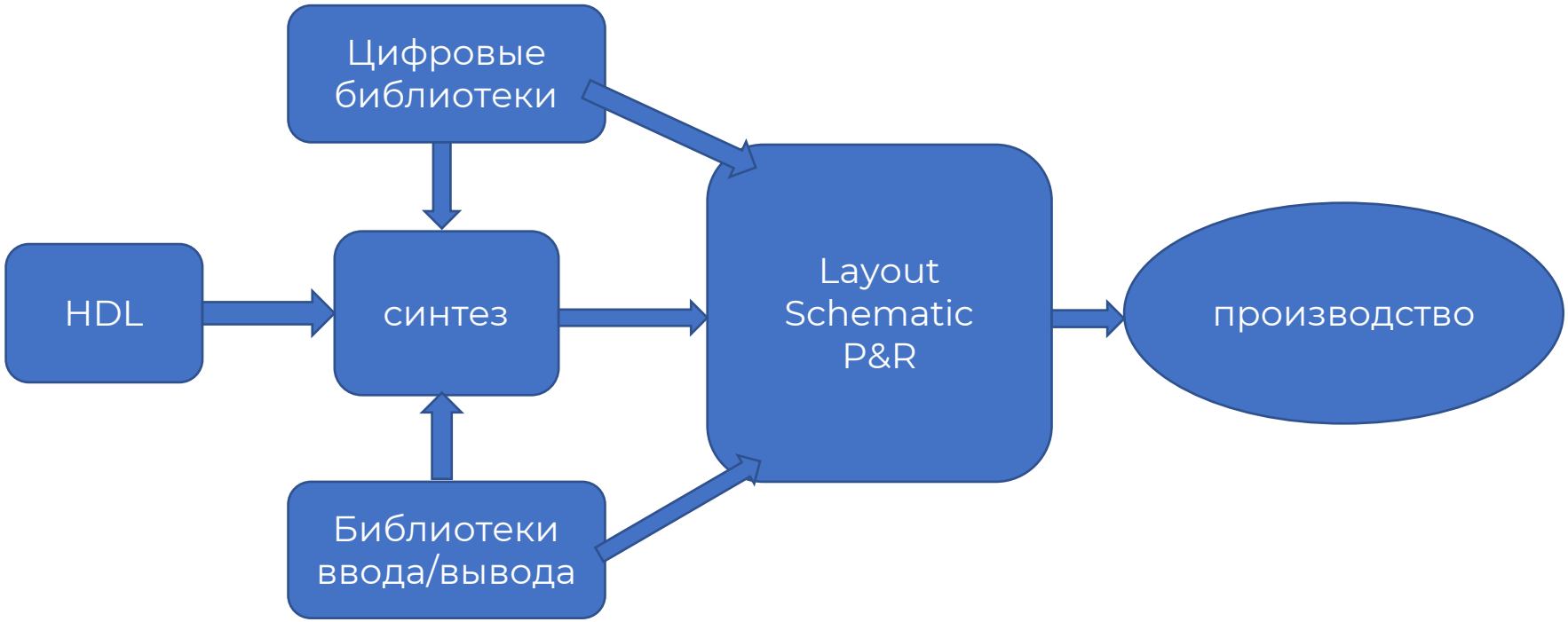
ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана

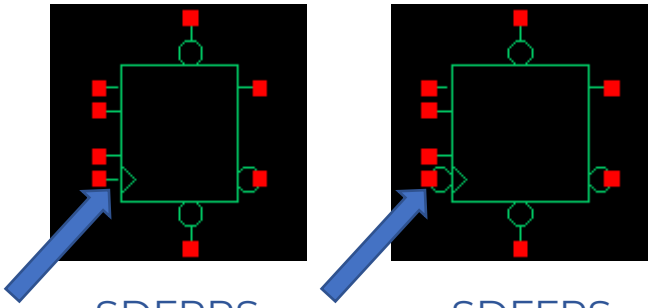
Выпускная квалификационная работа по курсу «Data Science». Разработка комплекса приложений для проверки графического изображения символа в библиотеках цифровых ячеек.

