Свёрточные нейронные сети: CIFAR10</h3>

Выполнила: Трофимова Екатерина Александровна, 20223

Теория CNN

В этом ноутбке мы посмотрим, насколько хорошо **CNN** будут предсказывать классы на более сложном датасете картинок **-- CIFAR10.**

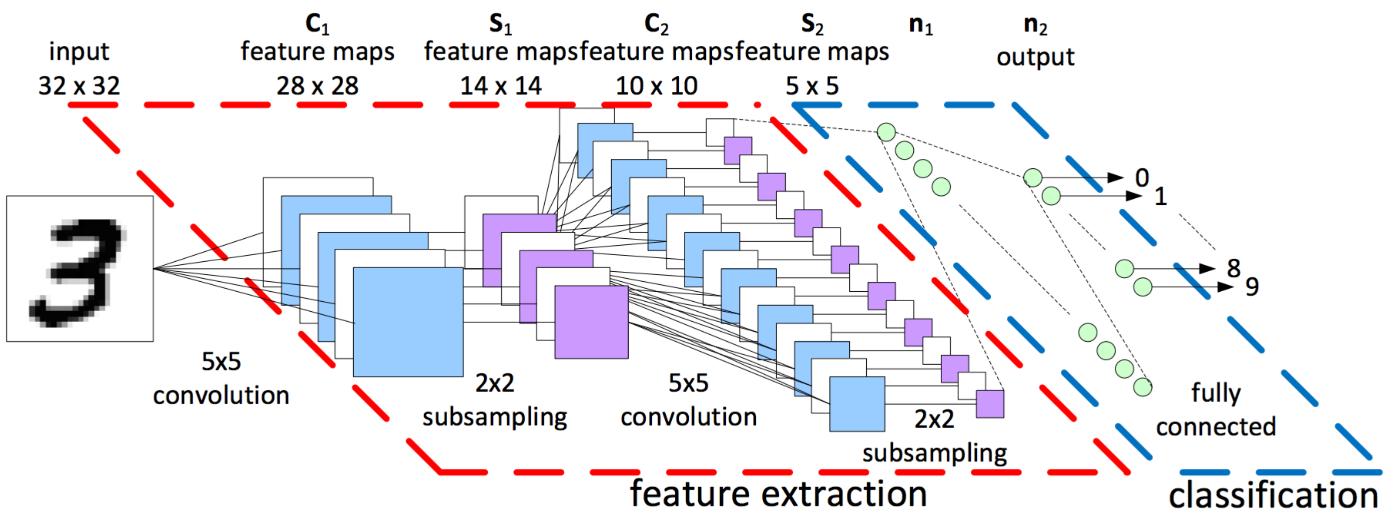
Внимание: Рассматривается ***задача классификации изображений*.**

***Свёрточная нейросеть (Convolutional Neural Network, CNN)* -** это многослойная нейросеть, имеющая в своей архитектуре помимо *полносвязных слоёв* (а иногда их может и не быть) ещё и **свёрточные слои (Conv Layers)** и **pooling-слои (Pool Layers).**

Собственно, название такое эти сети получили потому, что в основе их работы лежит операция **свёртки.**

Сразу же стоит сказать, что свёрточные нейросети **были придуманы прежде всего для задач, связанных с изображениями,** следовательно, на вход они тоже "ожидают" изображение.

- Например, вот так выглядит неглубокая свёрточная нейросеть, имеющая такую архитектуру: `Input -> Conv 5x5 -> Pool 2x2 -> Conv 5x5 -> Pool 2x2 -> FC -> Output`



Свёрточные нейросети (простые, есть и намного более продвинутые) почти всегда строятся по следующему правилу:

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*L -> FC
```

то есть:

1). ***Входной слой*: batch** картинок **--** тензор размера `(batch_size, H, W, C)` или `(batch_size, C, H, W)`

2). M блоков **(M** $\geq$ **0)** из свёрток и **pooling**-ов, причём именно в том порядке, как в формуле выше. Все эти M блоков вместе называют ***feature extractor*** свёрточной нейросети, потому что эта часть сети отвечает непосредственно за формирование новых, более сложных признаков поверх тех, которые подаются (то есть, по аналогии с **MLP**, мы опять же переходим к новому признаковому пространству, однако здесь оно строится аналогии с MLP, мы опять же переходим к новому признаковому пространству, однако здесь оно строится сложнее, чем в обычных многослойных сетях, поскольку используется операция свёртки)

3). L штук **FullyConnected**-слоёв (с активациями). Эту часть из L **FC**-слоёв называют ***classificator***, поскольку эти слои отвечают непосредственно за предсказание нужно класса (сейчас рассматривается задача классификации изображений).

Свёрточная нейросеть на **PyTorch**

Ещё раз напомним про основные компоненты нейросети:

- непосредственно, сама **архитектура** нейросети (сюда входят типы функций активации у каждого нейрона);
- начальная **инициализация** весов каждого слоя;
- метод **оптимизации** нейросети (сюда ещё входит метод изменения `learning_rate`);
- размер **батчей** (`batch_size`);
- количетсво **эпох** обучения (`num_epochs`);
- **функция потерь** (`loss`);
- тип **регуляризации** нейросети (`weight_decay`, для каждого слоя можно свой);

То, что связано с ***данными и задачей***:

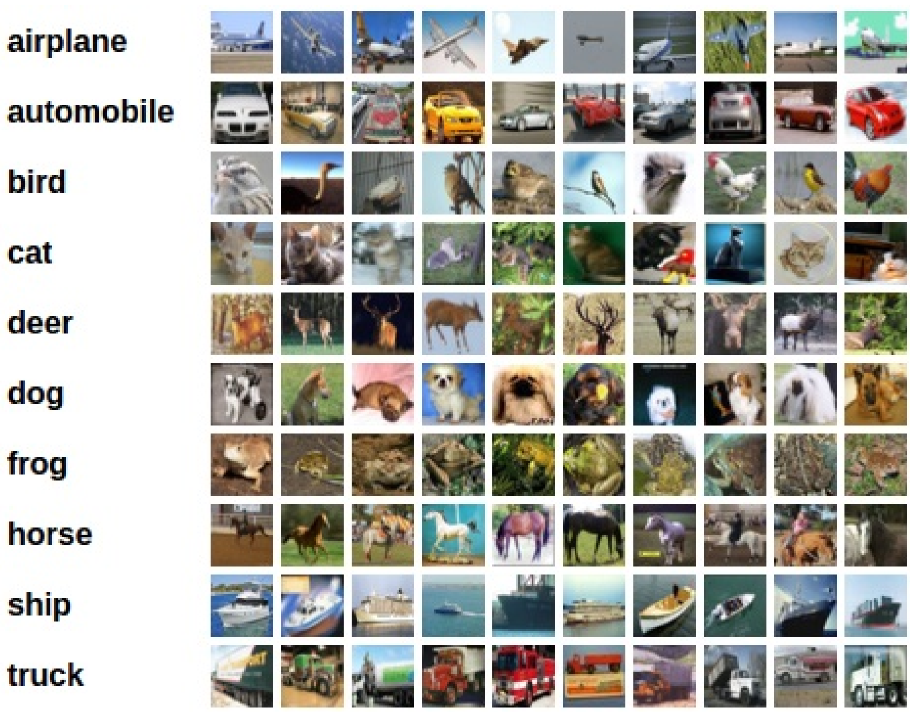
- само **качество** выборки (непротиворечивость, чистота, корректность постановки задачи);
- **размер** выборки;

Так как мы сейчас рассматриваем **архитектуру CNN**, то, помимо этих компонент, в свёрточной нейросети можно настроить следующие вещи:

- (в каждом **ConvLayer**) **размер фильтров (окна свёртки)** (`kernel_size`)
- (в каждом **ConvLayer**) **количество фильтров** (`out_channels`)
- (в каждом **ConvLayer**) размер **шага окна свёртки (stride)** (`stride`)
- (в каждом **ConvLayer**) **тип padding'а** (`padding`)

- (в каждом **PoolLayer**) размер окна **pooling'а** (`kernel_size`)
- (в каждом **PoolLayer**) шаг окна **pooling'а** (`stride`)
- (в каждом **PoolLayer**) тип **pooling'а** (`pool_type`)
- (в каждом **PoolLayer**) тип **padding'а** (`padding`)

CIFAR10



CIFAR10: это набор из **60k** картинок **32**x**32**x**3**, **50k** которых составляют обучающую выборку, и оставшиеся **10k** - тестовую. Классов в этом датасете **10:** `'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'`.

In [2]:

```
# !pip install torch torchvision
```

In [3]:

```
import torch
import torchvision
from torchvision import transforms
from tqdm import tqdm_notebook

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = torchvision.datasets.CIFAR10(root='../pytorch_data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4,
                                          shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='../pytorch_data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4,
                                         shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ../pytorch_data/ci
far-10-python.tar.gz


Extracting ../pytorch_data/cifar-10-python.tar.gz to ../pytorch_data
Files already downloaded and verified
```

In [ ]:

```
trainset.data
```

In [6]:

```
trainloader.dataset.train_list[0]
```

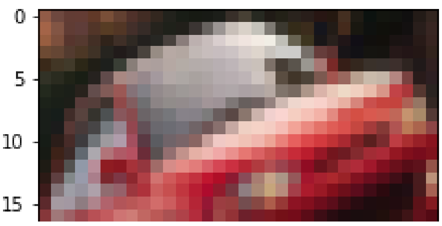
Out[6]:

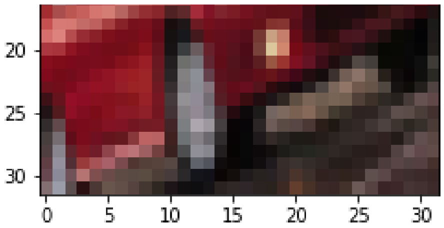
```
['data_batch_1', 'c99cafc152244af753f735de768cd75f']
```

In [7]:

```
# случайный индекс от 0 до размера тренировочной выборки
i = np.random.randint(low=0, high=50000)

plt.imshow(trainloader.dataset.data[i]);
```





CNN для предсказания на CIFAR10.

Напишем свёрточную нейросеть для предсказания на **CIFAR10**

In [8]:

```
# Подключение зависимостей

import torch.nn as nn
import torch.nn.functional as F
from tqdm import tqdm_notebook
from torch.optim import lr_scheduler
```

Вспомогательные функции

In [9]:

```
# Попытка ускорить вычисления за счет gpu

def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return torch.device('cpu')

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
```

In [10]:

```
device = get_default_device()
device
```

Out[10]:

```
device(type='cpu')
```

In [11]:

```
#trainloader = DeviceDataLoader(trainloader, device)
```

```
#testloader = DeviceDataLoader(testloader, device)
```

In [12]:

```
# Функция для обучения модели

def train(net, epoch_num = 5, learning_rate = 1e-3):

  loss_fn = torch.nn.CrossEntropyLoss()

  optimizer = torch.optim.Adam(net.parameters(), lr=learning_rate)
  # динамически изменяем LR
  scheduler = lr_scheduler.CosineAnnealingLR(optimizer, T_max=epoch_num)

  # итерируемся
  for epoch in tqdm_notebook(range(epoch_num)):

    scheduler.step()

    running_loss = 0.0
    for i, batch in enumerate(tqdm_notebook(trainloader)):
        # так получаем текущий батч
        X_batch, y_batch = batch

        # обнуляем веса
        optimizer.zero_grad()

        # forward + backward + optimize
        y_pred = net(X_batch)
        loss = loss_fn(y_pred, y_batch)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        # выводим качество каждые 2000 батчей
        if i % 2000 == 1999:
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

  print('Обучение закончено')
```

In [13]:

```
# Функция для проверки качества

def check_accuracy(net):
  class_correct = list(0. for i in range(10))
  class_total = list(0. for i in range(10))

  with torch.no_grad():
    for data in testloader:
        images, labels = data
        y_pred = net(images)
        _, predicted = torch.max(y_pred, 1)
        c = (predicted == labels).squeeze()
        for i in range(4):
            label = labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1

  avg_accuracy = 0

  for i in range(10):
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] / class_total
[i]))
    avg_accuracy += 100 * class_correct[i] / class_total[i]

  print('Avg accuracy %2d %%' % (avg_accuracy / 10))
```

In [14]:

```
# Функция для визуальной проверки результата

def visualize_result(net, index):
    image = testloader.dataset.data[index]
    plt.imshow(image)

    image = transform(image)  # не забудем отмасштабировать!

    y_pred = net(image.view(1, 3, 32, 32))
    _, predicted = torch.max(y_pred, 1)

    plt.title(f'Predicted: {classes[predicted.numpy()[0]]}')
```

Базовая архитектура

In [58]:

```
class SimpleConvNet(torch.nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(SimpleConvNet, self).__init__()
        # feature extractor
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5)
        # classificator
        self.fc1 = nn.Linear(5 * 5 * 16, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 5 * 5 * 16)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [59]:

```
net = SimpleConvNet()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.893
[1,  4000] loss: 1.628
[1,  6000] loss: 1.506
[1,  8000] loss: 1.471
[1, 10000] loss: 1.431
[1, 12000] loss: 1.359


[2,  2000] loss: 1.289
```

```
[2,  2000] loss: 1.289
[2,  4000] loss: 1.288
[2,  6000] loss: 1.268
[2,  8000] loss: 1.272
[2, 10000] loss: 1.248
[2, 12000] loss: 1.236

[3,  2000] loss: 1.152
[3,  4000] loss: 1.146
[3,  6000] loss: 1.132
[3,  8000] loss: 1.153
[3, 10000] loss: 1.139
[3, 12000] loss: 1.120

[4,  2000] loss: 1.031
[4,  4000] loss: 1.037
[4,  6000] loss: 1.026
[4,  8000] loss: 1.052
[4, 10000] loss: 1.049
[4, 12000] loss: 1.039

[5,  2000] loss: 0.947
[5,  4000] loss: 0.950
[5,  6000] loss: 0.949
[5,  8000] loss: 0.938
[5, 10000] loss: 0.947
[5, 12000] loss: 0.952

[6,  2000] loss: 0.837
[6,  4000] loss: 0.845
[6,  6000] loss: 0.862
[6,  8000] loss: 0.884
[6, 10000] loss: 0.867
[6, 12000] loss: 0.878

[7,  2000] loss: 0.798
[7,  4000] loss: 0.780
[7,  6000] loss: 0.796
[7,  8000] loss: 0.778
[7, 10000] loss: 0.776
[7, 12000] loss: 0.806

[8,  2000] loss: 0.717
[8,  4000] loss: 0.752
[8,  6000] loss: 0.720
[8,  8000] loss: 0.727
[8, 10000] loss: 0.732
[8, 12000] loss: 0.724

[9,  2000] loss: 0.671
[9,  4000] loss: 0.709
[9,  6000] loss: 0.685
[9,  8000] loss: 0.682
[9, 10000] loss: 0.712
[9, 12000] loss: 0.696

[10,  2000] loss: 0.699
[10,  4000] loss: 0.677
[10,  6000] loss: 0.690
[10,  8000] loss: 0.686
[10, 10000] loss: 0.661
[10, 12000] loss: 0.669
Обучение закончено
```

