

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Национальный исследовательский университет  
«Высшая школа экономики»**

**Факультет компьютерных наук  
Основная образовательная программа  
«Прикладная математика и информатика»**

# **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Программный проект  
по теме  
«Обнаружение болезней растений  
с использованием нейронных сетей»**

**Выполнил студент группы 194, 4 курса,  
Кочерин Никита Алексеевич**

**Руководитель ВКР:  
Соколов Евгений Андреевич**

**Соруководитель ВКР:  
Воронин Игорь Вадимович**

# Содержание

<b>1 Введение</b>	<b>3</b>
<b>2 Методология</b>	<b>3</b>
<b>3 Обзор литературы</b>	<b>3</b>
<b>4 Сверточная нейронная сеть</b>	<b>4</b>
<b>5 Обнаружение листьев</b>	<b>4</b>
<b>6 Обучение сверточной нейронной сети</b>	<b>5</b>
6.1 Сбор данных . . . . .	5
6.2 Архитектура нейронной сети . . . . .	5
<b>7 Результаты обучения сети для классификации изображений</b>	<b>6</b>
7.1 Большая . . . . .	6
7.2 Средняя . . . . .	6
7.3 Маленькая . . . . .	11
<b>8 Эксперименты со средней сеткой</b>	<b>11</b>
8.1 5 maxpool, 1 полносвязный . . . . .	11
8.2 7 maxpool, 4 полносвязных . . . . .	11
8.3 5 maxpool, 2 полносвязных . . . . .	15
8.4 6 maxpool, 3 полносвязных . . . . .	15
8.5 6 maxpool, 4 полносвязных . . . . .	15
8.6 5 maxpool, 4 полносвязных . . . . .	15
<b>9 Итоги экспериментов</b>	<b>20</b>
<b>10 Результаты</b>	<b>21</b>
<b>11 Руководство по использованию программы</b>	<b>22</b>
<b>12 Направления для дальнейших исследований</b>	<b>22</b>
<b>13 Заключение</b>	<b>23</b>

## Аннотация

В данном исследовании предлагается новый подход к обнаружению болезней растений по листу с помощью применения различных методов компьютерного зрения. Хотя было проведено множество исследований по автоматическому обнаружению болезней растений, некоторые из них страдают от недостатка результатов или слишком сложных методологий. Цель данного проекта - критически рассмотреть различные стратегии, применяемые для обнаружения болезней растений, и представить инновационные методы, чтобы внести свой вклад в эту область.

Проблема обнаружения болезней растений широко признана и является значительной проблемой, учитывая потенциальный вред, который одно зараженное растение может причинить всему урожаю. До появления техник компьютерного зрения не было автоматического решения для обнаружения болезней растений, используя только фотографии их листьев. Однако, благодаря последним достижениям в области компьютерного зрения, обнаружение болезней растений стало более возможным, и у него есть потенциал спасти множество растений от разрушительных заболеваний.

## Abstract

The current study proposes a novel approach for detecting plant diseases through the application of diverse computer vision methods. Although there have been numerous investigations on automatic detection of plant diseases, some suffer from lack of results or too complex methodologies. The objective of this project is to critically review various strategies employed for detecting plant diseases and present innovative methods to contribute to this field.

The issue of detecting plant diseases is a widely acknowledged and significant concern, given the potential harm that a single infected plant can inflict on an entire crop. Prior to the emergence of computer vision

techniques, there was no automatic solution for detecting plant diseases using only photos of leaves. However, with the recent advances in computer vision, the detection of plant diseases has become more feasible, and it has the potential to rescue many crops from devastating diseases.

## 1 Введение

Растения являются важной составляющей нашей экосистемы, обеспечивая пищу, кислород и эстетическую красоту. Однако, болезни растений представляют серьезную угрозу для глобальной продовольственной безопасности, так как они могут разрушать урожай и влиять на качество продуктов питания. Раннее обнаружение и идентификация болезней растений является ключевым моментом в смягчении их распространения и минимизации ущерба. С развитием технологий, нейронные сети стали мощным инструментом для обнаружения и диагностики болезней растений. Анализируя огромные объемы данных о растениях, нейронные сети могут точно определять и классифицировать болезни, что позволяет своевременно и эффективно вмешиваться. В этой статье мы рассмотрим использование нейронных сетей в обнаружении болезней растений и выделим наиболее перспективные применения этой технологии в сельском хозяйстве.

В этой статье мы будем обсуждать выращивание огурцов, популярную сельскохозяйственную деятельность в России. С годовым производством более 2 миллионов тонн, раннее обнаружение любых заболеваний является важным моментом из-за риска заражения всего урожая одним больным растением.

Учитывая огромное количество заболеваний огурцов, мы выделим здесь 5 наиболее распространенных из них[6]:

- Альтернариоз
- Антракноз
- Бактериоз
- Мучнистая роса
- Пероноспороз

Вы можете определить все эти заболевания, просто глядя на фотографии. Но мы хотим, чтобы это происходило автоматически. Преимущества автоматического определения заболеваний следующие:

1. Такая система позволяет определить болезнь на ранней стадии, пока она не успела распространиться, и ограничить вред для остального посева.

2. Благодаря внедрению этой технологии в промышленное производство овощей, можно существенно снизить затраты на агрономов, которых потребуется сильно меньше, и которые, к тому же, во многих спорных случаях смогут выносить вердикт удаленно, потому что все необходимые фотографии уже есть в системе.

3. Польза для небольших хозяйств тоже существенна: вместо долгого поиска симптомов болезней, нужно будет просто использовать эту технологию, которую можно оформить в виде приложения или web-сайта и определить заболевание, сделав всего несколько фотографий.

## 2 Методология

Процесс обнаружения заболеваний огурцов начинается с сбора нескольких изображений, каждое из которых содержит множество листьев. Затем наша программа идентифицирует каждый отдельный лист на этих изображениях и проводит анализ, чтобы определить вероятность заражения. Мы будем использовать сверточную нейронную сеть для этой цели, методология объяснена в следующем абзаце. После того, как мы определили вероятность заболевания для каждого листа, мы предложим соответствующие рекомендации, например, запрос более близкого фото, если диагноз неясен, или просто указание на то, заражен ли листок.

## 3 Обзор литературы

Область обнаружения болезней растений охватывает множество растительных видов и их заболеваний. Каждая статья по данной тематике фокусируется на конкретном растении и заболевании, которым оно страдает. Среди различных методов, используемых в этих статьях, методы на основе компьютерного зрения получили широкое распространение в последние годы.

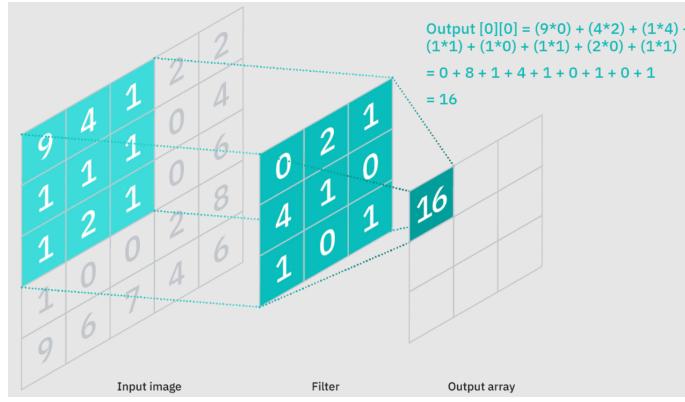


Рис. 1: Свёртка

Одна из таких статей, написанная Анандом [3], предлагает использование фильтра Габора для предобработки изображений, а затем сложной нейронной сети с несколькими этапами для обнаружения заболеваний. Хотя его подход показывает многообещающие результаты, сложность и количество этапов могут стать препятствием для практической реализации.

С другой стороны, Аллейников [2] предлагает более простой подход, используя только методы компьютерного зрения для обнаружения заболеваний. Однако его метод требует нескольких изображений одного и того же листа, что может быть не очень удобно. Кроме того, в его статье не сообщаются результаты и отсутствует подробное описание используемых сетей.

Таким образом, хотя обе статьи предлагают ценные идеи по использованию компьютерного зрения в области обнаружения болезней, все еще есть место для дальнейших исследований и разработки более практических и эффективных методов для обнаружения заболеваний растений.

## 4 Сверточная нейронная сеть

Сверточная нейронная сеть (CNN) — это тип алгоритмов глубокого обучения, часто используемый для распознавания изображений и задач компьютерного зрения.

Основными блоками CNN являются сверточные слои, которые выполняют математическую операцию свёртки на входном изображении. Операция свёртки (рис. 1) включает перемещение маленького окна, называемого фильтром или ядром, по входному изображению и вычисление скалярного произведения между фильтром и локальным участком изображения, который покрывается фильтром. Это создает новые признаки, где каждый элемент представляет ответ фильтра в определенном месте входного изображения.

С помощью сверточных слоев CNN может научиться извлекать все более сложные характеристики изображения. Выход последнего сверточного слоя затем разворачивается в вектор и передается в один или несколько полносвязных слоев, которые используются для классификации или регрессии.

Одна из первых архитектур сверточных сетей [2] была предложена в 1998 году Ле Куном в статье "Gradient-Based Learning Applied to Document Recognition" [4]. Она помогла улучшить качество классификации изображений, так как до этого в подобных задачах, в основном, использовались различные эвристики, вместо применения нейронных сетей.

Для улучшения производительности CNN часто используются различные техники, включая слои пулинга для уменьшения карт признаков, dropout слои для предотвращения переобучения и нормализацию по батчам для ускорения обучения. Кроме того, доступны предварительно обученные модели CNN, такие как VGG, ResNet и LeNet, которые можно настроить на новых наборах данных, чтобы достичь результатов с минимальными усилиями.

## 5 Обнаружение листьев

На этом этапе программы мы используем библиотеки OpenCV и ultralitics для Python. С их помощью мы обучили нейросеть на основе архитектуры YOLO [5] для обнаружения листьев на фотографиях. Для этого

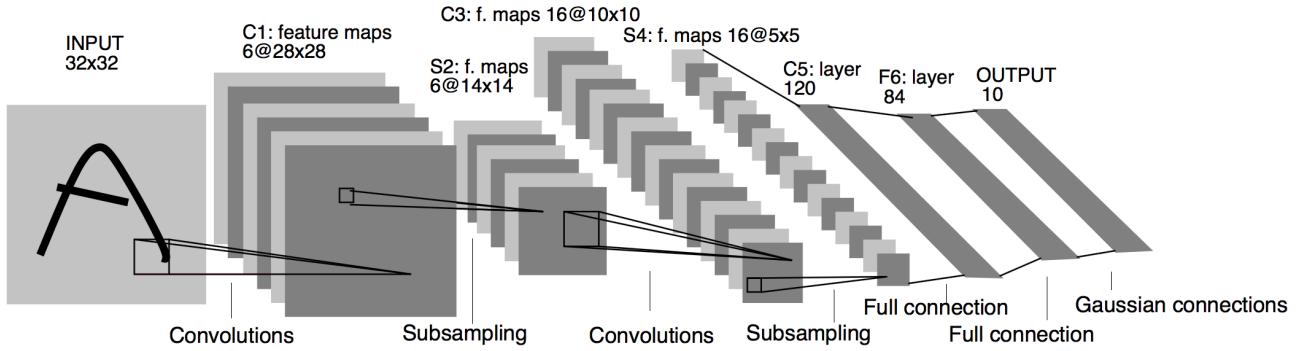


Рис. 2: LeNet



Рис. 3: Пример разметки данных с помощью labelimg

нужно было разметить данные, то есть указать, где есть листья на фотографиях (рис. 3), что мы сделали с помощью программы imagelabel [1]

## 6 Обучение сверточной нейронной сети

### 6.1 Сбор данных

Была собрана есть коллекция из 211 изображений листьев, на каждой из которых изображен один лист с какой то из 5 выбранных болезней, или же здоровый. Все фотографии разделены на 2 папки: **train** и **val**, на первой сетка тренируется, на второй валидируется. Внутри каждой из этих папок все изображения разделены на 6 классов: Альтернариоз, Антракноз, Бактериоз, Здоровые, Мучнистая роса, Пероноспороз.

