Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Национальный исследовательский Нижегородский государственный университет
им. Н.И. Лобачевского»

Институт Информационных Технологий, Математики и Механики

Направление: Прикладная математика и информатика

Магистерская программа: Компьютерные науки и приложения

ОТЧЕТ

по лабораторной работе №5

Тема:

«Применение переноса обучения для решения задачи, поставленной во второй лабораторной работе»

Выполнили: студенты группы 381803-4м
Котова О.А.

_____
Подпись
Лицов А.

_____
Подпись
Синицкая О.

_____
Подпись

Преподаватель: доцент, к.т.н. Кустикова
В.Д.

_____
Подпись

Нижний Новгород
2019

Оглавление

Постановка задачи

Цели

Цель настоящей работы состоит в том**,** чтобы исследовать возможности переноса обучения для решения целевой задачи**,** выбранной изначально для выполнения практических работ**.**

Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Поиск исходной задачи (близкой по смыслу к целевой задаче) и поиск натренированной модели для решения исходной задачи.
2. Выполнение трех типов экспериментов по переносу знаний (типы экспериментов описаны в лекции).

Тренировочные и тестовые наборы данных

Выбранная задача - Intel Image Classification: https://www.kaggle.com/puneet6060/intel-image-classification.

Исходные данные хранятся в директориях seg_pred, seg_test, seg_train в формате jpg и размера 150x150.

- seg_pred содержит 7301 изображений
- seg_test - 3000 изображений, которые распределены по папкам
  - buildings
  - forest
  - glacier
  - mountain
  - sea
  - street
- seg_train - 14034 изображений, которые распределены по папкам
  - buildings
  - forest
  - glacier
  - mountain
  - sea
  - street

Данные содержат около 25 тыс. цветных изображений размером 150x150, распределенных по 6 категориям: здания, лес, ледник, гора, море, улица. Изображения хранятся в формате jpg.

Тренировочная выборка содержит 14034 изображений.

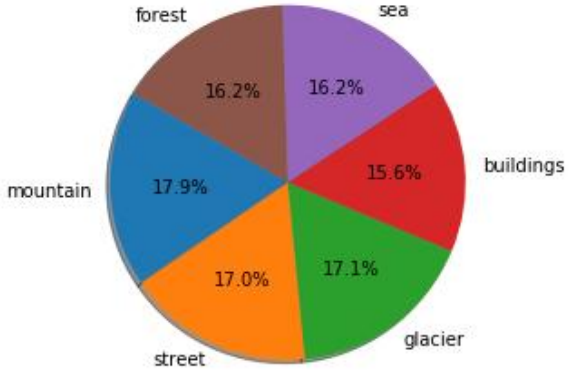
Тестовая выборка содержит 3000 изображений.

Размер каждого изображения 150x150.

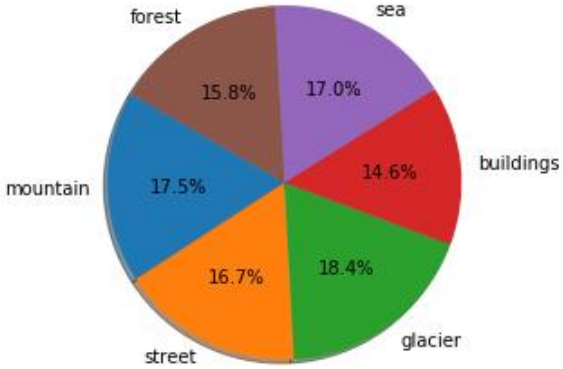
№	*Категории*	*Размер тренировочной выборки*	*Размер тестовой выборки*
1	mountain	2512	525

2	street	2382	501
3	glasier	2404	553
4	buildings	2191	437
5	sea	2274	510
6	forest	2271	474

Процентное соотношение категорий. Тренировочная выборка:



Процентное соотношение категорий. Тестовая выборка:



Метрика качества решения

Для оценки качества решения задачи выбрана метрика "Точность" ("Accuracy"). Она вычисляет, как часто прогнозы соответствуют меткам. Иными словами, частота с которой y_pred совпадает с y_true.