Посмотрим на **accuracy** на тестовом датасете:

In [60]:

```
check_accuracy(net)
```

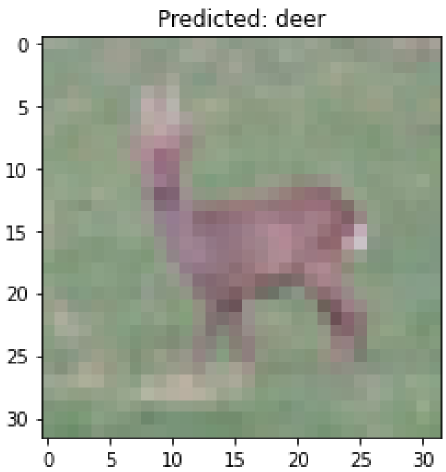
```
Accuracy of plane : 70 %
Accuracy of   car : 77 %
Accuracy of  bird : 50 %
Accuracy of   cat : 44 %
Accuracy of  deer : 58 %
Accuracy of   dog : 51 %
Accuracy of  frog : 74 %
Accuracy of horse : 71 %
Accuracy of  ship : 77 %
Accuracy of truck : 72 %
Avg accuracy 64 %
```

При базовой архитектуре наблюдается средняя точность в районе **64%** на **10** эпохах**.** Минимальная точность класса **44%.** Среднее время вычисления на эпоху **= 1:10**

Проверим работу нейросети визуально **(**позапускайте ячейку несколько раз**):**

In [37]:

```
i = np.random.randint(low=0, high=10000)
visualize_result(net, i)
```



Эксперименты с числом сверточных слоев и каналов

Попробуем просто добавить новый сверточный слой

In [68]:

```
class ConvNet_3CL(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3CL, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5)
        self.conv3 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=5)

        self.fc1 = nn.Linear(3 * 3 * 32, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = self.pool(x)
```

```
        x = x.view(-1, 3 * 3 * 32)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [69]:

```
net = ConvNet_3CL()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 2.017
[1,  4000] loss: 1.828
[1,  6000] loss: 1.682
[1,  8000] loss: 1.572
[1, 10000] loss: 1.531
[1, 12000] loss: 1.507

[2,  2000] loss: 1.458
[2,  4000] loss: 1.427
[2,  6000] loss: 1.412
[2,  8000] loss: 1.393
[2, 10000] loss: 1.382
[2, 12000] loss: 1.362

[3,  2000] loss: 1.302
[3,  4000] loss: 1.314
[3,  6000] loss: 1.281
[3,  8000] loss: 1.275
[3, 10000] loss: 1.275
[3, 12000] loss: 1.268

[4,  2000] loss: 1.191
[4,  4000] loss: 1.183
[4,  6000] loss: 1.194
[4,  8000] loss: 1.202
[4, 10000] loss: 1.178
[4, 12000] loss: 1.187

[5,  2000] loss: 1.083
[5,  4000] loss: 1.109
[5,  6000] loss: 1.101
[5,  8000] loss: 1.110
[5, 10000] loss: 1.115
[5, 12000] loss: 1.095

[6,  2000] loss: 1.014
[6,  4000] loss: 1.024
[6,  6000] loss: 1.003
[6,  8000] loss: 1.015
[6, 10000] loss: 1.026
[6, 12000] loss: 1.029
```

```
[7,  2000] loss: 0.938
[7,  4000] loss: 0.932
[7,  6000] loss: 0.966
[7,  8000] loss: 0.935
[7, 10000] loss: 0.929
[7, 12000] loss: 0.946


[8,  2000] loss: 0.861
[8,  4000] loss: 0.875
[8,  6000] loss: 0.898
[8,  8000] loss: 0.867
[8, 10000] loss: 0.883
[8, 12000] loss: 0.875


[9,  2000] loss: 0.829
[9,  4000] loss: 0.839
[9,  6000] loss: 0.834
[9,  8000] loss: 0.828
[9, 10000] loss: 0.844
[9, 12000] loss: 0.833


[10,  2000] loss: 0.800
[10,  4000] loss: 0.829
[10,  6000] loss: 0.820
[10,  8000] loss: 0.806
[10, 10000] loss: 0.832
[10, 12000] loss: 0.839
Обучение закончено
```

In [71]:

```
check_accuracy(net)
```

```
Accuracy of plane : 64 %
Accuracy of   car : 75 %
Accuracy of  bird : 44 %
Accuracy of   cat : 44 %
Accuracy of  deer : 54 %
Accuracy of   dog : 46 %
Accuracy of  frog : 72 %
Accuracy of horse : 66 %
Accuracy of  ship : 75 %
Accuracy of truck : 70 %
Avg accuracy 61 %
```

Добавление еще одного сверточного слоя с малым количеством каналов отрицательно сказалось на качестве обучения (средний результат ухудшился до **61%)** и времени обучения**.** Примерно на **9** эпохе процесс обучения застопорился**.** Попробуем теперь вернуться к **2**м сверточным слоям**,** но увеличим число каналов

In [80]:

```
class ConvNet_2Cl(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_2Cl, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=5)

        self.fc1 = nn.Linear(5 * 5 * 128, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
```

