

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРА-
ЦИИ

Федеральное государственное автономное образовательное учреждение выс-
шего образования

«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информацион-
ные технологии»

Магистерская программа: «Компьютерная графика»

Образовательный курс «Глубокое обучение»

ОТЧЕТ
по лабораторной работе №2

Разработка полностью связанной нейронной сети

Выполнили:
студенты групп 381806-2
и 381806-4М

Мурашов Михаил
Красильников Михаил
Повеликин Ростислав
Земцов Артем

Нижний Новгород
2019

Содержание

Цели	3
Задачи	4
Решаемая задача	5
Классы в датасете	7
Выбор библиотеки	9
Метрика качества решения задачи	9
Тренировочные и тестовые наборы данных	9
Конфигурации нейронных сетей	10
Кол-во слоев: 2	22
Кол-во нейронов: [512 - 256]	22
Функция активации: Tanh	22
Кол-во слоев: 2	23
Кол-во нейронов: [1024 - 512]	23
Функция активации: ReLu	23
Кол-во слоев: 2	24
Кол-во нейронов: [1024 - 512]	24
Функция активации: SeLu	24
Кол-во слоев: 2	25
Кол-во нейронов: [1024 - 512]	25
Функция активации: Tanh	25
Разработанные программы/скрипты	26
Результаты экспериментов	26
Анализ результатов	28

Цели

Цель настоящей работы состоит в том, чтобы получить базовые навыки работы с одной из библиотек глубокого обучения (Caffe, Torch, TensorFlow, MXNet или какая-либо другая библиотека на выбор студента) на примере полностью связанных нейронных сетей.

Задачи

Выполнение практической работы предполагает решение *следующих задач*:

1. Выбор библиотеки для выполнения практических работ курса.
2. Установка выбранной библиотеки на кластере (параметры аутентификации и инструкция по работе с кластером выложена в отдельной задаче в системе redmine).
3. Проверка корректности установки библиотеки. Разработка и запуск тестового примера сети, соответствующей логистической регрессии, для решения задачи классификации рукописных цифр набора данных MNIST (пример разобран в лекционных материалах).
4. Выбор практической задачи компьютерного зрения для выполнения практических работ.
5. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается выбранной библиотекой.
6. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой.
7. Обучение разработанных глубоких моделей.
8. Тестирование обученных глубоких моделей.
9. Публикация разработанных программ/скриптов в репозитории на GitHub.
10. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Решаемая задача

Была выбрана задача классификации животных. Количество классов - 37. Датасет: <https://www.kaggle.com/tanlikesmath/the-oxfordiiit-pet-dataset>

Архив обучающего набора имеет следующую структуру: директория с изображениями, имена которых представляют собой конкатенацию названия класса и порядковый номер изображения в классе - Class_<Number>.jpg. Таким образом, был написан python-скрипт, создания csv файла с двумя столбцами: наименование класса, а также относительный путь до изображения. На каждый класс приблизительно 200 изображений.

Формат изображения:

- Каждое изображение содержит животное
- Изображения хранятся в формате JPEG
- Размеры изображения варьируются от 137x103 до 3264x2448 пикселей
- Изображения необязательно имеют квадратную форму
- Животное не обязательно находится в центре изображения

В процессе выполнения лабораторной работы было выполнено множество различных комбинаций предобработки изображений. Ниже приведена таблица используемых параметров класса ImageDataGenerator.

Параметр	1	2	3	4	5	6
rescale	1/255	1/255	1/255	1/255	1/255	1/255
rotation range	2	1	4	0.5	2	1
width shift	1	2	5	0.3	0.1	-
height shift	-	1	3	2	1	1
zoom range	2	0.05	0.1	0.5	0.7	0.5
horizontal flip	-	-	true	true	true	-
vertical flip	true	-	-	-	-	true

Класс	Кол-во
japanese_chin	182
Birman	182
wheaten_terrier	182
english_cocker_spaniel	182
beagle	182
american_bulldog	182
boxer	182
samoyed	182
english_setter	182
havanese	182
leonberger	182
keeshond	182
newfoundland	182
basset_hound	182
great_pyrenees	182
german_shorthaired	182
scottish_terrier	181
shiba_inu	181
British_Shorthair	179
Persian	177
Maine_Coon	175
Siamese	175
Russian_Blue	175
staffordshire_bull_terrier	173
Sphynx	172
Ragdoll	168
yorkshire_terrier	167
Bengal	167

Класс	Кол-во
Abyssinian	167
american_pit_bull_terrrier	167
pug	160
Pomeranian	159
miniature_pinscher	157
chihuahua	155
Bombay	147
Egyptian_Mau	143

Классы в датасете

Пример изображений



Выбор библиотеки

Для выполнения лабораторных работ была выбрана библиотека Keras для языка программирования Python.

На этапе проверки корректности установки библиотеки выполнена разработка и запуск тестового примера сети для решения задачи классификации рукописных цифр набора данных MNIST. Достигнута точность 0.89.

Метрика качества решения задачи

В качестве метрики точности решения используется отношение правильно классифицированных знаков ко всем знакам в тестовой выборке:

$$Accuracy = \frac{Correct\ answers\ count}{Images\ count}$$

Тренировочные и тестовые наборы данных

В качестве тренировочной выборки используем 5345 изображений различных животных.

В качестве тестовой выборки используем 629 изображений животных.

Конфигурации нейронных сетей

В данной работе были рассмотрены шесть конфигураций полностью связанных нейронных сетей с 3-мя скрытыми слоями.

Активационная функция на слоях выбирается из следующих:

- $\tanh, f = \frac{e^s - e^{-s}}{e^s + e^{-s}}$
- $\text{selu}, (x) = \lambda (?)$
- $\text{relu}, f = \max(x, 0)$

На выходном слое:

- $\text{softmax}, f = \frac{e^{s_j}}{\sum_{j=1}^n e^{s_j}}$

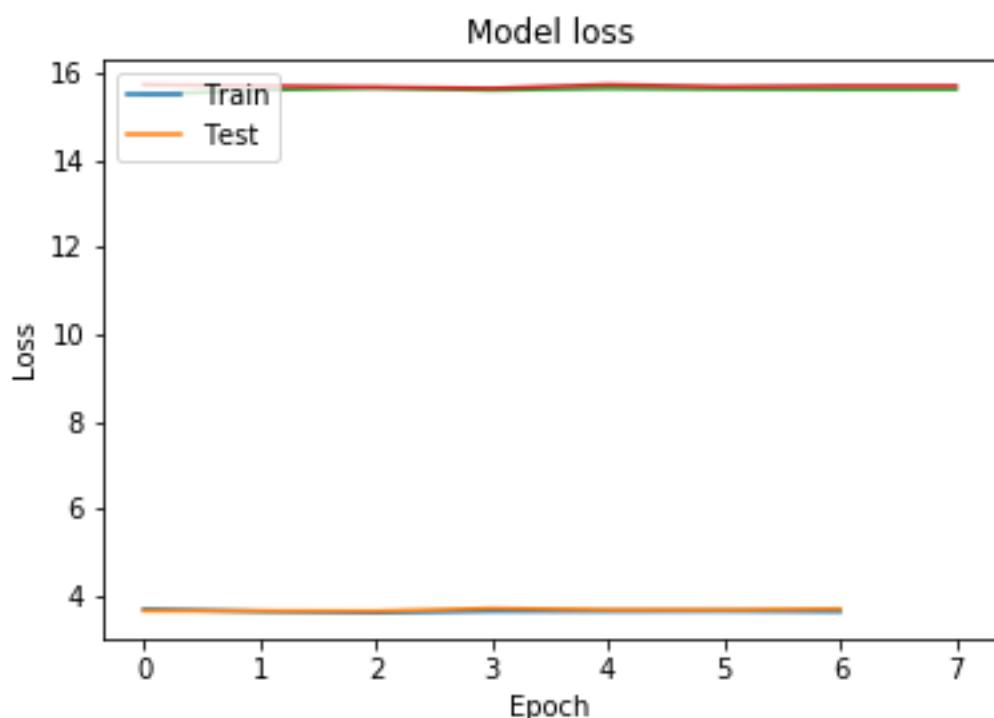
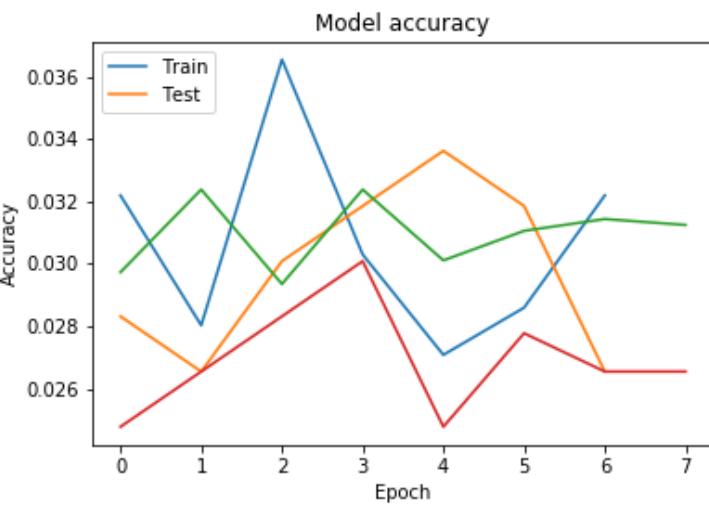
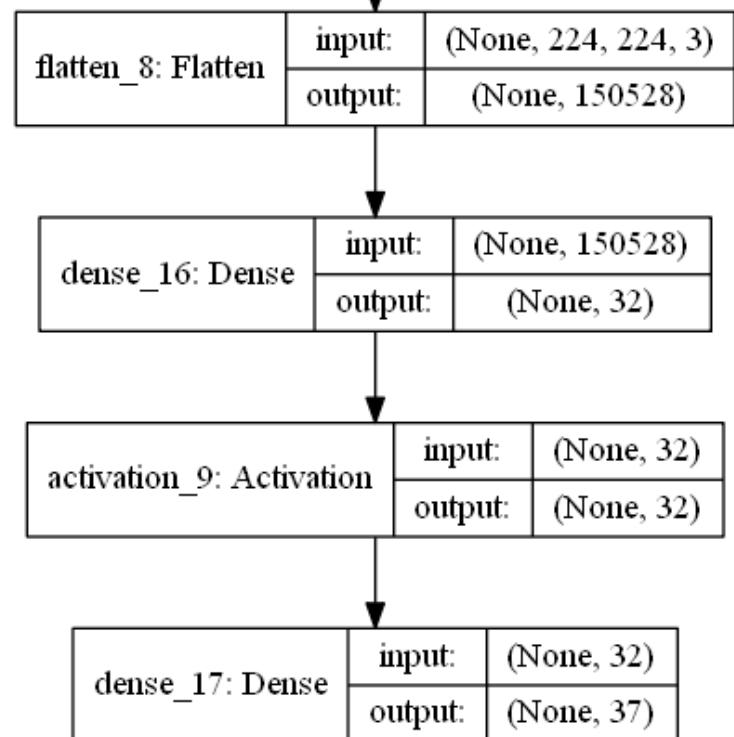
Для обучения был использован модифицированный метод градиентного спуска. Adam — adaptive moment estimation, оптимизационный алгоритм. Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков. Его реализация имеется в библиотеке Keras. Для начальной инициализации весов был использован Xavier Initialization