Слушатель: Мягков Олег Вячеславович

Библиотеки цифровых ячеек



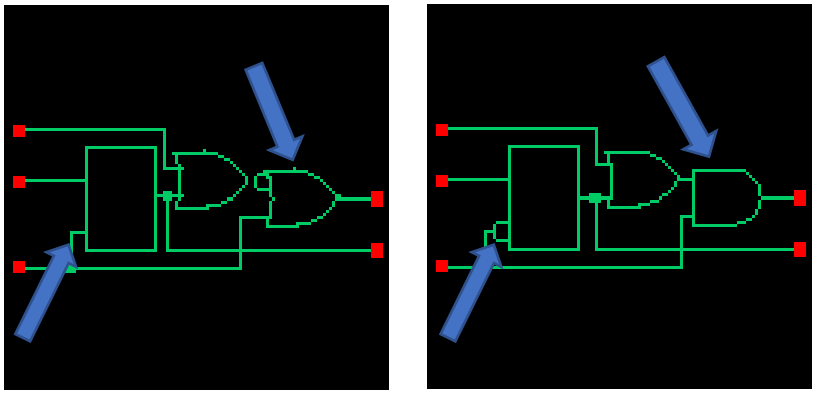
symbol/символ



SDFRRS SDFFRS

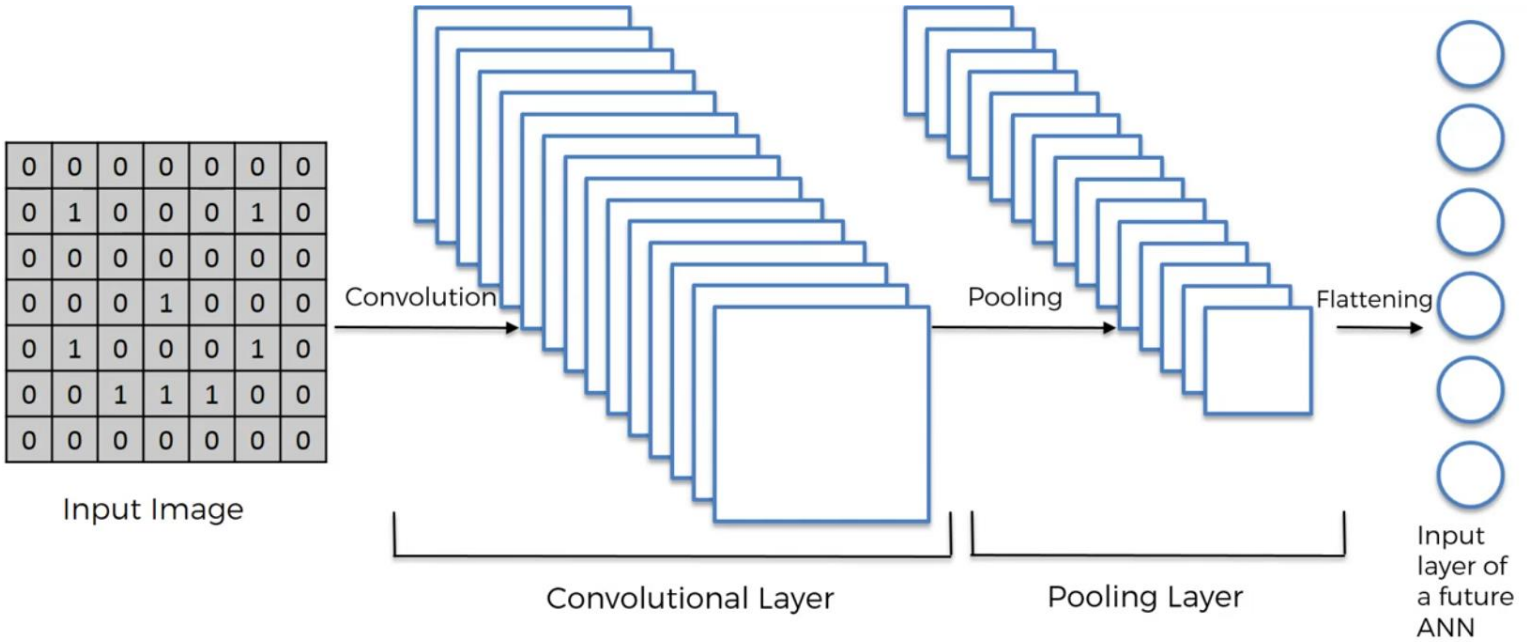
Обратите внимание, разница в символах представленных ячеек минимальна.

