СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во время выполнения выпускной квалификационной работы был разработан метод обнаружения сетевых атак с использованием многослойной нейронной сети.

Цель данной работы — демонстрация практической осуществимости спроектированного в ходе выполнения выпускной квалификационной работы метода обнаружения сетевых атак с использованием многослойной нейронной сети.

Для достижения поставленной цели необходимо решить следующие задачи:

— описать постановку задачи;

— привести ограничения на входные и выходные данные;

— описать технологии, с помощью которых был реализован метод обнаружения сетевых атак;

— провести анализ программного комплекса, реализующего интерфейс для взаимодействия с разработанным методом;

— исследовать разработанный метод на применимость.

1 Технологический раздел

В данном разделе приводится постановка задачи, описываются требования к разрабатываемому методу и программному комплексу. Рассматривается архитектура нейронной сети и структура программного обеспечения. Описываются данные для обучения модели.

1.1 Постановка задачи

На рисунке 1.1 приведена постановка задачи в виде IDEF0-диаграммы.

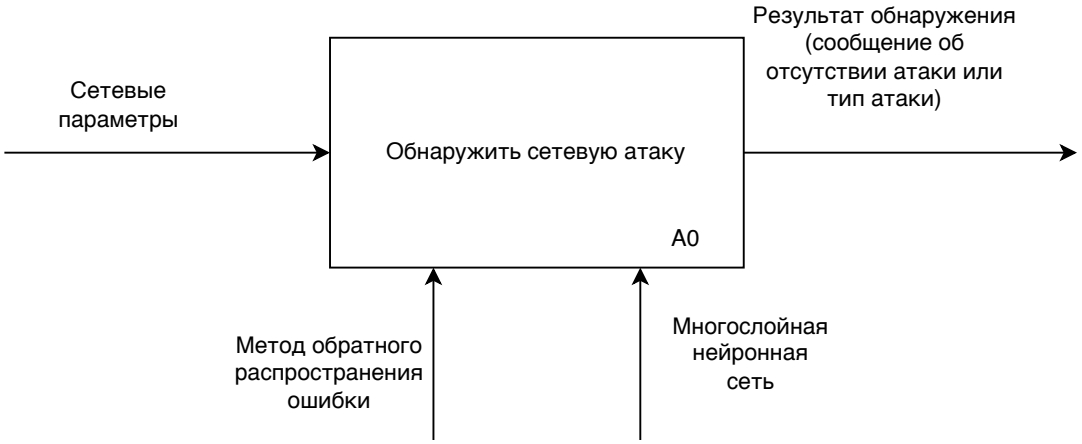

Рисунок 1.1 – Постановка задачи в виде IDEF0–диаграммы

1.2 Требования к разрабатываемому методу

Метод распознавания сетевых атак должен:

— принимать на вход данные о сетевых параметрах в формате CSV;

— определять наличие сетевой атаки во входных данных;

— определять вероятность принадлежности атаки к определенному классу.

1.3 Требования к разрабатываемому программному комплексу

Программный комплекс, реализующий интерфейс для разработанного метода, должен предоставлять:

— возможность загрузки параметров сетевого трафика через графический интерфейс;

— возможность просмотра информации о сетевом трафике;

— возможность просмотра информации о количестве атак и их типах в исходных данных.

1.4 Набор данных CICIDS2017

Набор данных CICIDS2017 содержит доброкачественные и самые современные распространенные атаки. Он также включает в себя результаты анализа сетевого трафика с маркированными потоками на основе временной метки, портов источника и назначения, протоколов и атак. Описание параметров приведено в таблице 1.1.

Таблица 1.1 – Описание параметров датасета CICIDS2017

№	**Название параметра**	**Описание**
1	Flow Duration	Продолжительность потока
2	Source Port	Порт источника
3	Destination Port	Порт назначения
4	Protocol ID	ID протокола
5	Total Fwd Packets	Всего пакетов в прямом направлении
6	Total Backward Packets	Всего пакетов в обратном направлении
7	Fwd Packet Length Min	Минимальный размер пакета в прямом направлении
8	Fwd Packet Length Max	Максимальный размер пакета в прямом направлении
9	Fwd Packet Length Mean	Средний размер пакета в прямом направлении
10	Fwd Packet Length Std	Размер стандартного отклонения пакета в прямом направлении
11	Fwd IAT Total	Общее время между двумя пакетами, отправленными в прямом направлении
12	Fwd IAT Mean	Среднее время между двумя пакетами, отправленными в прямом направлении

