**НИТУ «МИСиС»**

Квалификация (степень): **магистр**

Курс: **1**

Семестр: **1**

Дисциплина: Нейронные сети

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ**

на тему:

**«Свёрточные нейронные сети»**

**Выполнил:**

Группа: МПИ-20-4-2

Студент: Новицкий Д. А.

**Проверил:** доц., к.т.н. Курочкин И. И.

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва 2020**

**Оглавление**

[Постановка задачи 3](#_Toc61988261)

[Ход работы 3](#_Toc61988262)

[Результаты 3](#_Toc61988263)

[Входные/выходные данные 3](#_Toc61988264)

[Загрузка файлов 4](#_Toc61988265)

[Описание архитектуры свёрточной нейронной сети 5](#_Toc61988266)

[Описание работы программы 10](#_Toc61988267)

[Этапы работы программы 10](#_Toc61988268)

[Часть 1. Работа с датасетом cifar-10 10](#_Toc61988269)

[Часть 2. Работа с датасетом cifar-100 10](#_Toc61988270)

[Описание параметров свёрточной нейронной сети 11](#_Toc61988271)

[Результаты работы программы 12](#_Toc61988272)

[Список используемой литературы 21](#_Toc61988273)

# Постановка задачи

## Ход работы

1. Выбрать/разработать архитектуру сверточной нейронной сети (CNN) для классификации цветных изображений.

2. Решить задачу классификации изображений на эталонном датасете CIFAR-10. Настройку архитектуры CNN и параметров обучения проводить для получения наилучших результатов.

3. Полученные результаты (см. раздел "Результаты") сравнить с опубликованными результатами для аналогичных/похожих архитектур и дать на них ссылку.

4. Используя ту же CNN и полученный набор весов дообучить CNN на классах из одного суперкласса (по вариантам) эталонного датасета CIFAR-100. Задача классификации решается на классах CIFAR-10 + классы из одного суперкласса CIFAR-100.

5. Провести сравнительный анализ результатов (см. раздел "Результаты") с полученными результатами из п.3.

## Результаты

1. Описание архитектуры сверточной нейронной сети, параметры слоев и настройки обучения (к примеру, типы слоев, количество нейронов в каждом слое, функции активации, начальные значения весов).

2. Метрики качества: Accuracy, Precision, Recall, F1-measure, матрица ошибок (confusion matrix).

3. Динамика обучения (Accuracy/loss) для обучающего множества.

4. Метрики качества по завершению обучения должны быть определены отдельно для обучающего и тестового множества.

## Входные/выходные данные

1. Входные данные (датасеты) в виде текстового файла. (к примеру, экспорт таблицы из Excel в формате TXT или CSV).

2. Выходные данные (результаты) сохраняются в виде HTML, PDF или DOCX отчета (текст и графики).

## Загрузка файлов

1. Ссылка на проект в github/gitlab.

2. Методика разделения датасетов на обучающее и тестовое множества, если это разделение отлично от стандартного.

3. Текстовый файл со сценариями запуска (к примеру: myapp.exe dataset1 parameters).

4. Отчет по ЛР с приведенными результатами и пояснениями (к примеру, если Вы получили какой-то график, приведите ниже сценарий и параметры запуска вычислительного приложения).

# Описание архитектуры свёрточной нейронной сети

Рассмотрим реализацию свёрточной нейронной сети в keras. Начнем с того, что сверточный слой в двумерном случае реализуется с помощью класса: [1]

keras.layers.Conv2D(filters,

kernel\_size, strides=(1, 1),..)

где **filters** – число фильтров (каналов);

**kernel\_size** – размер ядра каждого фильтра;

**strides** – шаг сканирования фильтров по осям плоскости (по умолчанию один пиксел).

В качестве примера рассмотрим задачу классификации рукописных цифр. Каждая такая цифра представлена в виде изображения 28\*28 пикселей в градациях серого (рис. 1).