Подобные ошибки могут Возникать когда инженер Создает одну ячейку из другой



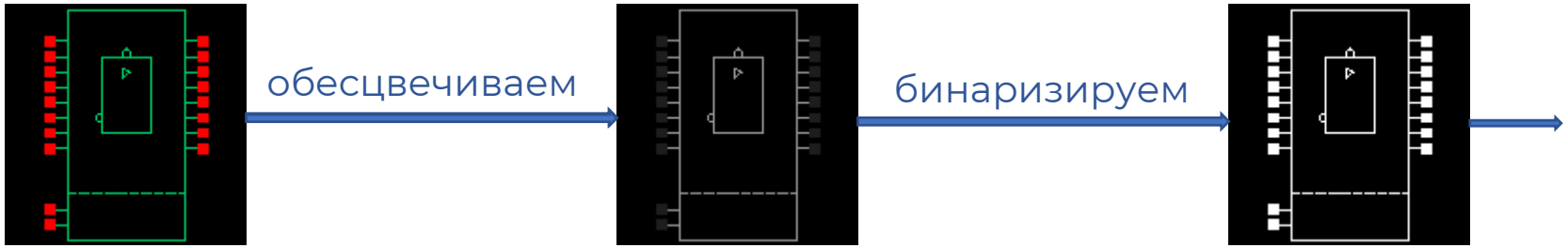
LSOGCN LSOGCP

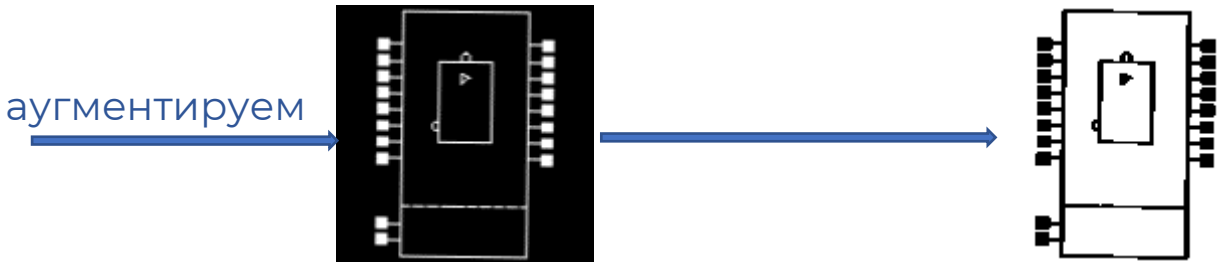
Сверточные нейросети



На данном историческом этапе, сверточные нейросети являются практически безальтернативным выбором для анализа изображений

Подготовка изображений





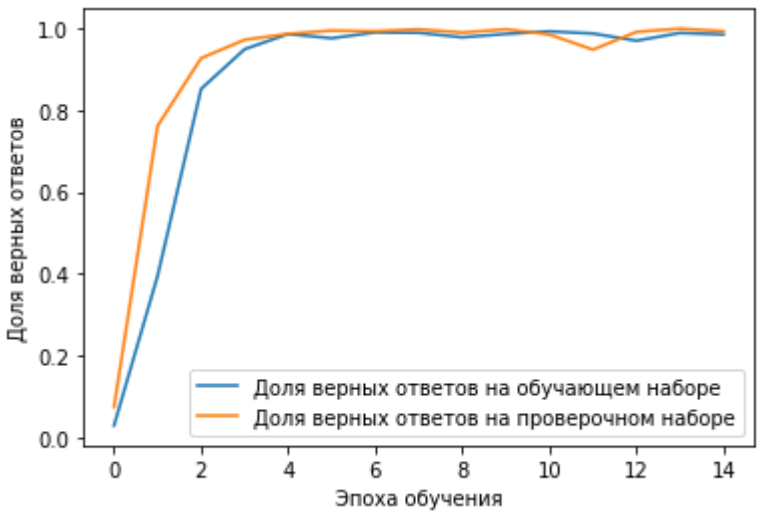
- ✓ Создаем по 30 изображений для каждого класса в тренировочном наборе
- ✓ Создаем по 3 изображения каждого класса в тестовом наборе.