$$accuracy(y_{pred}, y_{true}) = \frac{1}{N}\sum_{i=1}^{N} 1(y_{pred_i} == y_{true_i})$$

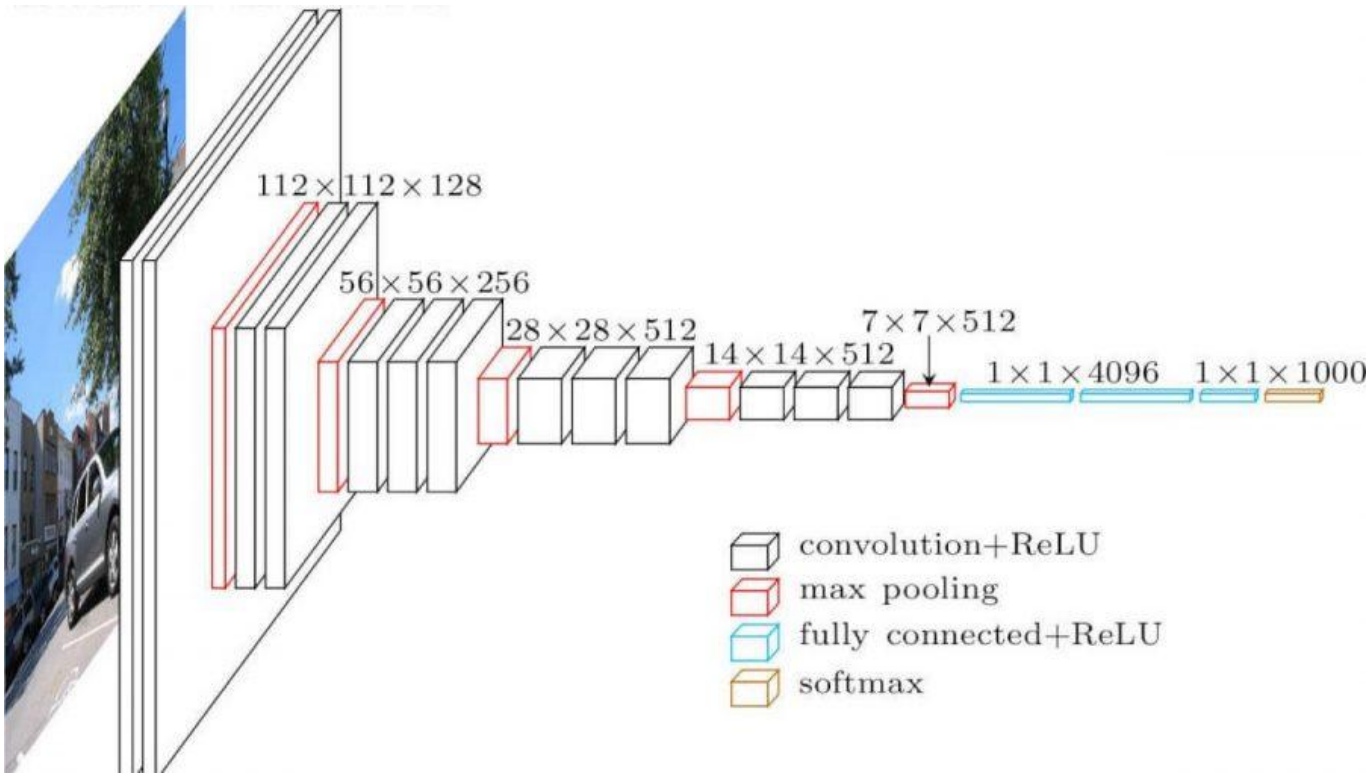
Разработанные программы

Lab3.ipynb – скрипт для обучения свёрточных нейронных сетей.

Исходная модель

В качестве исходной модели для переноса обучения будем использовать **VGG16** (модель сверточной нейронной сети) на задаче ImageNet. Датасет состоит из более чем 14 миллионов изображений, принадлежащих к 1000 классам.

Архитектура VGG16 представлена на рисунке ниже.



