МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

ПРОФИЛЬ ИННОВАЦИОННЫЕ IT-ПРОЕКТЫ

НАПРАВЛЕНИЕ ПРИКЛАДНАЯ ИНФОРМАТИКА

ГРУППА МПИ-20-4-2

Отчёт по лабораторной работе №4

ПО КУРСУ: «Нейронные сети и машинное обучение»

СТУДЕНТКА Денисова Е.А.

ПРЕПОДАВАТЕЛЬ Курочкин И. И.

Москва, 2020г.

В ходе данной лабораторной работы была выбрана и реализована архитектура сверточной нейронной сети для классификации цветных изображений и решена задача классификации изображений на эталонном датасете CIFAR-10.

CIFAR-10 состоит из 60 000 цветных фотографий размером 32 x 32 пикселя. Датасет содержит 10 классов:

- 0 – самолет;
- 1 – автомобиль;
- 2 – птица;
- 3 – кошка;
- 4 – олень;
- 5 – собака;
- 6 – лягушка;
- 7 – лошадь;
- 8 – корабль;
- 9 – грузовик.

При загрузке данного датасета он состоит из обучающего и тестового множеств, где размер обучающего множества составляет 50000 фотографий, а тестового – 10000 фотографий.

При подготовке данных производится горячее кодирование вектора классов. Данное кодирование преобразовывает категориальные данный в числовую форму. При данном кодировании целое число, обозначающее класс, преобразовывается в двоичный вектор из 10 элементов, где число «1» стоит на позиции, номер которой соответствует номеру класса, а остальные числа в векторе являются нулями. После этого проводится нормировка значений пикселей, поскольку они представляют собой значения от 0 до 255. При нормировке они масштабируются до [0, 1].

Для реализации была выбрана архитектура сверточных нейронных сетей «VGG3» с улучшением в виде регуляризации отсева. Описание данной архитектуры дано в статье: «How to Develop a CNN From Scratch for CIFAR-10 Photo Classification», опубликованной на Machine Learning Mastery в мае 2019. Ссылка на статью будет дана в конце отчета. Данная архитектура показала наивысший уровень производительности в конкурсе «ILSVRC 2014».

Архитектура предполагает наложение сверточных слоев с фильтрами 3x3, за которыми следует слой максимального объединения. Вместе данные слои образуют блок и далее данные блоки могут повторяться и количество фильтров в каждом блоке

увеличивается с глубиной сети. Например, для первых четырех блоков модели значения: 32, 64, 128, 256. В данной работе реализована архитектура с тремя блоками.

Для повышения качества классификации в данной работе используется метод отсева. В данном методе из нейронной сети случайным образом удаляются узлы, что заставляет оставшиеся узлы адаптироваться к измененным условиям.

Слои нейронной сети и их параметры:

1. Conv2D – сверточный слой. Количество фильтров в слое – 32, размер фильтров 3x3, функция активации – «relu», заполнение – «same»(вывод имеет ту же длину и ширину, что и ввод), форма входных данных – (32, 32, 3), инициализатор для матрицы весов ядра(определяет способ установки начальных случайных весов) – «he_uniform»(задает значения весов в пределах [-limit; limit], где limit = sqrt(6/fan_in), где fan_in – число входящих экземпляров в тензор весов);
2. Conv2D – сверточный слой. Количество фильтров в слое – 32, размер фильтров 3x3, функция активации – «relu», заполнение – «same», инициализатор для матрицы весов ядра– «he_uniform»;
3. MaxPooling2D – слой максимального объединения для 2D данных. Уменьшает разрешение входного представления, беря максимальное значение в окне, размер которого определен в параметре «pool_size». Pool size - (2,2).
4. Dropout – слой сброса. Случайным образом устанавливает для входных единиц значение 0 с частотой rate. Rate = 0.2;
5. Conv2D – сверточный слой. Количество фильтров в слое – 64, размер фильтров 3x3, функция активации – «relu», заполнение – «same», инициализатор для матрицы весов ядра– «he_uniform»;
6. Conv2D – сверточный слой. Количество фильтров в слое – 64, размер фильтров 3x3, функция активации – «relu», заполнение – «same», инициализатор для матрицы весов ядра– «he_uniform»;
7. MaxPooling2D – слой максимального объединения для 2D данных. Pool size - (2,2).
8. Dropout – слой сброса. Rate = 0.2;
9. Conv2D – сверточный слой. Количество фильтров в слое – 128, размер фильтров 3x3, функция активации – «relu», заполнение – «same», инициализатор для матрицы весов ядра– «he_uniform»;