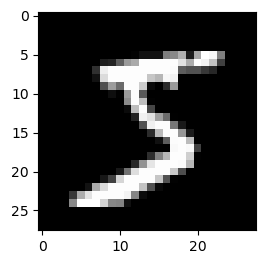


Рис. . Пример изображения рукописной цифры размером 28\*28 пикселей

Первый сверточный слой определим так:

model = keras.Sequential

([

    Conv2D(32, (3,3), padding='same', activation='relu', input\_shape=(28, 28, 1)),

…

])

Здесь указано 32 фильтра с ядрами 3х3 пиксела каждый. Затем, параметр padding = «same» означает, что выходная карта признаков на каждом канале должна быть той же размерностью, что и исходное изображение, т.е. 28х28 элементов. Фактически, это означает добавление значений на границах двумерных данных (обычно нулей), чтобы центр ядра фильтра мог размещаться над граничными элементами (рис. 2):

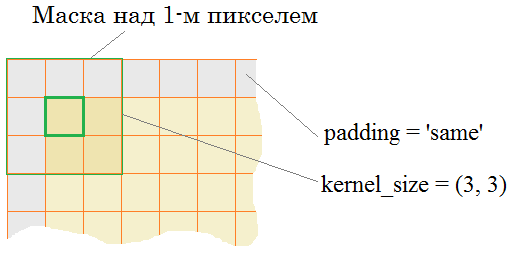


Рис. . Визуализация маски

Следующий параметр – это функция активации ReLu и формат входных данных в виде изображений 28х28 пикселей с одним цветовым каналом (градации серого).

Следующий слой в соответствии с концепцией СНС должен укрупнять масштаб полученных признаков. Как мы говорили на предыдущем занятии, для этого чаще всего используется операция MaxPooling (рис. 3):

keras.layers.MaxPooling2D(pool\_size=(2, 2), strides=None, padding=«valid», data\_format=None)

Где **pool\_size** – размер окна, в котором выбирается максимальное значение;

**strides** – шаг сканирования по координатам плоскости;

**padding**=«valid» – не добавлять нулевых значений на границах (соответственно рамка не смещается за пределы поля признаков);

**data\_format** – формат входных данных (об этом поговорим чуть позже).

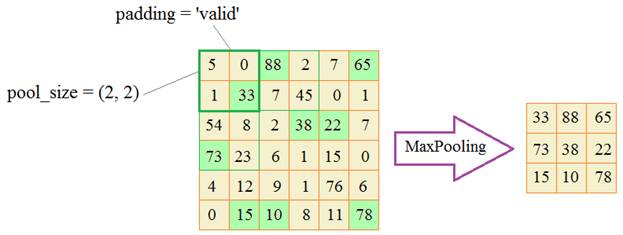


Рис. . Визуализация работы операции MaxPooling

Добавим в модель операцию (слой) MaxPooling2D:

model = keras.Sequential

([

    Conv2D(32, (3,3), padding='same', activation='relu', input\_shape=(28, 28, 1)),

    MaxPooling2D((2, 2), strides=2),

…

])

По аналогии пропишем еще два таких слоя:

model = keras.Sequential

([

    Conv2D(32, (3,3), padding='same', activation='relu', input\_shape=(28, 28, 1)),

    MaxPooling2D((2, 2), strides=2),

    Conv2D(64, (3,3), padding='same', activation='relu'),

    MaxPooling2D((2, 2), strides=2),

…

])

Здесь следующий слой свертки содержит уже 64 фильтра, то есть, на выходе будем иметь 64 канала. После операции MaxPooling2D каждая карта признаков уменьшается до размера 7х7 элементов.

Таким образом, все свертки сделаны. Далее, нужно вытянуть полученный тензор

7х7х64 в единый вектор. Это выполняется с помощью специального слоя:

Keras.layers.Flatten(data\_format=None)

И, затем, подать его на полносвязную сеть из 128 нейронов и 10 нейронов выходного слоя. Получаем следующую архитектуру СНС для распознавания рукописных цифр:

model = keras.Sequential

([

    Conv2D(32, (3,3), padding='same', activation='relu', input\_shape=(28, 28, 1)),

    MaxPooling2D((2, 2), strides=2),

    Conv2D(64, (3,3), padding='same', activation='relu'),

    MaxPooling2D((2, 2), strides=2),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(10,  activation='softmax')

])

Выведем структуру этой сети и посмотрим на число весовых коэффициентов в каждом слое:

print(model.summary())

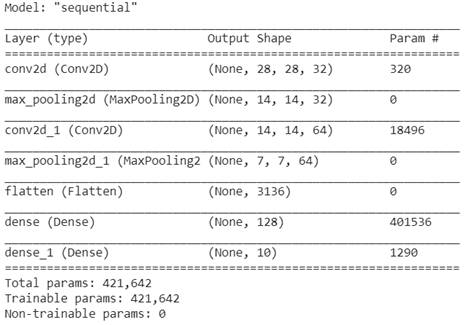


Рис. . Результат работы операции print(model.summary())