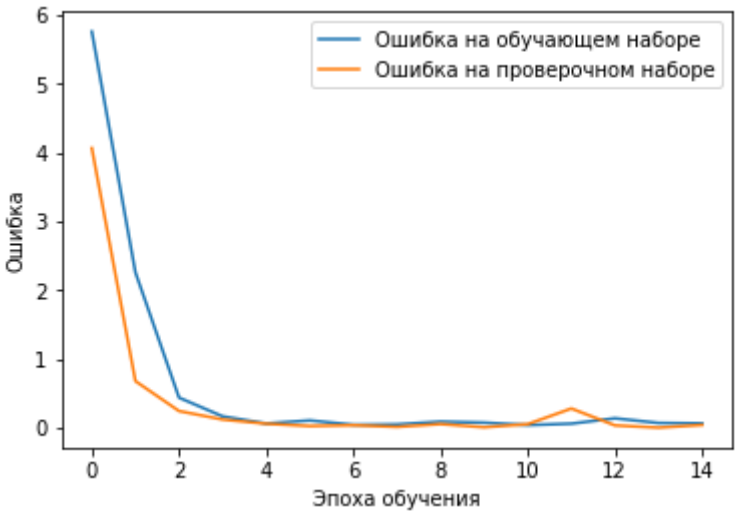
Fruits_360_CNN

В самом начале работы я воспользовался готовой архитектурой нейросети которую нашел в интернете - Fruits_360_CNN.

https://github.com/Horea94/Fruit-Images-Dataset

Именно с этой нейросетью я вступлю в заочное соревнование.

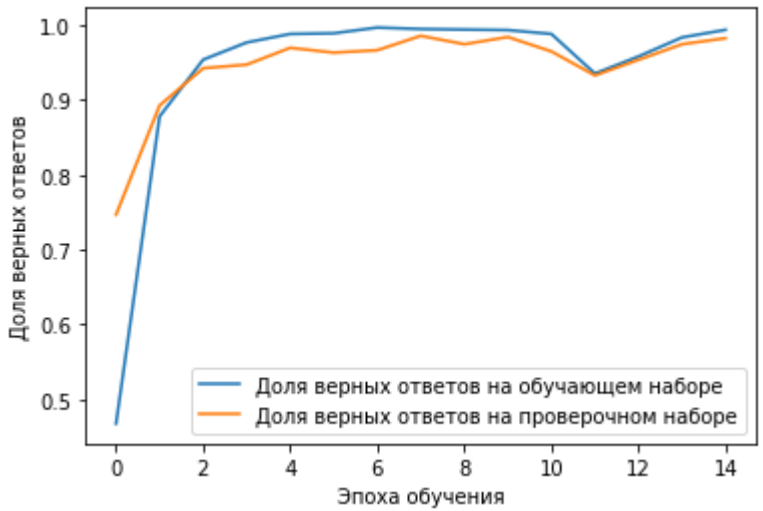


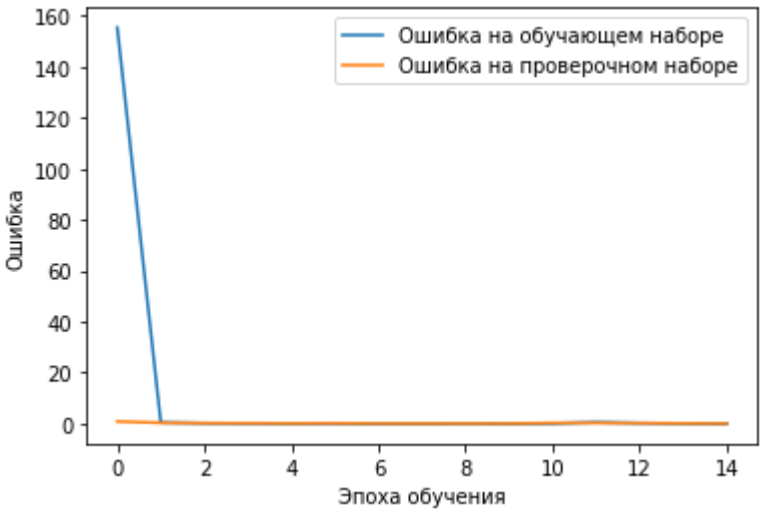


Точность на тестовом наборе 99.04%

Простейшая сверточная нейросеть

```
model_simple.add(Conv2D(16, (5, 5), padding='same',          #Сверточный слой
                 input_shape=(128, 128, 1), activation='relu'))
model_simple.add(MaxPooling2D(pool_size=(2, 2)))             # Слой подвыборки
model_simple.add(Flatten())                                  # Полносвязная часть
model_simple.add(Dense(256, activation='relu'))
model_simple.add(Dense(208, activation='softmax'))           # Выходной слой
```