10. Conv2D – сверточный слой. Количество фильтров в слое – 128, размер фильтров 3x3, функция активации – «relu», заполнение – «same», инициализатор для матрицы весов ядра– «he_uniform»;
11. MaxPooling2D – слой максимального объединения для 2D данных. Pool size - (2,2).
12. Dropout – слой сброса. Rate = 0.2;
13. Flatten – сглаживающий слой;
14. Dense – глубоко связанный слой, что означает, что каждый нейрон в данном слое получает данные от всех нейронов предыдущего слоя. Число нейронов – 128, функция активации – «relu», инициализатор для матрицы весов ядра– «he_uniform»;
15. Dropout – слой сброса. Rate = 0.2;
16. Dense – глубоко связанный слой. Число нейронов равно числу классов, в данном случае – 10, функция активации – «softmax».

Вывод слоев модели:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_14 (Dropout)         (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_15 (Dropout)         (None, 4, 4, 128)         0
```

```
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               262272
_________________________________________________________________
dropout_16 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
```

Также в модели для оптимизации используется стохастический градиентный спуск со скоростью обучения 0,001 и импульсом 0,9(параметр, который ускоряет градиентный спуск в соответствующем направлении). В качестве функции потерь используется категориальная кросс-энтропия, так как классификация является мультиклассовой.

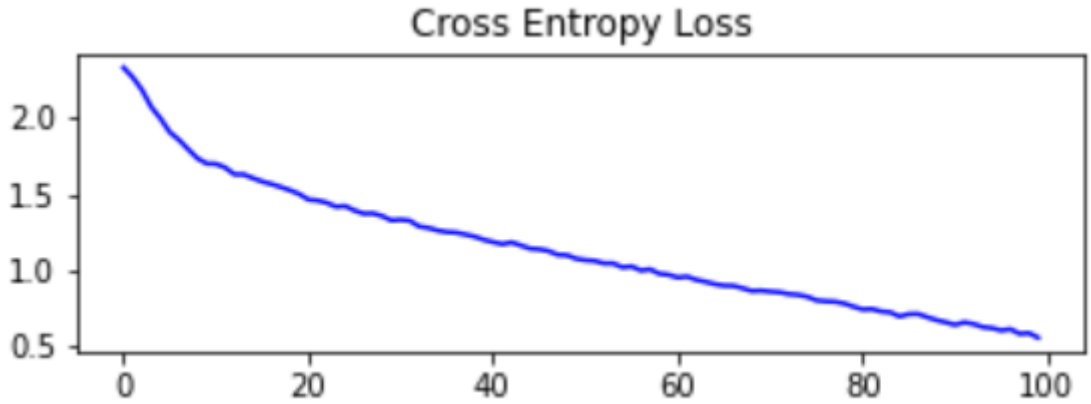
Метрики качества для обучающего множества:

Accuracy	0.825
Precision	0.751
Recall	0.736
F1-measure	0.743

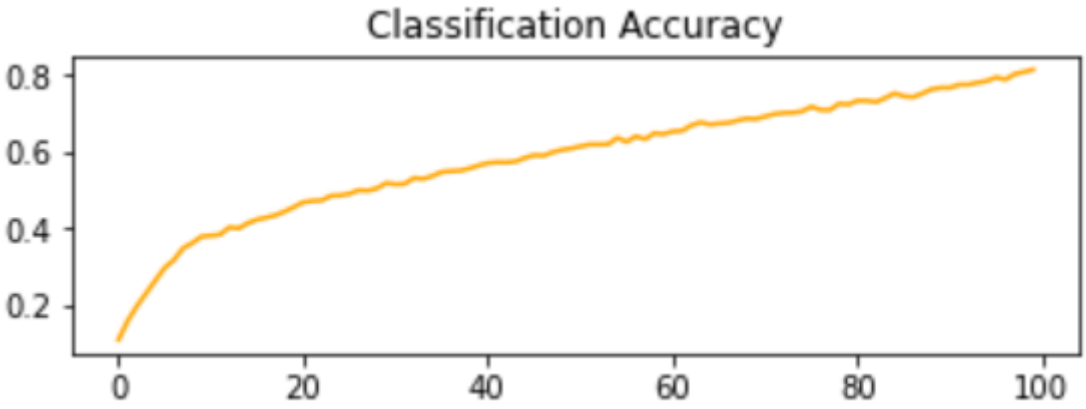
Метрики качества для тестового множества:

Accuracy	0.804
Precision	0.723
Recall	0.704
F1-measure	0.713

Динамика loss для обучающего множества:



Динамика accuracy для обучающего множества:



В рассматриваемой статье для оценки качества обучения модели использовалась только метрика «accuracy». Получившееся значение accuracy в статье:

```
1 > 83.450
```

График динамики loss в статье для обучающего и тестового множества (синим отмечено обучающее множество, оранжевым – тестовое):

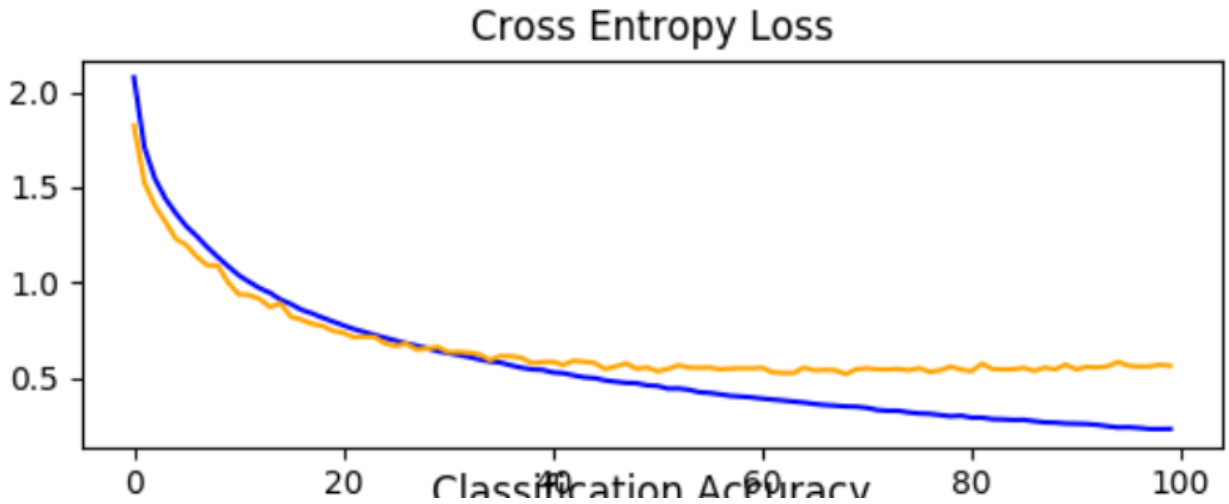
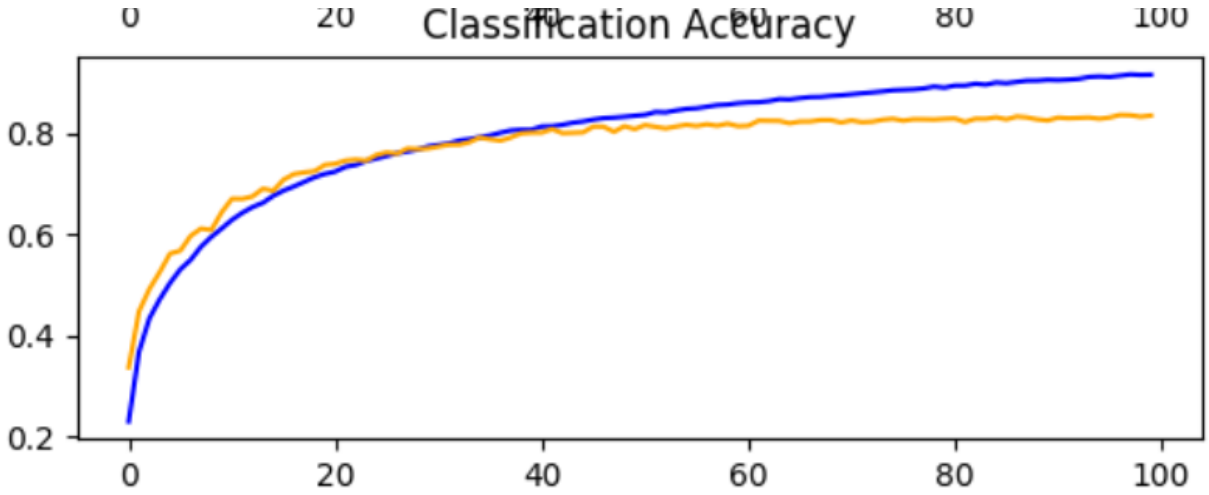


График динамики accuracy в статье для обучающего и тестового множества (синим отмечено обучающее множество, оранжевым – тестовое):



Далее необходимо было провести дообучение нейронной сети на классах из одного суперкласса датасета CIFAR-100. Этот набор данных похож на CIFAR-10, за исключением того, что он содержит 100 классов, каждый из которых содержит 600 изображений. На каждый класс приходится 500 обучающих изображений и 100 тестовых изображений. 100 классов в CIFAR-100 сгруппированы в 20 суперклассов. Каждое изображение имеет метку «точный» (класс, к которому оно принадлежит) и метку «грубое» (суперкласс, к которому оно принадлежит). В данном варианте лабораторной работы необходимо было взять суперкласс «people», классы: «baby», «boy», «girl», «man», «woman». Для подготовки данных также была проведена нормировка пикселей и они были отмасштабированы до значений в отрезке [0, 1].