Конфигурация сети

Input (224 * 224 RGB image)
Conv (3,3), 64
Conv (3,3), 64
Maxpooling(2,2), 2
Conv (3,3), 128, relu
Conv (3,3), 128, relu
Maxpooling(2,2), 2
Conv (3,3), 256, relu
Conv (3,3), 256, relu

Conv (1,1), 256, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu
Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu
Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2
FC – 4096, relu
FC – 4096, relu
FC – 1000, softmax

Формат данных, предоставляющийся на вход сети

С помощью класса ImageDataGenerator и его метода flow_from_directory() генерируем пакеты. Данные возвращаются в формате (x, y), где x, y - numpy массивы.

Форма x: (batch_size, 150, 150, 3).
Форма y: (batch_size, 6).

Методу fit_generator подается на вход генератор данных в формате (x, y). Сети подается на вход массив numpy формата (150, 150, 3), который "сглаживается" сетью с помощью метода Flatten().

Описание экспериментов и конфигурации

Эксперимент 1

Использование структуры глубокой модели, построенной для решения исходной «Задачи А», с целью обучения аналогичной модели для решения «Задачи В»

- Предполагается, что модель, построенная для решения исходной задачи, обучается на данных, подготовленных для решения целевой задачи
- При этом веса модели инициализируются случайным образом
- Эксперимент реализует перенос знаний для родственных доменов

Т. е. в этом эксперименте используем только структуру модели. Чтобы применить данную модель VGG16 к нашей задаче, заменим на последнем слое количество выходов с 1000 на 6.

Конфигурация 1:

Input (150 * 150 RGB image)
Conv (3,3), 64
Conv (3,3), 64
Maxpooling(2,2), 2
Conv (3,3), 128, relu
Conv (3,3), 128, relu
Maxpooling(2,2), 2
Conv (3,3), 256, relu
Conv (3,3), 256, relu
Conv (1,1), 256, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu
Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu

Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2
FC – 4096, relu
FC – 4096, relu
FC – **6**, softmax

Эксперимент 2

Использование модели, построенной для решения исходной «Задачи А», в качестве фиксированного метода извлечения признаков при построении модели, решающей «Задачу В»

- Идея данного подхода состоит в том, чтобы удалить из глубокой модели классификатор (последние полностью связанные слои) и рассматривать начальную часть сети как метод выделения признаков
- Взамен старого классификатора можно поместить новый классификатор (например, другой набор полностью связанных слоев или машину опорных векторов) и обучить его на признаках, построенных с использованием начальной части сети
- Эксперимент реализует перенос признакового описания

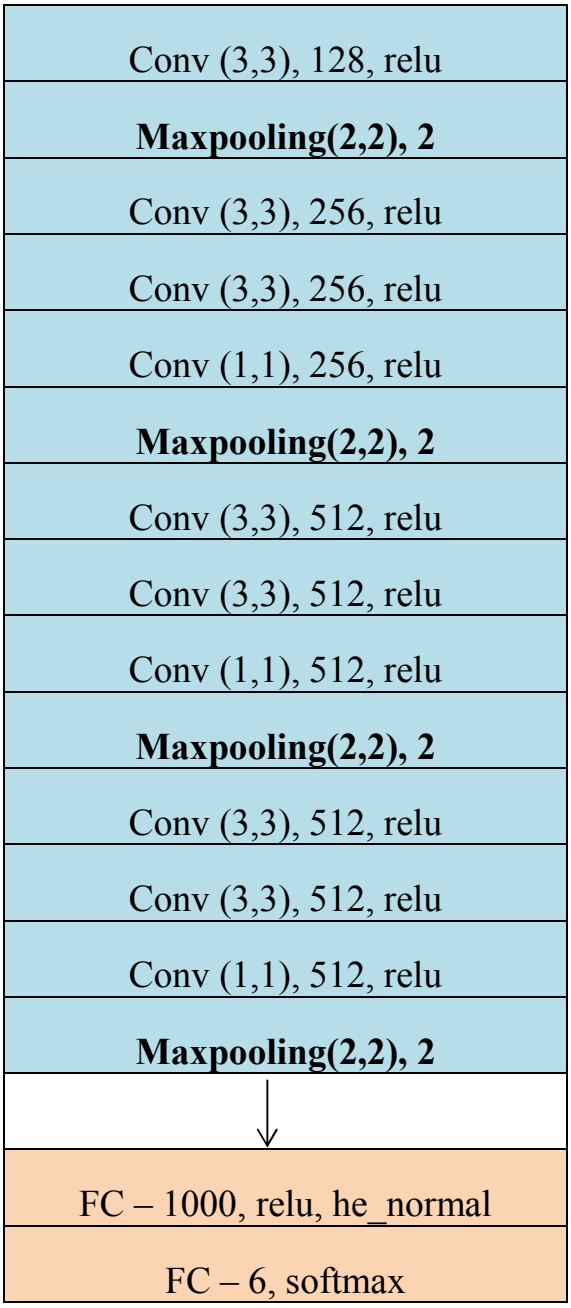
Во втором эксперименте возьмем сверточную основу VGG16, обученную на наборе ImageNet.