Кол-во слоев: 1

Кол-во нейронов: 32

Функция активации: ReLu

1716176181704

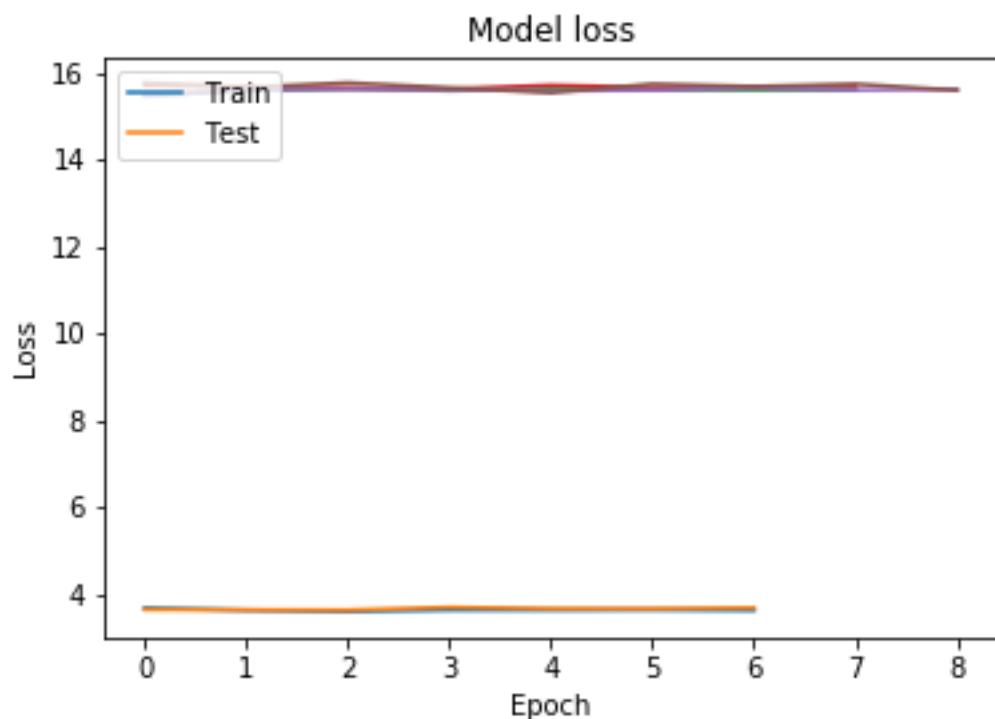
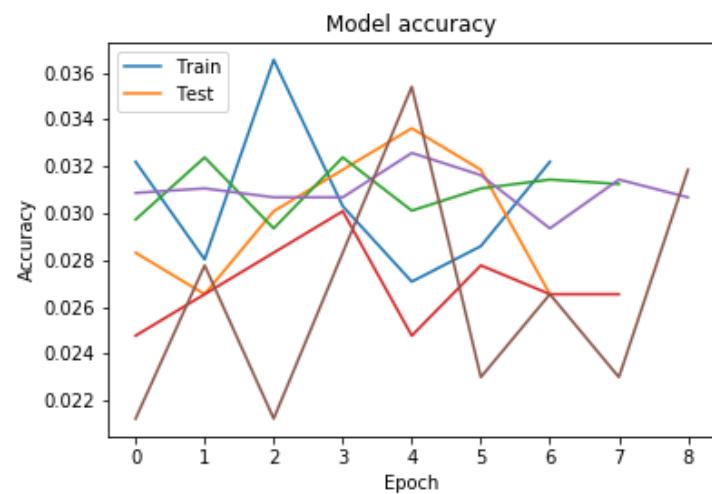
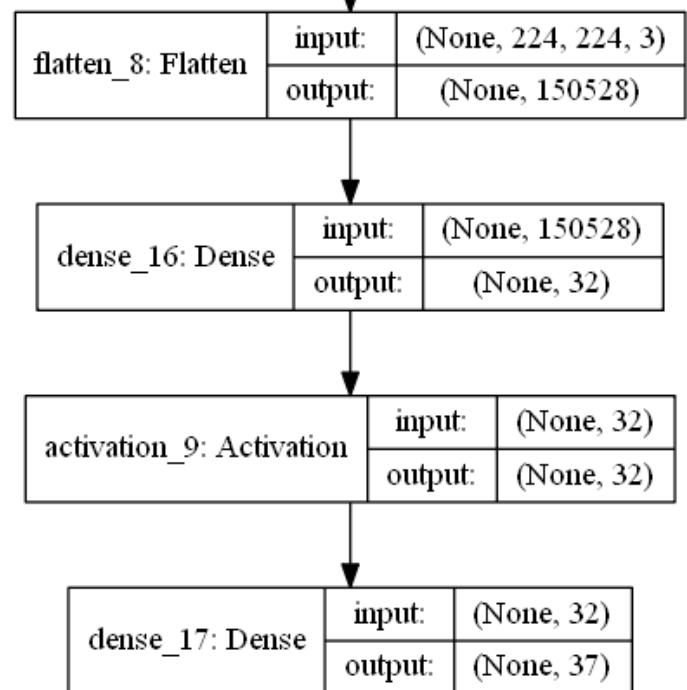


Кол-во слоев: 1

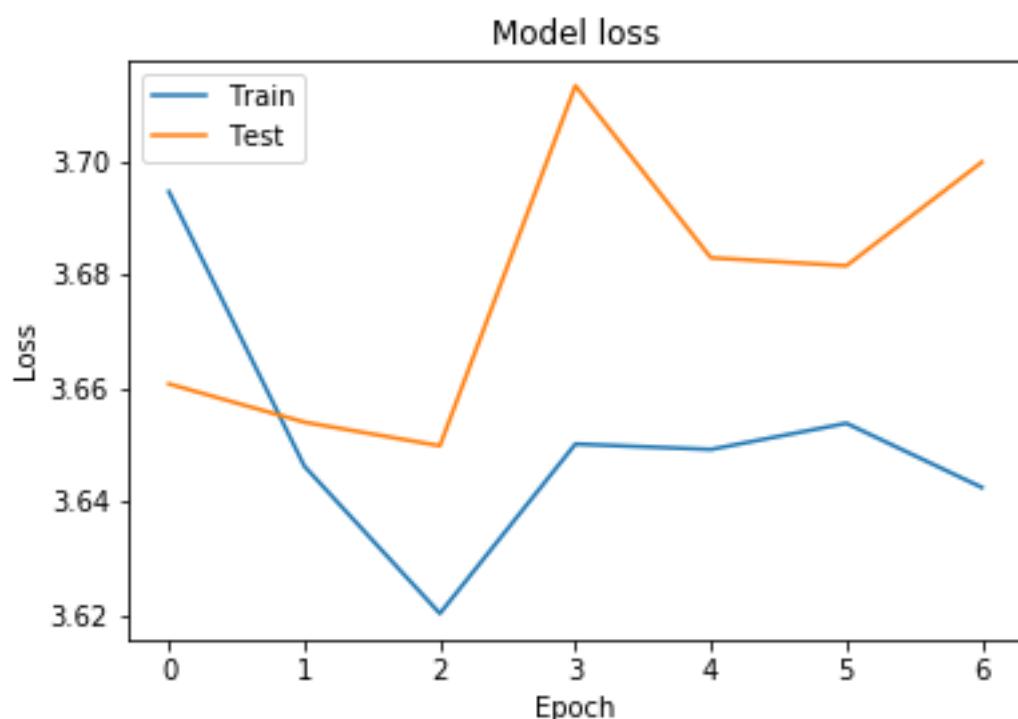
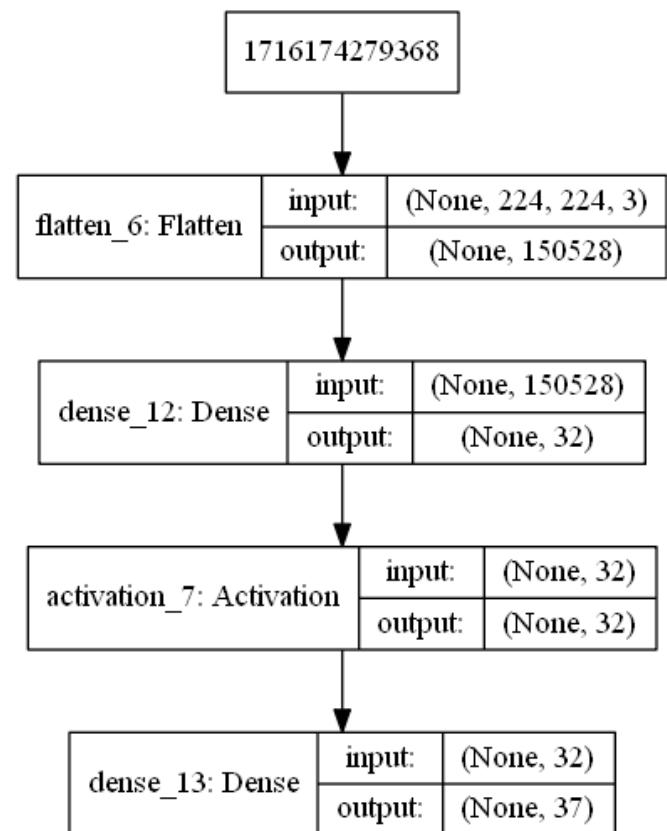
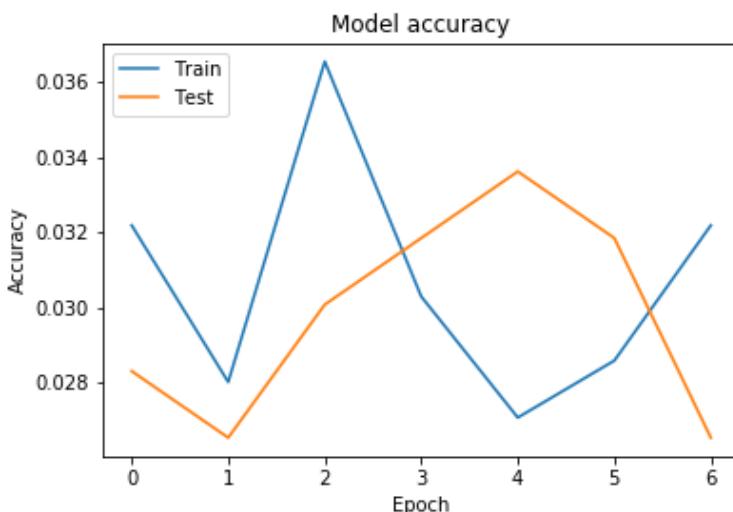
Кол-во нейронов: 32