```
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 5 * 5 * 128)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [81]:

```
net = ConvNet_2Cl()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.861
[1,  4000] loss: 1.567
[1,  6000] loss: 1.458
[1,  8000] loss: 1.388
[1, 10000] loss: 1.348
[1, 12000] loss: 1.317

[2,  2000] loss: 1.222
[2,  4000] loss: 1.187
[2,  6000] loss: 1.164
[2,  8000] loss: 1.159
[2, 10000] loss: 1.151
[2, 12000] loss: 1.104

[3,  2000] loss: 1.024
[3,  4000] loss: 1.002
[3,  6000] loss: 1.007
[3,  8000] loss: 0.995
[3, 10000] loss: 1.008
[3, 12000] loss: 1.017

[4,  2000] loss: 0.864
[4,  4000] loss: 0.889
[4,  6000] loss: 0.897
[4,  8000] loss: 0.899
[4, 10000] loss: 0.908
[4, 12000] loss: 0.894

[5,  2000] loss: 0.777
[5,  4000] loss: 0.773
[5,  6000] loss: 0.783
[5,  8000] loss: 0.784
[5, 10000] loss: 0.789
[5, 12000] loss: 0.785

[6,  2000] loss: 0.658
[6,  4000] loss: 0.674
[6,  6000] loss: 0.672
[6,  8000] loss: 0.669
[6, 10000] loss: 0.690
```

```
[6, 12000] loss: 0.668

[7,  2000] loss: 0.566
[7,  4000] loss: 0.575
[7,  6000] loss: 0.579
[7,  8000] loss: 0.581
[7, 10000] loss: 0.569
[7, 12000] loss: 0.551

[8,  2000] loss: 0.475
[8,  4000] loss: 0.489
[8,  6000] loss: 0.466
[8,  8000] loss: 0.501
[8, 10000] loss: 0.490
[8, 12000] loss: 0.495

[9,  2000] loss: 0.431
[9,  4000] loss: 0.418
[9,  6000] loss: 0.420
[9,  8000] loss: 0.431
[9, 10000] loss: 0.427
[9, 12000] loss: 0.436

[10,  2000] loss: 0.400
[10,  4000] loss: 0.420
[10,  6000] loss: 0.408
[10,  8000] loss: 0.403
[10, 10000] loss: 0.405
[10, 12000] loss: 0.416
Обучение закончено
```

In [82]:

```
check_accuracy(net)
```

```
Accuracy of plane : 76 %
Accuracy of   car : 83 %
Accuracy of  bird : 59 %
Accuracy of   cat : 51 %
Accuracy of  deer : 64 %
Accuracy of   dog : 61 %
Accuracy of  frog : 81 %
Accuracy of horse : 75 %
Accuracy of  ship : 82 %
Accuracy of truck : 82 %
Avg accuracy 71 %
```

Увеличение числа каналов положительно сказалось на средней точности **- 71%** против базовых **64%.** Но обучение в рамках эпохи теперь идет гораздо дольше**.** Попробуем одновременно увеличить число сверточных слоев и число каналов

In [24]:

```
class ConvNet_3CL_CH(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3CL_CH, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=5)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=5)

        self.fc1 = nn.Linear(256, 120) # 1 x 1 x 256
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

```
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [25]:

```
net = ConvNet_3CL_CH()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.913
[1,  4000] loss: 1.647
[1,  6000] loss: 1.544
[1,  8000] loss: 1.488
[1, 10000] loss: 1.459
[1, 12000] loss: 1.390


[2,  2000] loss: 1.330
[2,  4000] loss: 1.299
[2,  6000] loss: 1.279
[2,  8000] loss: 1.257
[2, 10000] loss: 1.230
[2, 12000] loss: 1.227


[3,  2000] loss: 1.139
[3,  4000] loss: 1.139
[3,  6000] loss: 1.127
[3,  8000] loss: 1.101
[3, 10000] loss: 1.111
[3, 12000] loss: 1.075


[4,  2000] loss: 0.996
[4,  4000] loss: 1.012
[4,  6000] loss: 1.014
[4,  8000] loss: 0.997
[4, 10000] loss: 0.991
[4, 12000] loss: 0.973


[5,  2000] loss: 0.875
[5,  4000] loss: 0.875
[5,  6000] loss: 0.890
[5,  8000] loss: 0.896
[5, 10000] loss: 0.892
[5, 12000] loss: 0.902


[6,  2000] loss: 0.777
```

```
[6,  4000] loss: 0.783
[6,  6000] loss: 0.794
[6,  8000] loss: 0.776
[6, 10000] loss: 0.798
[6, 12000] loss: 0.769


[7,  2000] loss: 0.678
[7,  4000] loss: 0.694
[7,  6000] loss: 0.672
[7,  8000] loss: 0.679
[7, 10000] loss: 0.680
[7, 12000] loss: 0.703


[8,  2000] loss: 0.607
[8,  4000] loss: 0.589
[8,  6000] loss: 0.600
[8,  8000] loss: 0.596
[8, 10000] loss: 0.588
[8, 12000] loss: 0.591


[9,  2000] loss: 0.529
[9,  4000] loss: 0.520
[9,  6000] loss: 0.524
[9,  8000] loss: 0.544
[9, 10000] loss: 0.561
[9, 12000] loss: 0.528


[10,  2000] loss: 0.506
[10,  4000] loss: 0.518
[10,  6000] loss: 0.514
[10,  8000] loss: 0.518
[10, 10000] loss: 0.534
[10, 12000] loss: 0.513
Обучение закончено
```

In [26]:

```
check_accuracy(net)
```

```
Accuracy of plane : 75 %
Accuracy of   car : 81 %
Accuracy of  bird : 54 %
Accuracy of   cat : 49 %
Accuracy of  deer : 67 %
Accuracy of   dog : 59 %
Accuracy of  frog : 79 %
Accuracy of horse : 73 %
Accuracy of  ship : 81 %
Accuracy of truck : 78 %
Avg accuracy 70 %
```

Средняя точность стала чуть ниже **- 70%** против **71%** на двух слоях**.** При этом сильно возросло время обучения**.** Теперь попробуем изменить размер ядра свертки для случая с двумя слоями

In [17]:

```
class ConvNet_2Cl_3KS(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_2Cl_3KS, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)

        self.fc1 = nn.Linear(6 * 6 * 128, 120)
        self.fc2 = nn.Linear(120, 84)
```

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

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 6 * 6 * 128)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [18]:

```
net = ConvNet_2Cl_3KS()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.801
[1,  4000] loss: 1.463
[1,  6000] loss: 1.395
[1,  8000] loss: 1.290
[1, 10000] loss: 1.256
[1, 12000] loss: 1.211


[2,  2000] loss: 1.151
[2,  4000] loss: 1.105
[2,  6000] loss: 1.077
[2,  8000] loss: 1.052
[2, 10000] loss: 1.047
[2, 12000] loss: 1.039


[3,  2000] loss: 0.930
[3,  4000] loss: 0.924
[3,  6000] loss: 0.934
[3,  8000] loss: 0.945
[3, 10000] loss: 0.940
[3, 12000] loss: 0.934


[4,  2000] loss: 0.805
[4,  4000] loss: 0.819
[4,  6000] loss: 0.839
[4,  8000] loss: 0.839
[4, 10000] loss: 0.847
[4, 12000] loss: 0.812


[5,  2000] loss: 0.714
[5,  4000] loss: 0.723
[5,  6000] loss: 0.730
[5,  8000] loss: 0.736
[5, 10000] loss: 0.749
[5, 12000] loss: 0.738
```

```
[6,  2000] loss: 0.632
[6,  4000] loss: 0.627
[6,  6000] loss: 0.637
[6,  8000] loss: 0.647
[6, 10000] loss: 0.659
[6, 12000] loss: 0.649


[7,  2000] loss: 0.557
[7,  4000] loss: 0.558
[7,  6000] loss: 0.559
[7,  8000] loss: 0.555
[7, 10000] loss: 0.575
[7, 12000] loss: 0.564


[8,  2000] loss: 0.511
[8,  4000] loss: 0.479
[8,  6000] loss: 0.505
[8,  8000] loss: 0.505
[8, 10000] loss: 0.491
[8, 12000] loss: 0.496


[9,  2000] loss: 0.465
[9,  4000] loss: 0.453
[9,  6000] loss: 0.462
[9,  8000] loss: 0.467
[9, 10000] loss: 0.473
[9, 12000] loss: 0.448


[10,  2000] loss: 0.439
[10,  4000] loss: 0.446
[10,  6000] loss: 0.455
[10,  8000] loss: 0.442
[10, 10000] loss: 0.450
[10, 12000] loss: 0.457
Обучение закончено
```