13	Fwd IAT Std	Время стандартного отклонения между двумя пакетами, отправленными в прямом направлении
14	Fwd IAT Max	Максимальное время между двумя пакетами, отправленными в прямом направлении
15	Fwd IAT Min	Минимальное время между двумя пакетами, отправленными в прямом направлении
16	Fwd IAT Mean	Среднее время между двумя пакетами, отправленными в прямом направлении
17	SYN Flag Count	Количество пакетов с SYN
18	ACK Flag Count	Количество пакетов с ACK
19	FIN Flag Count	Количество пакетов с FIN
20	CWE Flag Count	Количество пакетов с CWE
21	Bwd Packet Length Min	Минимальный размер пакета в обратном направлении
22	Bwd Packet Length Max	Максимальный размер пакета в обратном направлении
23	Bwd Packet Length Mean	Средний размер пакета в обратном направлении
24	Bwd Packet Length Std	Размер стандартного отклонения пакета в обратном направлении
25	Bwd IAT Total	Общее время между двумя пакетами, отправленными в обратном направлении
26	Bwd IAT Mean	Среднее время между двумя пакетами, отправленными в обратном направлении
27	Bwd IAT Std	Время стандартного отклонения между двумя пакетами, отправленными в обратном направлении
28	Bwd IAT Max	Максимальное время между двумя пакетами, отправленными в обратном направлении
29	Bwd IAT Min	Минимальное время между двумя пакетами, отправленными в обратном направлении

30	Bwd IAT Mean	Среднее время между двумя пакетами, отправленными в обратном направлении
31	Attack Label	Метка атаки

1.5 Структура разрабатываемого программного комплекса

На рисунке 1.2 представлена диаграмма IDEF0 обучения модели.

