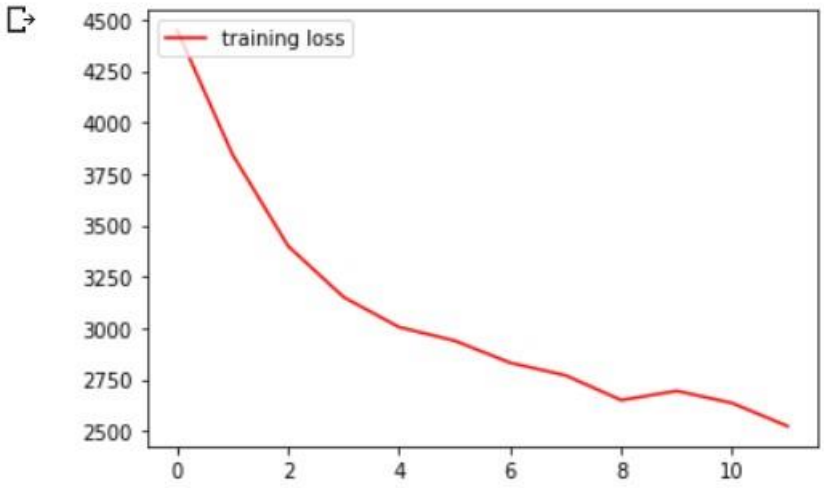
SIT789 - Applications of Computer Vision and Speech Processing

Credit Task 5.1: Deep learning for Computer Vision

```
[44] plt.plot(loss_history, label = 'training loss', color = 'r')
     plt.legend(loc = "upper left")
     plt.show()
```




```
[45] PATH = './cifar_net.pth'
     torch.save(net.state_dict(), PATH)
```

```
[46] dataiter = iter(testloader)
     images, labels = dataiter.next()
     # print images
     imshow(torchvision.utils.make_grid(images))
     print('GroundTruth: ', ' '.join('%5s' % classes[labels[j]] for j in range(4)))
```


```
GroundTruth:    cat  ship  ship plane
```

Time on local machine (with already using GPU on performance[gaming] laptop)

In [18]:
```
import time
start_time = time.time()
correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, groundtruth_labels = data
outputs = net(images)
_, predicted_labels = torch.max(outputs.data, 1)
total += groundtruth_labels.size(0)
correct += (predicted_labels == groundtruth_labels).sum().item()
print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / total))
print("Testing time GPU is in %s seconds ---" % (time.time() - start_time))
```

```
Accuracy of the network on the 10000 test images: 50 %
Testing time GPU is in 3.003812789916992 seconds ---
```

In [19]:
```
import time
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))
start_time = time.time()
with torch.no_grad():
    for data in testloader:
        images, groundtruth_labels = data
        outputs = net(images)
        _, predicted_labels = torch.max(outputs, 1)
        c = (predicted_labels == groundtruth_labels).squeeze()
        for i in range(4):
            label = groundtruth_labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1
for i in range(10):
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] / class_total[i]))

print("Testing time is in %s seconds ---" % (time.time() - start_time))
```

```
Accuracy of plane : 63 %
Accuracy of   car : 64 %
Accuracy of  bird : 49 %
Accuracy of   cat : 38 %
Accuracy of  deer : 43 %
Accuracy of   dog : 45 %
Accuracy of  frog : 70 %
Accuracy of horse : 55 %
Accuracy of  ship : 44 %
Accuracy of truck : 53 %
Testing time is in 5.318864107131958 seconds ---
```

In [25]:
```
import time
start_time = time.time()
correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, groundtruth_labels = data
outputs = net(images)
_, predicted_labels = torch.max(outputs.data, 1)
total += groundtruth_labels.size(0)
correct += (predicted_labels == groundtruth_labels).sum().item()
print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / total))
print("Testing time CPU is in %s seconds ---" % (time.time() - start_time))
```

```
Accuracy of the network on the 10000 test images: 50 %
Testing time CPU is in 2.874325752258301 seconds ---
```

In [26]:
```
import time
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))
start_time = time.time()
with torch.no_grad():
    for data in testloader:
        images, groundtruth_labels = data
        outputs = net(images)
        _, predicted_labels = torch.max(outputs, 1)
        c = (predicted_labels == groundtruth_labels).squeeze()
        for i in range(4):
            label = groundtruth_labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1
for i in range(10):
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] / class_total[i]))

print("Testing time is in %s seconds ---" % (time.time() - start_time))
```

```
Accuracy of plane : 63 %
Accuracy of   car : 64 %
Accuracy of  bird : 49 %
Accuracy of   cat : 38 %
Accuracy of  deer : 43 %
Accuracy of   dog : 45 %
Accuracy of  frog : 70 %
Accuracy of horse : 55 %
Accuracy of  ship : 44 %
Accuracy of truck : 53 %
Testing time is in 4.936636209487915 seconds ---
```

Time on Google Colab

```
t_time = time.time()

ect = 0
l = 0
 torch.no_grad():
for data in testloader:
    images, groundtruth_labels = data[0].to(device), data[1].to(device)
    outputs = net(images)
    _, predicted_labels = torch.max(outputs.data, 1)
    total += groundtruth_labels.size(0)
    correct += (predicted_labels == groundtruth_labels).sum().item()
t(device)
t('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / tota

t("Testing time is in %s seconds ---" % (time.time() - start_time))
```

```
cpu
Accuracy of the network on the 10000 test images: 54 %
Testing time is in 14.577147245407104 seconds ---
```

```
start_time = time.time()

correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, groundtruth_labels = data[0].to(device), data[1].to(device)
        outputs = net(images)
        _, predicted_labels = torch.max(outputs.data, 1)
        total += groundtruth_labels.size(0)
        correct += (predicted_labels == groundtruth_labels).sum().item()
print(device)
print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct /

print("Testing time is in %s seconds ---" % (time.time() - start_time))
```

```
cuda:0
Accuracy of the network on the 10000 test images: 55 %
Testing time is in 6.533771514892578 seconds ---
```

Confusion matrix:

```
In [21]: cm = confusion_matrix(groundtruth_labels, predicted_labels)
```

```
In [22]: print(cm)
```

```
[[0 0 1 0 0 0]
 [0 0 0 0 1 0]
 [0 0 0 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 1]]
```