Функция активации: SeLu

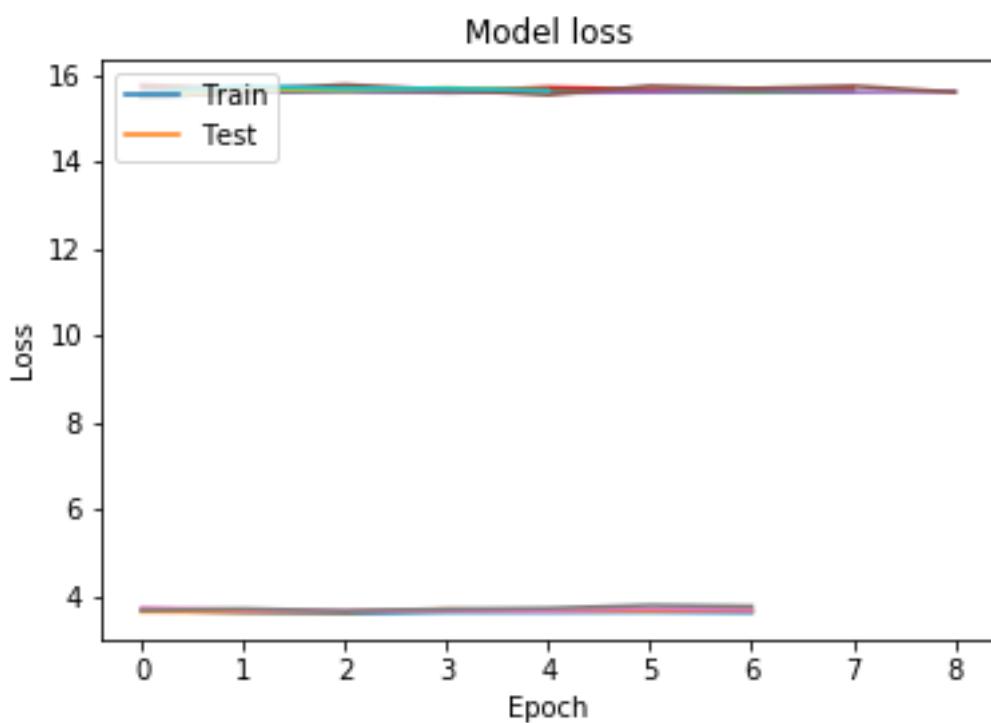
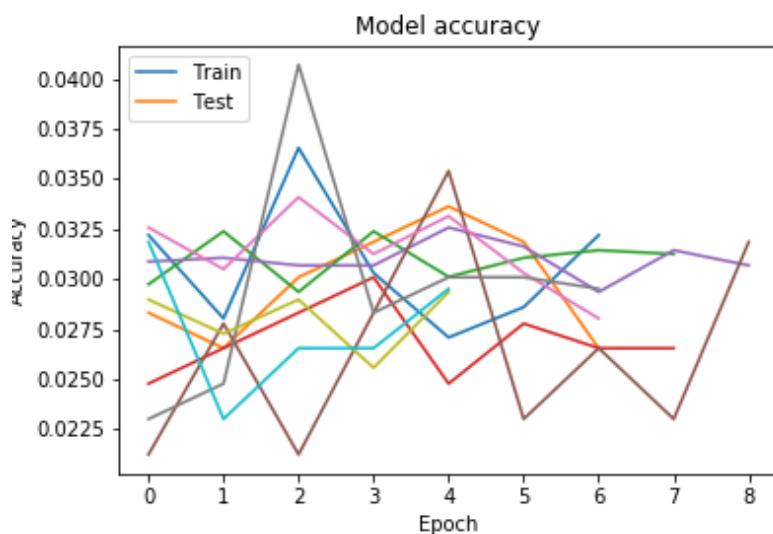
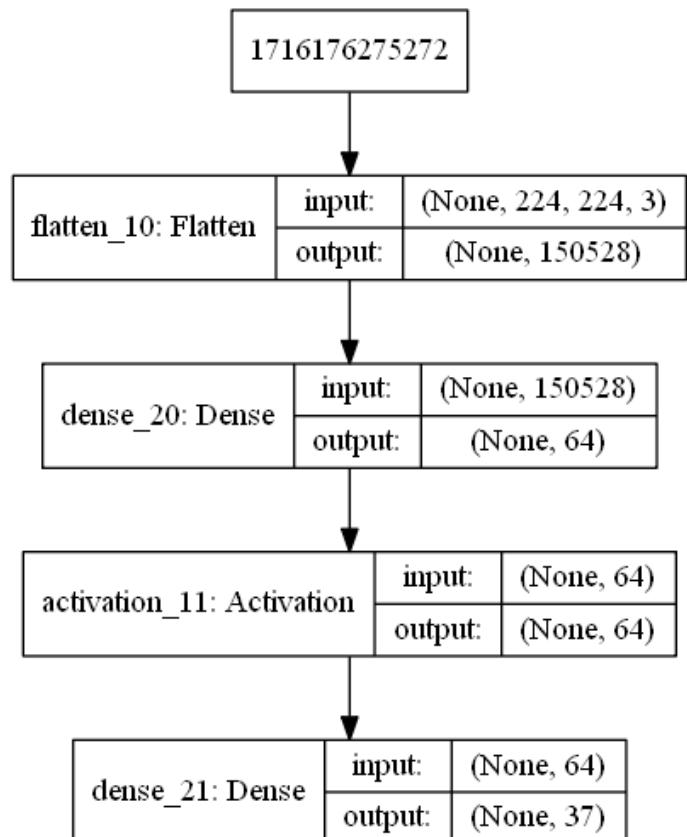
1716176181704



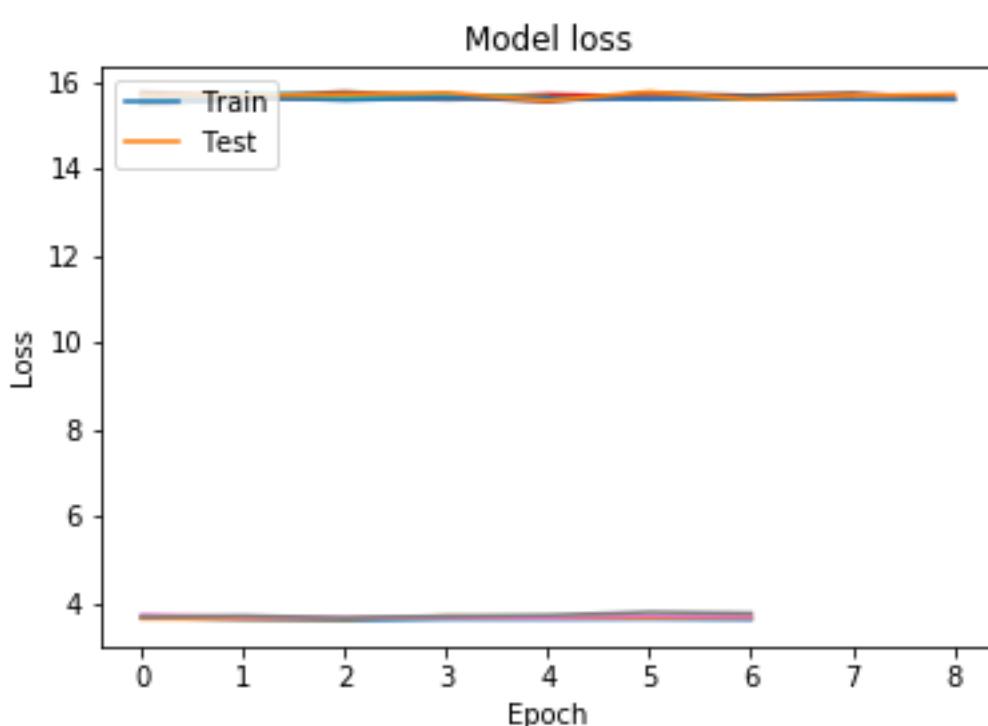
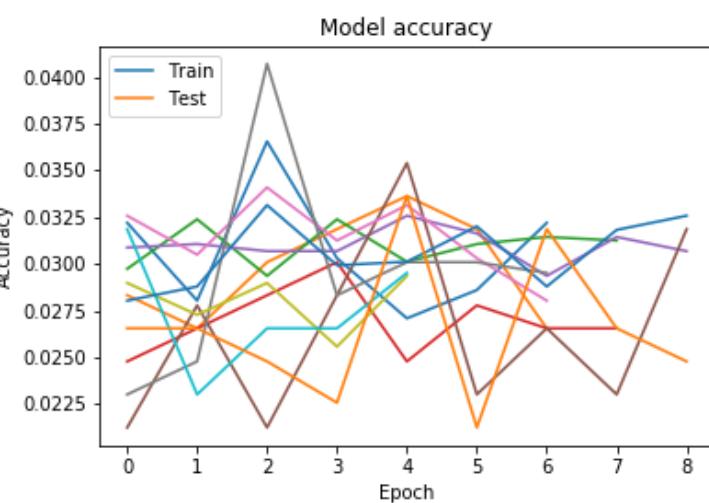
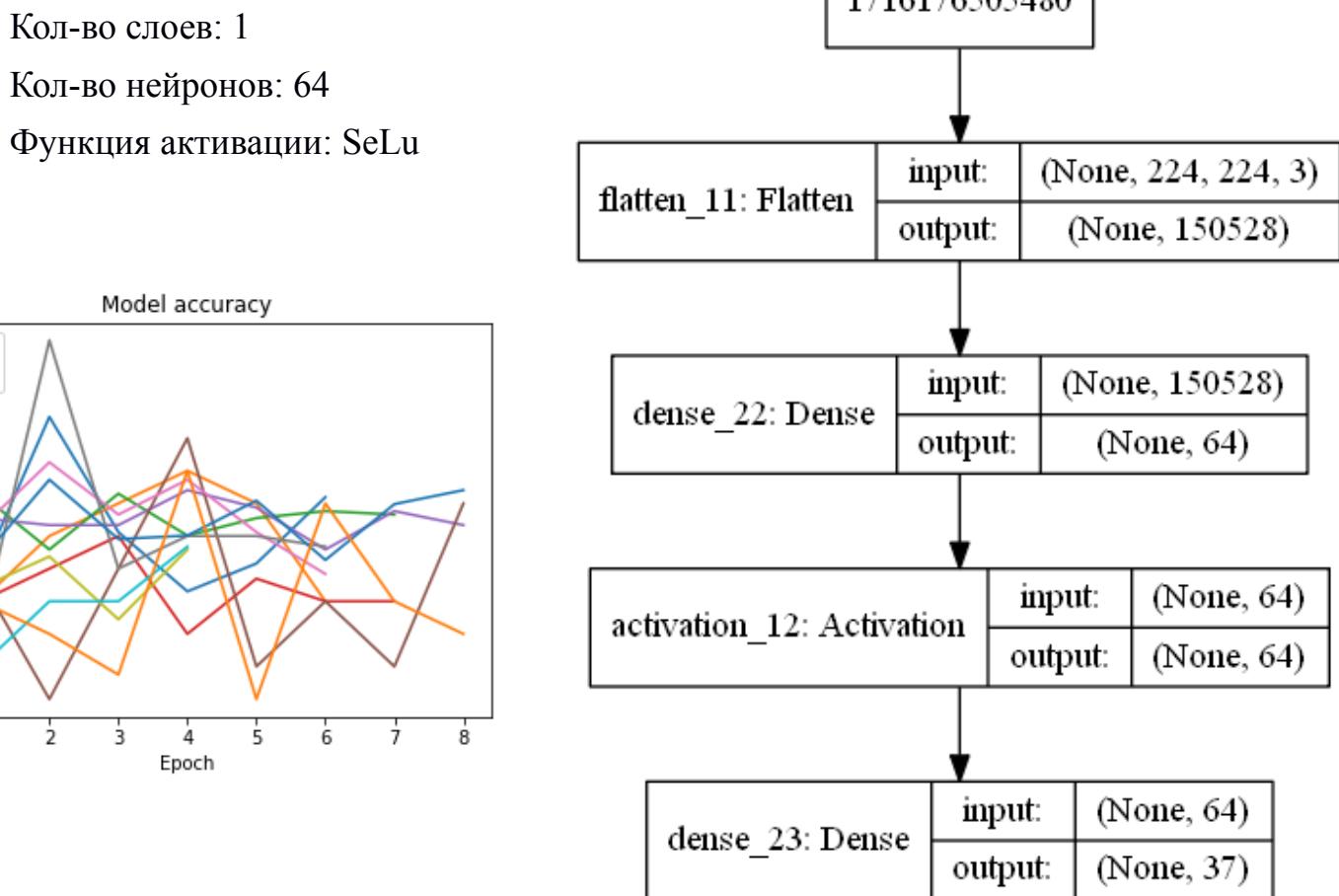
Кол-во слоев: 1
Кол-во нейронов: 32
Функция активации: Tanh



