Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет Информатики и систем управления
Кафедра Прикладная математика и информатика

Лабораторная работа
по курсу «Машинное обучение»

«Нейронные сети. Распознавание лиц на фото»

Выполнил:
Студент группы ИУ9-31М
Беляев Антон

Проверила:
Криндач О.

Москва, 2019

Содержание

Постановка задачи 3
Используемые инструменты 4
Ход работы 4
Модель 6
Тестирование 8
Вывод 9

Постановка задачи

Необходимо реализовать алгоритм компьютерного зрения с использованием нейронных сетей для решения одной из следующих задач:

- классификация лиц
- определение поведения водителя
- классификация помещений

Можно выбрать любую подходящую модель нейронной сети, в том числе и обученную заранее.

Далее необходимо протестировать алгоритм на выбранном наборе данных и привести результаты.

Используемые инструменты

В качестве задачи была выбрана **классификация лиц на изображении.**

В качестве набора данных было выбрано подмножество известного датасета LFW, содержащего изображения лиц знаменитостей, а именно - изображения 3х людей:

- Вильям (Билл) Саймон (класс 0)
- Дженнифер Энистон (класс 1)
- Леброн Джеймс (класс 2)

Таким образом, задача сводится к классификации изображений по 3м классам с использованием нейронных сетей.

В качестве используемых инструментов были выбраны

- Язык Python 3.7
- Библиотека компьютерного зрения OpenCV
- Библиотека машинного обучения PyTorch

Ход работы

На листинге ниже представлена программа

```
import zipfile
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import torch.nn as nn
from torch.optim import Adam
from torch.utils.data import random_split
from torchvision.transforms import transforms
from torch.utils.data import DataLoader
import torchvision.datasets as datasets
from torchvision.utils import make_grid

with zipfile.ZipFile('data 2.zip', 'r') as zip_ref:
    zip_ref.extractall('.')


transformations = transforms.Compose([
    transforms.RandomResizedCrop(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

```
total_dataset = datasets.ImageFolder('data', transform=transformations)
dataset_loader = DataLoader(dataset=total_dataset, batch_size=100)
# --> {'Bill_Simon': 0, 'Jennifer_Aniston': 1, 'LeBron_James': 2}

train_size = int(0.8 * len(total_dataset))
test_size = len(total_dataset) - train_size
train_dataset, test_dataset = random_split(total_dataset, [train_size, test_size])

train_dataset_loader = DataLoader(dataset=train_dataset, batch_size=10)
test_dataset_loader = DataLoader(dataset=test_dataset, batch_size=10)

class FaceClassifierX(nn.Module):

    def __init__(self, num_classes=5):
        super(FaceClassifierX, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=12, kernel_size=3, stride=1,
padding=1)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2)
        self.conv2 = nn.Conv2d(in_channels=12, out_channels=24, kernel_size=3, stride=1,
padding=1)
        self.relu2 = nn.ReLU()
        self.lf = nn.Linear(in_features=32 * 32 * 24, out_features=num_classes)

    def forward(self, input):
        output = self.conv1(input)
        output = self.relu1(output)
        output = self.maxpool1(output)
        output = self.conv2(output)
        output = self.relu2(output)
        output = output.view(-1, 32 * 32 * 24)
        output = self.lf(output)
        return output

model = FaceClassifierX(num_classes=3)
optimizer = Adam(model.parameters())
loss_fn = nn.CrossEntropyLoss()

def train_and_build(n_epoches):
    for epoch in range(n_epoches):
        model.train()
        for i, (images, labels) in enumerate(train_dataset_loader):
            optimizer.zero_grad()
            outputs = model(images)
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()

# запускаем на 30 эпох
train_and_build(30)
```

```
model.eval()
test_acc_count = 0
for k, (test_images, test_labels) in enumerate(test_dataset_loader):
    test_outputs = model(test_images)
    _, prediction = torch.max(test_outputs.data, 1)
    test_acc_count += torch.sum(prediction == test_labels.data).item()

test_accuracy = test_acc_count / len(test_dataset)