Теперь обучим эту сеть и посмотрим на результаты ее работы. Вначале подключим необходимые библиотеки, загрузим обучающую и тестовую выборки и стандартизируем входные данные:

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

*# Библиотека базы выборок Mnist*

**from** tensorflow.keras.datasets **import** mnist

**from** tensorflow **import** keras

**from** tensorflow.keras.layers **import** Dense, Flatten, Dropout, Conv2D, MaxPooling2D

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

*# Стандартизация входных данных*

x\_train = x\_train / 255

x\_test = x\_test / 255

y\_train\_cat = keras.utils.to\_categorical(y\_train, 10)

y\_test\_cat = keras.utils.to\_categorical(y\_test, 10)

Параметры компиляции и обучения мы определим стандартным образом:

model.compile(optimizer='adam',

             loss='categorical\_crossentropy',

             metrics=['accuracy'])

his = model.fit(x\_train, y\_train\_cat, batch\_size=32, epochs=5, validation\_split=0.2)

Точность классификации для обучающего множества и валидации составила 99%. Для проверки точности на тестовом множестве используется следующая функция:

model.evaluate(x\_test, y\_test\_cat)

Здесь тоже имеем значение 99%.

# Описание работы программы

## Этапы работы программы

### Часть 1. Работа с датасетом cifar-10

1. Загрузка и предобработка датасета cifar-10.

1.1. Нормализация значений.

1.2. Конвертация входных данных в категориальные.

1. Создание модели свёрточной нейронной сети.

2.1. Задание количества слоёв и их параметров.

2.2. «Вытягивание» полученного тензора.

2.3. Задание параметров входного и выходного слоёв для нейронной сети.

2.3. Вывод общей информации о построенной модели.

1. Компиляция и обучение модели до получения наилучшего результата.

3.1. Компиляция модели.

3.2. Обучение модели с сохранением истории обучения.

1. Вывод и визуализация результатов обучения.

4.1. Визуализация функции потерь (loss) и метрики точности (accuracy) для обучающего и тестового множеств в зависимости от эпохи обучения.

4.2. Вывод полученных метрик для обучающего множества на экран (precision, recall, f1-score, confusion matrix).

4.3. Вывод полученных метрик для тестового множества на экран (precision, recall, f1-score, confusion matrix).

1. Сохранение модели в файл.

### Часть 2. Работа с датасетом cifar-100

1. Загрузка и предобработка датасетов cifar-10 и cifar-100.

1.1. Загрузка датасета cifar-10.

1.2. Загрузка датасета cifar-100.

1.3. Объединение данных датасета cifar-10 с датасетом cifar-100 (по вариантам).

1.4. «Перевод» данных в тип numpy array.

1.5. Нормализация значений.

1.6. Конвертация входных данных в категориальные.

1. Загрузка и проверка модели свёрточной нейронной сети.

2.1. Загрузка сохранённой модели cifar-10.

2.2. Проверка корректности загруженной модели.

1. Компиляция и обучение модели до получения наилучшего результата.

3.1. Компиляция модели.

3.2. Обучение модели с сохранением истории обучения.

1. Вывод и визуализация результатов обучения.

4.1. Визуализация функции потерь (loss) и метрики точности (accuracy) для обучающего и тестового множеств в зависимости от эпохи обучения.

4.2. Вывод полученных метрик для обучающего множества на экран (precision, recall, f1-score, confusion matrix).

4.3. Вывод полученных метрик для тестового множества на экран (precision, recall, f1-score, confusion matrix).

## Описание параметров свёрточной нейронной сети

В данной работе используется свёрточная нейронная сеть с шестью свёрточными слоями. Рассмотрим подробнее каждый из слоёв.

1. Первый слой (двумерный свёрточный слой)
   * Количество фильтров: 64
   * Размер ядра: (3, 3)
   * Формат входных данных: (32, 32, 3)
   * Функция активации: «relu»
2. Второй слой (укрупнение масштабов признаков)
   * Размер окна: (2, 2)
3. Третий слой (двумерный свёрточный слой)
   * Количество фильтров: 128
   * Размер ядра: (3, 3)
   * Функция активации: «relu»
4. Четвёртый слой (укрупнение масштабов признаков)
   * Размер окна: (2, 2)
5. Первый слой (двумерный свёрточный слой)
   * Количество фильтров: 128
   * Размер ядра: (3, 3)
   * Функция активации: «relu»
6. Шестой слой (укрупнение масштабов признаков)
   * Размер окна: (2, 2)

Далее происходит вытягивание полученного тензора в единый вектор, которая затем подаётся на полносвязную сеть из 512 нейронов и 10 нейронов выходного слоя (с функциями активации «relu» и «softmax» соответственно).