### 6.2 Архитектура нейронной сети

Здесь мы представляем базовую архитектуру, которую можно улучшить при необходимости для повышения качества.

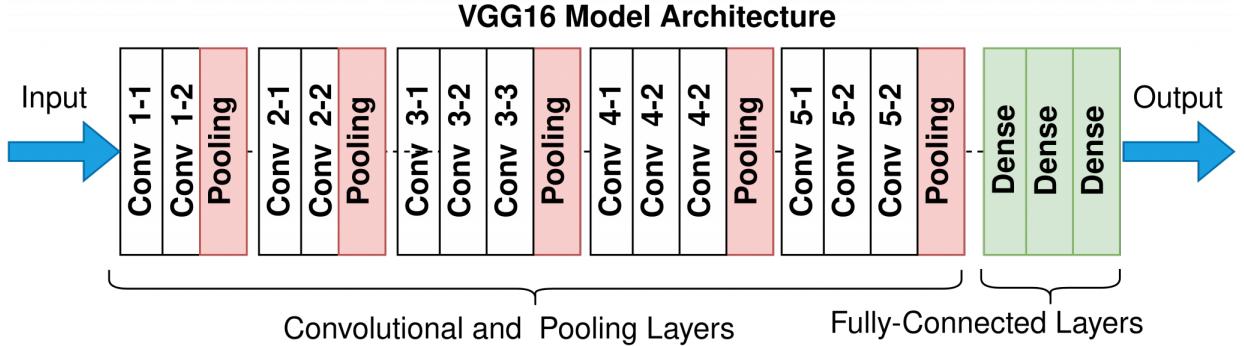


Рис. 4: VGG

В качестве прототипа мы возьмем архитектуру VGG (4), введенную в 2014 году в статье "Very Deep Convolutional Networks for Large-Scale Image Recognition" [7].

Опишем главный элемент архитектуры – VGG-блок. Один VGG-блок в общем случае состоит из следующих последовательных слоев: Convolution – BatchNorm – ReLU – Convolution – BatchNorm – ReLU – MaxPooling. Все свертки берутся с ядром 3x3, MaxPooling с ядром 2x2. Таким образом, после одного VGG-блока картинка уменьшается в 2 раза.

После всех VGG-блоков применяется 2 полносвязных слоя для итоговой классификации

На выходе нейронной сети будет 6 нейронов, затем функция softmax и кросс-энтропия для вычисления вероятностей. Softmax для выходов  $y_1 \dots y_n$  является функцией, которая преобразует  $y_i$  в  $\frac{e^{y_i}}{\sum_j e^{y_j}}$ . После Softmax применяется кросс-энтропия - это функция, которая при заданных входах  $Softmax(y_1) \dots Softmax(y_n)$  и значениях целевой переменной  $t_1 \dots t_n$ , где  $t_i$  – индикатор правильного класса (то есть, если объект при обучении имеет класс j, соответствующий ему вектор t будет состоять из нулей при одной единице на позиции j), имеет выход  $CE = -\sum_i t_i \log(Softmax(y_i))$ . Эта функция потерь будет использоваться для оптимизации, а вероятности каждого класса будут получены из предыдущего слоя, потому что сумма выходов после функции softmax равна единице.

Для экспериментов будем использовать 3 различных размера сетки: большая –  $21 * 10^6$  параметров и 7 VGG блоков, средняя –  $9.6 * 10^6$  параметров и 5 VGG блоков и маленькая –  $2.2 * 10^6$  параметров и 3 VGG блока.

## 7 Результаты обучения сети для классификации изображений

Для лучшей читаемости графиков лосса и accuracy, я разделил их на 2 части для каждой модели – первые 220 эпох и следующие 60

### 7.1 Большая

Количество параметров –  $21 * 10^6$

Accuracy – 0.68

Итоговые результаты оказались не такими хорошими, т.к из-за большого количества параметров, сетка переобучилась. Ниже графики для лосса и accuracy (accuracy-1 5, accuracy-2 6, loss-1 7, loss-2 8)

### 7.2 Средняя

Количество параметров –  $9.6 * 10^6$

Accuracy – 0.77

Средняя сетка оказалась самой удачной, так как из опробованных, у нее наилучшее качество, при этом, она не сильно больше маленькой, так что инференс будет довольно быстрым. Поэтому, дальнейшие эксперименты, будем проводить с этой архитектурой.

Графики – (accuracy-1 9, accuracy-2 10, loss-1 11, loss-2 12)

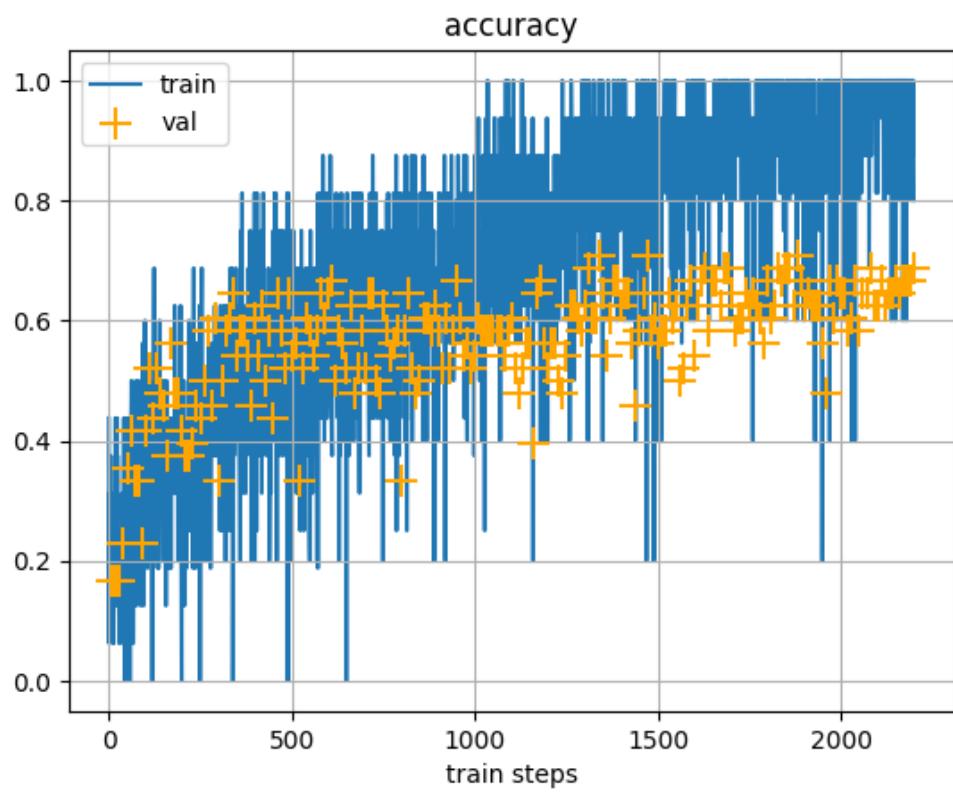


Рис. 5: Accuracy first 220 epochs(Big model)

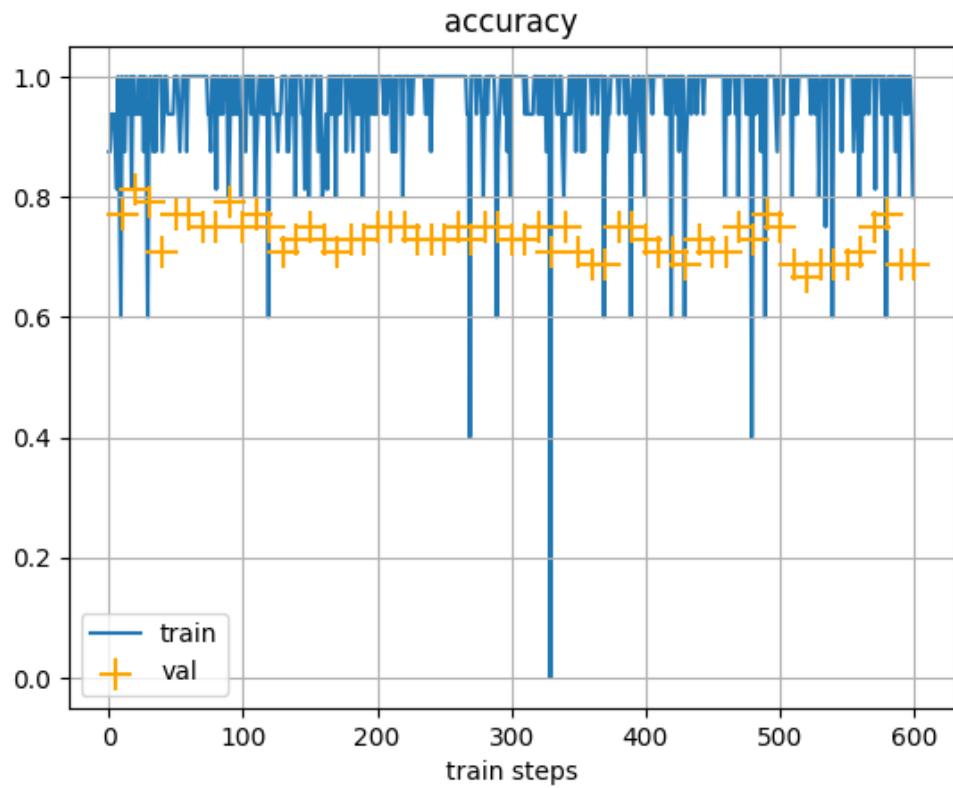


Рис. 6: Accuracy last 60 epochs(Big model)