test_accuracy
# —> 0.8888888888888888
```

Выборка была разбита в пропорции 80:20 - train:test. Модель показала высокую точность предсказания - почти **90%!**

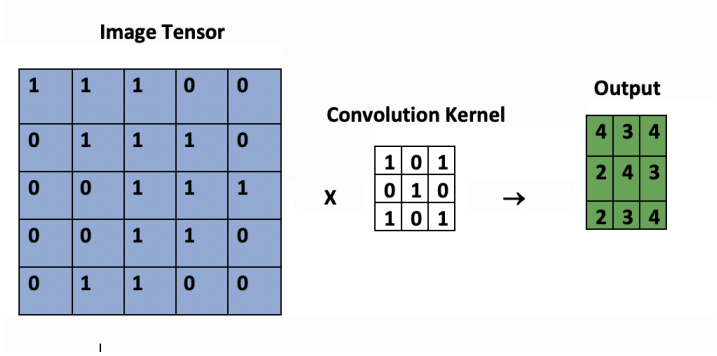
Модель

Т.к. изображение содержит множество «скрытой» информации (скрытых фич), то для извлечения подобных фич необходимо использовать одновременно комбинацию свертки и max-pooling’а.

Свертка математически:

$$f * g = \sum_{a} f(a).g(a)$$

Свертка наглядно:

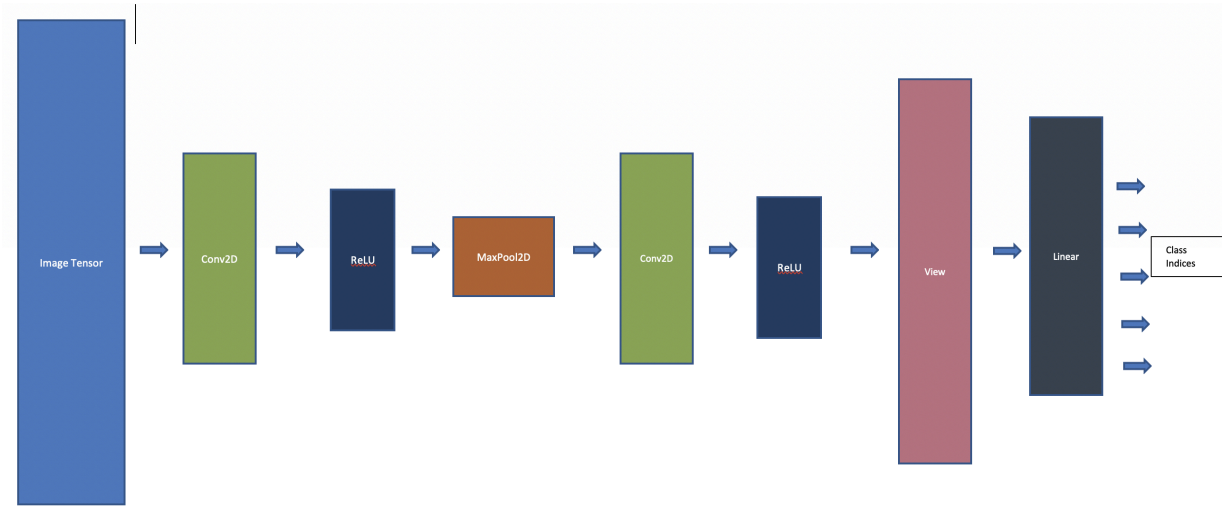


Слой **активационной функции ReLU** для обработки нелинейного выхода предыдущей функции:

$$f(x) = max(0, x)$$

Слой **Max-pooling**’а следует за ReLU и выбирает наибольшее значение из представленных:

Далее следует линейная функция.
В итоге **архитектура используемой нейронной сети** может быть представлена следующим образом:



Данные, на которых обучена модель были нормализованы и преобразованы в изображения размером 64x64 пиксела. Представление их в виде сетки выглядит следующим образом:

Тестирование

Далее необходимо протестировать модель на тестовом изображении Дженнифер Энистон. Это реализовано на листинге ниже

```
test_image = Image.open(«/content/jen_sample_img.jpg»)
test_image_tensor = transformations(test_image).float()
test_image_tensor = test_image_tensor.unsqueeze_(0)
output = model(test_image_tensor)

classified_as_idx = output.data.numpy().argmax()
print(classified_as_idx)
# —> 1

for name, idx in total_dataset.class_to_idx.items():
    if idx == classified_as_idx:
        print(name)
        break
# —> Jennifer_Aniston
```

Полученный класс - 1. Таким образом, поданное на вход изображение **классифицировано верно!**

Вывод

В ходе работы была выполнена задача классификации изображения людей.

Точность предсказания полученной модели высокая и, ожидаемо, тестовое изображение было классифицировано верно.