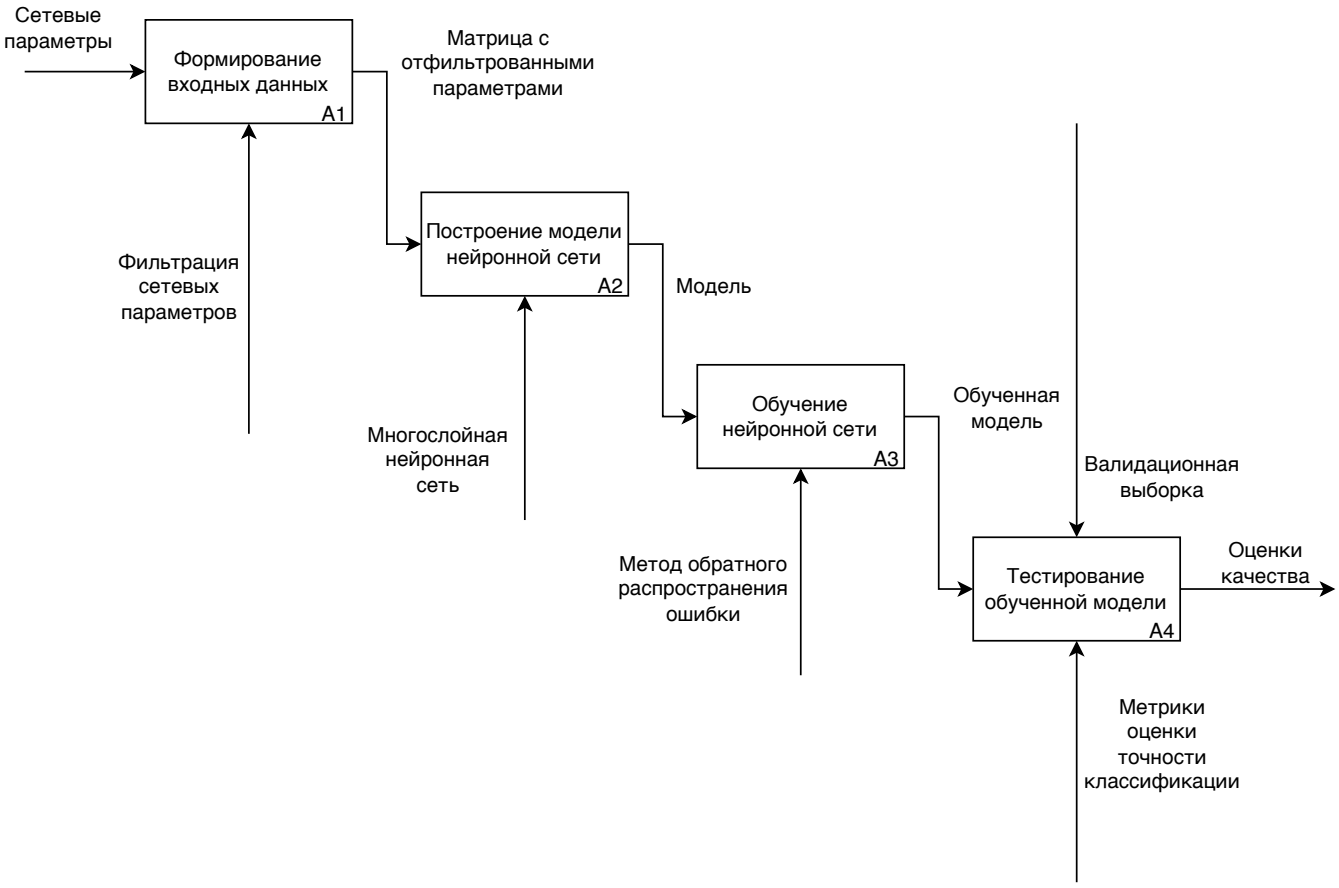

Рисунок 1.2 – IDEF0–диаграмма обучения модели

1.6 Структура многослойной нейронной сети

Поскольку некоторые семейства атак в сети проще обнаружить, чем другие, архитектура многослойной нейронной сети будет модифицирована так, чтобы для легко классифицируемых образцов сеть не должна была оценивать все слои.

Такая архитектура основана на предположении о том, что нейронные сети с большим количеством слоев могут обучаться все более сложным функциям, которые, в свою очередь, требуются только для классификации некоторых конкретных, заведомо трудноклассифицируемых образцов.

В результате сеть строится таким образом, что к каждому слою подключается дополнительный набор нейронов (копия выходных нейронов), позволяющий осуществлять предсказания на каждом уровне.