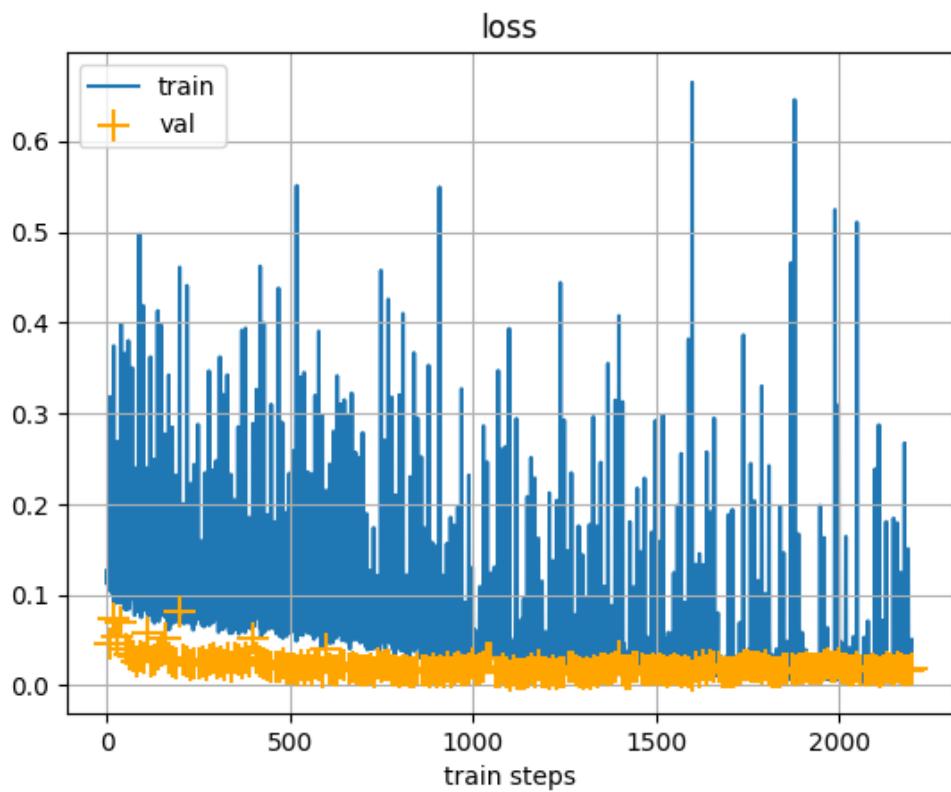


Рис. 7: Loss first 220 epochs(Big model)

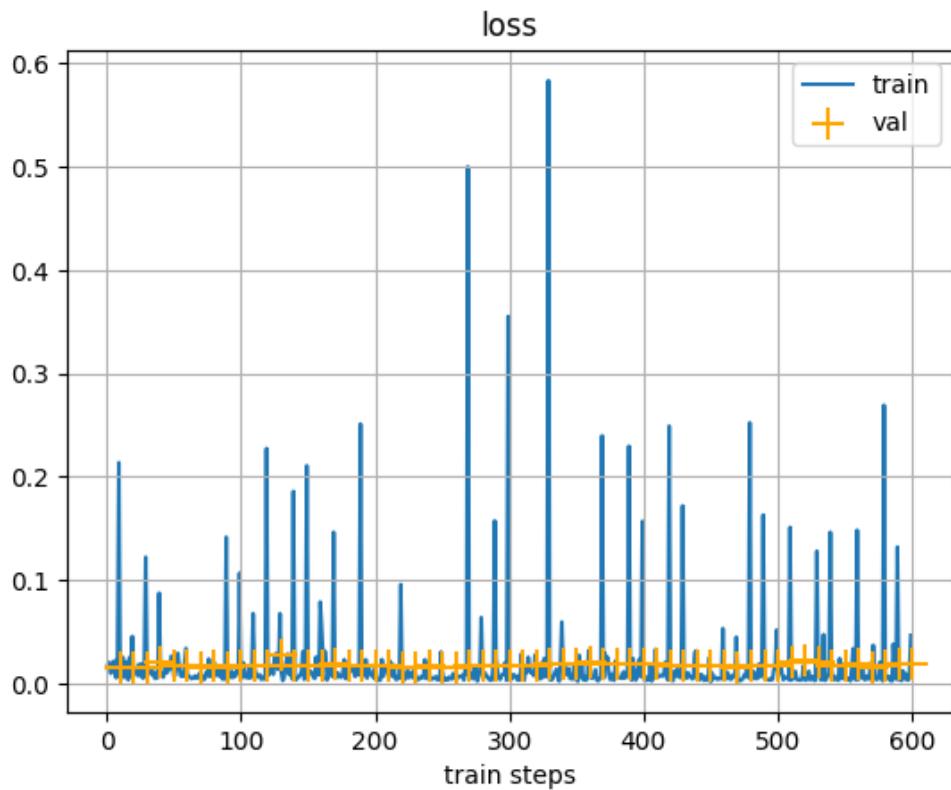


Рис. 8: Loss last 60 epochs(Big model)

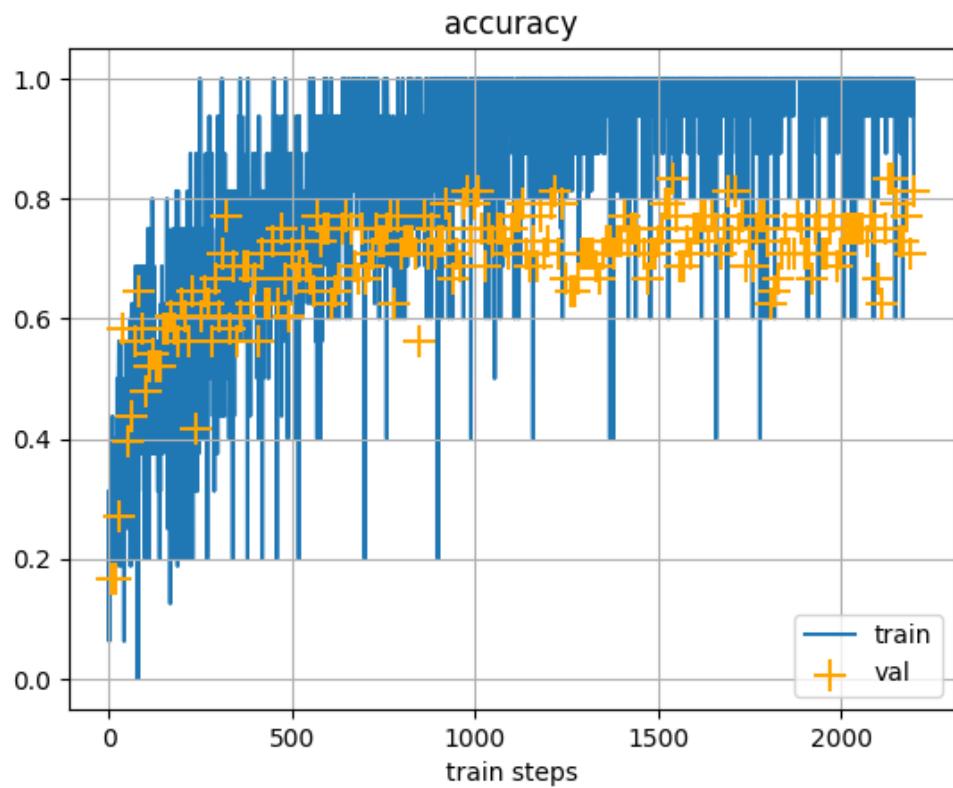


Рис. 9: Accuracy first 220 epochs(Medium model)

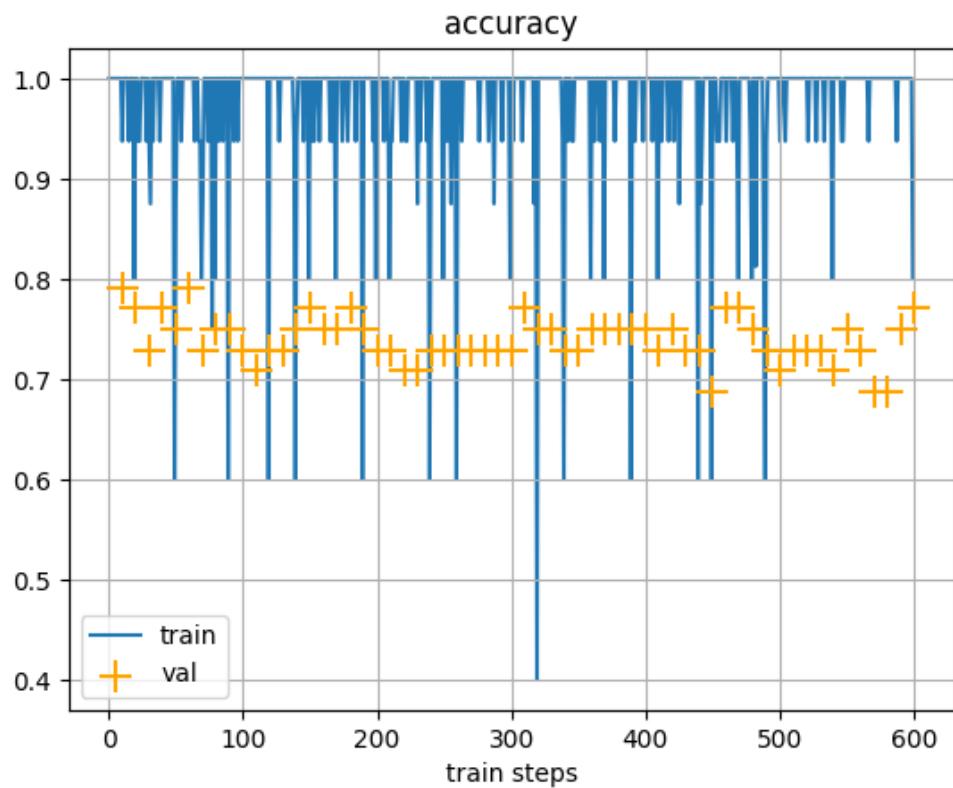


Рис. 10: Accuracy last 60 epochs(Medium model)

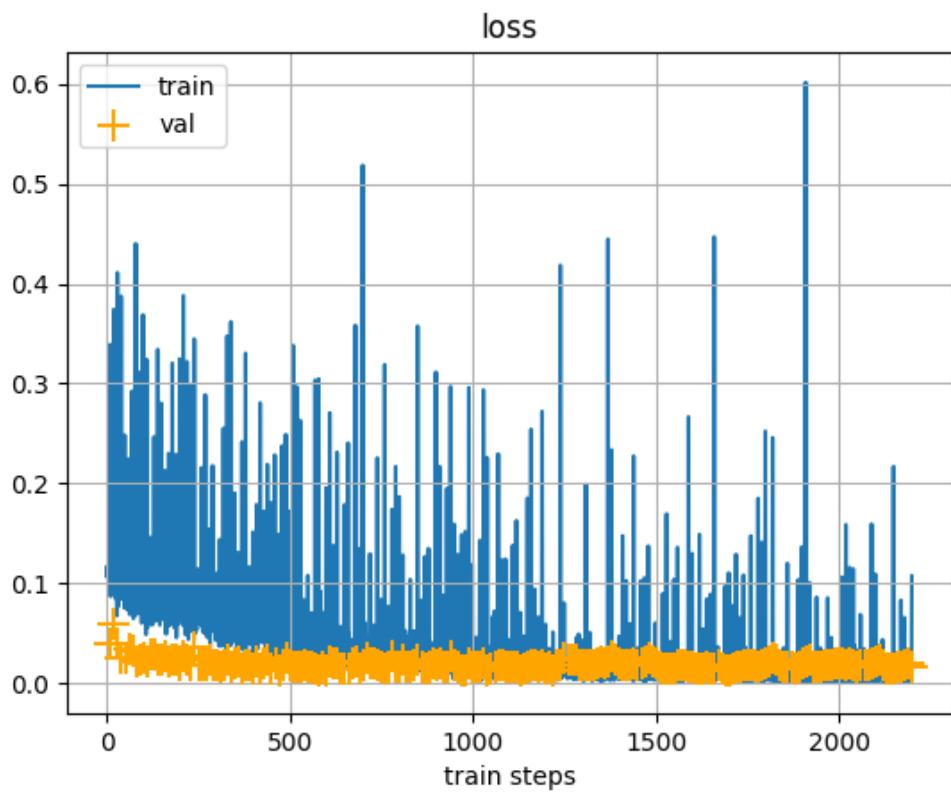


Рис. 11: Loss first 220 epochs(Medium model)