In [19]:

```
check_accuracy(net)
```

```
Accuracy of plane : 74 %
Accuracy of   car : 83 %
Accuracy of  bird : 57 %
Accuracy of   cat : 52 %
Accuracy of  deer : 65 %
Accuracy of   dog : 63 %
Accuracy of  frog : 77 %
Accuracy of horse : 77 %
Accuracy of  ship : 81 %
Accuracy of truck : 81 %
Avg accuracy 71 %
```

Для двух слоев изменение размера ядра свертки не дало существенных изменений**.** Теперь посмотрим на **3**х слоях

In [27]:

```
class ConvNet_3CL_CH_3KS(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3CL_CH_3KS, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)
```

```
        self.fc1 = nn.Linear(2 * 2 * 256, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [28]:

```
net = ConvNet_3CL_CH_3KS()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.869
[1,  4000] loss: 1.537
[1,  6000] loss: 1.431
[1,  8000] loss: 1.371
[1, 10000] loss: 1.279
[1, 12000] loss: 1.251


[2,  2000] loss: 1.146
[2,  4000] loss: 1.095
[2,  6000] loss: 1.078
[2,  8000] loss: 1.065
[2, 10000] loss: 1.071
[2, 12000] loss: 1.057


[3,  2000] loss: 0.934
[3,  4000] loss: 0.945
[3,  6000] loss: 0.943
[3,  8000] loss: 0.919
[3, 10000] loss: 0.924
[3, 12000] loss: 0.914


[4,  2000] loss: 0.791
[4,  4000] loss: 0.812
[4,  6000] loss: 0.798
[4,  8000] loss: 0.824
[4, 10000] loss: 0.808
[4, 12000] loss: 0.813


[5,  2000] loss: 0.692
[5,  4000] loss: 0.697
[5,  6000] loss: 0.693
```

```
[5,  8000] loss: 0.722
[5, 10000] loss: 0.702
[5, 12000] loss: 0.689

[6,  2000] loss: 0.589
[6,  4000] loss: 0.567
[6,  6000] loss: 0.565
[6,  8000] loss: 0.579
[6, 10000] loss: 0.593
[6, 12000] loss: 0.611

[7,  2000] loss: 0.471
[7,  4000] loss: 0.479
[7,  6000] loss: 0.489
[7,  8000] loss: 0.469
[7, 10000] loss: 0.478
[7, 12000] loss: 0.490

[8,  2000] loss: 0.375
[8,  4000] loss: 0.401
[8,  6000] loss: 0.403
[8,  8000] loss: 0.379
[8, 10000] loss: 0.406
[8, 12000] loss: 0.399

[9,  2000] loss: 0.351
[9,  4000] loss: 0.339
[9,  6000] loss: 0.323
[9,  8000] loss: 0.336
[9, 10000] loss: 0.339
[9, 12000] loss: 0.343

[10,  2000] loss: 0.318
[10,  4000] loss: 0.321
[10,  6000] loss: 0.315
[10,  8000] loss: 0.324
[10, 10000] loss: 0.319
[10, 12000] loss: 0.321
Обучение закончено
```

In [29]:

```
check_accuracy(net)
```

```
Accuracy of plane : 79 %
Accuracy of   car : 85 %
Accuracy of  bird : 60 %
Accuracy of   cat : 54 %
Accuracy of  deer : 70 %
Accuracy of   dog : 61 %
Accuracy of  frog : 79 %
Accuracy of horse : 75 %
Accuracy of  ship : 82 %
Accuracy of truck : 84 %
Avg accuracy 73 %
```

А вот на **3**х слоях уже наблюдается небольшой прирост**: 73%** против **70%.** Минимальная точность возрослад о **54%.**

Добавление слоев и каналов позволило обогатить пространство признаков и улучшить тем самым результат классификации

Эксперименты с пулингом и нормализацией

Теперь попробуем поменять тип пулинга с **max** на **avg**

In [20]:

```
class ConvNet_3Cl_3KS_AvgPool(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3Cl_3KS_AvgPool, self).__init__()

        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)

        self.fc1 = nn.Linear(2 * 2 * 256, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [21]:

```
net = ConvNet_3Cl_3KS_AvgPool()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.924
[1,  4000] loss: 1.625
[1,  6000] loss: 1.478
[1,  8000] loss: 1.390
[1, 10000] loss: 1.328
[1, 12000] loss: 1.277


[2,  2000] loss: 1.203
[2,  4000] loss: 1.143
[2,  6000] loss: 1.126
[2,  8000] loss: 1.073
[2, 10000] loss: 1.062
[2, 12000] loss: 1.025


[3,  2000] loss: 0.942
[3,  4000] loss: 0.919
[3,  6000] loss: 0.919
[3,  8000] loss: 0.892
[3, 10000] loss: 0.890
[3, 12000] loss: 0.898
```

```
[4,  2000] loss: 0.772
[4,  4000] loss: 0.784
[4,  6000] loss: 0.778
[4,  8000] loss: 0.781
[4, 10000] loss: 0.763
[4, 12000] loss: 0.778

[5,  2000] loss: 0.659
[5,  4000] loss: 0.666
[5,  6000] loss: 0.687
[5,  8000] loss: 0.645
[5, 10000] loss: 0.661
[5, 12000] loss: 0.657

[6,  2000] loss: 0.552
[6,  4000] loss: 0.547
[6,  6000] loss: 0.556
[6,  8000] loss: 0.566
[6, 10000] loss: 0.556
[6, 12000] loss: 0.574

[7,  2000] loss: 0.466
[7,  4000] loss: 0.458
[7,  6000] loss: 0.473
[7,  8000] loss: 0.475
[7, 10000] loss: 0.464
[7, 12000] loss: 0.472

[8,  2000] loss: 0.398
[8,  4000] loss: 0.388
[8,  6000] loss: 0.401
[8,  8000] loss: 0.375
[8, 10000] loss: 0.390
[8, 12000] loss: 0.400

[9,  2000] loss: 0.347
[9,  4000] loss: 0.355
[9,  6000] loss: 0.355
[9,  8000] loss: 0.349
[9, 10000] loss: 0.340
[9, 12000] loss: 0.338

[10,  2000] loss: 0.332
[10,  4000] loss: 0.332
[10,  6000] loss: 0.332
[10,  8000] loss: 0.330
[10, 10000] loss: 0.340
[10, 12000] loss: 0.334
Обучение закончено
```

In [22]:

```
check_accuracy(net)
```

```
Accuracy of plane : 79 %
Accuracy of   car : 84 %
Accuracy of  bird : 64 %
Accuracy of   cat : 57 %
Accuracy of  deer : 74 %
Accuracy of   dog : 64 %
Accuracy of  frog : 82 %
Accuracy of horse : 78 %
Accuracy of  ship : 85 %
Accuracy of truck : 83 %
Avg accuracy 75 %
```