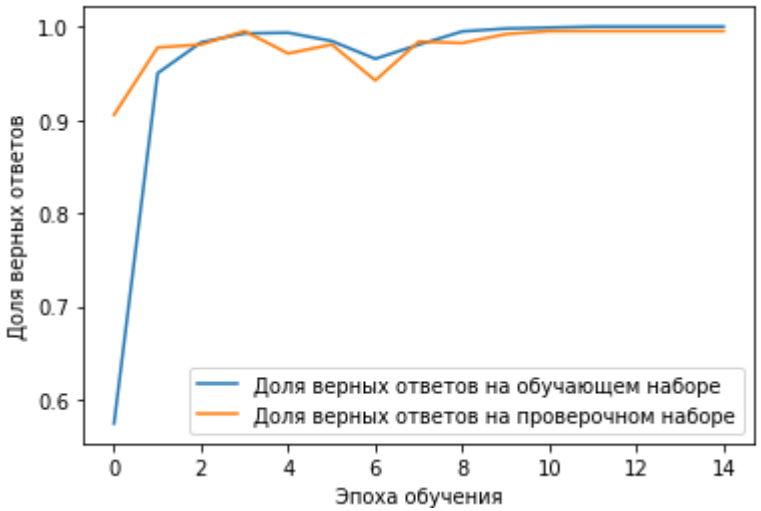


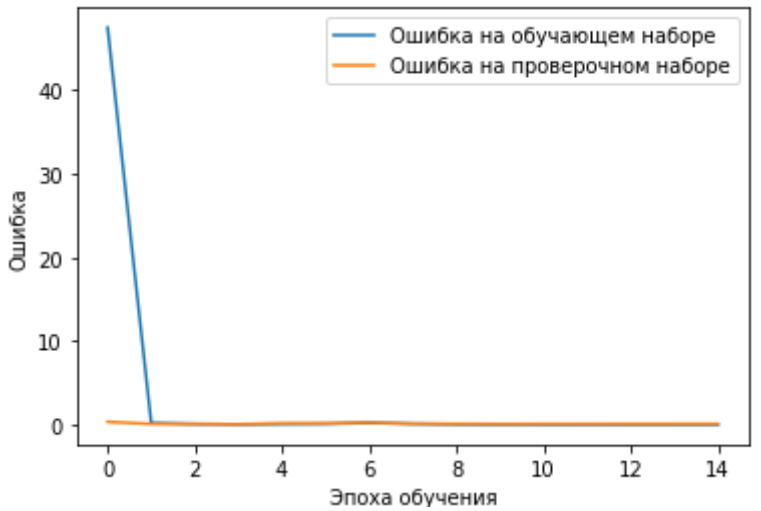


Точность на тестовом наборе 98.88%

Сверточная нейросеть с двумя сверточными слоями

```
model_s16_32.add(Conv2D(16, (5, 5), padding='same',      #первый слой свертки
        input_shape=(128, 128, 1), activation='relu'))
model_s16_32.add(MaxPooling2D(pool_size=(2, 2)))
model_s16_32.add(Conv2D(32, (3, 3), padding='same',      #второй слой свертки
        input_shape=(128, 128, 1), activation='relu'))
model_s16_32.add(MaxPooling2D(pool_size=(2, 2)))
model_s16_32.add(Flatten())                              #полносвязанная часть
model_s16_32.add(Dense(512, activation='relu'))
model_s16_32.add(Dense(208, activation='softmax'))
```