Данная архитектура представлена на рисунке 1.3

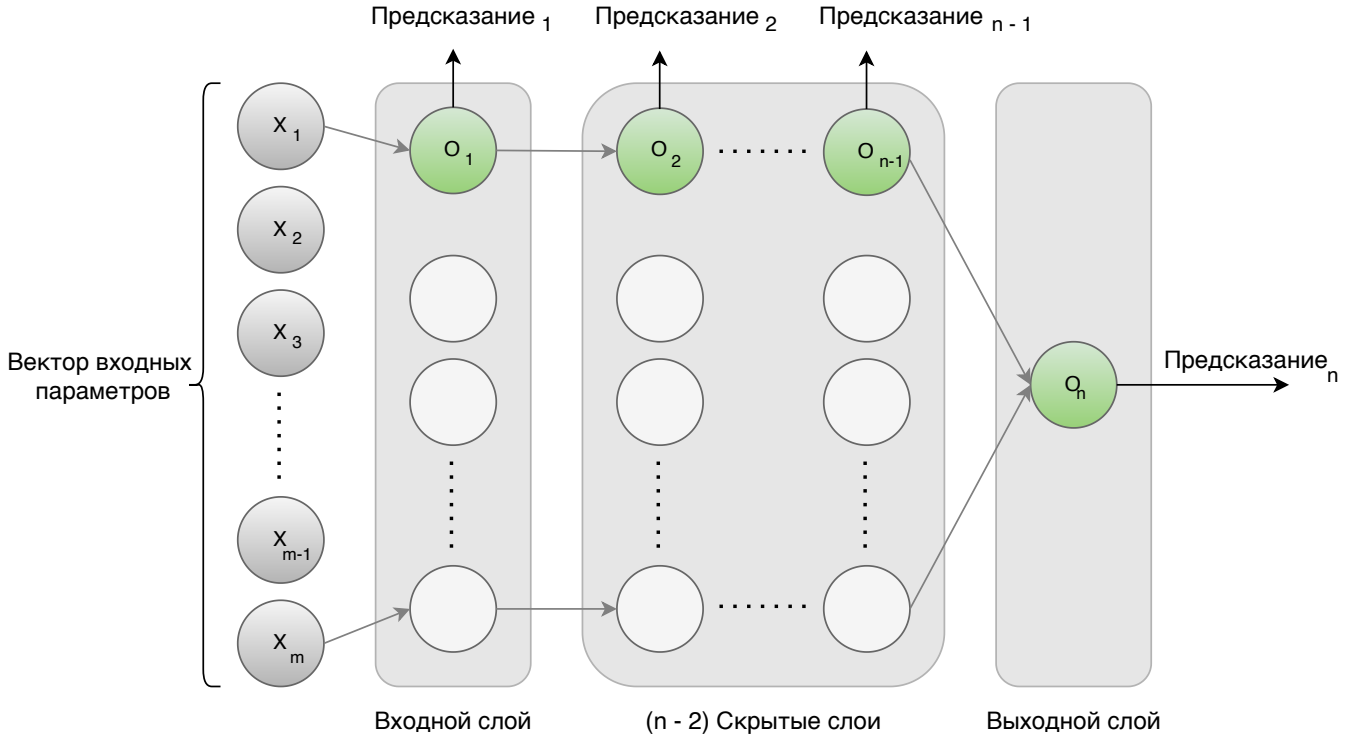

Рисунок 1.3 – Модификация многослойной нейронной сети

Исходные нейроны показаны серым цветом, а выходные нейроны (для каждого слоя) — зеленым. Нейронная сеть продолжает оценивать слои один за другим и выдает значение достоверности прогноза на каждом слое. Значение достоверности определяется как величина, которая отражает, насколько сеть уверена в принадлежности образца к определенному классу. Это число от 0,5 до 1, которое получается путем применения сигмовидной функции к выходному нейрону.

Оценивая значение достоверности, нейронная сеть определяет, следует ли обрабатывать дополнительные слои или полученное в данный момент значение достоверности достаточно высоко. Эта модификация гарантирует,

что более простые образцы будут классифицироваться на ранних слоях, а более сложные образцы будут передаваться в более глубокие слои.

1.7 Средства реализации программного комплекса

1.7.1 Выбор языка программирования

Для написания программного обеспечения будет использоваться язык программирования Python [1] версии 3.11.

Данный выбор обусловлен следующими факторами:

— наличие большого количества библиотек для работы с нейронными сетями;

— кроссплатформенность;

— автоматическое управление памятью.

1.7.2 Используемые библиотеки

При разработке программного обеспечения использовались следующие библиотеки:

— PyTorch [2] — библиотека, для создания и обучения модели нейронной сети. Основным преимуществом PyTorch является гибкость. PyTorch предоставляет множество инструментов для разработки своих моделей, включая выбор оптимизаторов, функций потерь и слоев;

— PyQt5 [3] — библиотека для создания графического интерфейса;

— Scikit-learn [4] — библиотека включает в себя подготовку данных для последующей классификации, а также различные алгоритмы машинного обучения и поддерживает взаимодействие с NumPy;

— Numpy [5] — это библиотека, которая предоставляет функциональность для работы с многомерными массивами и матрицами. Она используется для научных вычислений, обработки данных и машинного обучения;

— Matplotlib [6] — библиотека для визуализация результатов экспериментов и исследований, создание диаграмм и др.

1.8 Минимальные требования к вычислительной системе

Для того, чтобы воспользоваться программным обеспечением, потребуется ЭВМ с предустановленным интерпретатором Python версии 3.11. Также необходимо скачать и установить все использованные в проекте библиотеки.

Требования к использованию определенной операционной системы отсутствуют поскольку ПО реализовано на языке Python, который является кроссплатформенным.

1.9 Формат входных данных

Входными данными являются текстовые файл в формате CSV. CSV — формат текстовых данных в котором каждая строка — это отдельная строка таблицы, а столбцы отделены один от другого специальными символами-разделителями — запятыми. Файл CSV содержит информацию о сетевых параметрах. Файлы данного формата могут быть открыты на любой операционной системе в любом программном обеспечении для работы с текстом.

1.10 Формат выходных данных

Выходные данные модуля классификации можно условно разделить на два типа:

— прогнозирование класса атаки для каждой строки из входного файла;

— визуализация количества атак принадлежащих к различным классов сетевых атак.

Полученные выходные данные выводятся на графический интерфейс программного обеспечения

1.11 Структура разработанного ПО

Структура разработанного ПО изображена на рисунке 1.4.