Смена типа пулинга с **max** на **avg** еще немного улучшила результат**:** средняя точность **75%** против **73%,** минимальная - **57%** против **54%.** Т.е. положение признака оказалось немного важнее его нличия**.** Попробуем добавить нормализацию

In [23]:

```
class ConvNet_3Cl_3KS_AvgPool_BN(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3Cl_3KS_AvgPool_BN, self).__init__()

        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.bn1 = nn.BatchNorm2d(64)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.bn2 = nn.BatchNorm2d(128)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)
        self.bn3 = nn.BatchNorm2d(256)

        self.fc1 = nn.Linear(2 * 2 * 256, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.bn1(F.relu(self.conv1(x)))
        x = self.pool(x)
        x = self.bn2(F.relu(self.conv2(x)))
        x = self.pool(x)
        x = self.bn3(F.relu(self.conv3(x)))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [24]:

```
net = ConvNet_3Cl_3KS_AvgPool_BN()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.862
[1,  4000] loss: 1.600
[1,  6000] loss: 1.479
[1,  8000] loss: 1.357
[1, 10000] loss: 1.289
[1, 12000] loss: 1.241


[2,  2000] loss: 1.110
[2,  4000] loss: 1.081
[2,  6000] loss: 1.094
[2,  8000] loss: 1.068
[2, 10000] loss: 1.066
[2, 12000] loss: 1.029
```

```
[3,  2000] loss: 0.943
[3,  4000] loss: 0.903
[3,  6000] loss: 0.941
[3,  8000] loss: 0.920
[3, 10000] loss: 0.896
[3, 12000] loss: 0.898

[4,  2000] loss: 0.770
[4,  4000] loss: 0.777
[4,  6000] loss: 0.787
[4,  8000] loss: 0.769
[4, 10000] loss: 0.794
[4, 12000] loss: 0.782

[5,  2000] loss: 0.660
[5,  4000] loss: 0.668
[5,  6000] loss: 0.651
[5,  8000] loss: 0.677
[5, 10000] loss: 0.651
[5, 12000] loss: 0.649

[6,  2000] loss: 0.547
[6,  4000] loss: 0.538
[6,  6000] loss: 0.551
[6,  8000] loss: 0.546
[6, 10000] loss: 0.535
[6, 12000] loss: 0.542

[7,  2000] loss: 0.451
[7,  4000] loss: 0.444
[7,  6000] loss: 0.451
[7,  8000] loss: 0.452
[7, 10000] loss: 0.446
[7, 12000] loss: 0.438

[8,  2000] loss: 0.377
[8,  4000] loss: 0.371
[8,  6000] loss: 0.381
[8,  8000] loss: 0.376
[8, 10000] loss: 0.374
[8, 12000] loss: 0.366

[9,  2000] loss: 0.321
[9,  4000] loss: 0.342
[9,  6000] loss: 0.330
[9,  8000] loss: 0.343
[9, 10000] loss: 0.323
[9, 12000] loss: 0.337

[10,  2000] loss: 0.322
[10,  4000] loss: 0.319
[10,  6000] loss: 0.319
[10,  8000] loss: 0.331
[10, 10000] loss: 0.325
[10, 12000] loss: 0.329
Обучение закончено
```

In [25]:

```
check_accuracy(net)
```

```
Accuracy of plane : 81 %
Accuracy of   car : 85 %
Accuracy of  bird : 65 %
Accuracy of   cat : 57 %
Accuracy of  deer : 73 %
Accuracy of   dog : 64 %
```

```
Accuracy of  frog : 81 %
Accuracy of horse : 81 %
Accuracy of  ship : 83 %
Accuracy of truck : 82 %
Avg accuracy 75 %
```

Нормализация не повлияла на результат

Эксперимент с функцией активации

Обычно в качестве функции активации сверточных слоев используют функцию **ReLU.** Рассматривать функции активации вроде сигмоидной или тангенциальной мы не будем**,** т.к**.** они приводят к проблемам с затуханием или увеличением градиентов**.** Вместо этого попробуем использовать **ELU,** которая сохраняет преимущества **ReLU** и помогает избежать проблемы умирающего **ReLU**

In [27]:

```
class ConvNet_3Cl_3KS_AvgPool_ELU(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3Cl_3KS_AvgPool_ELU, self).__init__()

        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)

        self.fc1 = nn.Linear(2 * 2 * 256, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.elu(self.conv1(x))
        x = self.pool(x)
        x = F.elu(self.conv2(x))
        x = self.pool(x)
        x = F.elu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.elu(self.fc1(x))
        x = F.elu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [30]:

```
net = ConvNet_3Cl_3KS_AvgPool_ELU()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.932
[1,  4000] loss: 1.686
```

```
[1,  4000] loss: 1.[illegible]
[1,  6000] loss: 1.577
[1,  8000] loss: 1.484
[1, 10000] loss: 1.433
[1, 12000] loss: 1.373

[2,  2000] loss: 1.255
[2,  4000] loss: 1.227
[2,  6000] loss: 1.192
[2,  8000] loss: 1.178
[2, 10000] loss: 1.161
[2, 12000] loss: 1.134

[3,  2000] loss: 0.965
[3,  4000] loss: 0.994
[3,  6000] loss: 0.978
[3,  8000] loss: 0.978
[3, 10000] loss: 0.967
[3, 12000] loss: 0.973

[4,  2000] loss: 0.803
[4,  4000] loss: 0.784
[4,  6000] loss: 0.800
[4,  8000] loss: 0.815
[4, 10000] loss: 0.808
[4, 12000] loss: 0.828

[5,  2000] loss: 0.639
[5,  4000] loss: 0.634
[5,  6000] loss: 0.651
[5,  8000] loss: 0.655
[5, 10000] loss: 0.642
[5, 12000] loss: 0.645

[6,  2000] loss: 0.480
[6,  4000] loss: 0.465
[6,  6000] loss: 0.501
[6,  8000] loss: 0.501
[6, 10000] loss: 0.508
[6, 12000] loss: 0.495

[7,  2000] loss: 0.340
[7,  4000] loss: 0.340
[7,  6000] loss: 0.340
[7,  8000] loss: 0.358
[7, 10000] loss: 0.359
[7, 12000] loss: 0.369

[8,  2000] loss: 0.235
[8,  4000] loss: 0.242
[8,  6000] loss: 0.251
[8,  8000] loss: 0.240
[8, 10000] loss: 0.253
[8, 12000] loss: 0.231

[9,  2000] loss: 0.190
[9,  4000] loss: 0.170
[9,  6000] loss: 0.178
[9,  8000] loss: 0.174
[9, 10000] loss: 0.180
[9, 12000] loss: 0.181

[10,  2000] loss: 0.163
[10,  4000] loss: 0.160
[10,  6000] loss: 0.145
[10,  8000] loss: 0.164
[10, 10000] loss: 0.156
[10, 12000] loss: 0.162
Обучение закончено
```

Обучение закончено

In [46]:

```
check_accuracy(net)
```

```
Accuracy of plane : 78 %
Accuracy of   car : 83 %
Accuracy of  bird : 63 %
Accuracy of   cat : 55 %
Accuracy of  deer : 66 %
Accuracy of   dog : 64 %
Accuracy of  frog : 78 %
Accuracy of horse : 74 %
Accuracy of  ship : 81 %
Accuracy of truck : 80 %
Avg accuracy 72 %
```