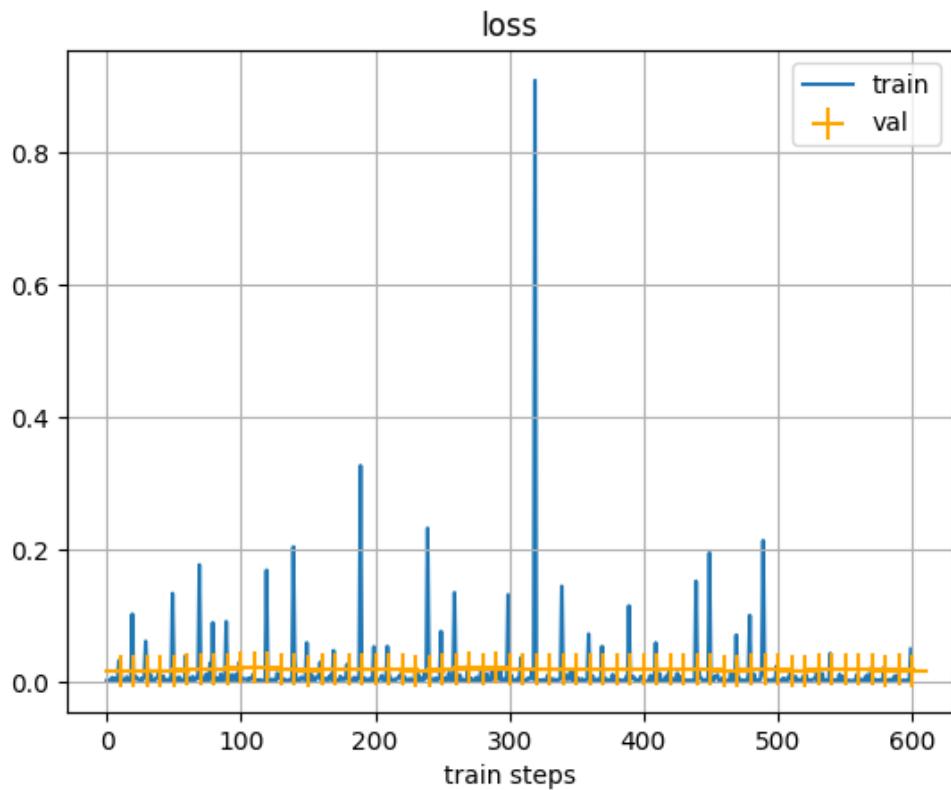


Рис. 12: Loss last 60 epochs(Medium model)

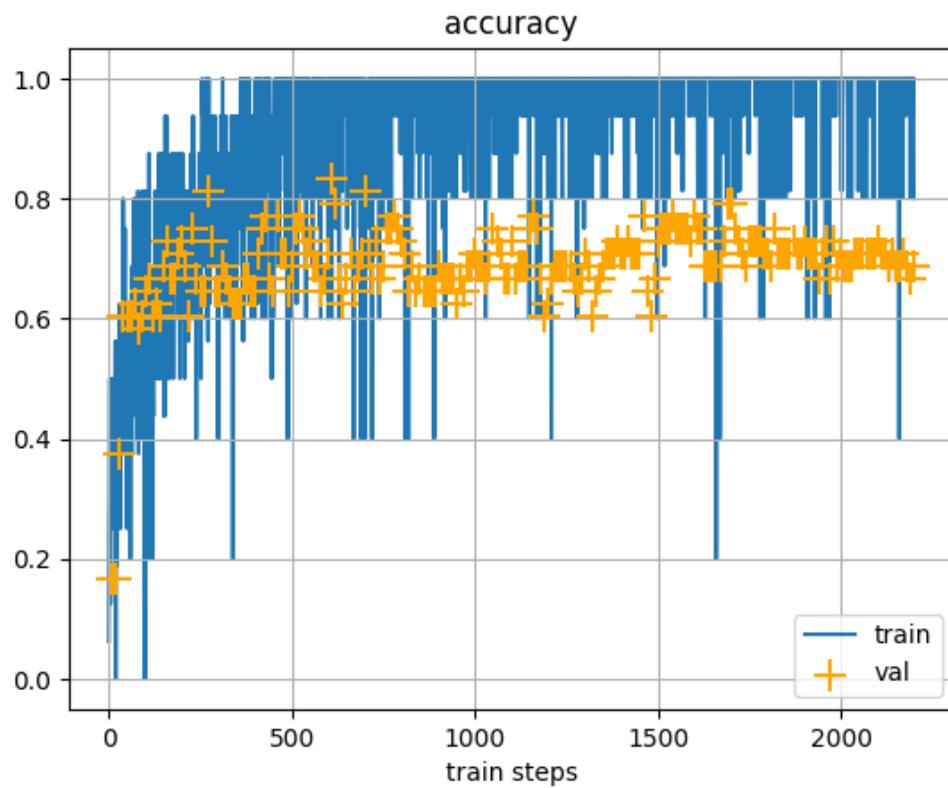


Рис. 13: Accuracy first 220 epochs(Small model)

### 7.3 Маленькая

Количество параметров –  $2.2 * 10^6$

Accuracy – 0.7

Сетка не смогла так хорошо обучиться как средняя, но она пригодится, если нужен очень быстрый инфренс.

Графики – (accuracy-1 13, accuracy-2 14, loss-1 15, loss-2 16)

## 8 Эксперименты со средней сеткой

В этом разделе испытываются разные конфигурации, основанные на средней сетке. Исследуется влияние количества maxpool и полносвязных слоев. В исходном варианте 3 полносвязных и 5 maxpool слоев. Эксперименты упорядочены по количеству параметров в нейросети.

### 8.1 5 maxpool, 1 полносвязный

$1.2 * 10^6$  параметров

accuracy – 0.58

Графики – (accuracy-1 17, accuracy-2 18)

### 8.2 7 maxpool, 4 полносвязных

$2.8 * 10^6$  параметров

accuracy – 0.75

Графики – (accuracy-1 19, accuracy-2 20)

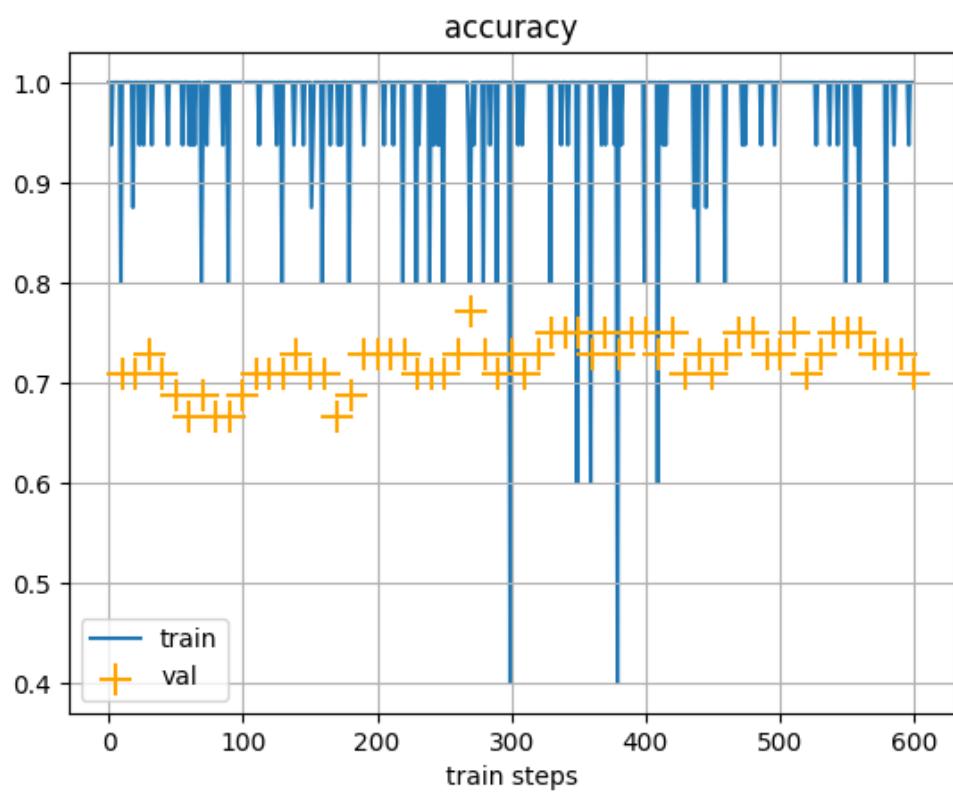


Рис. 14: Accuracy last 60 epochs(Small model)

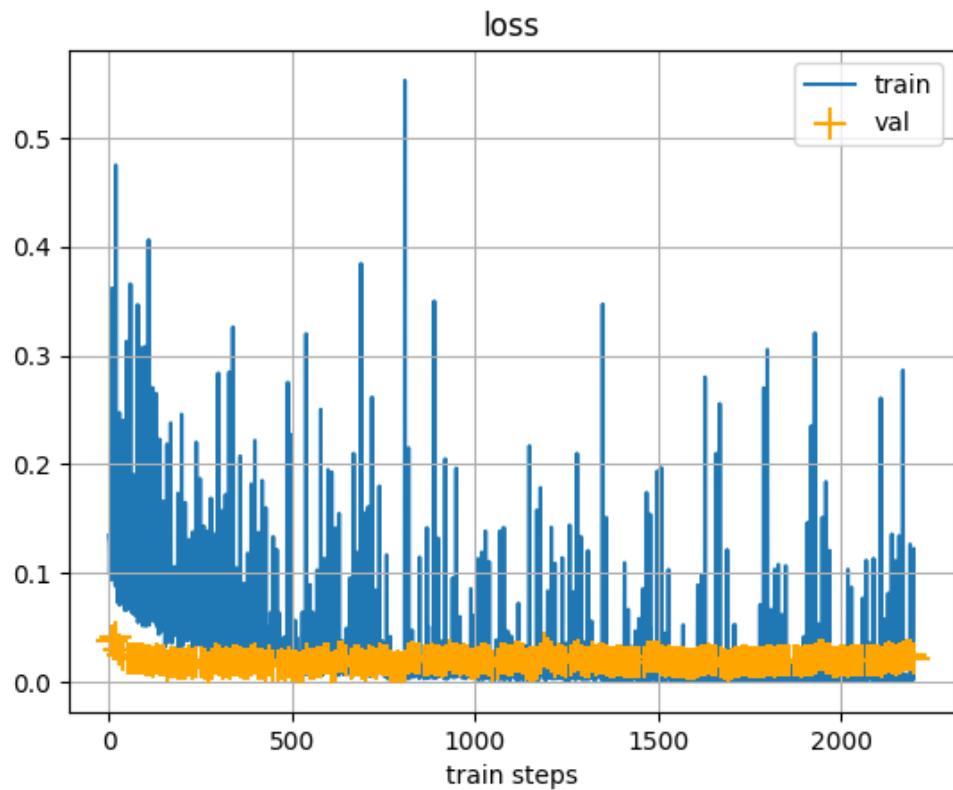


Рис. 15: Loss first 220 epochs(Small model)