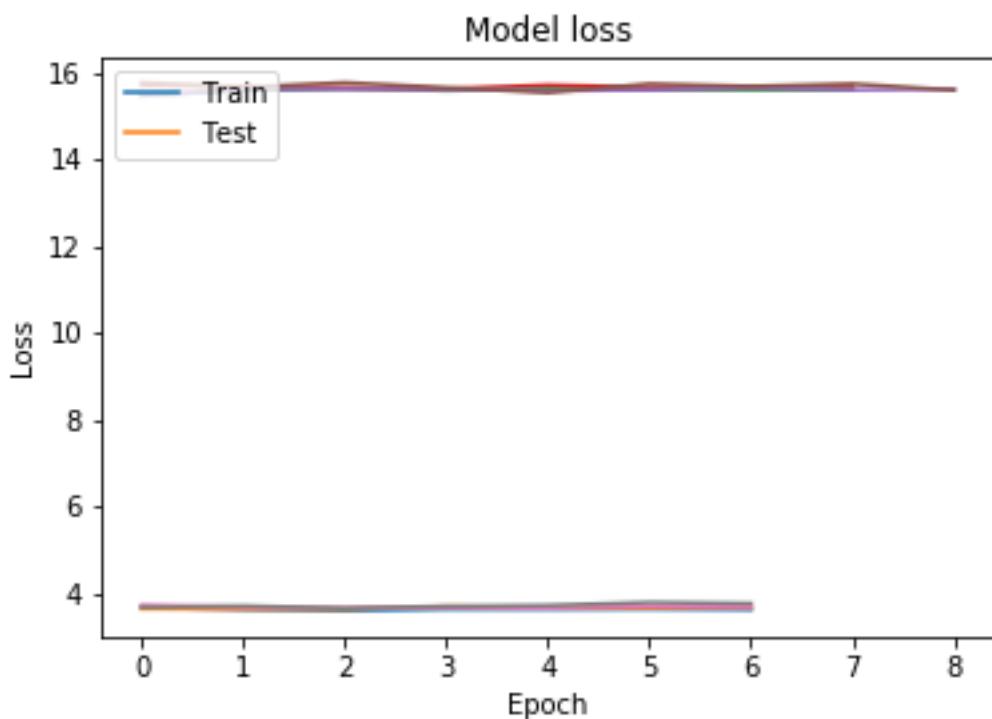
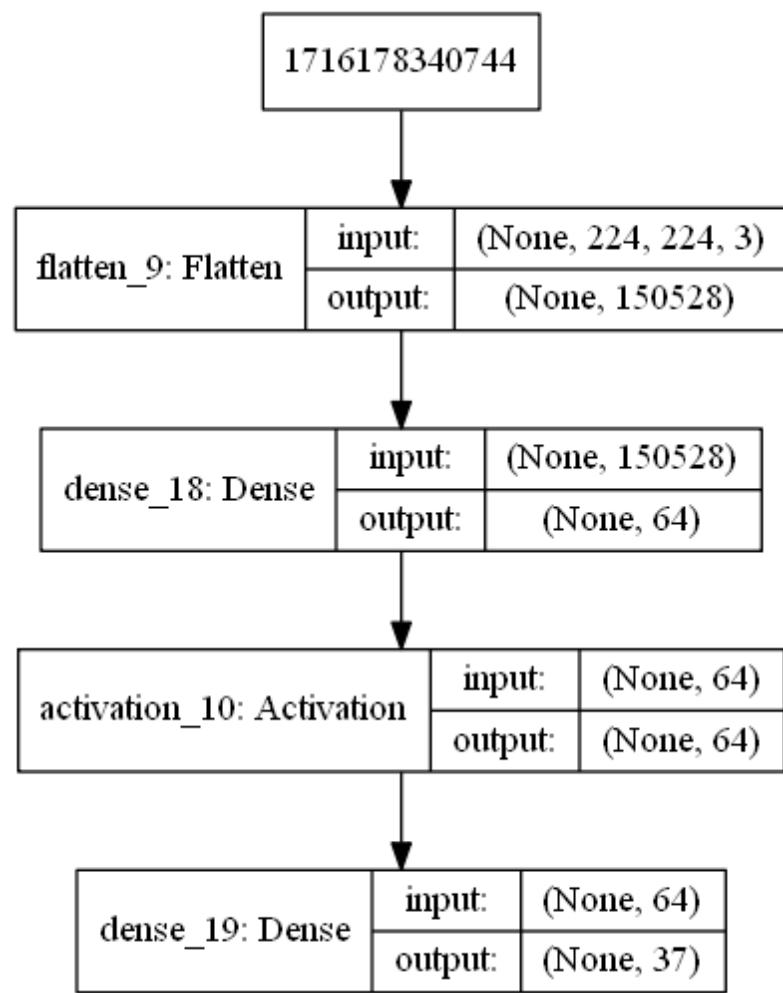
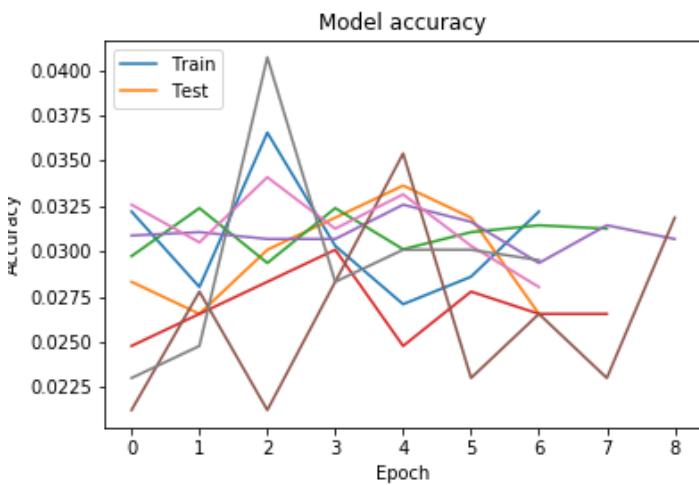
Кол-во слоев: 1
Кол-во нейронов: 64
Функция активации: ReLu



Кол-во слоев: 1
Кол-во нейронов: 64
Функция активации: SeLu



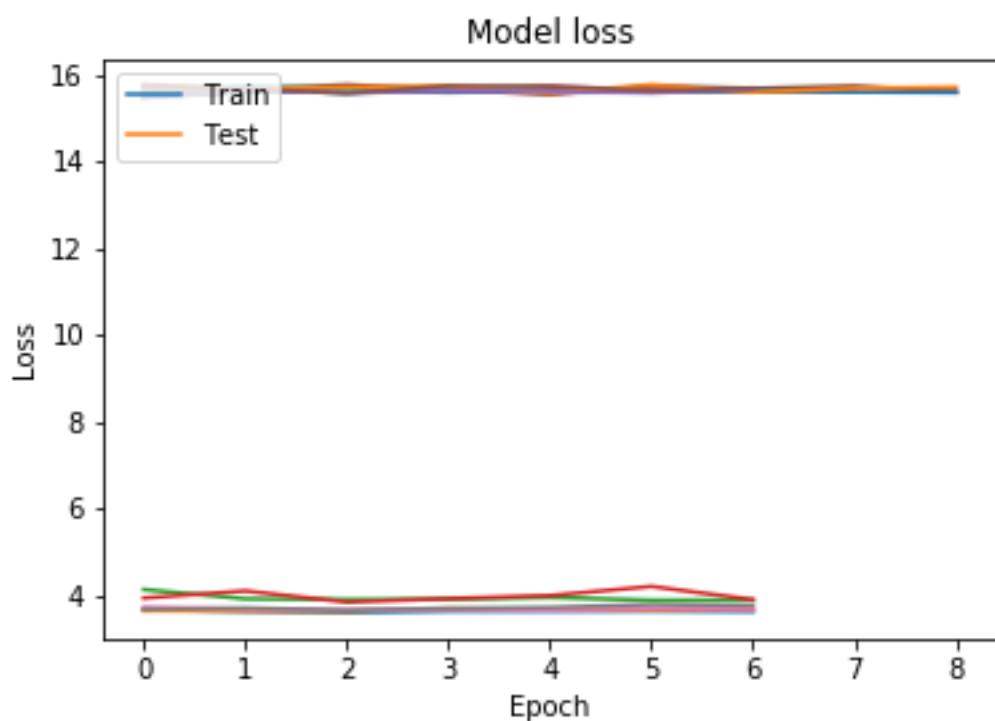
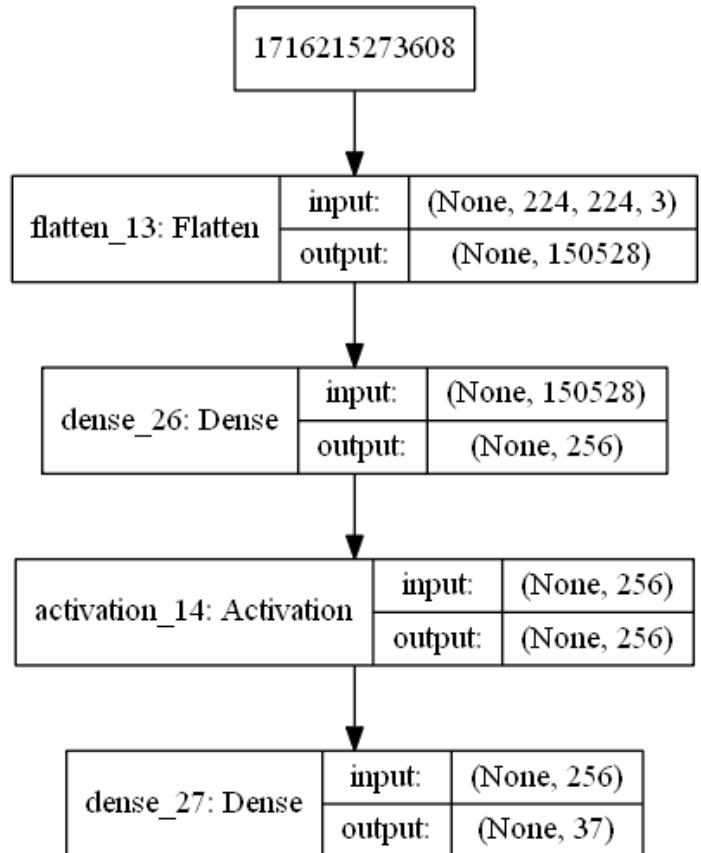
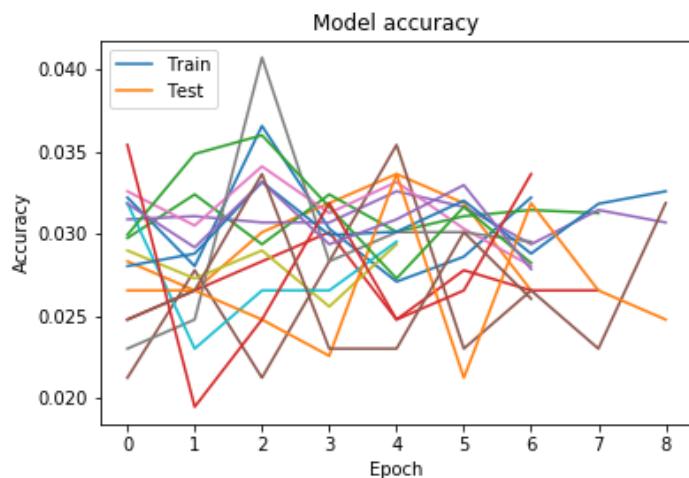
Кол-во слоев: 1
Кол-во нейронов: 64
Функция активации: Tanh



