МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

домашнее задание
по дисциплине «Методы машинного обучения»

ИСПОЛНИТЕЛЬ: Ли Яцзинь
ФИО
группа ИУ5И-21М
подпись"___

ПРЕПОДАВАТЕЛЬ: Гапанюк_Ю .Е

Москва - 2024

Задания

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

- выбор задачи;
- теоретический этап;
- практический этап.

Выбор задачи

Классификация изображений(Image Classification)

Обучите полностью подключенную нейронную сеть классифицировать 3 типа изображений в наборе данных CIFAR100, а затем улучшите точность на тестовых выборках.Теоретический этап

Классификация изображений — это фундаментальная задача распознавания изображений, цель которой — понять и классифицировать изображение в целом под определенным ярлыком. В отличие от обнаружения объектов, которое включает в себя классификацию и определение местоположения нескольких объектов на изображении, классификация изображений обычно относится к изображениям одного объекта. Когда классификация становится очень подробной или достигает уровня экземпляра, ее часто называют поиском изображений, который также включает поиск похожих изображений в большой базе данных.

Теоретический этап

Классификация изображений — важная задача в области компьютерного зрения, целью которой является классификация изображений в один из разных классов или классов на основе их содержания. Процесс классификации изображений включает в себя извлечение полезных функций из изображений и использование этих функций для определения категории, к которой принадлежит изображение. Ниже приведены некоторые важные теоретические знания, необходимые для классификации изображений:

Извлечение признаков. В классификации изображений извлечение признаков является ключевым шагом. Особенностями изображения могут быть края, текстуры, цветовые гистограммы и т. д. Традиционные методы извлечения признаков включают SIFT (масштабно-инвариантное преобразование признаков), SURF (устойчивое ускорение признаков), HOG (гистограмма ориентированных градиентов) и т. д. В последние годы методы, основанные на глубоком обучении (например, сверточные нейронные сети), стали основным методом извлечения характеристик изображений.

Структура сети: каждый нейрон полностью связанной нейронной сети связан со всеми нейронами предыдущего слоя, поэтому количество его параметров очень велико. В задачах классификации изображений размерность входного изображения очень высока, а это означает, что требуется очень большое количество нейронов скрытого слоя, что приводит к резкому увеличению количества параметров модели.

Предварительная обработка данных. Перед классификацией изображений изображения обычно необходимо предварительно обработать, чтобы сделать их пригодными для обучения модели. Предварительная обработка может включать масштабирование изображения, обрезку, нормализацию,

увеличение (усиление данных) и другие операции для повышения способности модели к обобщению и уменьшения переобучения.

Обучение и оптимизация модели. После выбора подходящей структуры модели ее необходимо обучить с использованием аннотированных данных изображения. В процессе обучения функция потерь обычно используется для измерения разницы между предсказаниями модели и реальными метками, а алгоритм оптимизации (например, градиентный спуск) используется для настройки параметров модели для минимизации функции потерь.

Оценка и настройка. После завершения обучения модели вам необходимо оценить модель, чтобы понять ее эффективность на невидимых данных. Обычно используемые показатели оценки включают точность, точность, полноту, значение F1 и т. д. На основе результатов оценки модель можно настроить, например настроить гиперпараметры, добавить обучающие данные, улучшить структуру модели и т. д.

Практический этап

1. Загрузка и распаковка набора данных CIFAR100

```
!wget https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
!tar -xvzf cifar-100-python.tar.gz
```

```
--2024-06-07 12:54:57-- https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
Resolving www.cs.toronto.edu (www.cs.toronto.edu)... 128.100.3.30
Connecting to www.cs.toronto.edu (www.cs.toronto.edu)|128.100.3.30|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 169001437 (161M) [application/x-gzip]
Saving to: 'cifar-100-python.tar.gz'

cifar-100-python.ta 100%[===================>] 161.17M  90.0MB/s    in 1.8s

2024-06-07 12:54:59 (90.0 MB/s) - 'cifar-100-python.tar.gz' saved [169001437/169001437]

cifar-100-python/
cifar-100-python/file.txt~
cifar-100-python/train
cifar-100-python/test
cifar-100-python/meta
```