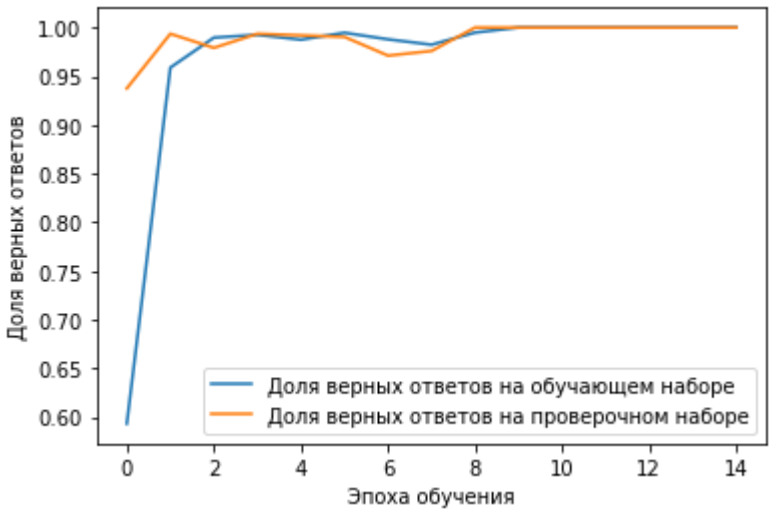


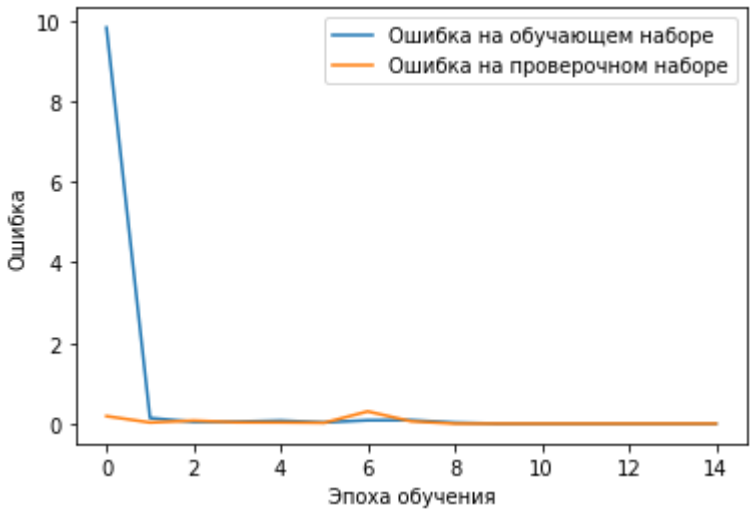


Точность на тестовом наборе 99.68%

Сверточная нейросеть с тремя сверточными слоями

```
model_s16_32_64.add(Conv2D(16, (5, 5), padding='same',      #первый слой свертки
          input_shape=(128, 128, 1), activation='relu'))
model_s16_32_64.add(MaxPooling2D(pool_size=(2, 2)))
model_s16_32_64.add(Conv2D(32, (3, 3), padding='same',      #второй слой свертки
          input_shape=(128, 128, 1), activation='relu'))
model_s16_32_64.add(MaxPooling2D(pool_size=(2, 2)))
model_s16_32_64.add(Conv2D(64, (3, 3), padding='same',      #третий слой свертки
          input_shape=(128, 128, 1), activation='relu'))
model_s16_32_64.add(MaxPooling2D(pool_size=(2, 2)))
model_s16_32_64.add(Flatten())                              #полносвязанная часть
model_s16_32_64.add(Dense(512, activation='relu'))
model_s16_32_64.add(Dense(208, activation='softmax'))
```