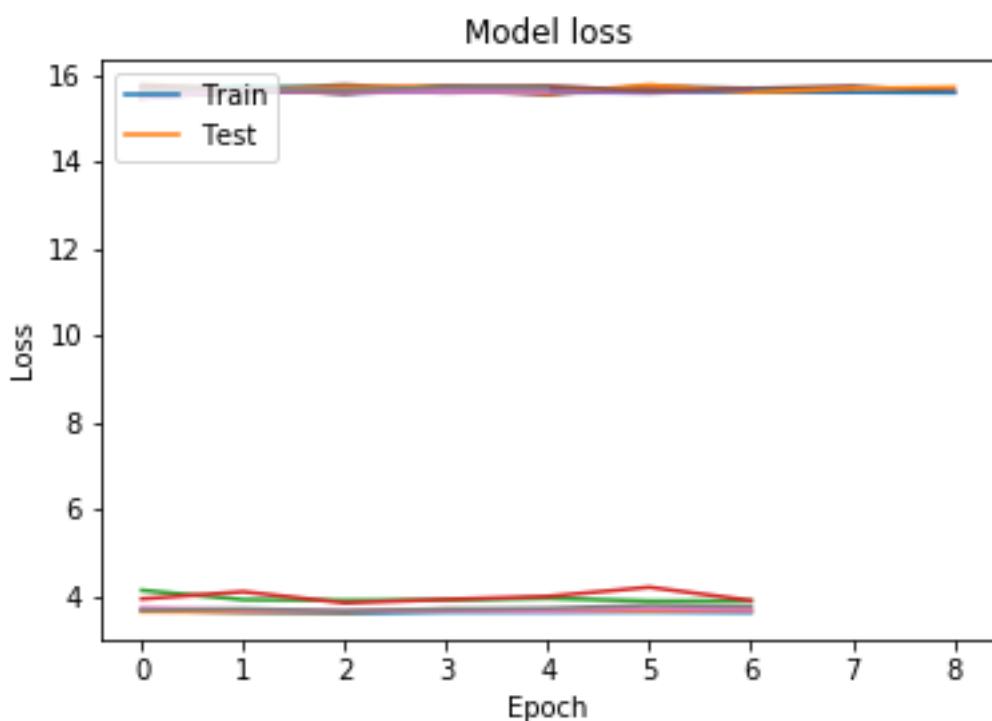
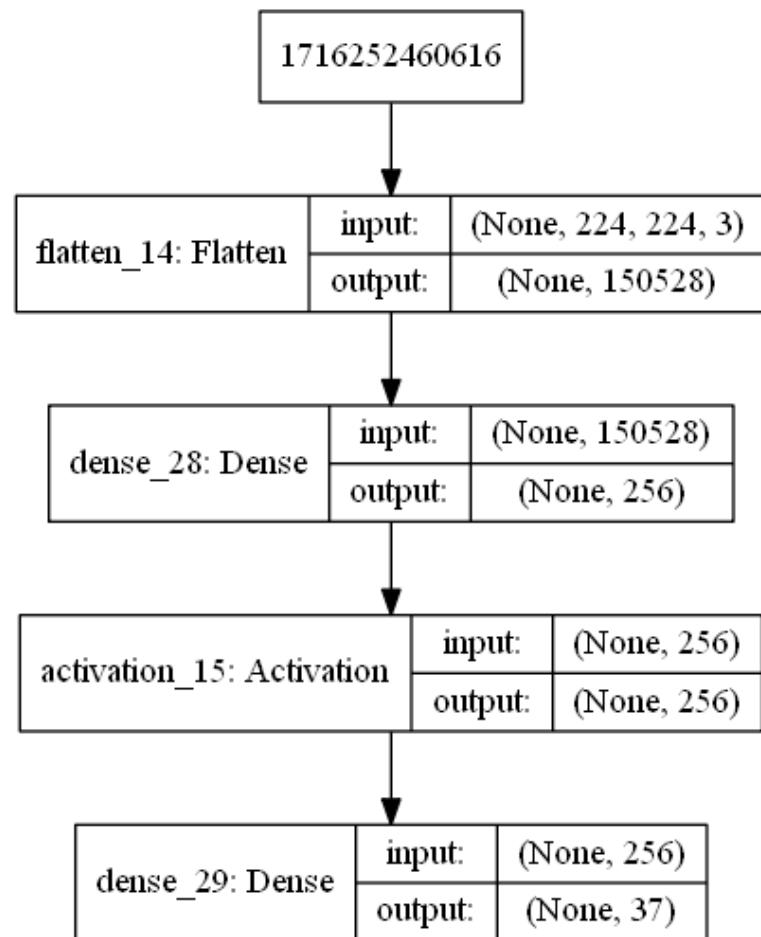
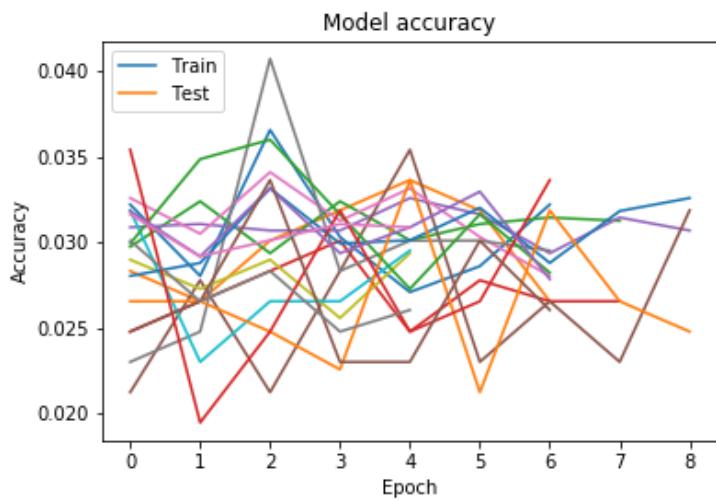
Кол-во слоев: 1

Кол-во нейронов: 256

Функция активации: ReLu



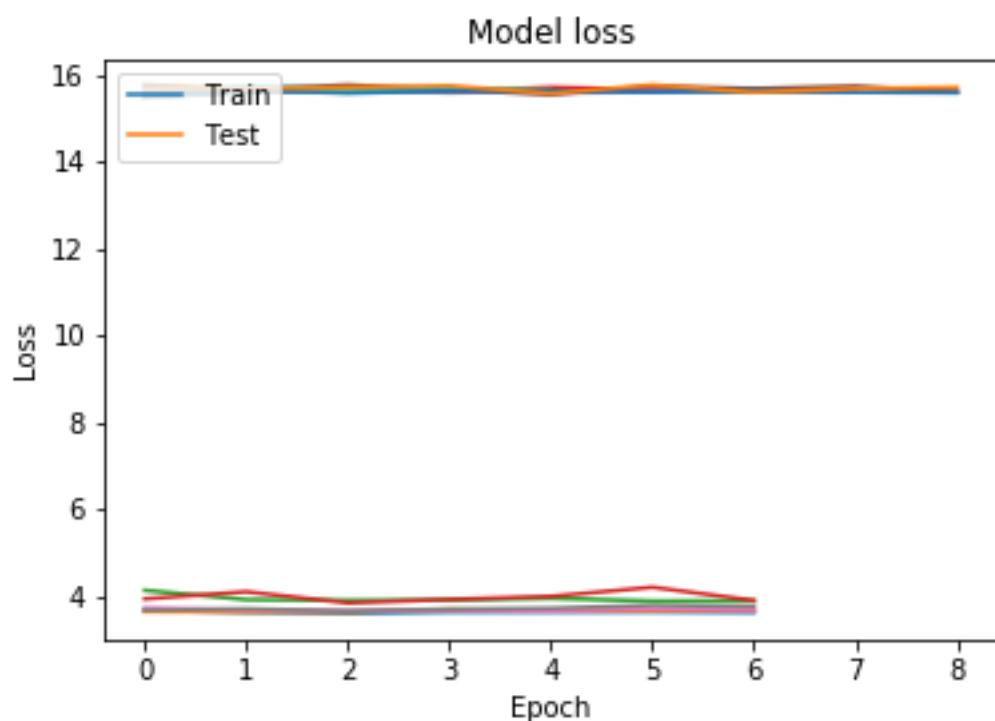
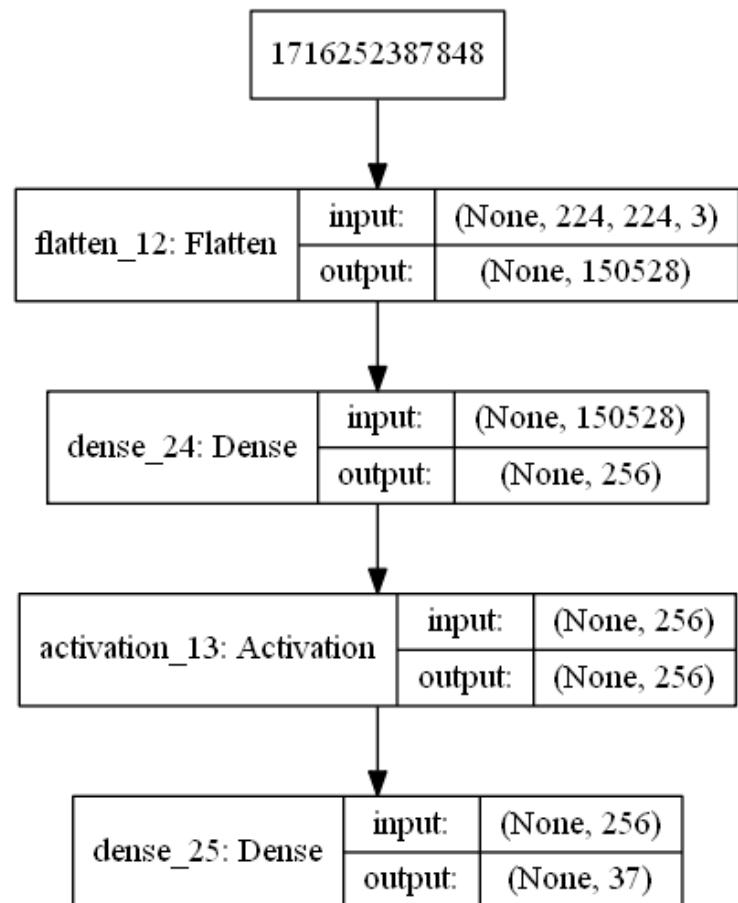
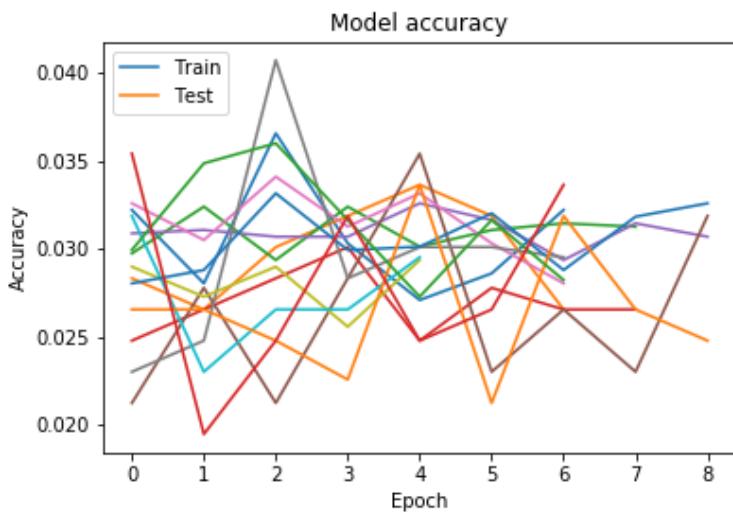
Кол-во слоев: 1
Кол-во нейронов: 256
Функция активации: SeLu