Пропустим наш набор данных через предварительно обученную сверточную основу VGG16, таким образом выделив признаки.

Далее возьмем полносвязный классификатор и обучим его на полученных признаках.

Конфигурация 2:

Input (150 * 150 RGB image)
Conv (3,3), 64
Conv (3,3), 64
Maxpooling(2,2), 2
Conv (3,3), 128, relu



Эксперимент 3

Тонкая настройка параметров модели, построенной для решения исходной «Задачи А», с целью решения «Задачи В»

- Последние слои глубокой модели, соответствующие классификатору, который решает «Задачу А», заменяются новым классификатором (например, набором полностью связанных слоев с другим количеством выходов)
- Полученная модель обучается как единая система
- Эксперимент реализует перенос обучения на основе экземпляров

Последние слои глубокой модели VGG16, соответствующие классификатору, который решает ImageNet, заменим новым полносвязным классификатором с количеством выходов 6.

И обучим модель как единую систему.

Конфигурация 3:

Input (150 * 150 RGB image)
Conv (3,3), 64
Conv (3,3), 64
Maxpooling(2,2), 2
Conv (3,3), 128, relu
Conv (3,3), 128, relu
Maxpooling(2,2), 2
Conv (3,3), 256, relu
Conv (3,3), 256, relu
Conv (1,1), 256, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu
Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2
Conv (3,3), 512, relu
Conv (3,3), 512, relu
Conv (1,1), 512, relu
Maxpooling(2,2), 2

FC – 1000, relu, he_normal
FC – 6, softmax

Результаты экспериментов

В таблице приведены конфигурация системы и программное обеспечение, с помощью которых проводилось обучение и тестирование построенных моделей.

Параметры	*Версия*
GPU	Tesla P100, having 3584 CUDA cores, 16GB(16.28GB Usable) GDDR6 VRAM Tesla P100 Spec Sheet
Python	3.7.5
TensorFlow	2.0.0

Параметры обучения:

Количество эпох	20
Размер пачки	128

Результаты экспериментов:

Модель	1 Использование только структуры глубокой модели	2 Извлечение признаков и полносвязный классификатор	3 Тонкая настройка
Батч	128	128	128
Количество эпох	20	20	20
Общее время (сек)	1081	74	602
Точность (Accuracy) на тренировочном наборе, %	73.20	98.29	**98.95**
Ошибка на тренировочном наборе	0.73	0.04	0.03
Точность (Accuracy) на тестовом наборе, %	72.87	86.47	**88.10**
Ошибка на тестовом наборе	0.70	0.76	0.63

Выводы

В ходе выполнения лабораторной работы была получена модель 3, которая позволяет решать выбранную практическую задачу с достаточно высокими показателями качества. Была спроектирована и разработана программная реализация, позволяющая обучать различные конфигурации нейронных сетей. С помощью полученной реализации были произведены эксперименты на выбранном наборе данных. Во время экспериментов была измерена ошибка классификации. Полученные результаты отражены в настоящем отчете.