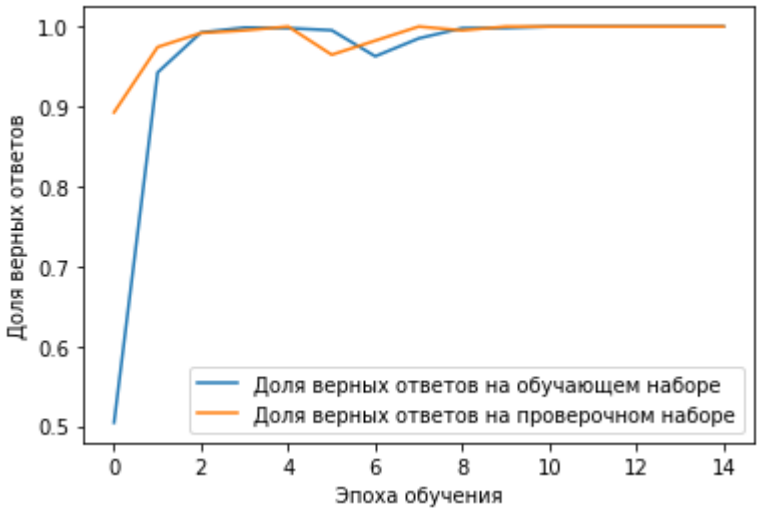


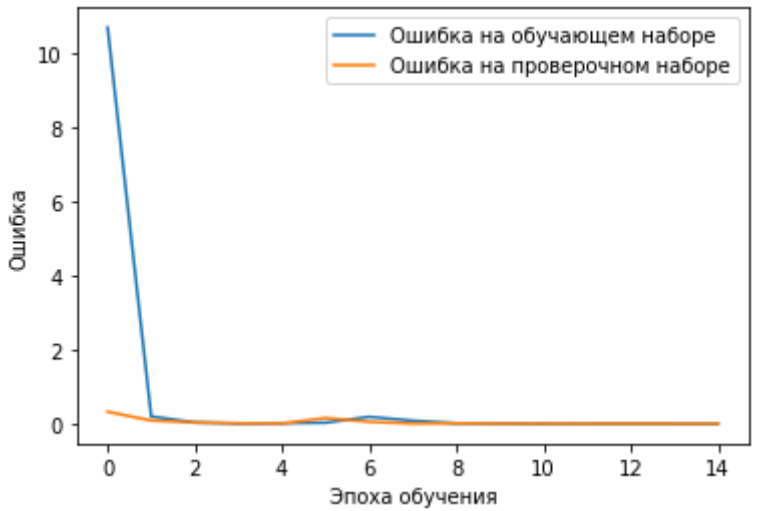


Точность на тестовом наборе 100.00%

Сверточная нейросеть с тремя сверточными слоями и слоем дропаут

```
model_f7_5_3_drop.add(Conv2D(16, (7, 7), padding='same',          # Сверточный слой
        input_shape=(128, 128, 1), activation='relu'))
model_f7_5_3_drop.add(MaxPooling2D(pool_size=(2, 2)))
model_f7_5_3_drop.add(Conv2D(32, (5, 5), padding='same',          # Сверточный слой
        input_shape=(128, 128, 1), activation='relu'))
model_f7_5_3_drop.add(MaxPooling2D(pool_size=(2, 2)))
model_f7_5_3_drop.add(Conv2D(64, (3, 3), padding='same',          # Сверточный слой
        input_shape=(128, 128, 1), activation='relu'))
model_f7_5_3_drop.add(MaxPooling2D(pool_size=(2, 2)))
model_f7_5_3_drop.add(Flatten())                                  #Полносвязная часть
model_f7_5_3_drop.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))                                           #Слой дропаут
model_f7_5_3_drop.add(Dense(208, activation='softmax'))           # Выходной слой, 208
```





Точность на тестовом наборе 100.00%

Сводная таблица результатов обучения нейросетей

Модель	Точность	Модель	Точность
Fruits_360_CNN	99.04%	model_s16_32_64	100.00%
model_simple	98.88%	model_s16_32_64_drop	99.84%
model_s32	98.72%	model_f7_5_3_drop	100.00%
model_s16_32	99.68%	model_f11_7_3_drop	98.88%

- Fruits_360_CNN - 4 слоя свертки, 2 слоя дропаут
- model_simple – 1 слой свертки
- model_s32 – 1 слой свертки но увеличенно количество фильтров
- model_s16_32 – 2 слоя свертки
- model_s16_32_64 – 3 слоя свертки
- model_s16_32_64_drop – 3 слоя свертки и слой дропаут
- model_f7_5_3_drop – 3 слоя свертки и слой дропаут изменен размер фильтров
- model_f11_7_3_drop – 3 слоя свертки и слой дропаут изменен размер фильтров

Набор утилит для проверки символов цифровых библиотек

На основе проведенных исследований был создан комплекс утилит для проверки символов цифровых библиотек.
В состав комплекса входит три утилиты.
Утилиты имеют интерфейс командной строки.

- Утилита ImgPrep_V_1_0_0.py – предназначена для подготовки набора изображений для дальнейшего обучения нейросети.
- Утилита Classificator_V_1_0_0.py – это нейросеть, обучаемая на подготовленном наборе изображений. Результатом работы утилиты является, сохраненная в файл, обученная нейросеть и несколько служебных файлов.
- Утилита check_simbol_pic_V_1_0_0.py – предназначена для проверки символов цифровой библиотеки. Результаты проверки выводятся в терминал и сохраняются в файл

Готов ответить на ваши вопросы

Спасибо за внимание