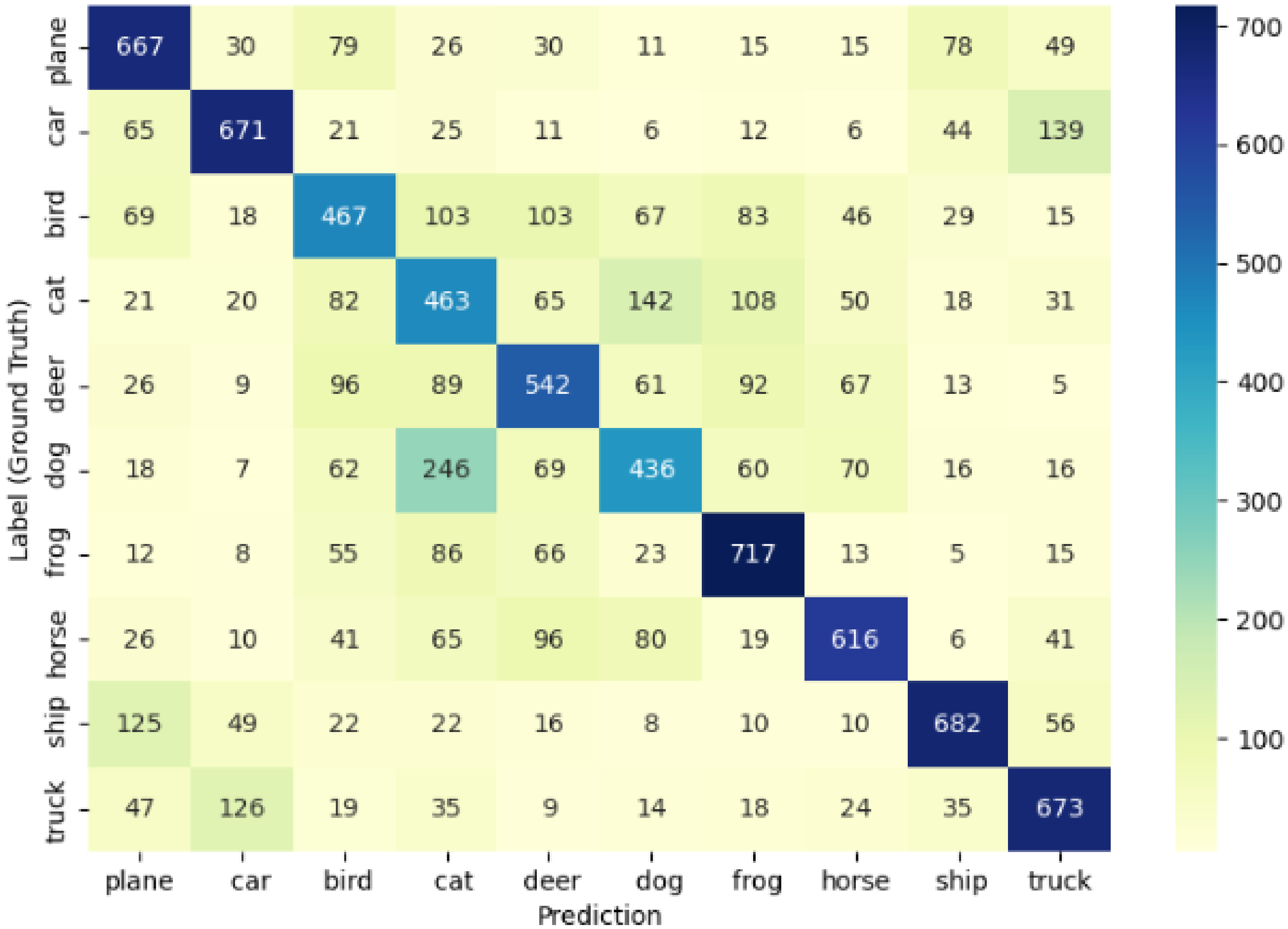
Fluctuations: There are fluctuations in the loss values, especially in the validation loss, which may indicate overfitting of the model.

Convergence: If the losses on the training and validation sets converge and stabilize, it's a good sign that the model is not overfitting. Gap Between Curves: significant and non-decreasing gap between the train loss and validation loss, it may indicate overfitting of the model.



Accuracy Dynamics: The graph shows that training accuracy continues to rise, while validation accuracy stabilizes after initial growth. This may indicate that the model is learning well from the data, but it could also be a sign of overfitting that we discussed earlier -> (6. Model Training)

8. Evaluate Test Data



Confusion Matrix: The presented confusion matrix visualizes the performance of a classification model in predicting various classes, such as airplane, car, bird, etc.

Accuracy and Misclassifications: The matrix helps understand the accuracy of the model and its misclassifications.

Model Analysis: Based on the matrix, one can assess which classes the model predicts well and which ones it struggles with.

```
Test accuracy: 59.340%
```

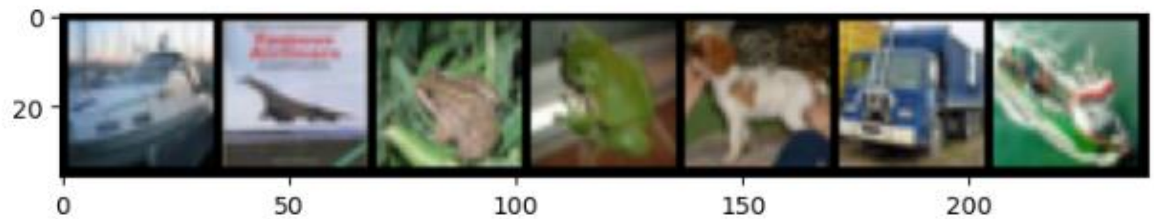
Definitely not the best result, unfortunately I couldn’t do better.

9. Examine the Misses:

```
Number of misses: 4066
```

```
Index: tensor([ 2,  3,  4,  7, 12, 14, 15])
Label: [    8     0     6     6     5     9     8]
Class:  ship plane  frog  frog   dog truck  ship

Guess: [    1     1     4     2     4     1     0]
Class:   car   car  deer  bird  deer   car plane
```



- *Types of Errors: The model tends to make more mistakes in classifying images representing airplanes and frogs. This may indicate the difficulty in distinguishing between these classes due to their similarity in appearance.*
- *Unpredictable Errors: Some images, for example, images with indices 2 and 12, were misclassified by the model even though they have relatively straightforward interpretations even for the human eye. This may indicate insufficient complexity of the model or issues in the training process, such as overfitting or underfitting.*
- *Incorrect Predictions: Some images, for example, images with indices 3 and 7, were incorrectly classified as airplanes and birds, respectively, even though they actually belonged to the same class - frogs. This may indicate imbalance in the training dataset or the difficulty of training the model for certain classes.*

By examining the misses, we delving into the nuanced intricacies of the model's performance. This process serves a twofold purpose: first, it offers valuable insights into the specific patterns or instances where the model struggles, potentially highlighting areas for refinement or augmentation. Second, it provides an opportunity for iterative learning, allowing for the refinement of the model through targeted adjustments or feature engineering based on the identified shortcomings. In essence, delving into the misses is a strategic endeavor aimed at enhancing the model's overall efficacy and ensuring its adaptability.

Conclusion:

We can observe that we have managed to train the machine using this dataset, although we haven't achieved high accuracy. The likelihood of the network recognizing an image is greater than the likelihood of making an error. I understand that there are optimization and improvement methods such as adding layers to the model, etc., and I hope to implement them in the future.

Thank you for your attention and goodbye!
Valentine.