Кол-во слоев: 1

Кол-во нейронов: 256

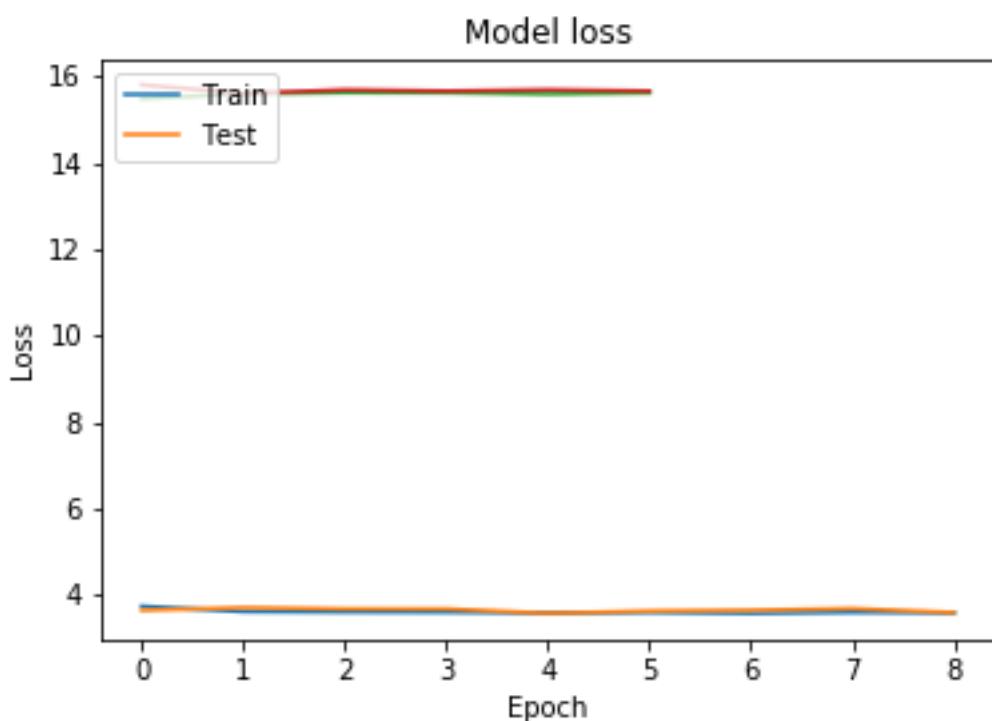
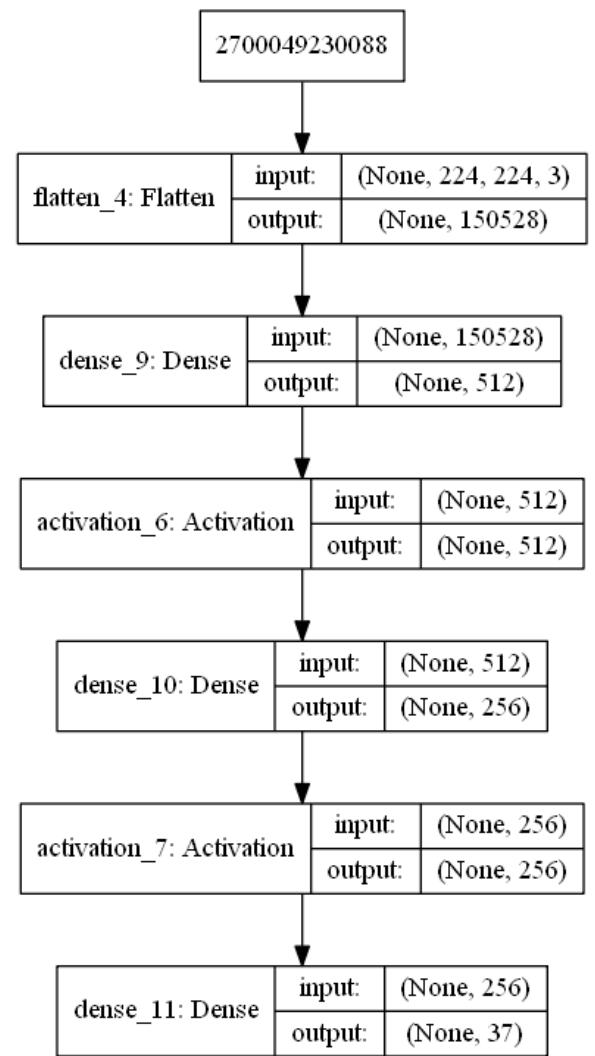
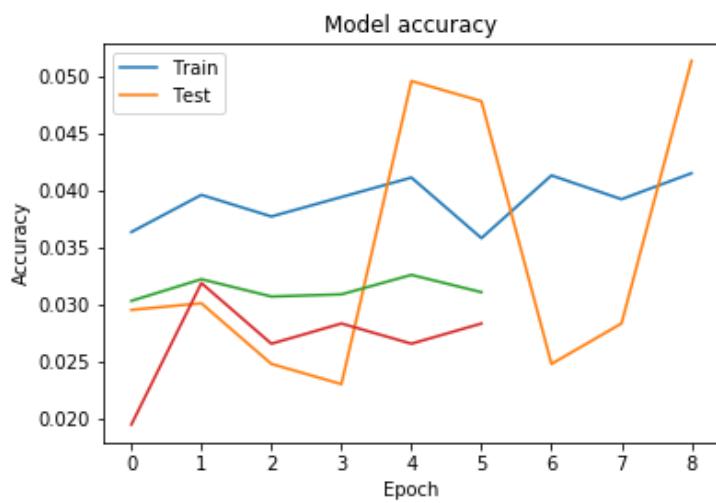
Функция активации: Tanh



Кол-во слоев: 2

Кол-во нейронов: [512 - 256]

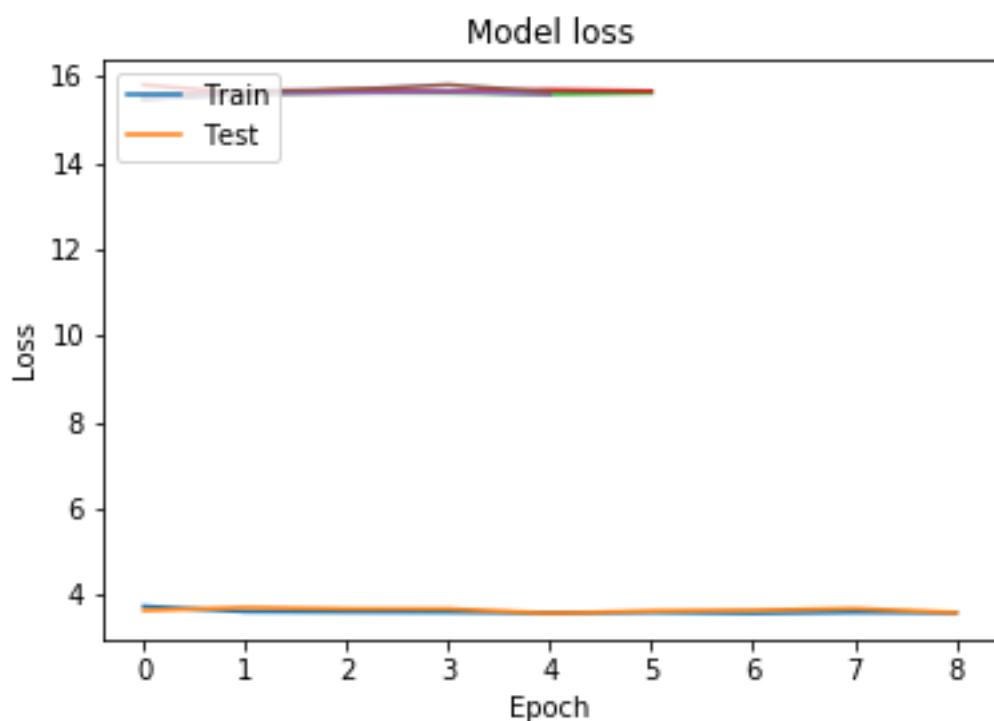
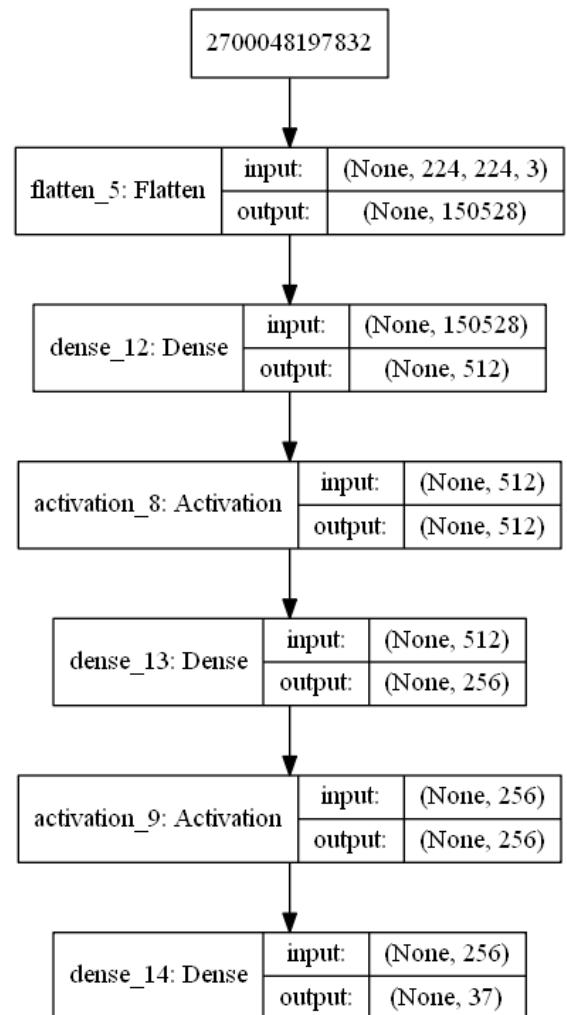
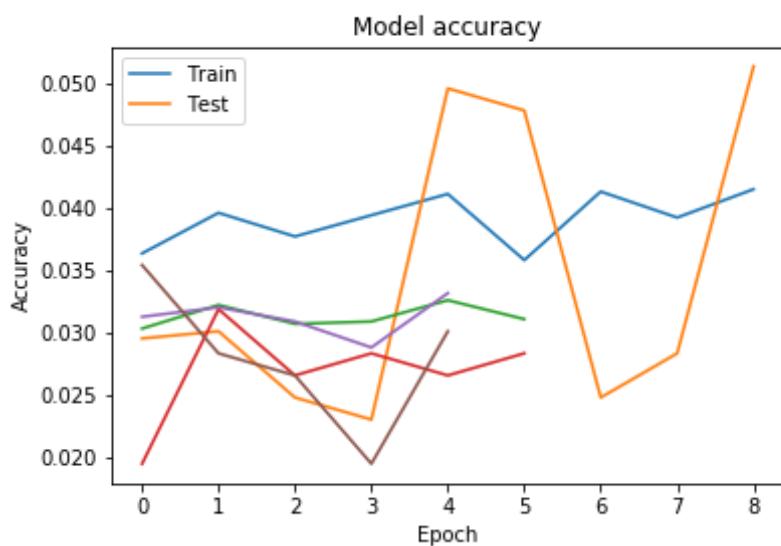
Функция активации: ReLu



Кол-во слоев: 2

Кол-во нейронов: [512 - 256]

