

Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»



Отчет
Лабораторная работа № 4
По курсу «Проектирование интеллектуальных систем»

Вариант 9

ИСПОЛНИТЕЛЬ:

Попов Илья Андреевич
Группа ИУ5-23М

"__" _____ 2022 г.

ПРЕПОДАВАТЕЛЬ:

Канев А.И.

"__" _____ 2022 г.

Москва 2022

Задание

По заданию выбрать свои классы, загрузить предобученную модель по варианту, заморозить веса модели и провести дообучение на своих классах набора данных. Параметры аугментации использовать из лабораторной работы номер 3.

Сравнить результаты и качество обученных моделей для первых четырех лабораторных работ.

Выполнение

```

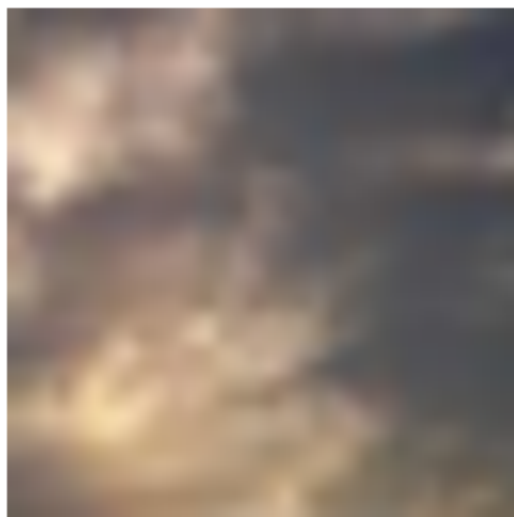
▶ with open('cifar-100-python/train', 'rb') as f:
    data_train = pickle.load(f, encoding='latin1')
with open('cifar-100-python/test', 'rb') as f:
    data_test = pickle.load(f, encoding='latin1')

# Здесь указать ваши классы по варианту!!!
CLASSES = [23, 9, 39]

train_X = data_train['data'].reshape(-1, 3, 32, 32)
train_X = np.transpose(train_X, [0, 2, 3, 1]) # NCHW -> NHWC
train_y = np.array(data_train['fine_labels'])
mask = np.isin(train_y, CLASSES)
train_X = train_X[mask].copy()
train_y = train_y[mask].copy()
train_y = np.unique(train_y, return_inverse=1)[1]
del data_train

test_X = data_test['data'].reshape(-1, 3, 32, 32)
test_X = np.transpose(test_X, [0, 2, 3, 1])
test_y = np.array(data_test['fine_labels'])
mask = np.isin(test_y, CLASSES)
test_X = test_X[mask].copy()
test_y = test_y[mask].copy()
test_y = np.unique(test_y, return_inverse=1)[1]
del data_test
Image.fromarray(train_X[42]).resize((256,256))

```



Загружаем датасет, делим на обучающую и тестовую выборки

```

def __init__(self, X, y, transform=None, p=0.0):
    assert X.size(0) == y.size(0)
    super(Dataset, self).__init__()
    self.X = X
    self.y = y
    self.transform = transform
    self.prob = p

def __len__(self):
    return self.y.size(0)

def __getitem__(self, index):
    x = self.X[index]
    if self.transform and np.random.random() < self.prob:
        x = self.transform(x.permute(2, 0, 1)/255.).permute(1, 2, 0)*255.
    y = self.y[index]
    return x, y

transform = T.Compose([
    T.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.2, hue=0.0),
    T.RandomAffine(degrees=15, translate=(0.1, 0.1), scale=(0.8, 1.2),
        shear=5),
])

Image.fromarray((transform(torch.Tensor(train_X[42]).permute(2, 0, 1)/255.).\
    permute(1, 2, 0).numpy()*255.).astype(np.uint8)).\
    resize((256, 256))

```




Используем аугментацию

```
model = torch.hub.load("chenyaofa/pytorch-cifar-models",
                      "cifar100_mobilenetv2_x0_5",
                      #"cifar100_resnet20",
                      pretrained=True)

model.to(device)
new_model = nn.Sequential(
    Normalize([0.5074,0.4867,0.4411],[0.2011,0.1987,0.2025]),# https://blog.jovian.ai/image-classification-of-cifar100-dataset-using-pytorch-8b7145242df1
    model
).to(device)
print(new_model(torch.rand(1, 32, 32, 3).to(device)))
summary(new_model, input_size=(32, 32, 3))
new_model
```

Downloading: "<https://github.com/chenyaofa/pytorch-cifar-models/archive/master.zip>" to /root/.cache/torch/hub/master.zip
Downloading: "https://github.com/chenyaofa/pytorch-cifar-models/releases/download/mobilenetv2/cifar100_mobilenetv2_x0_5-9f915757.pt" to /root/.cache/torch/hub/checkpoints/cifar100_mobilenetv2_x0_5-9f915757.pt