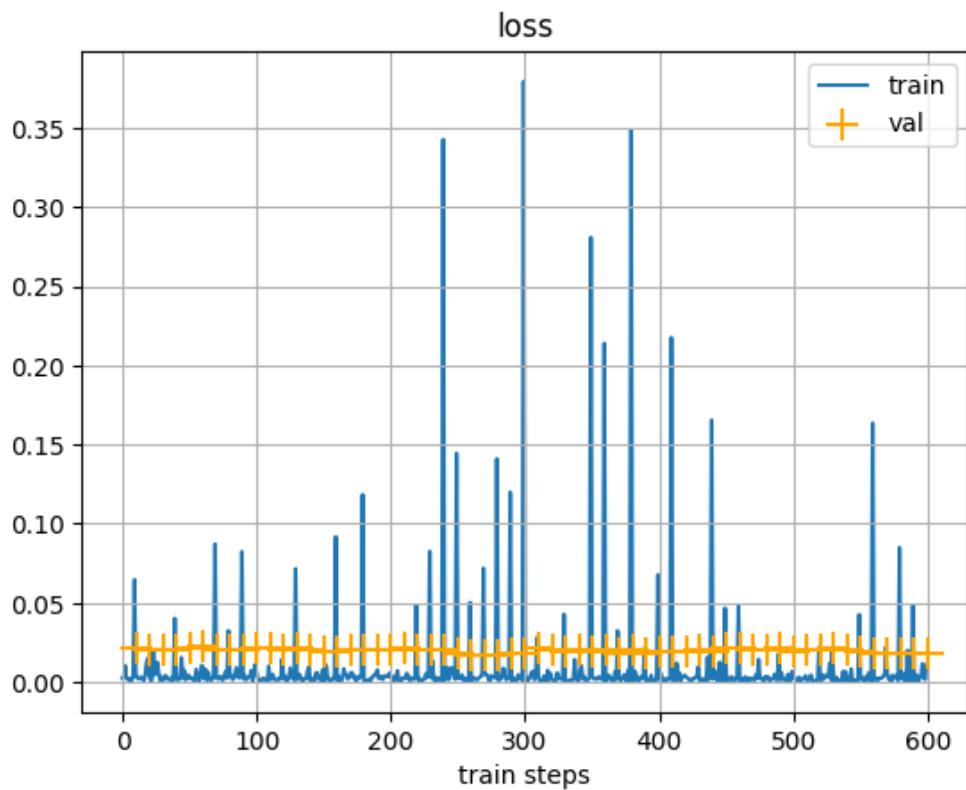


Рис. 16: Loss last 60 epochs(Small model)

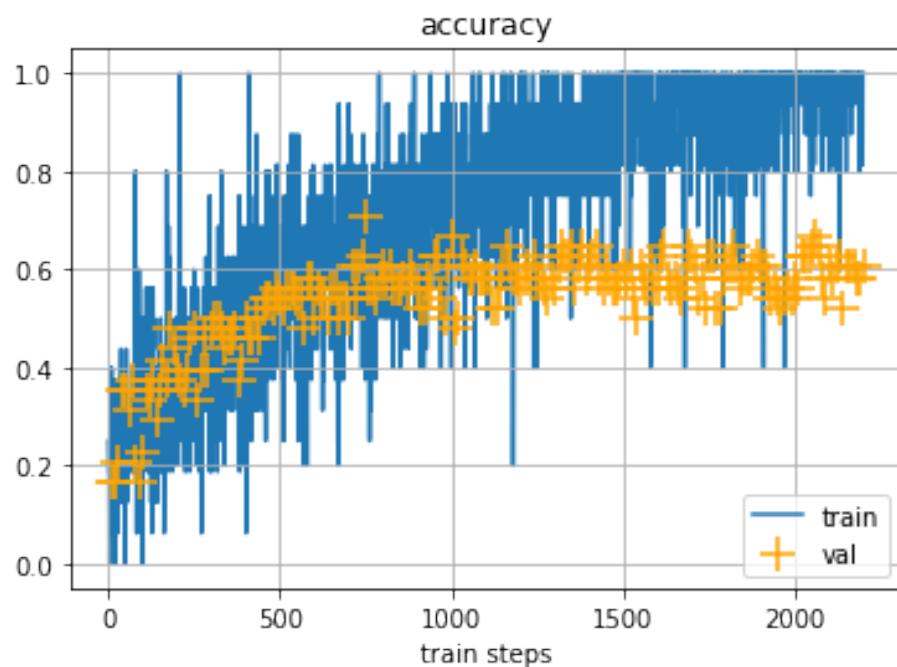


Рис. 17: Accuracy first 220 epochs(5 maxpool, 1 fully-connected)

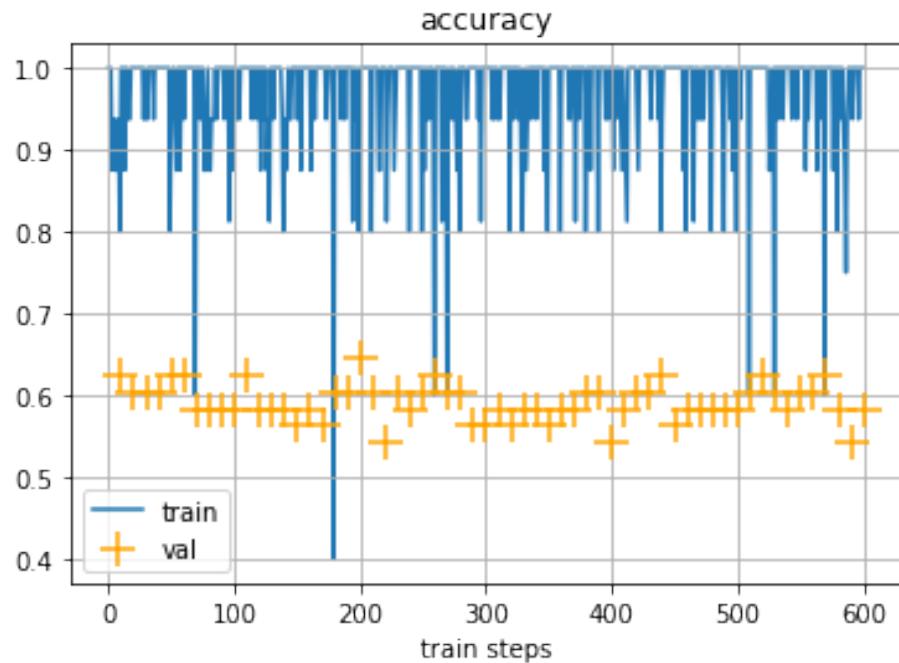


Рис. 18: Accuracy last 60 epochs(5 maxpool, 1 fully-connected)

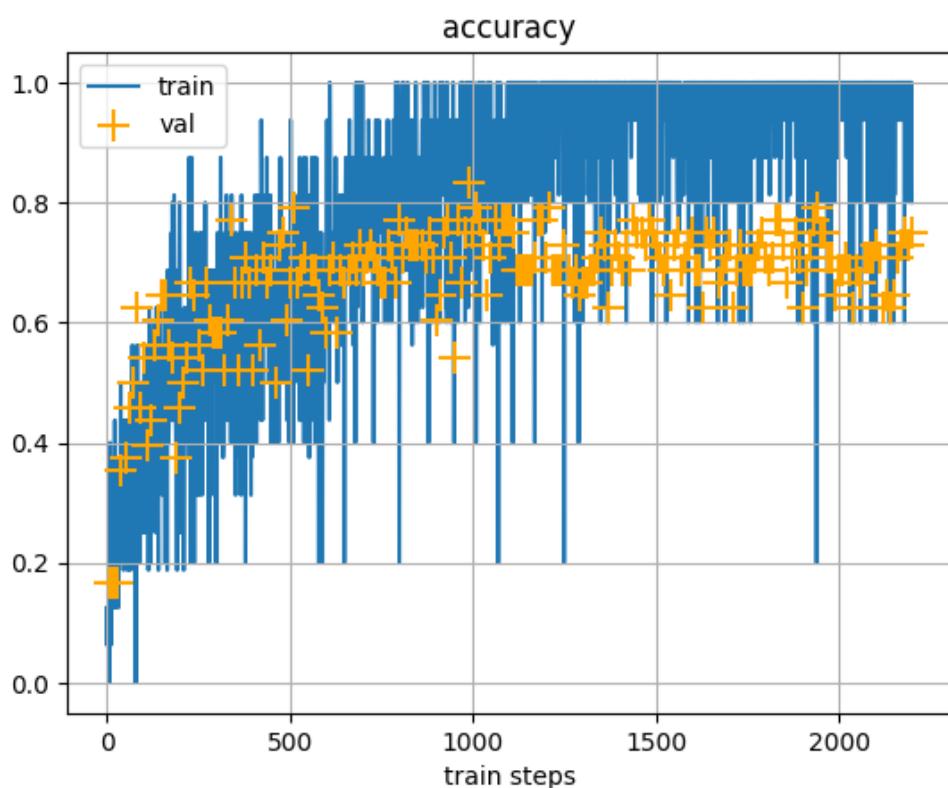


Рис. 19: Accuracy first 220 epochs(7 maxpool, 4 fully-connected)