Несмотря на то, что значение функции потерь теперь меньше, точность тоже упала. **ELU** не дает нам выигрыша на текущей архитектуре

Эксперимент с числом полносвязных слоев

Начнем с простого - уберем **1** слой

In [14]:

```
class ConvNet_3Cl_3KS_AvgPool_2Fl(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3Cl_3KS_AvgPool_2Fl, self).__init__()

        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)

        self.fc1 = nn.Linear(2 * 2 * 256, 512)
        self.fc3 = nn.Linear(512, 10)

    def forward(self, x):
        x = F.elu(self.conv1(x))
        x = self.pool(x)
        x = F.elu(self.conv2(x))
        x = self.pool(x)
        x = F.elu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.elu(self.fc1(x))
        x = self.fc3(x)
        return x
```

In [15]:

```
net = ConvNet_3Cl_3KS_AvgPool_2Fl()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
```

```
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 2.004
[1,  4000] loss: 1.805
[1,  6000] loss: 1.700
[1,  8000] loss: 1.636
[1, 10000] loss: 1.590
[1, 12000] loss: 1.529

[2,  2000] loss: 1.410
[2,  4000] loss: 1.411
[2,  6000] loss: 1.376
[2,  8000] loss: 1.355
[2, 10000] loss: 1.306
[2, 12000] loss: 1.327

[3,  2000] loss: 1.140
[3,  4000] loss: 1.174
[3,  6000] loss: 1.159
[3,  8000] loss: 1.146
[3, 10000] loss: 1.167
[3, 12000] loss: 1.126

[4,  2000] loss: 0.886
[4,  4000] loss: 0.953
[4,  6000] loss: 0.900
[4,  8000] loss: 0.969
[4, 10000] loss: 0.936
[4, 12000] loss: 0.958

[5,  2000] loss: 0.654
[5,  4000] loss: 0.692
[5,  6000] loss: 0.686
[5,  8000] loss: 0.695
[5, 10000] loss: 0.737
[5, 12000] loss: 0.715

[6,  2000] loss: 0.439
[6,  4000] loss: 0.430
[6,  6000] loss: 0.449
[6,  8000] loss: 0.449
[6, 10000] loss: 0.460
[6, 12000] loss: 0.463

[7,  2000] loss: 0.232
[7,  4000] loss: 0.229
[7,  6000] loss: 0.235
[7,  8000] loss: 0.247
[7, 10000] loss: 0.247
[7, 12000] loss: 0.247

[8,  2000] loss: 0.109
[8,  4000] loss: 0.108
[8,  6000] loss: 0.104
[8,  8000] loss: 0.095
[8, 10000] loss: 0.104
[8, 12000] loss: 0.110

[9,  2000] loss: 0.050
[9,  4000] loss: 0.048
[9,  6000] loss: 0.049
[9,  8000] loss: 0.049
[9, 10000] loss: 0.050
[9, 12000] loss: 0.044
```

```
[10,  2000] loss: 0.036
[10,  4000] loss: 0.035
[10,  6000] loss: 0.035
[10,  8000] loss: 0.034
[10, 10000] loss: 0.029
[10, 12000] loss: 0.035
Обучение закончено
```

In [16]:

```
check_accuracy(net)
```

```
Accuracy of plane : 74 %
Accuracy of   car : 80 %
Accuracy of  bird : 62 %
Accuracy of   cat : 53 %
Accuracy of  deer : 66 %
Accuracy of   dog : 59 %
Accuracy of  frog : 76 %
Accuracy of horse : 76 %
Accuracy of  ship : 82 %
Accuracy of truck : 79 %
Avg accuracy 70 %
```

И наоборот - добавим 1 слой

In [17]:

```
class ConvNet_3Cl_3KS_AvgPool_4Fl(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(ConvNet_3Cl_3KS_AvgPool_4Fl, self).__init__()

        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3)

        self.fc1 = nn.Linear(2 * 2 * 256, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 10)

    def forward(self, x):
        x = F.elu(self.conv1(x))
        x = self.pool(x)
        x = F.elu(self.conv2(x))
        x = self.pool(x)
        x = F.elu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 2 * 2 * 256)
        x = F.elu(self.fc1(x))
        x = F.elu(self.fc2(x))
        x = F.elu(self.fc3(x))
        x = self.fc4(x)
        return x
```

In [18]:

```
net = ConvNet_3Cl_3KS_AvgPool_4Fl()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
```

```
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 2.077
[1,  4000] loss: 1.844
[1,  6000] loss: 1.739
[1,  8000] loss: 1.624
[1, 10000] loss: 1.540
[1, 12000] loss: 1.471

[2,  2000] loss: 1.806
[2,  4000] loss: 1.525
[2,  6000] loss: 1.382
[2,  8000] loss: 1.323
[2, 10000] loss: 1.313
[2, 12000] loss: 1.270

[3,  2000] loss: 1.257
[3,  4000] loss: 1.153
[3,  6000] loss: 1.158
[3,  8000] loss: 1.152
[3, 10000] loss: 1.090
[3, 12000] loss: 1.101

[4,  2000] loss: 1.003
[4,  4000] loss: 0.963
[4,  6000] loss: 0.961
[4,  8000] loss: 0.964
[4, 10000] loss: 1.006
[4, 12000] loss: 0.934

[5,  2000] loss: 0.785
[5,  4000] loss: 0.841
[5,  6000] loss: 0.800
[5,  8000] loss: 0.804
[5, 10000] loss: 0.811
[5, 12000] loss: 0.808

[6,  2000] loss: 0.641
[6,  4000] loss: 0.629
[6,  6000] loss: 0.633
[6,  8000] loss: 0.634
[6, 10000] loss: 0.647
[6, 12000] loss: 0.668

[7,  2000] loss: 0.515
[7,  4000] loss: 0.485
[7,  6000] loss: 0.472
[7,  8000] loss: 0.474
[7, 10000] loss: 0.471
[7, 12000] loss: 0.474

[8,  2000] loss: 0.355
[8,  4000] loss: 0.351
[8,  6000] loss: 0.352
[8,  8000] loss: 0.343
[8, 10000] loss: 0.360
[8, 12000] loss: 0.341

[9,  2000] loss: 0.265
[9,  4000] loss: 0.266
[9,  6000] loss: 0.265
```

```
[9,  8000] loss: 0.260
[9, 10000] loss: 0.276
[9, 12000] loss: 0.268


[10,  2000] loss: 0.242
[10,  4000] loss: 0.244
[10,  6000] loss: 0.236
[10,  8000] loss: 0.237
[10, 10000] loss: 0.239
[10, 12000] loss: 0.244
Обучение закончено
```

In [19]:

```
check_accuracy(net)
```

```
Accuracy of plane : 78 %
Accuracy of   car : 82 %
Accuracy of  bird : 60 %
Accuracy of   cat : 56 %
Accuracy of  deer : 68 %
Accuracy of   dog : 55 %
Accuracy of  frog : 79 %
Accuracy of horse : 74 %
Accuracy of  ship : 80 %
Accuracy of truck : 79 %
Avg accuracy 71 %
```