## Результаты работы программы

Приведём визуализацию датасета cifar-10, чтобы убедиться, что датасет загружен корректно (рис. 5).

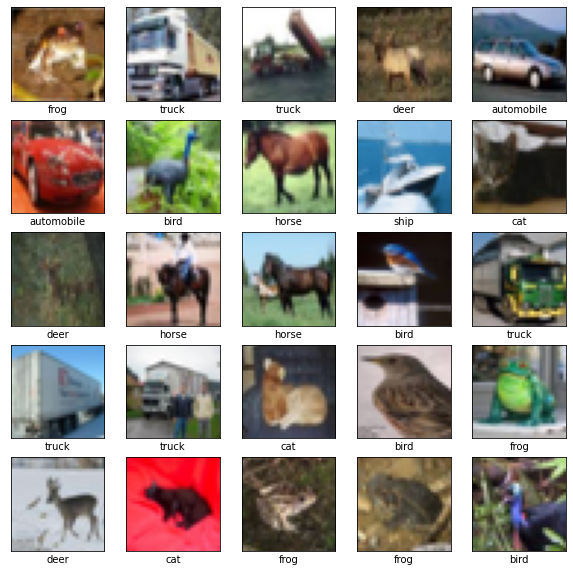


Рис. . Визуализация датасета cifar-10

Информация по созданной модели свёрточной нейронной сети представлена на рисунке 6.

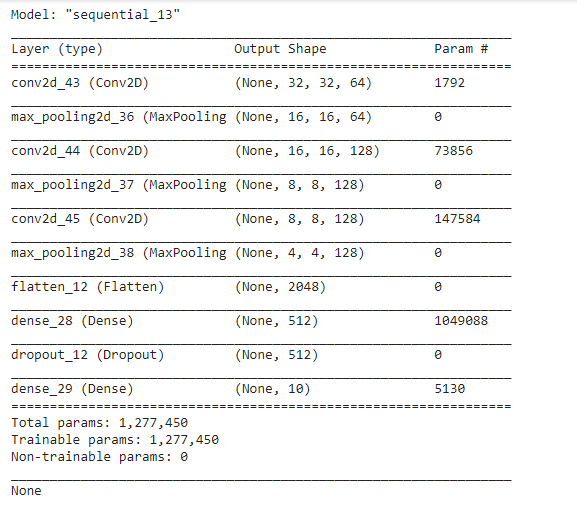


Рис. . Информация по модели свёрточной нейронной сети

Визуализация зависимости значений потерь (loss) и метрики точности от эпохи обучения представлена на рисунке 7.

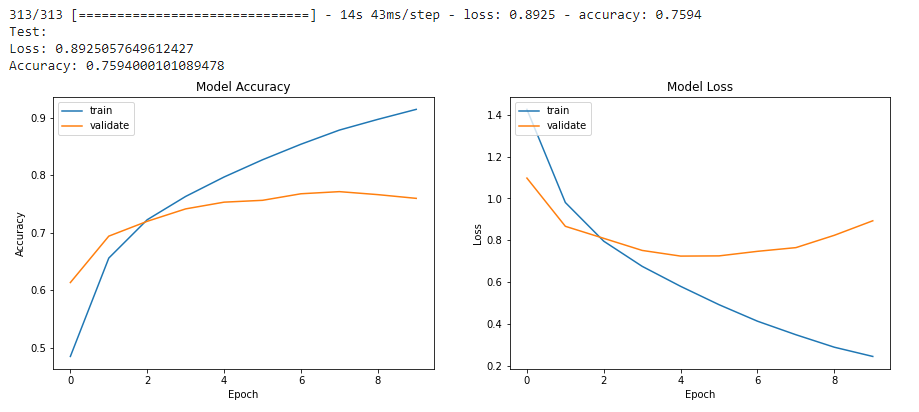


Рис. . Зависимость значений потерь (loss) и метрики точности (accuracy) от эпохи обучения

Значения метрик precision, recall и f1-score, а также матрица ошибок (confusion matrix) для обучающего множества представлена на рисунках 8-9.