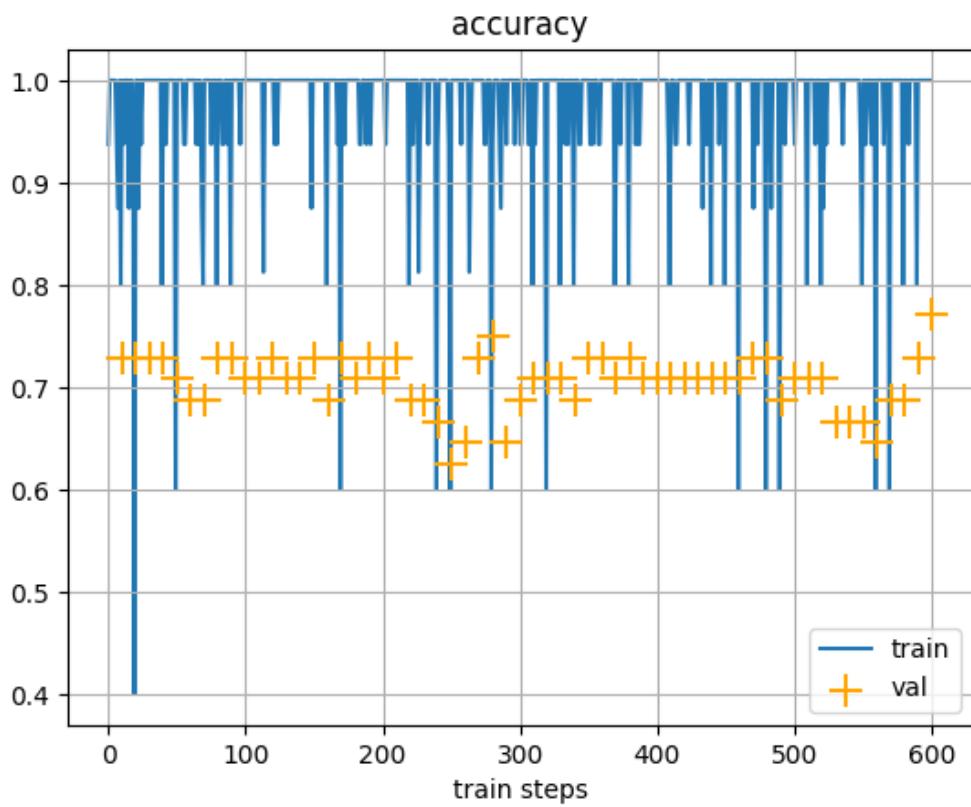


Рис. 20: Accuracy last 60 epochs(7 maxpool, 4 fully-connected)

### 8.3 5 maxpool, 2 полносвязных

$3.2 \times 10^6$  параметров

accuracy – 0.7

Графики – (accuracy-1 [21](#), accuracy-2 [22](#))

### 8.4 6 maxpool, 3 полносвязных

$3.3 \times 10^6$  параметров

accuracy – 0.73

Графики – (accuracy-1 [23](#), accuracy-2 [24](#))

### 8.5 6 maxpool, 4 полносвязных

$5.9 \times 10^6$  параметров

accuracy – 0.82

Графики – (accuracy-1 [25](#), accuracy-2 [26](#))

### 8.6 5 maxpool, 4 полносвязных

$18.5 \times 10^6$  параметров

accuracy – 0.77

Графики – (accuracy-1 [27](#), accuracy-2 [28](#))

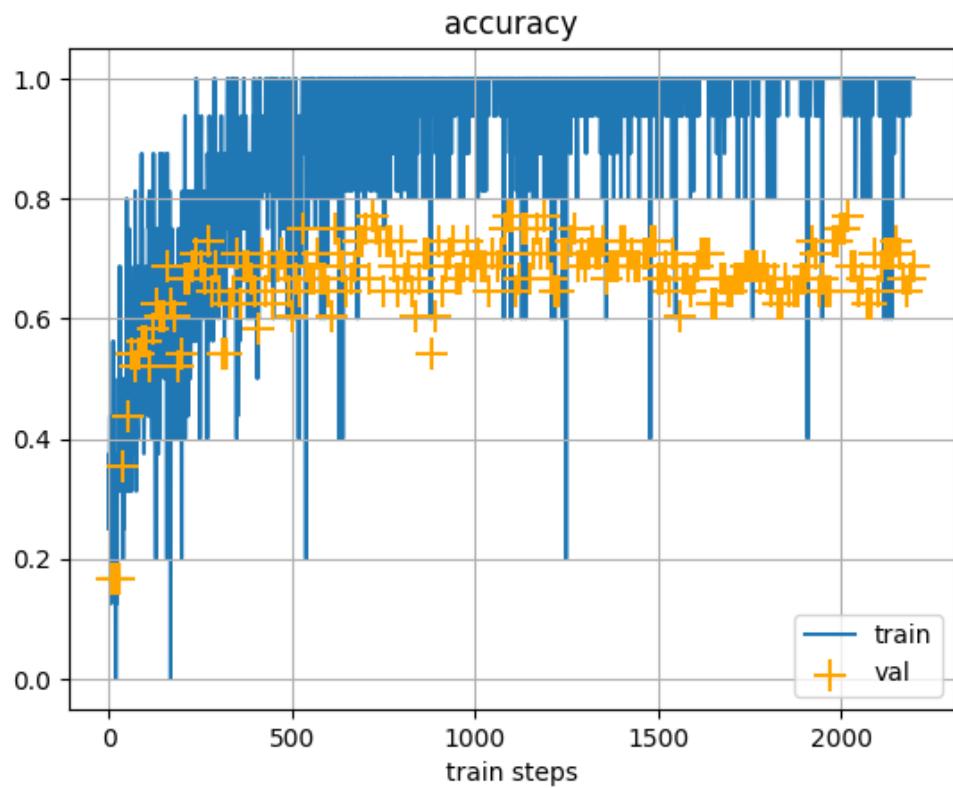


Рис. 21: Accuracy first 220 epochs(5 maxpool, 2 fully-connected)

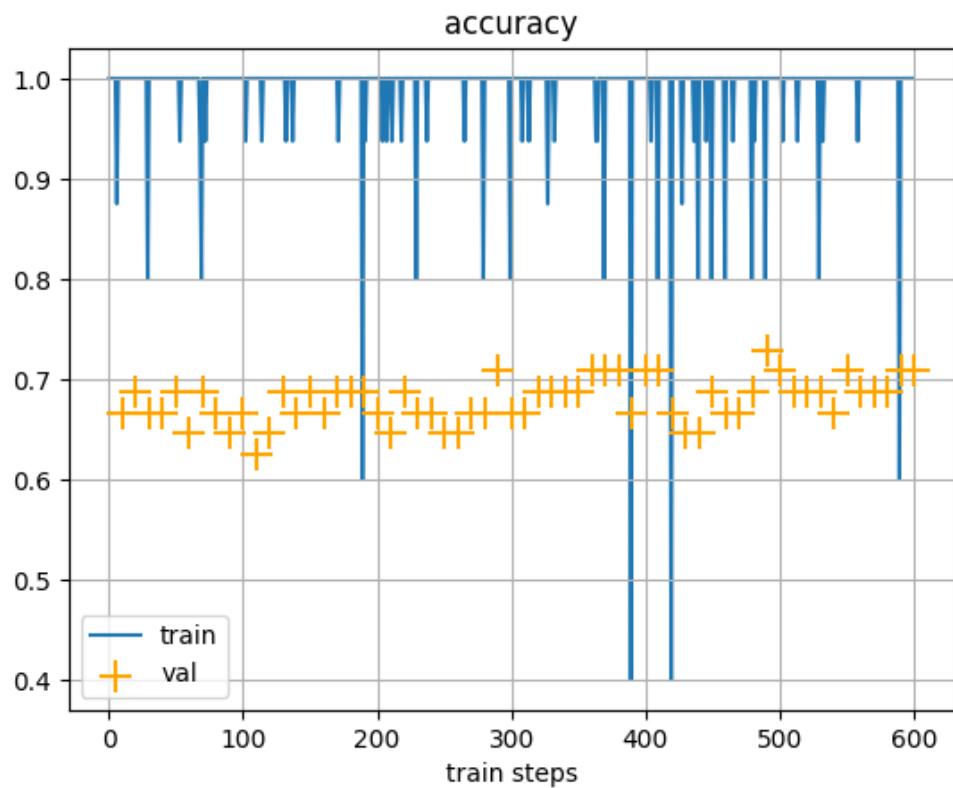


Рис. 22: Accuracy last 60 epochs(5 maxpool, 2 fully-connected)

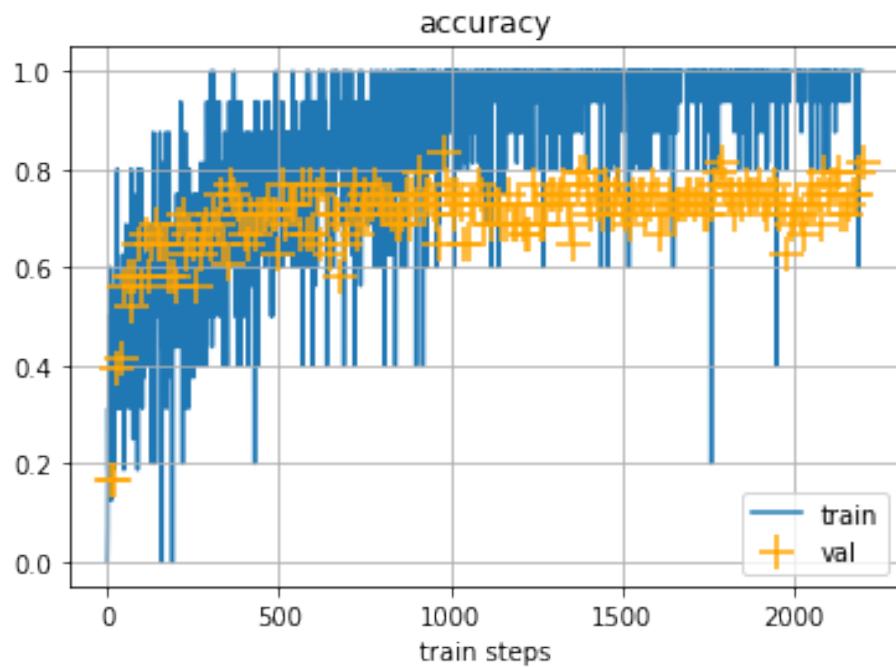


Рис. 23: Accuracy first 220 epochs(6 maxpool, 3 fully-connected)

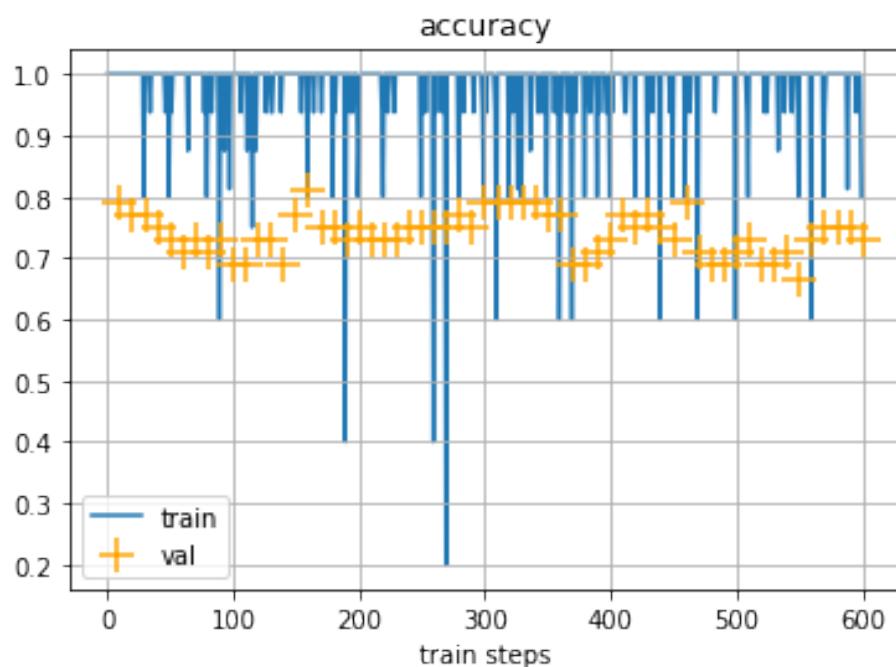


Рис. 24: Accuracy last 60 epochs(6 maxpool, 3 fully-connected)