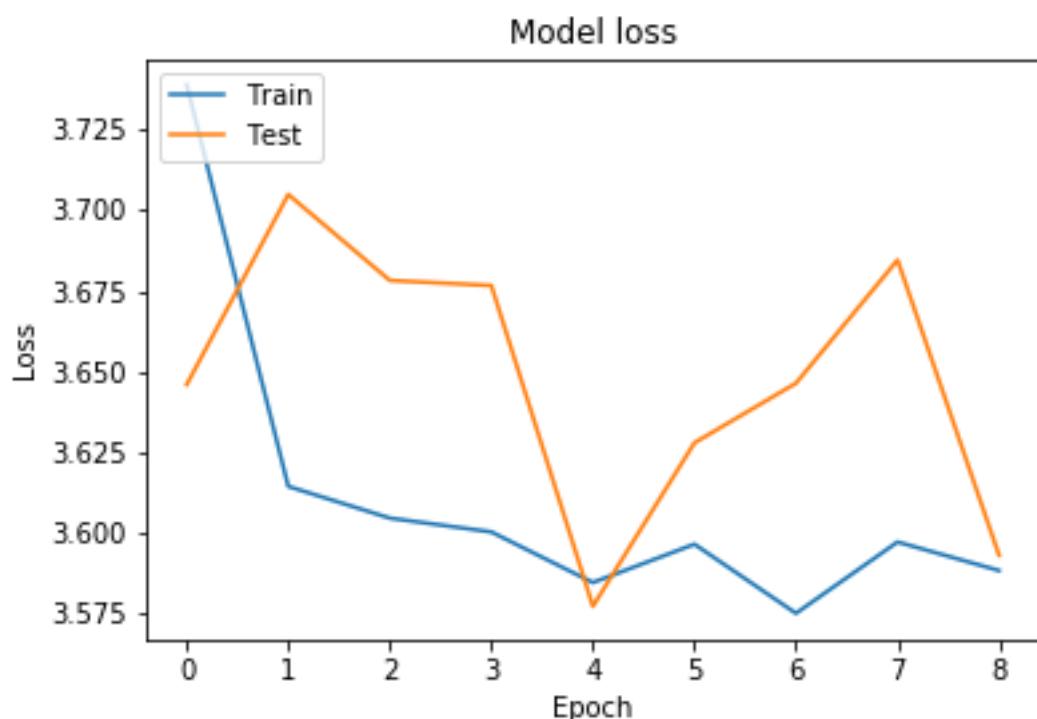
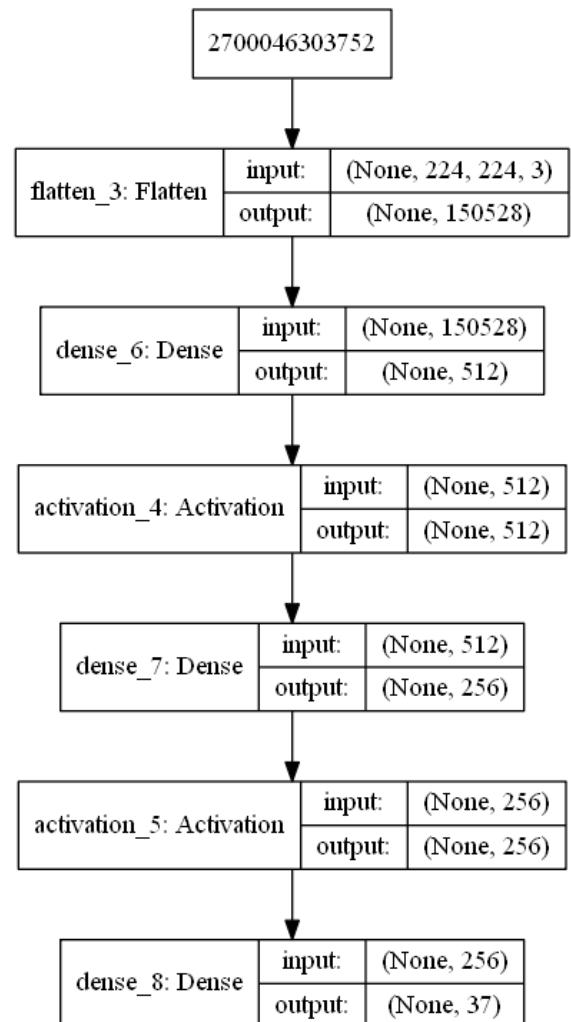
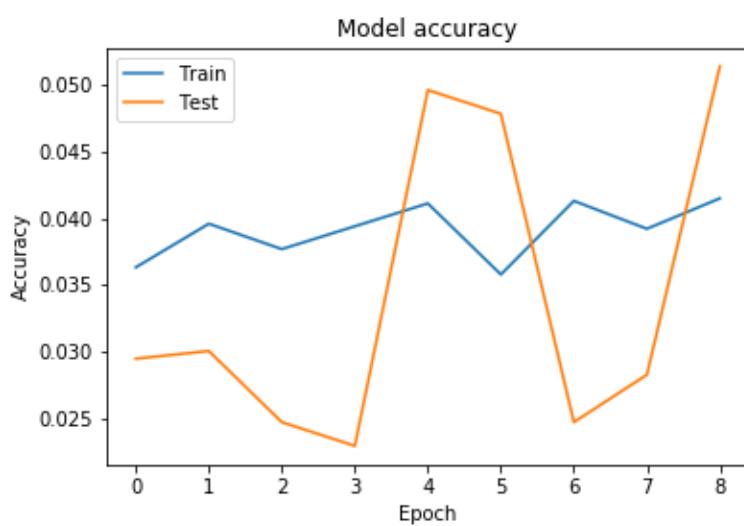
Функция активации: SeLu



Кол-во слоев: 2

Кол-во нейронов: [512 - 256]

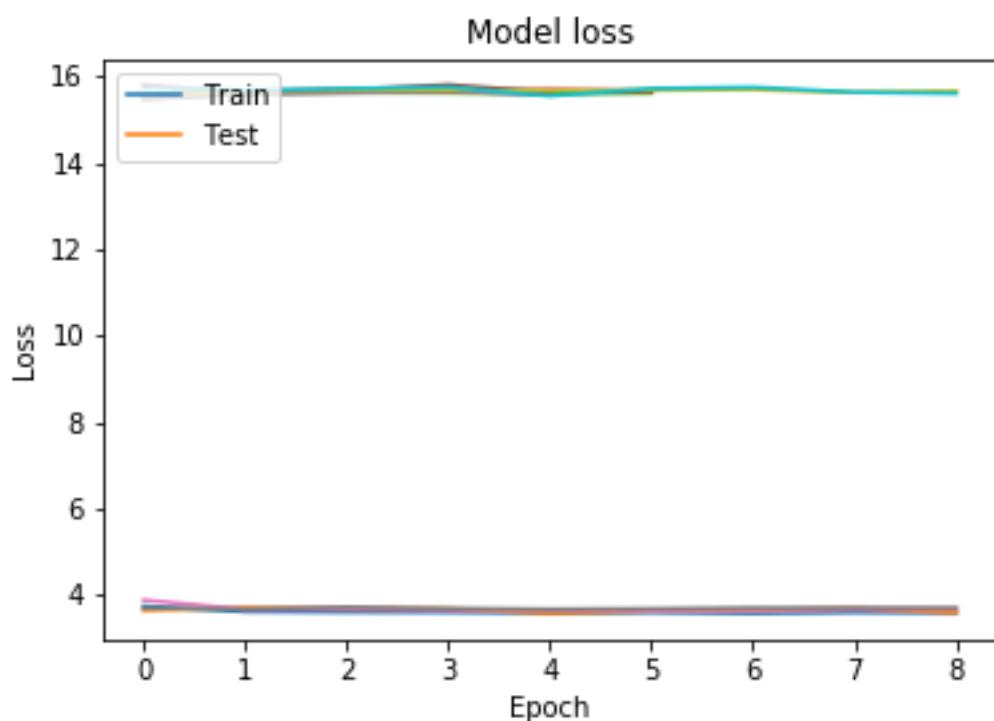
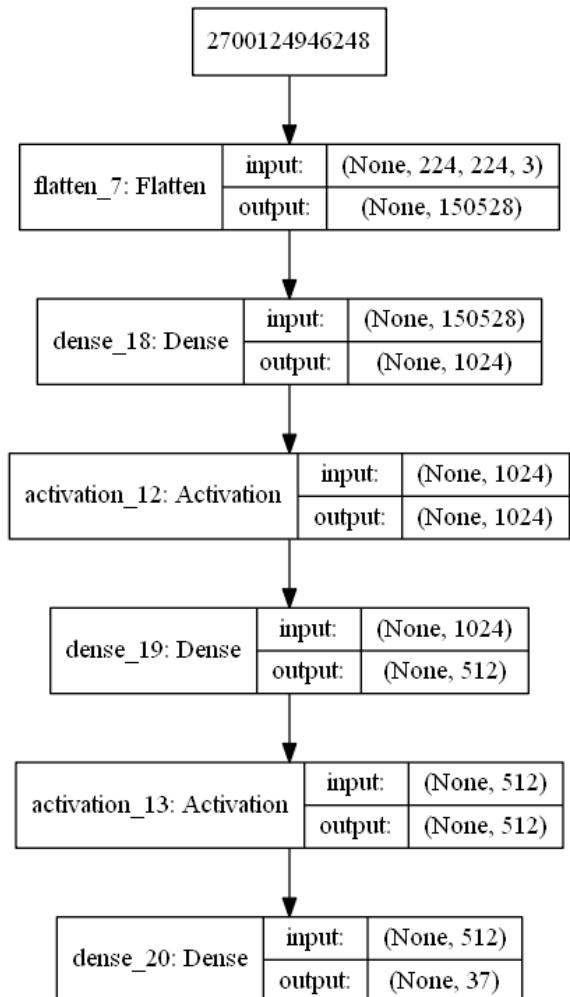
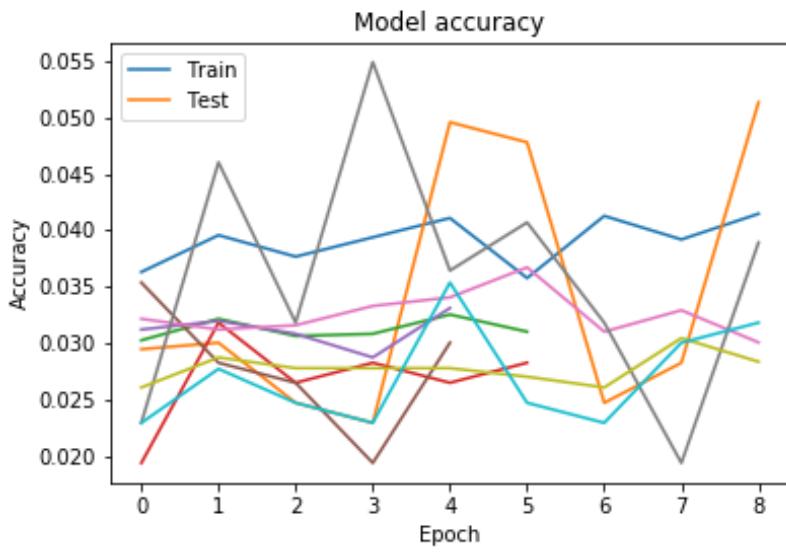
Функция активации: Tanh



Кол-во слоев: 2

Кол-во нейронов: [1024 - 512]

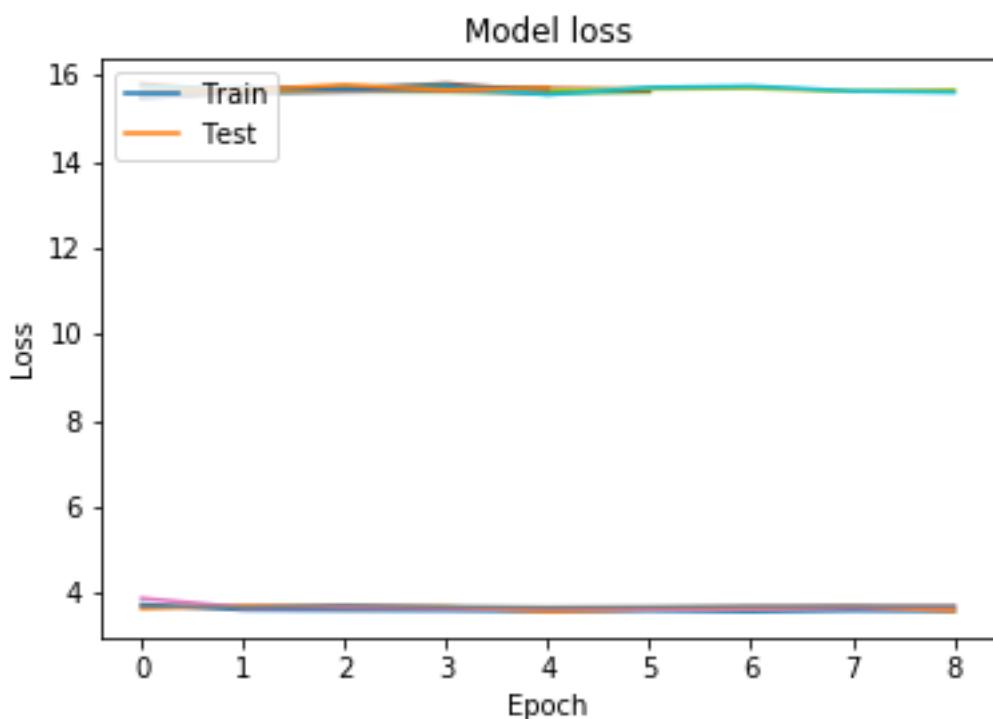
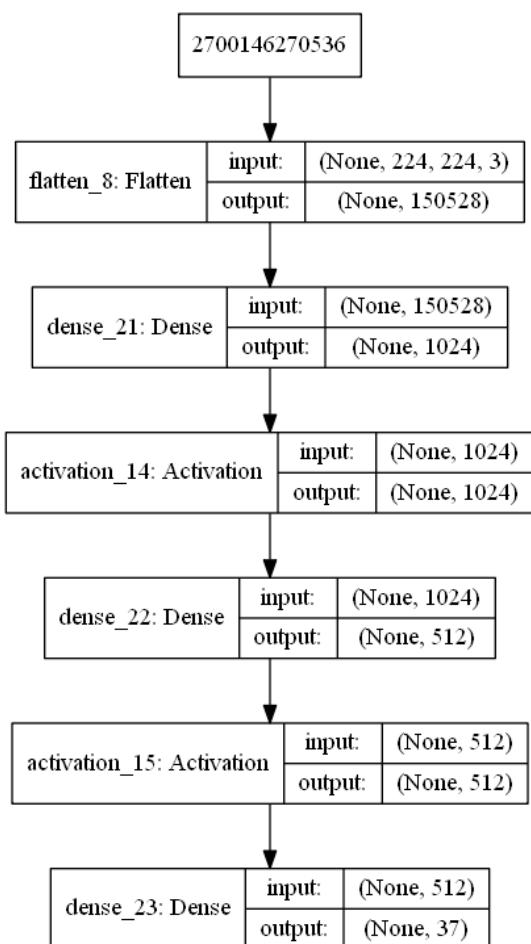
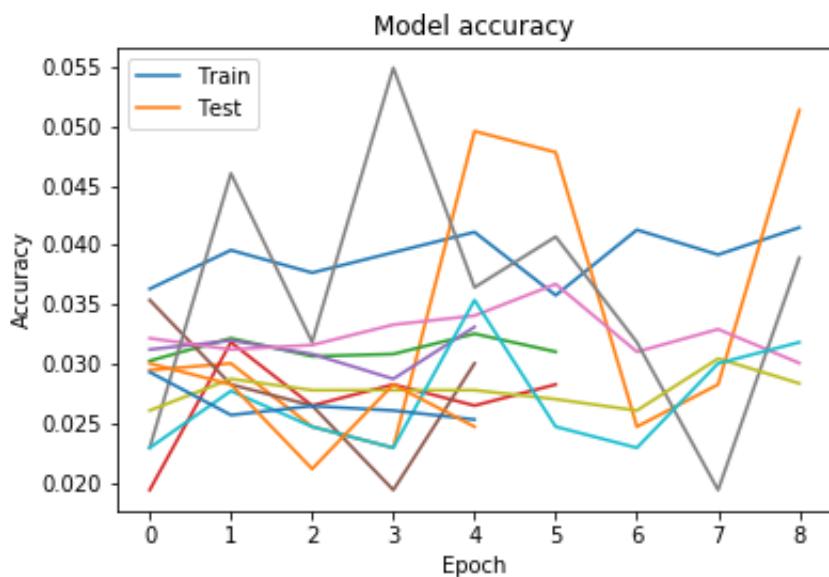
Функция активации: ReLu