Для переобучения нейронной сети использовалось трансферное обучение. Для этого изменяется последний слой сети для того, чтобы он имел нужное количество классов (в данном случае – 15), и происходит дообучение сети. При дообучении сети замораживаются все сверточные слои, слои максимального объединения и некоторые из слоев сброса для того, чтобы они не изменяли свои ранее найденные веса. Свободными остаются только глубоко связанные слои.

Метрики качества для обучающего множества:

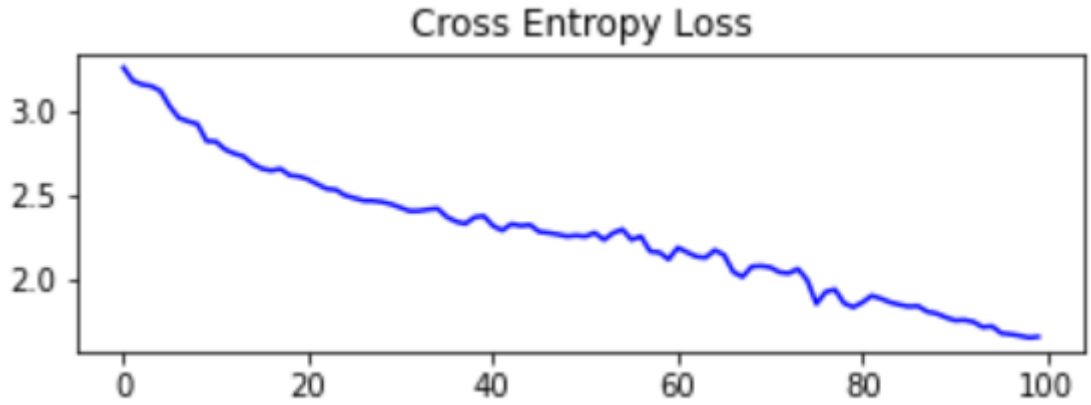
Accuracy	0.717
Precision	0.673
Recall	0.648
F1-measure	0.660

Метрики качества для тестового множества:

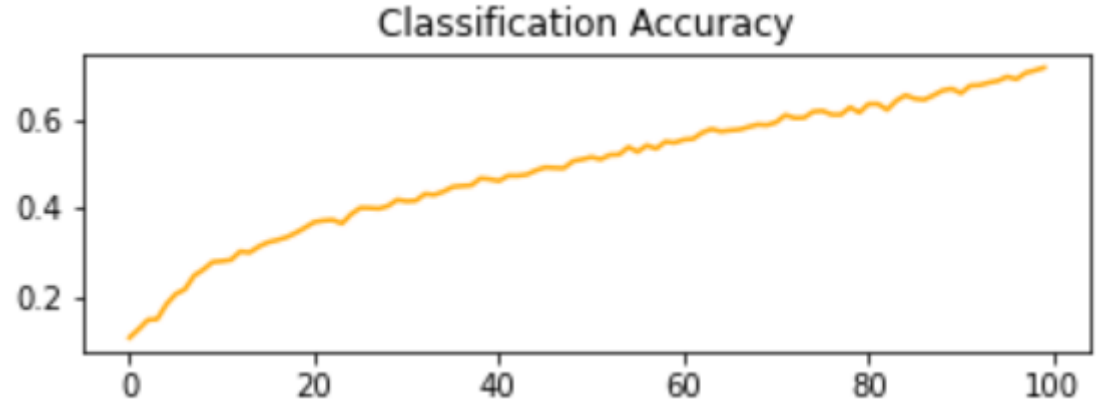
Accuracy	0.671
Precision	0.634
Recall	0.607

F1-measure	0.620

Динамика loss для обучающего множества:



Динамика accuracy для обучающего множества:



В данном случае можно наблюдать, что значения метрик после добавления классов из суперкласса из датасета CIFAR100 значения метрик ухудшились.