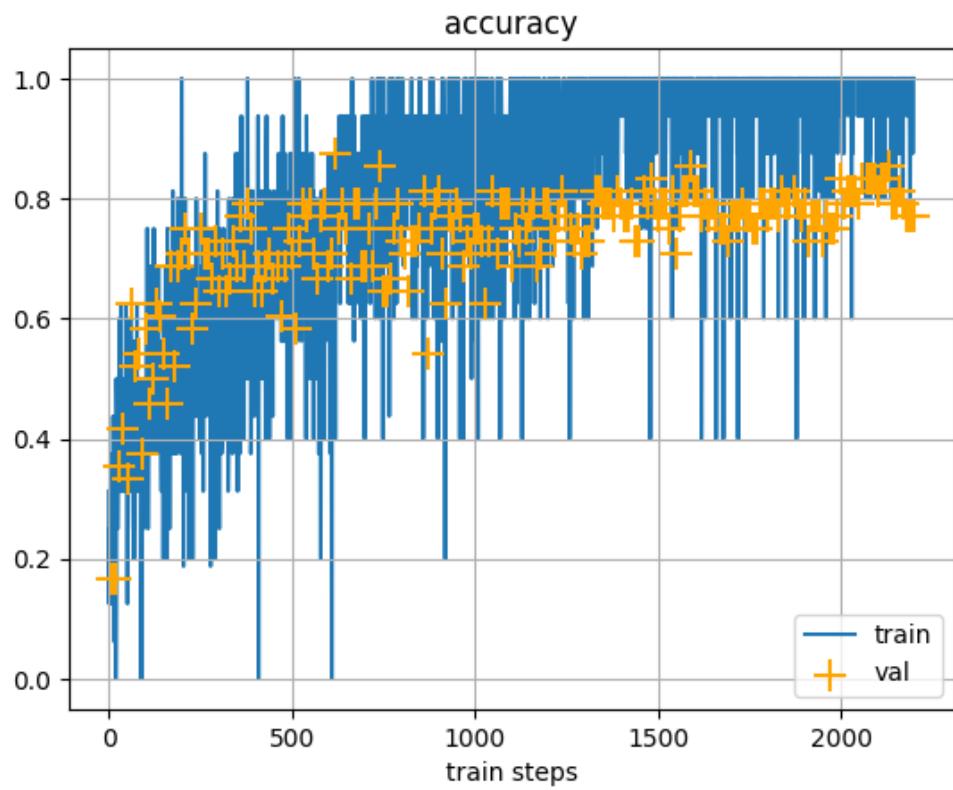


Рис. 25: Accuracy first 220 epochs(6 maxpool, 4 fully-connected)

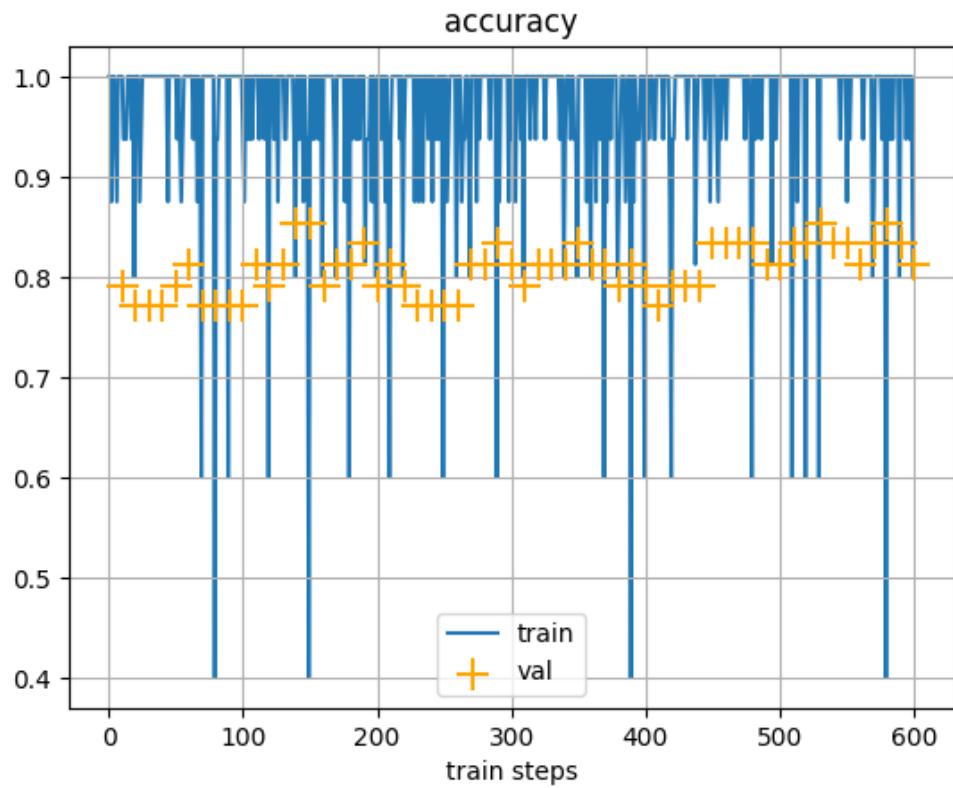


Рис. 26: Accuracy last 60 epochs(6 maxpool, 4 fully-connected)

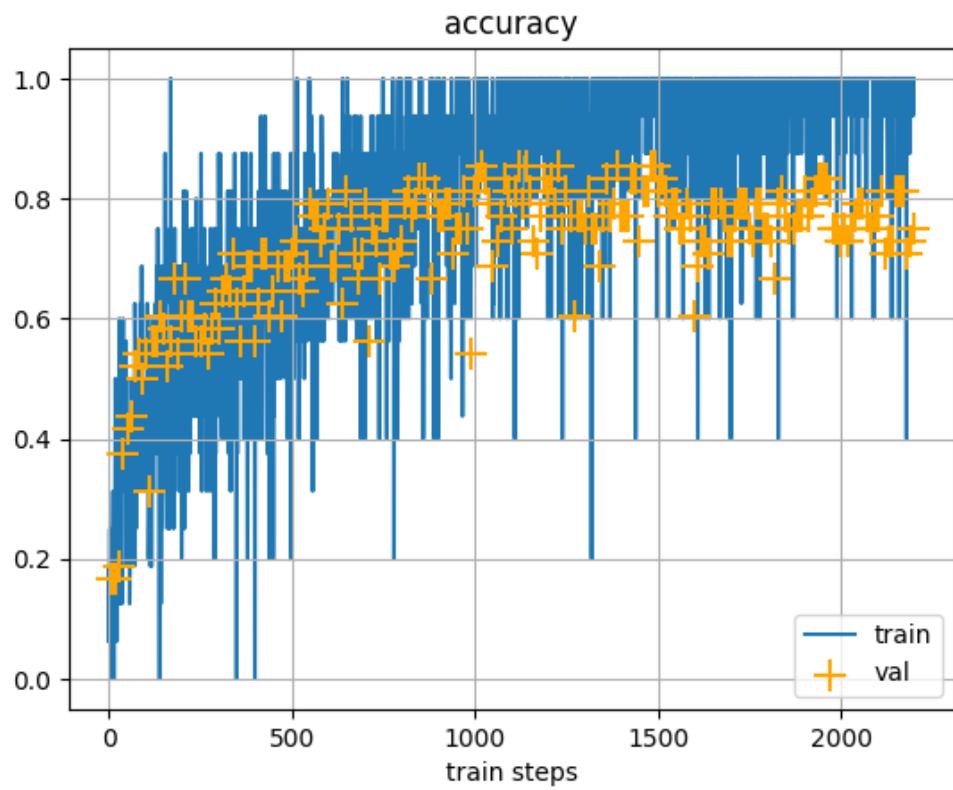


Рис. 27: Accuracy first 220 epochs(5 maxpool, 4 fully-connected)

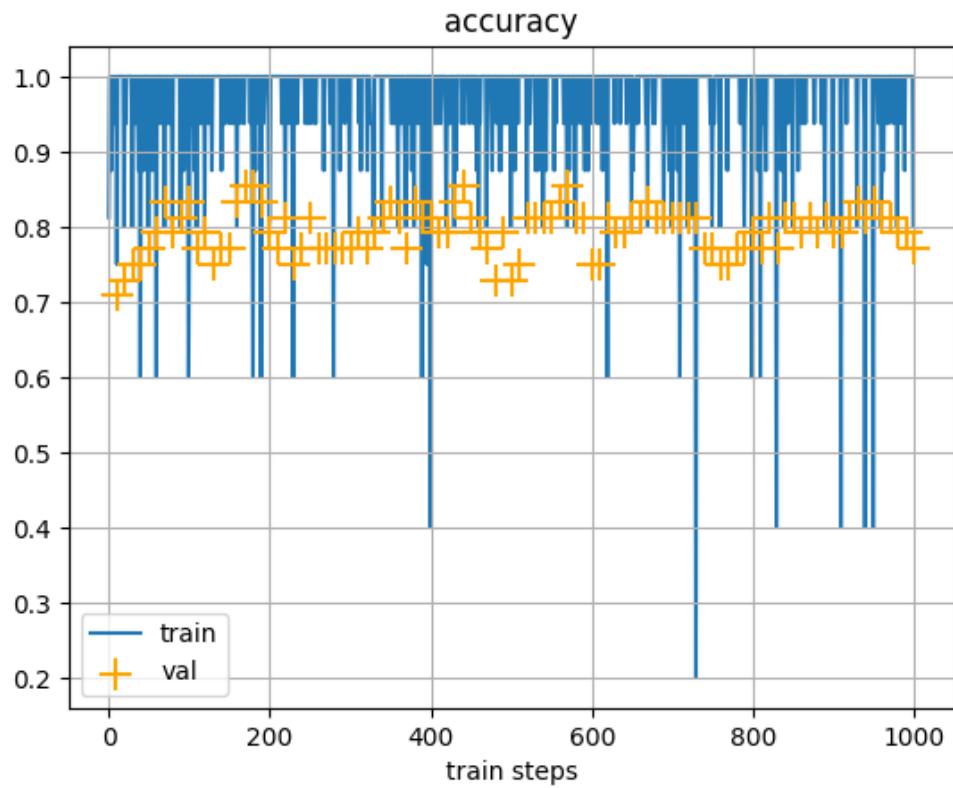


Рис. 28: Accuracy last 60 epochs(5 maxpool, 4 fully-connected)