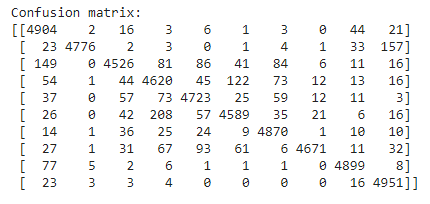


Рис. . Матрица ошибок (confusion matrix) для обучающего множества

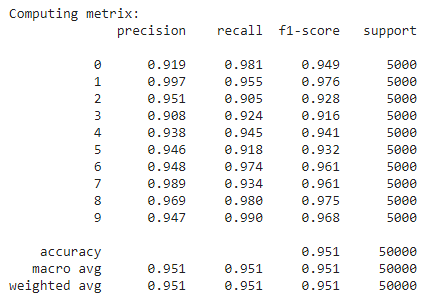


Рис. . Метрики precision, recall и f1-score для обучающего множества

Значения метрик precision, recall и f1-score, а также матрица ошибок (confusion matrix) для тестового множества представлена на рисунках 10-11.

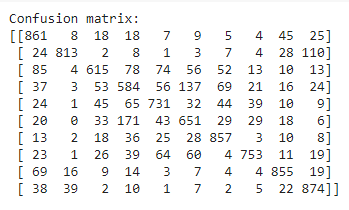


Рис. . Матрица ошибок (confusion matrix) для тестового множества

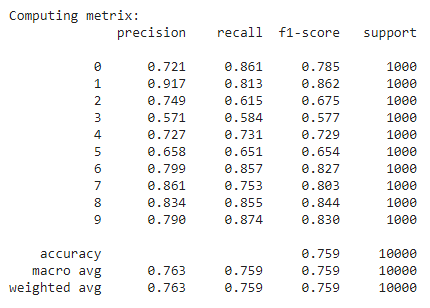


Рис. . Метрики precision, recall и f1-score для тестового множества

Информация по изменённой модели свёрточной нейронной сети для работы с датасетами cifar-10 и cifar-100 представлена на рисунке 12.

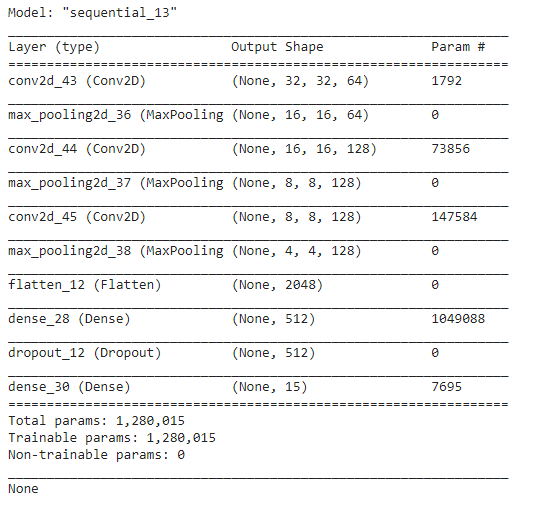


Рис. . Информация по изменённой модели свёрточной нейронной сети

Визуализация зависимости значений потерь (loss) и метрики точности от эпохи обучения представлена на рисунке 13.

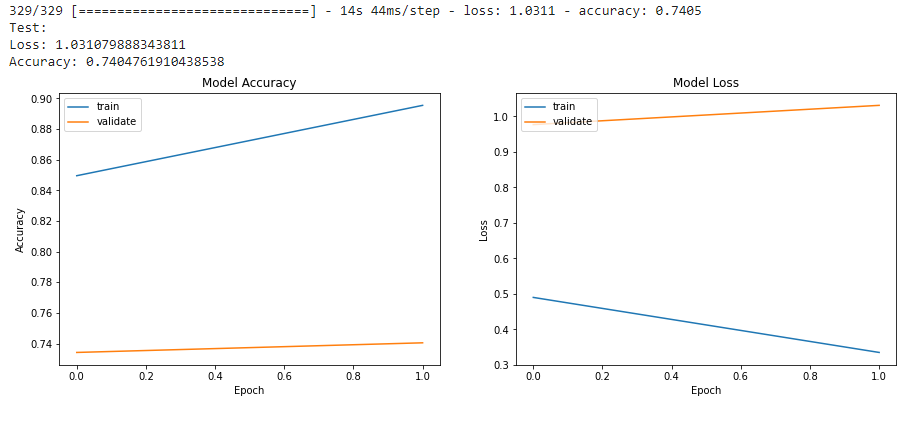


Рис. 13. Зависимость значений потерь (loss) и метрики точности (accuracy) от эпохи обучения

Значения метрик precision, recall и f1-score, а также матрица ошибок (confusion matrix) для обучающего множества представлена на рисунках 14-15.