Рисунок 1-Код и результаты

2.Создание Pytorch DataLoader'a

Создание Pytorch DataLoader'a

```
batch_size = 128
dataloader = {}
for (X, y), part in zip([(train_X, train_y), (test_X, test_y)],
                        ['train', 'test']):
        tensor_x = torch.Tensor(X)
        tensor_y = F.one_hot(torch.Tensor(y).to(torch.int64),
                             num_classes=len(CLASSES))/1.
        dataset = TensorDataset(tensor_x, tensor_y) # создание объекта датасета
        dataloader[part] = DataLoader(dataset, batch_size=batch_size, shuffle=True) # создание экземпляр
dataloader
```

Loading...

```
{'train': <torch.utils.data.dataloader.DataLoader at 0x7a1328c65ba0>,
 'test': <torch.utils.data.dataloader.DataLoader at 0x7a12450ccf10>}
```

Рисунок 2-Код и результаты создание Pytorch DataLoader'a

2. Обучение модели по эпохам

```
EPOCHS = 250
steps_per_epoch = len(dataloader['train'])
steps_per_epoch_val = len(dataloader['test'])
for epoch in range(EPOCHS):  # проход по набору данных несколько раз
    running_loss = 0.0
    model.train()
    for i, batch in enumerate(dataloader['train'], 0):
        # получение одного минибатча: batch это двуэлементный список из [inputs, labels]
        inputs, labels = batch

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

        # прямой + обратный проходы + оптимизация
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        #loss = F.cross_entropy(outputs, labels)
        loss.backward()
        optimizer.step()

        # для подсчёта статистик
        running_loss += loss.item()
    print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / steps_per_epoch:.3f}')
    running_loss = 0.0
    model.eval()
    with torch.no_grad(): # отключение автоматического дифференцирования
        for i, data in enumerate(dataloader['test'], 0):
            inputs, labels = data

            outputs = model(inputs)
            loss = criterion(outputs, labels)
            running_loss += loss.item()
    print(f'[{epoch + 1}, {i + 1:5d}] val loss: {running_loss / steps_per_epoch_val:.3f}')
print('Обучение закончено')
...
[214,    12] loss: 0.073
```

Рисунок 3-Код и результаты обучение модели по эпохам

3.Проверка качества модели по классам на обучающей и тестовой выборках

```
for part in ['train', 'test']:
    y_pred = []
    y_true = []
    with torch.no_grad(): # отключение автоматического дифференцирования
        for i, data in enumerate(dataloader[part], 0):
            inputs, labels = data

            outputs = model(inputs).detach().numpy()
            y_pred.append(outputs)
            y_true.append(labels.numpy())
        y_true = np.concatenate(y_true)
        y_pred = np.concatenate(y_pred)
        print(part)
        print(classification_report(y_true.argmax(axis=-1), y_pred.argmax(axis=-1),
                                    digits=4, target_names=list(map(str, CLASSES))))
        print('-'*50)
```

Рисунок 4-Код проверка качества модели

train

	precision	recall	f1-score	support
36	0.9840	0.9860	0.9850	500
62	0.9881	1.0000	0.9940	500
90	0.9980	0.9840	0.9909	500
accuracy			0.9900	1500
macro avg	0.9900	0.9900	0.9900	1500
weighted avg	0.9900	0.9900	0.9900	1500

test

	precision	recall	f1-score	support
36	0.7889	0.7100	0.7474	100
62	0.7664	0.8200	0.7923	100
90	0.8738	0.9000	0.8867	100
accuracy			0.8100	300
macro avg	0.8097	0.8100	0.8088	300
weighted avg	0.8097	0.8100	0.8088	300

Рисунок 5- результаты обучение модели по эпохам

5.Визуализация весов

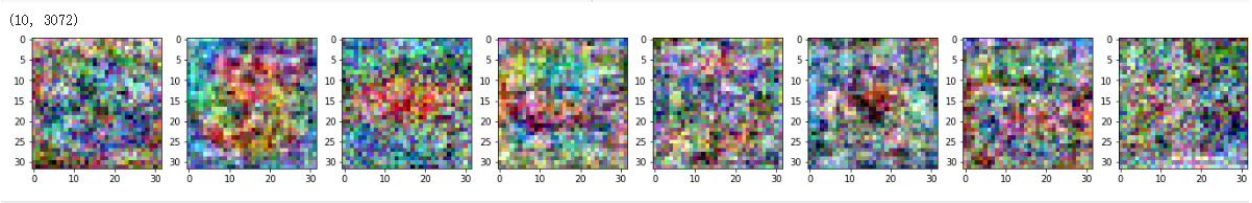

Рисунок 6- результаты визуализация весов