## 9 Итоги экспериментов

Ниже приведена сводная таблица со всеми результатами экспериментов

Модель	Количество параметров	Accuracy	Ссылки на графики
Большая	$21 * 10^6$	0.68	5, 6, 7, 8
Средняя	$9.6 * 10^6$	0.77	9, 10, 11, 12
Маленькая	$2.2 * 10^6$	0.7	13, 14, 15, 16
5 maxpool, 1 полносвязный	$1.2 * 10^6$	0.58	17, 18
7 maxpool, 4 полносвязных	$2.8 * 10^6$	0.75	19, 20
5 maxpool, 2 полносвязных	$3.2 * 10^6$	0.7	21, 22
6 maxpool, 3 полносвязных	$3.3 * 10^6$	0.73	23, 24
6 maxpool, 4 полносвязных	$5.9 * 10^6$	0.82	25, 26
5 maxpool, 4 полносвязных	$18.5 * 10^6$	0.77	27, 28

Оказалось, что лучшая модель – 6 maxpool, 4 полносвязных. Ноутбук с экспериментами выложен в репозиторий диплома. Для наглядности 29 – архитектура наилучшей модели.

```
model = nn.Sequential([
    nn.Conv2d(3, 16, 3, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    nn.Conv2d(16, 16, 3, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    #1block
    nn.Conv2d(16, 32, 3, padding=1),
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.Conv2d(32, 32, 3, padding=1),
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    #2block
    nn.Conv2d(32, 64, 3, padding=1),
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.Conv2d(64, 64, 3, padding=1),
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    #3block
    nn.Conv2d(64, 128, 3, padding=1),
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.Conv2d(128, 128, 3, padding=1),
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    #4block
    nn.Conv2d(128, 256, 3, padding=1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.Conv2d(256, 256, 3, padding=1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.MaxPool2d(2, 2),
    #5block
    nn.Flatten(),
    nn.Linear(256 * 4 * 4, 1024),
    nn.BatchNorm1d(1024),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(1024, 512),
    nn.BatchNorm1d(512),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(512, 128),
    nn.BatchNorm1d(128),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(128, 6),
])
```

Рис. 29: Best architecture(6 maxpool, 4 fully-connected)

## 10 Результаты

1. Изначально было собрано 73 изображения огурцов из различных теплиц. На каждой из фотографий присутствует один или несколько листьев.

2. Для обнаружения листьев огурцов на собранных изображениях был реализован детектор объектов на основе YOLO из Python-библиотек OpenCV и ultralitics.

3. Несмотря на то, что существует несколько статей о растительных болезнях, таких как статьи Anand.H.Kulkarni и Ashwil Patil R.K. "Применение техники обработки изображений для обнаружения болезней растений"[\[3\]](#) и Aleinikov A.F. "Метод неинвазивного определения грибковых заболеваний клубники"[\[2\]](#), которые сосредоточены на различных растениях, они могут предложить ценные идеи, которые могут быть применены к нашей работе. В частности, они могут предоставить информацию о распространенных болезнях растений, а также о симптомах и признаках, характеризующих эти болезни.

4. После обнаружения листьев огурцов на собранных изображениях, была собрана выборка из 211 фотографий, на каждой из которых для листа указана одна из 5 болезней, или же, он помечен как здоровый. Для обучения нейросети для классификации изображений, вся выборка была разделена на 2: train и val – для обучения и проверки качества.

5. С помощью помеченного набора данных было обучено несколько моделей на основе PyTorch с разными архитектурами. Каждая архитектура немного отличается от других для того, чтобы определить наилучшую. Во время обучения были использованы различные техники, такие как dropout и batch-norm слои, для регуляризации нейронной сети и предотвращения переобучения.

6. Наконец, после обучения, каждая из моделей была проверена на отдельном наборе данных, чтобы измерить ее точность.

Примеры работы программы: [30](#), [31](#)

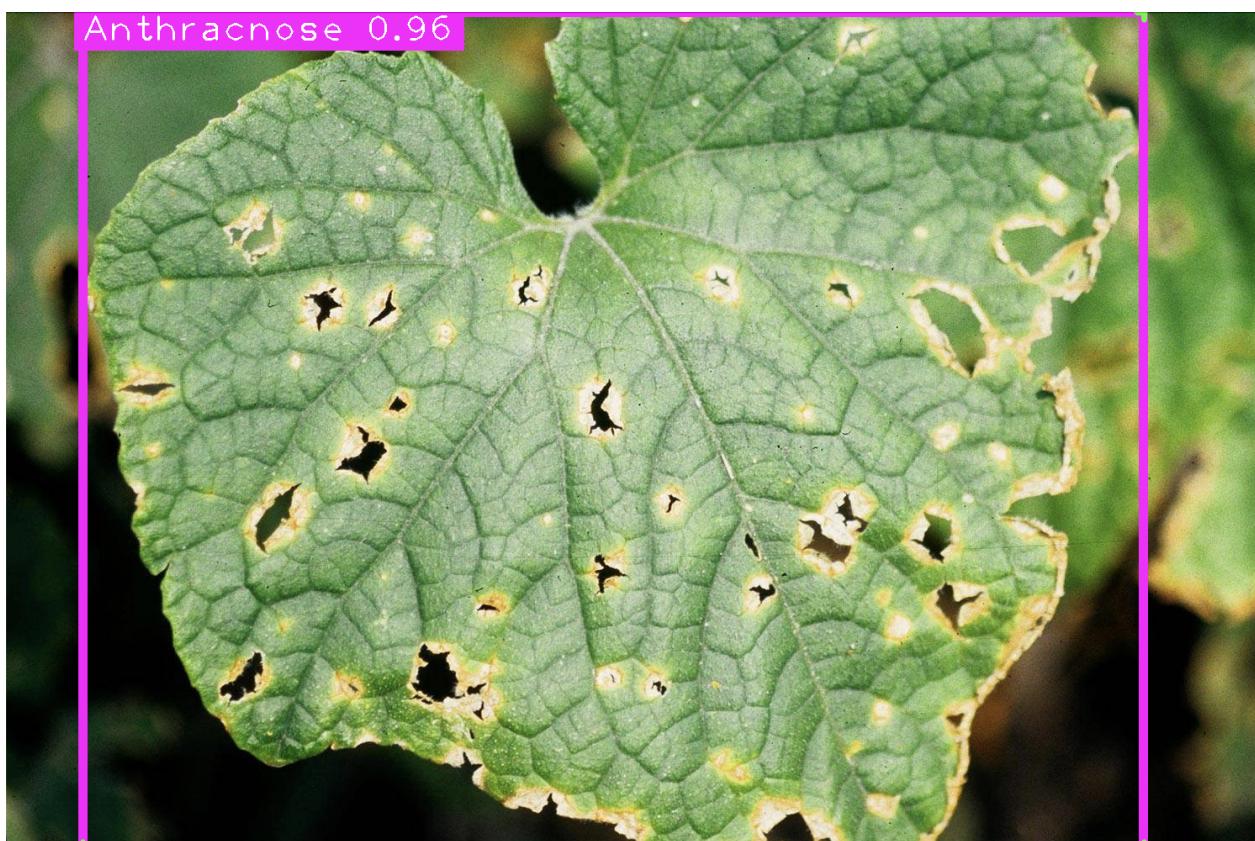


Рис. 30: Example 1



Рис. 31: Example 2

## 11 Руководство по использованию программы

Весь код программы и ноутбук, который использовался для обучения выложен на гитхаб

<https://github.com/nick15431543/diplom>

В этом репозитории есть 7 файлов:

1. requirements.txt – необходимые библиотеки
2. model.py – файл, описывающий архитектуру нейросети-классификатора
3. best\_model – файл с весами нейросети-классификатора
4. yolo – файл с весами нейросети-детектора листьев
5. dir\_counter.py – программа, с помощью которой можно считать количество листов на фотографиях из конкретной директории и определять количество больных.
6. one\_photo.py – программа, которая принимает одну фотографию на вход и показывает на ней листы и вероятности их заболевания.
7. Illness\_recognition\_train.ipynb – ноутбук, в котором проводились эксперименты

Также есть 2 фотографии для примера работы one\_photo.py

Для того, чтобы воспользоваться этими двумя программами:

1. Скачать репозиторий
2. В зависимости от того, какой режим использования предполагается, выбрать один из двух файлов(one\_photo.py и dir\_counter.py) и заменить в указанных там местах директории на те, которые есть локально.

## 12 Направления для дальнейших исследований

Эта технология имеет несколько перспективных направлений для дальнейшего развития:

1. Разработка приложения и веб-сайта, которые облегчат работу с данной программой для обычных пользователей. Создание интуитивного интерфейса и инструкций поможет пользователям более эффективно использовать эту технологию и получать нужную информацию о болезнях растений. Это поможет даже тем, кто не обладает глубокими знаниями в области разведения овощей, справиться с проблемами, связанными с определением и лечением болезней растений.

2. Взаимодействие с представителями отрасли разведения овощей для лучшего понимания проблем, с которыми они сталкиваются при определении болезней. Коммуникация с профессионалами и специалистами в этой области позволит разработчикам технологии получить ценные знания о конкретных болезнях и особенностях их диагностики. Это поможет совершенствовать модель и обеспечивать более точное определение болезней растений.

3. Улучшение качества модели с помощью расширения обучающей выборки. Добавление новых болезней и увеличение количества фотографий, используемых для обучения модели, поможет ей стать более точной и надежной в определении различных состояний болезней растений. Расширение обучающей выборки также может включать разные стадии развития болезней и использование фотографий из разных теплиц. Это поможет модели стать более адаптивной к различным условиям и ситуациям, с которыми могут столкнуться производители овощей.

В итоге, развитие указанных направлений позволит сделать эту технологию более доступной и полезной для обычных пользователей, а также поможет улучшить точность и надежность модели при определении болезней растений. Это приведет к повышению эффективности разведения овощей и улучшению качества урожая.

## 13 Заключение

В заключение, использование техники компьютерного зрения для обнаружения заболеваний растений в последние годы показало большой потенциал. Методы, описанные в различных статьях, такие как фильтр Габора, предложенный Анандом, и подход на основе компьютерного зрения, предложенный Аллейниковым, предлагают многообещающие решения для этой актуальной проблемы. Однако, все еще есть необходимость в дальнейшем исследовании и развитии для улучшения точности и эффективности этих методов. Кроме того, важно учитывать факторы, такие как экономическая эффективность и практичность при выборе метода обнаружения заболеваний растений. Тем не менее, с продолжающимся развитием компьютерного зрения и машинного обучения, будущее обнаружения заболеваний растений выглядит многообещающим.

## Список литературы

- [1] <https://github.com/samuelwestlake/imagelabel?ysclid=lgqoqicf15175119187>.
- [2] Alleinikov.A.F. Method of noninvasive determination of fungal diseases of strawberry. 2018.
- [3] Anand.H.Kulkarni and R.K. Ashwil Patil. Applying image processing technique to detect plant diseases. *IJMER*, 2012.
- [4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- [5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016.
- [6] Jon Rowland. The six common cucumber diseases you need to know about. <https://greenhousecenter.net>, 2021.
- [7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.