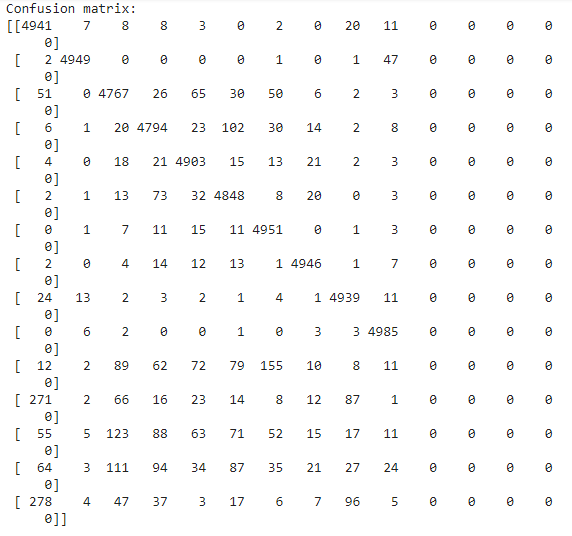


Рис. 14. Матрица ошибок (confusion matrix) для обучающего множества

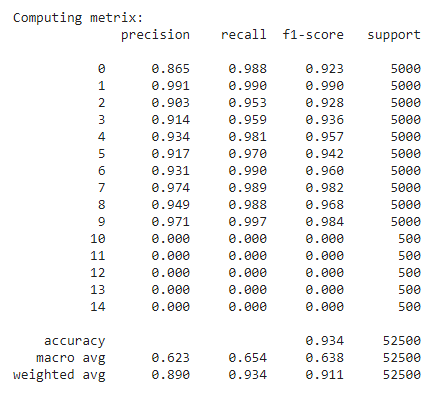


Рис. 15. Метрики precision, recall и f1-score для обучающего множества

Значения метрик precision, recall и f1-score, а также матрица ошибок (confusion matrix) для тестового множества представлена на рисунках 16-17.

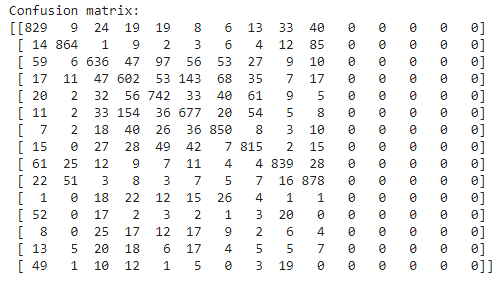


Рис. 16. Матрица ошибок (confusion matrix) для тестового множества

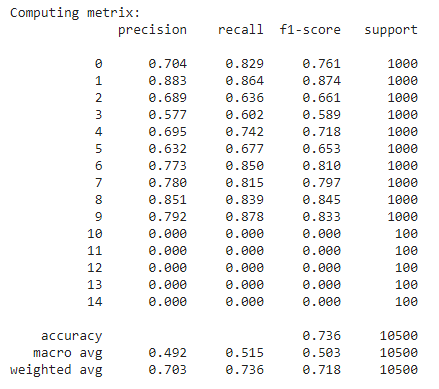


Рис. 17. Метрики precision, recall и f1-score для тестового множества

Исходя из аналогичных реализованных моделей свёрточной нейронной сети, протестированных на датасете cifar-10 [2], можно сделать вывод о том, что полученные в данной работе результаты классификации с достаточной степенью точности можно назвать примерно совпадающими с результатами, опубликованными в литературных источниках.

# Список используемой литературы

1. Про программирование – Делаем свёрточную нейронную сеть в Keras. URL: <https://proproprogs.ru/neural_network/delaem-svertochnuyu-neyronnuyu-set-v-keras> (дата обращения: 16.01.21).
2. Nie Jinliang. CIFAR10 IMAGE CLASSIFICATION BASED ON RESNET. Russia, St. Petersburg, Peter the Great St. Petersburg Polytechnic University.