Машинное обучение — Итоговая аттестация

Мы с вами видели, как можно классифицировать изображения в градациях серого (не имеющие информации о цвете) на 10 классов (пример: https://www.kaggle.com/code/amyjang/tensorflow-mnist-cnn-tutorial). Теперь предлагается более сложная задача для другого датасета.

Датасет: https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz.

Задача такая: обучить классификатор цветных изображений с помощью свёрточной (не полносвязной архитектуры) сети искусственных нейронов.

➡️ **Файлы .ipynb, .py, структуру папок (можно с данными — обязательно, если данные не в корне) и описание конвейера выгрузите на ваш удалённый репозиторий, предоставьте ссылку на него.**

Требования:

1. Объяснить, какие элементы вашей сети зависят от количества цветов, какие — от количества классов.
2. Обучить модель. Объяснить место в модели каждого слоя, обосновать выбор гиперпараметров (можно ссылаться на примеры кода в Сети, но в любом случае нужно объяснить словами).
3. Сравнить качество предсказания при обучении на 20 широких классах с предсказаниями при обучении на 100 узких классах, обобщив предсказания по узким меткам до метки их широкого класса в соответствии с таблицей:

Метка широкого класса	Метка узкого класса
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

4. Исследовать с помощью графиков метрики предсказания для каких узких классов более всего отличаются от метрик их более широких классов. Выдвинуть предположение о причине возможного отличия.
5. У вас должен получиться .ipynb файл с объяснениями, оформленными в отдельных текстовых блоках.
6. После готовности блокнота .ipynb преобразуйте его в программу на Питоне (.py) так, чтобы после обучения модели она автоматически экспортировалась бы в файл (см. например, https://www.tensorflow.org/guide/keras/serialization_and_saving).
7. Дополнительно:
   - Опишите конвейер, который бы запускал программу обучения, и при успешном формировании файла модели выгружал бы его как артефакт в репозиторий локально развёрнутого GitLab (это было задание модуля 4 по проектированию и развертке)

- Добавьте в ваш локальный репозиторий специальную ветку, в которой хранились бы только файлы с данными, чтобы при обновлении этой ветки инициировалось бы повторное обучение модели на новых файлах данных.

🧐 Особенности:

Вам придётся разобраться со структурой этого датасета, который хранит изображения не в файлах (см. https://www.tensorflow.org/datasets/catalog/cifar100 и код https://github.com/tensorflow/datasets/blob/master/tensorflow_datasets/image_classification/cifar.py)

> 💡 Изображения в этом датасете уже цветные (3 канала на один пиксел, вместо одного — важно для размера свёртки), и есть два набора обучающих меток (один делит изображения на 20 классов, другой на 100 — важно для понимания структуры данных).

▼ Примеры команд для работы с репозиториями:

▼ 🌐 Работа с двумя репозиториями

Сначала необходимо авторизоваться на всех расположениях, с репозиториями которых будем работать: локальном и gitlab.com с помощью `glab auth login`

Представимся:

```
git config --global user.name "Иванов Джон Йоханович"
git config --global user.email "your@ema.il"
```

Инициализируем локальный репозиторий

`git init --initial-branch=main`

Переименуем текущую ветку (обычно она вначале называется `origin`), чтобы можно было название `origin` использовать для другого расположения:

```
#Если начинаем с материалов локального репозитория
git remote rename origin local
#Если начинаем с материалов удалённого репозитория
git remote rename origin shared
```

> `glab` supports multiple authenticated GitLab instances and automatically detects the authenticated hostname from the remotes available in the working Git directory.
> https://gitlab.com/gitlab-org/cli

```
#Добавим удалённое расположение под именем origin (оно может ещё на этот момент не существовать)
git remote add origin git@gitlab.com:<your_gitlab.com_username>/<your_repo_name>.git
git remote add local git@127.0.0.1:root/<your_repo_name>.git
git remote -v
> local git@127.0.0.1:root/<your_repo_name>.git (fetch)
> local git@127.0.0.1:root/<your_repo_name>.git (push)
> origin  git@gitlab.com:<your_gitlab.com_username>/<your_repo_name>.git (fetch)
> origin  git@gitlab.com:<your_gitlab.com_username>/<your_repo_name>.git (push)

#Создать удалённый репозиторий с помощью glab, если он не был создан ранее (теперь glab позволит работать с хранилищем на gitlab.com тоже)
glab repo list

git add .
git commit -m "Commit message"
# Сделаем удалённый репозиторий с именем origin ведущим для всех веток локального репозитория и отправим в него изменения
```

```
git push --set-upstream origin --all
#git push --set-upstream origin --tags
```

```
adam@y4:~/dev/les_final$ git push --set-upstream origin --all
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To gitlab.com:ar20al17ms/les_final.git
   03ccc40..44279ee  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

```
#Теперь отправим содержание локального (в смысле файловой системы) репозитория в репозиторий локально развёрнутого Gitlab.
git push local
```

```
adam@y4:~/dev/les_final$ git push local
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 326.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To 172.17.0.1:root/les_final.git
   44279ee..36ab983  main -> main
```

Теперь мы можем выбирать, в какой репозиторий публиковать изменения.