В обоих случаях точность классификации снизилась**.** Для текущей архитектуры оптимальным является наличие **3** линейных слоев

Сильная архитектура, которая уже была (!) в ноутбуке

Попробуем обучить ещё более сильную нейросеть**:**

In [15]:

```
class StrongConvNet(nn.Module):
    def __init__(self):
        # вызов конструктора класса nn.Module()
        super(StrongConvNet, self).__init__()

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.dropout = nn.Dropout(p=0.2)

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=8, kernel_size=5)
        self.bn1 = nn.BatchNorm2d(8)
        self.conv2 = nn.Conv2d(in_channels=8, out_channels=16, kernel_size=1)
        self.bn2 = nn.BatchNorm2d(16)
        self.conv3 = nn.Conv2d(in_channels=16, out_channels=16, kernel_size=3)
        self.bn3 = nn.BatchNorm2d(16)
        self.conv4 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=1)
        self.bn4 = nn.BatchNorm2d(32)
        self.conv5 = nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3)
        self.bn5 = nn.BatchNorm2d(32)

        self.fc1 = nn.Linear(4 * 4 * 32, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.bn1(F.relu(self.conv1(x)))
        x = self.pool(x)
        x = self.bn2(F.relu(self.conv2(x)))
        x = self.bn3(F.relu(self.conv3(x)))
        x = self.pool(x)
        x = self.bn4(F.relu(self.conv4(x)))
        x = self.bn5(F.relu(self.conv5(x)))
```

```
#         print(x.shape)
        x = x.view(-1, 4 * 4 * 32)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

Обучим:

In [16]:

```
net = StrongConvNet()
train(net, learning_rate=0.001, epoch_num=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if sys.path[0] == '':
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.858
[1,  4000] loss: 1.665
[1,  6000] loss: 1.570
[1,  8000] loss: 1.522
[1, 10000] loss: 1.473
[1, 12000] loss: 1.398

[2,  2000] loss: 1.300
[2,  4000] loss: 1.293
[2,  6000] loss: 1.291
[2,  8000] loss: 1.292
[2, 10000] loss: 1.263
[2, 12000] loss: 1.233

[3,  2000] loss: 1.156
[3,  4000] loss: 1.149
[3,  6000] loss: 1.156
[3,  8000] loss: 1.148
[3, 10000] loss: 1.144
[3, 12000] loss: 1.148

[4,  2000] loss: 1.049
[4,  4000] loss: 1.031
[4,  6000] loss: 1.073
[4,  8000] loss: 1.057
[4, 10000] loss: 1.026
[4, 12000] loss: 1.039

[5,  2000] loss: 0.974
[5,  4000] loss: 0.984
[5,  6000] loss: 0.998
[5,  8000] loss: 0.966
[5, 10000] loss: 0.951
[5, 12000] loss: 0.955

[6,  2000] loss: 0.905
[6,  4000] loss: 0.910
[6,  6000] loss: 0.907
```

```
[6,  8000] loss: 0.909
[6, 10000] loss: 0.898
[6, 12000] loss: 0.913

[7,  2000] loss: 0.852
[7,  4000] loss: 0.838
[7,  6000] loss: 0.847
[7,  8000] loss: 0.861
[7, 10000] loss: 0.830
[7, 12000] loss: 0.860

[8,  2000] loss: 0.807
[8,  4000] loss: 0.825
[8,  6000] loss: 0.827
[8,  8000] loss: 0.797
[8, 10000] loss: 0.819
[8, 12000] loss: 0.827

[9,  2000] loss: 0.775
[9,  4000] loss: 0.797
[9,  6000] loss: 0.779
[9,  8000] loss: 0.781
[9, 10000] loss: 0.802
[9, 12000] loss: 0.791

[10,  2000] loss: 0.782
[10,  4000] loss: 0.793
[10,  6000] loss: 0.773
[10,  8000] loss: 0.795
[10, 10000] loss: 0.765
[10, 12000] loss: 0.764
Обучение закончено
```

In [17]:

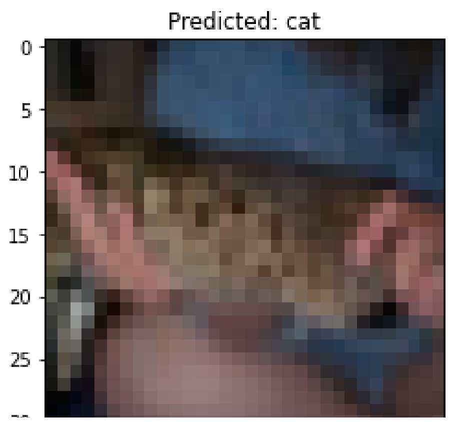
```
check_accuracy(net)
```

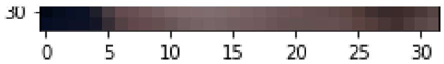
```
Accuracy of plane : 67 %
Accuracy of   car : 77 %
Accuracy of  bird : 50 %
Accuracy of   cat : 48 %
Accuracy of  deer : 60 %
Accuracy of   dog : 51 %
Accuracy of  frog : 72 %
Accuracy of horse : 69 %
Accuracy of  ship : 75 %
Accuracy of truck : 73 %
Avg accuracy 64 %
```

Посмотрим визуально на работу нейросети:

In [ ]:

```
i = np.random.randint(low=0, high=10000)
visualize_result(i)
```





Лучшая архитектура

Итого: наилучший результат **75%** точности в среднем, максимальной точности в **85%** для отдельного класса и минимальной точности в **57%** для отдельного класса. Результат был достигнут при **3**х сверточных слоях с увеличенным числом каналов и меньшим ядром свертки для получения большего пространства признаков (что приводит также к сильному увеличению времени обучения), с **avg** пулингом, **ReLU** в качестве функции активации и **3**мя линейными полносвязными слоями в качестве классификатора.

Даже обучив более глубокую и прокаченную **(BatchNorm, Dropout)** нейросеть на этих данных мы видим, что качество нас всё ещё не устраивает, в реальной жизни необходимо ошибаться не больше, чем на **5%**, а часто и это уже много. Как же быть, ведь свёрточные нейросети должны хорошо классифицировать изображения?

К сожалению, обучение нейросети с нуля на не очень большой выборке (а здесь она именно такая) часто приводит к переобучению, что плохо сказывается на тестовом качестве.

Для того, чтобы получить более качественную модель, часто **до**обучают сильную нейросеть, обученную на **ImageNet**, то есть используют технику **Transfer Learning.** О ней речь пойдёт далее в нашем курсе.

Полезные ссылки

1). *Примеры написания нейросетей на **PyTorch** (официальные туториалы) (на английском):*
https://pytorch.org/tutorials/beginner/pytorch_with_examples.html#examples
https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

2). Курс Стэнфорда: http://cs231n.github.io/

3). Практически исчерпывающая информация по основам свёрточных нейросетей (из **cs231n**) (на английском):

http://cs231n.github.io/convolutional-networks/
http://cs231n.github.io/understanding-cnn/
http://cs231n.github.io/transfer-learning/

4). Видео о **Computer Vision** от **Andrej Karpathy:** https://www.youtube.com/watch?v=u6aEYuemt0M