Кол-во слоев: 2

Кол-во нейронов: [1024 - 512]

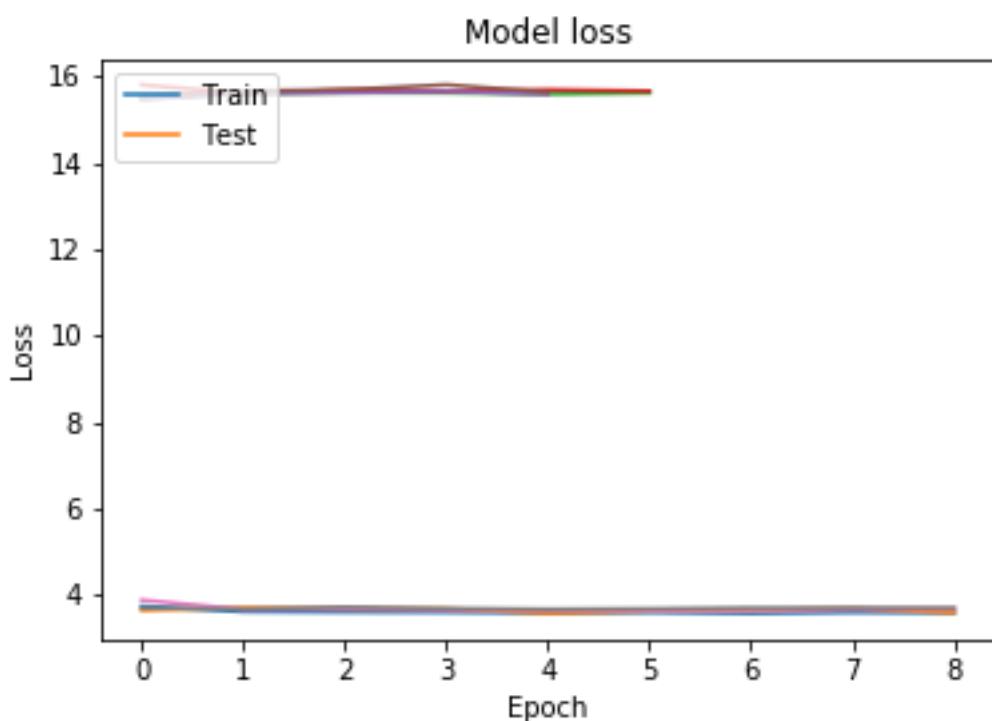
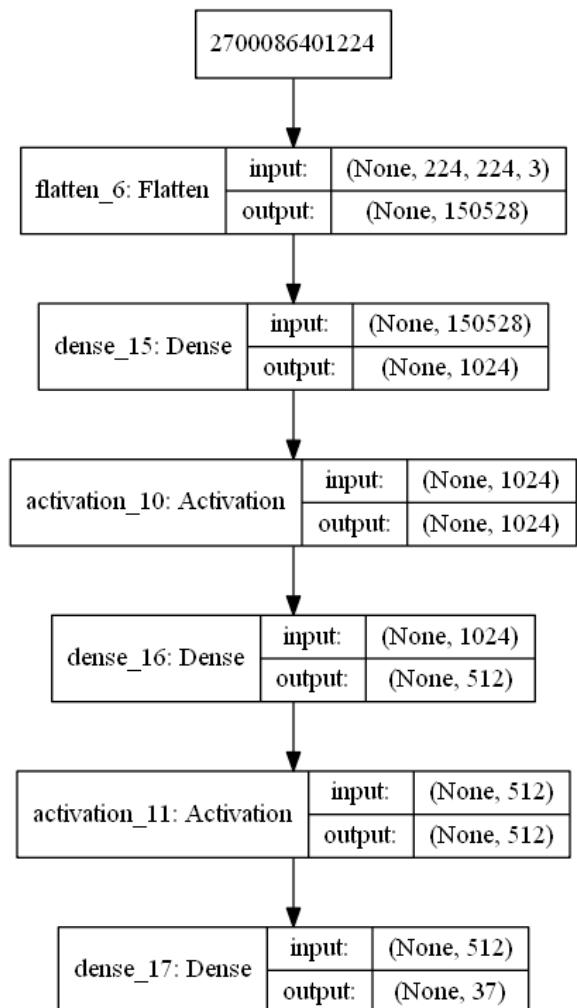
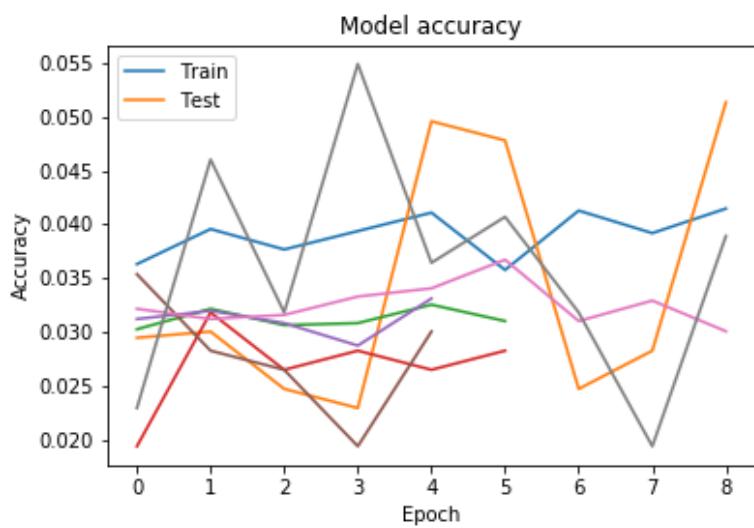
Функция активации: SeLu



Кол-во слоев: 2

Кол-во нейронов: [1024 - 512]

Функция активации: Tanh



Разработанные программы/скрипты

Был разработан python-скрипт `create_csv.py`, обрабатывающий массив изображений и создающий csv-файл с двумя столбцами: `category` и `path`. Первый столбец содержит информацию вида `class_name`, второй столбец содержит относительный путь до изображения. Затем был разработан .ipynb-скрипт реализующий обучение нейронной сети. Для генерации тренировочных и валидационных данных был использован упомянутый выше csv-файл. Для предобработки изображений из нашей тренировочной выборки использовался класс `ImageDataGenerator`, определяющий конфигурацию подготовки данных. В процессе обучения нейронной сети были испробованы следующие параметры класса: `rescale`, `rotation range`, `horizontal and vertical flip`, `width and height shift`. В свою очередь предобработка валидационных данных заключалась в применении `rescale`. Далее проходит обучение двух наших моделей с применением различных функций активации, таких как: гиперболический тангенс, `ReLU` и `SeLu`. Первая модель является однослоиной, вторая - двухслойной.

Результаты экспериментов

Параметры обучения:

- Функция ошибки: '`categorical_crossentropy`',
- Оптимизационный алгоритм: '`adam`' - <https://keras.io/optimizers/>,
- `batch_size`: 64,
- Количество эпох: 100 (использовался callback `EarlyStopping`, `patience = 4`)
- Скорость обучения: 0.001.

Параметры PC:

- Операционная система: Windows 10
- CPU: Intel Core i5-6200U (2.3 GHz)
- Процессор: NVidia 940M (2 GB)

Оперативная память: 8 GB

№	Количе- ство скрытых слоев	Количе- ство ней- ронов на скрытых слоях	Функции актива- ции	Результат		
				Точность на тренировоч- ном множе- стве	Точность на тесто- вом множе- стве	Время обуче- ния, с
1	1	64	tanh	0.0617	0.0602	306

Анализ результатов

Нейронные сети с функцией активации ReLU показали результаты лучше, чем с сигмоидальной функцией активации.

Однако, результат полностью связанных сетей не получится сильно улучшить, так как когда мы преобразуем изображение в линейную цепочку байт, мы что-то безвозвратно теряем. Причем с каждым слоем эта потеря только усугубляется. Мы теряем топологию изображения, т.е. взаимосвязь между отдельными его частями. Кроме того, задача распознавания подразумевает умение нейросети быть устойчивой к небольшим сдвигам, поворотам и изменению масштаба изображения, т.е. она должна извлекать из данных некие инварианты, не зависящие от углов, под которым сделано фото знака.

При тестировании с большим количеством нейронов было немного проще использовать ReLU функцию активации, но в целом на время данной функция не сильно влияет на время обучения, так как оно сильно зависит от количества нейронов в сети.

Все же стоит отметить преимущества ReLU:

1. Вычисление сигмоиды и гиперболического тангенса требует ресурсоёмких операций, таких как возведение в степень, в то время как ReLU не подвержен насыщению.
2. Применение ReLU существенно повышает скорость стохастического градиентного спуска по сравнению с сигмоидой и гиперболическим тангенсом. Это обусловлено линейным характером и отсутствием насыщения данной функции.

Также имеется недостаток:

1. ReLU не всегда достаточно надёжны и в процессе обучения могут выходить из строя. Например, большой градиент, проходящий через ReLU, может привести к такому обновлению весов, что данный нейрон никогда больше не активируется. Если это произойдет, то, начиная с данного момента, градиент, проходящий через этот нейрон, всегда будет равен нулю. Соответственно, данный нейрон будет необратимо выведен из строя. Например, при слишком большой скорости обучения, может оказаться, что до 40% ReLU никогда не активируются. Эта проблема решается посредством выбора надлежащей скорости обучения.

Выводы:

Лучший результат - 0.12% точности был получен в 6 модели, за счет подбора оптимального количества нейронов для данных параметров обучения. Если взять большее количество нейронов, то время обучения и процент ошибок сильно ухудшался.

Столь низкий показатель точности обусловлен слишком «разными» изображениями в изучаемом датасете. Ниже приведены несколько изображений, сгруппированных по своим классам. Можно заметить, что основные отличительные признаки классификации на каждом изображении затруднены в изучении.

