Многослойная нейронная сеть

Задание: создать модель используя нейронную сеть с полносвязными слоями, подобрать гиперпараметры и протестировать на тестовой выборке.

Порядок выполнения:

1. Загрузить базу данных по варианту;
2. Разделить данные на выборки: обучающую, проверочную и тестовую;
3. Провести предварительную обработку данных;
4. Используя фреймворк PyTorch создать нейронную сеть;
5. Обучить нейронную сеть;
6. Подобрать гиперпараметры;
7. Оценить результаты лучшей модели на тестовой выборке.

Ссылки:

1. Официальная документация
2. Туториал по PyTorch

Варианты

Вариант1 - Kuzushiji-MNIST Unown-MNIST на github2

Вариант2 - DermaMNIST MedMNIST v2 на github3

Вариант3 - CINIC-10 CINIC-10 на github4

Вариант4 - Simpsons-MNIST-Grayscale Simpsons-MNIST на Github5

In [ ]:
```
# Импортируйте PyTorch и необходимые модули для работы с фреймворком
```

In [ ]:
```
# Загрузите данные

# Ваш код

# Создайте объекты DataLoader для ваших выборок, размер пакета (batch) выбирите

train_dataloader =  # DataLoader()
val_dataloader = # DataLoader()
test_dataloader = # DataLoader()
```

In [ ]:
```
# Напишите ответ какого ранга тензор, содержащий 1 пакет ваших данных и укажите

# Ваш ответ
```

In [ ]:
```
# Код для получения доступных устройств, которые потом можно задействовать при о

device = (
    "cuda"
```

```
    if torch.cuda.is_available()
    else "mps"
    if torch.backends.mps.is_available()  # Для mps device enables high-performa
    else "cpu"
)
print(f"Используется {device}")
```

In [ ]:
```
# Напишите вашу реализацию NeuralNetwork для нейронной сети
# Не забудьте преобразовать ваши данные в одномерный тензор с помощью flatten()

class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()

        # Ваш код
        pass

    def forward(self, x):

        # Ваш код
        pass
```

In [ ]:
```
# Для экземпляра вашей нейронной сети можно использовать метод .to(device), чтоб
# Если вы используете .to(device) для модели, то не забудьте батчи и метки тоже

model = NeuralNetwork()  # .to(device)
print(model)
```

In [ ]:
```
# попробуйте сперва обучить модель с этими гиперпараметрами
learning_rate = 1e-3
epochs = 3

# выбирите функцию потерь для вашей задачи
loss_fn = None

# выбирите один из оптимизаторов и укажите подходящие параметры
optimizer = None
```

In [ ]:
```
def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    # Переключим модель в режим обучния
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        # Вычислите предсказание, потери и метрики точности
        # Сохраните значение потерь и точности для построения графика
        # Ваш код

        # Обратное распространение ошибки, не забудьте обнулить градиенты
        # Ваш код


def test_loop(dataloader, model, loss_fn):
    # Переключим модель в режим оценки
    model.eval()
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    test_loss, correct = 0, 0

    # Оценка модели в контексте torch.no_grad() гарантирует, что в тестовом режи
```

```
    with torch.no_grad():
        for X, y in dataloader:
            # Напишите код для расчета предсказаний, потерь и метрики точности
            # Сохраните значение потерь и точности для построения графика
            # Ваш код
```

In [ ]:
```
for t in range(epochs):
    train_loop(train_dataloader, model, loss_fn, optimizer)
    test_loop(test_dataloader, model, loss_fn)

# Постройте графики потерь и точности на обучающей и валидационной выборках.
```

Подбор гиперпараметров

In [ ]:
```
# Задание: подберите гиперпараметры сети используя валидационную выборку.
# Попробуйте разные варианты архитектуры (глубина, ширина), количество эпох, ско
# Возможно нормализация данных поможет повысить точность.
# Из не менее 6 вариантов модели выбирите лучшую и сохраните в best_net

# Ваш код

best_net = None # сохраните лучший экземпляр вашей сети
```

Установка модуля для оценки модели

С помощью модуля torchinfo оцените размер вашей модели и количество параметров

torchinfo на github

Команды для выполнения в терминале

```
pip install torchinfo

conda install -c conda-forge torchinfo

mamba install -c conda-forge torchinfo
```

Если необходимо выполнить команду в самой среде Jupyter то перед командой добавьте !, например:

```
!pip install torchinfo
```

In [ ]:
```
from torchinfo import summary
summary(best_net, input_size=(None))  # в input_size укажите правильные размерно
```

Запуск на тестовой выборке

Осталось получить финальные показатели точности для лучшей модели на **тестовой выборке**

In [ ]:
```
test_acc =  # Ваш код
print('Точность на тестовой выборке: ', test_acc)
```

In [ ]:
```
# Постройте матрицу смешения (confusion matrix)
# Опишите гипотезу о возможных проблемах вашей модели
```

In [ ]:
```
# Можете сохранить модель и переиспользовать ее в будущем.

# torch.save(model, 'model.pth')
```

Контрольные вопросы

1. Что такое гиперпараметры модели?
2. Как можно осуществлять подбор гиперпараметров?
3. Зачем данные передавать пакетами (batch)?
4. На что влияет размер пакета (batch)?
5. Расскажите основные этапы цикла обучения (train loop)
6. Из каких модулей строится модель в Pytorch?
7. Какая или какие функции потерь подходят для задачи классификации?