100%  329M/3.29M [00:00<00:00. 32.2MB/s]

tensor([[-0.0471, -0.2091, -0.0345, 0.0660, -0.0173, -0.7410, -0.2336, 0.3607,
 -0.2076, 0.2140, 0.3131, 0.5753, -0.2791, 0.1509, -0.2442, 0.0022,
 -0.0719, -0.0160, 0.0671, 1.0598, -0.2804, 0.1315, 0.7659, -0.3351,
 -0.1126, 0.3381, 0.3588, 0.2555, -0.1446, 1.5141, -1.0294, -0.2153,
 1.2840, 0.8810, 0.2457, -0.1732, -0.5684, 0.5262, 0.3299, 0.4783,
 0.8978, -0.1241, 0.7717, -0.7854, 0.3129, 0.7681, 0.6000, -0.5473,
 -0.4595, -0.0075, 0.2230, 0.4779, -1.3132, -0.0848, -0.2005, 0.4222,
 -0.7398, 0.4536, -1.0340, -0.0454, -1.0101, -0.2008, -0.6573, 0.1348,
 0.5917, 0.9609, 0.5598, -0.1173, -0.7242, 0.1011, -0.6924, -1.4242,
 0.9028, 0.1655, -0.0584, 0.3919, -0.0692, 0.2560, 0.5414, 0.0704,
 0.3177, -1.3315, 0.1001, -0.8979, -0.0527, 0.3684, -0.3609, 1.2117,
 0.4496, 0.0103, -0.2423, 0.3682, -0.3062, -0.2262, -0.7917, -1.4358,
 0.0384, 0.0372, 0.0976, -0.3527]], device='cuda:0',
 grad_fn=<AddmmBackward0>)

Layer (type)	Output Shape	Param #
Normalize-1	[-1, 3, 32, 32]	0
Conv2d-2	[-1, 16, 32, 32]	432
BatchNorm2d-3	[-1, 16, 32, 32]	32
ReLU6-4	[-1, 16, 32, 32]	0
Conv2d-5	[-1, 16, 32, 32]	144
BatchNorm2d-6	[-1, 16, 32, 32]	32

Загружаем предобученную модель «mobilenetv2_x0_5» по варианту

```
[ ] print("Обучаемые параметры:")
keep_last = 2
total = len(*new_model.named_parameters())
params_to_update = []
for i, (name, param) in enumerate(new_model.named_parameters()):
    if i < total - keep_last:
        param.requires_grad = False
    else:
        params_to_update.append(param)
        print("\t",name)
summary(new_model, input_size=(32, 32, 3))
```

Обучаемые параметры:
1.classifier.1.weight
1.classifier.1.bias

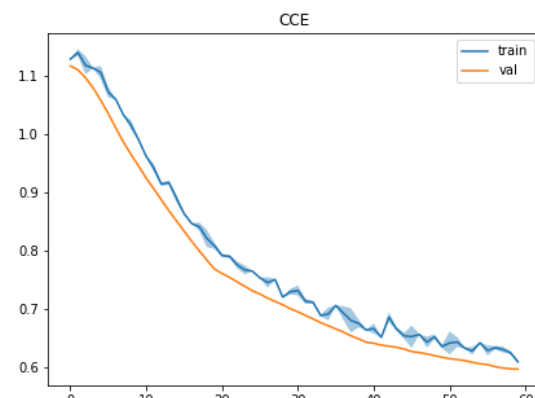
Заморозка весов предобученной модели

```
[ ] # добавляем сглаживание целевых меток, это увеличит значение функции потерь
# но полученная модель будет более устойчивой к выбросам в обучающей выборке
criterion = nn.CrossEntropyLoss(label_smoothing=0)
# используется SGD с momentum и L2-регуляризацией весов
optimizer = optim.SGD(params_to_update, lr=3e-4, momentum=0.9,
                        weight_decay=1e-5)
# добавляем постепенное уменьшение шага обучения каждые 200 эпох
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)
```

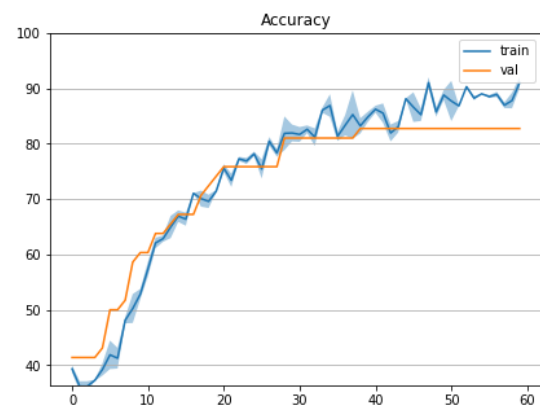
```
[ ] EPOCHS = 60
REDRAW EVERY = 10
steps_per_epoch = len(dataloader['train'])
steps_per_epoch_val = len(dataloader['test'])
# NEW
pbar = tqdm(total=EPOCHS*steps_per_epoch)
losses = []
losses_val = []
passed = 0
# для создания чекпоинта
best_acc = 0
checkpoint_path = 'cifar_cnn_fine.pth'
for epoch in range(EPOCHS): # проход по набору данных несколько раз
    tmp = []
    new_model.train()
    for i, batch in enumerate(dataloader['train'], 0):
        # получение одного минибатча; batch это двухэлементный список из [inputs, labels]
        inputs, labels = batch
        # на GPU
        inputs, labels = inputs.to(device), labels.to(device)

        # очищение прошлых градиентов с прошлой итерации
        optimizer.zero_grad()
```

Эпоха: 60
 Лучшая доля правильных ответов: 82.75862121582031
 Текущая доля правильных ответов: 82.75862121582031
 100% 120/120 [00:10<00:00, 11.20it/s]



Обучение закончено за 98.66965699195862 секунд



Дообучаем не замороженные параметры

Гперпараметры и результат:

Label smoothing 0

Weight decay $1e-5$

Accuracy 0.82