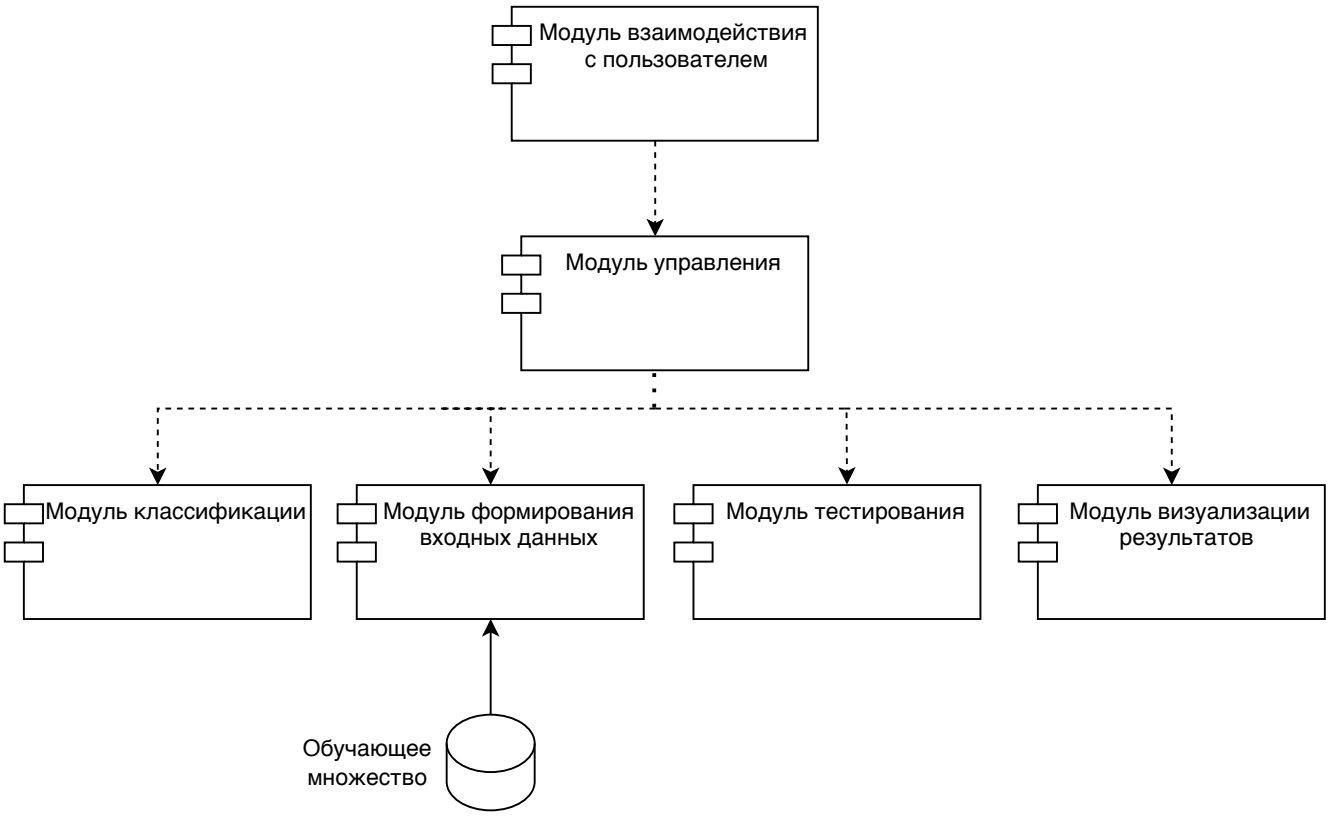

Рисунок 1.4 – Диаграмма структуры ПО

Каждый модуль, который изображен на диаграмме, содержит в себе сгруппированные по функциональному значению соответствующие функции и классы. Модуль взаимодействия с пользователем отвечает за пользовательский интерфейс. Модуль управления связывает модули классификации, формирования данных, визуализации результатов и тестирования а также отвечает за координацию их работы.

1.12 Пользовательский интерфейс

Графический интерфейс разработан при помощи библотеки PyQt5, предоставляющей набор классов и методов для работы с компонентами интерфейса. На рисунке 1.5 представлен пользовательский интерфейс.

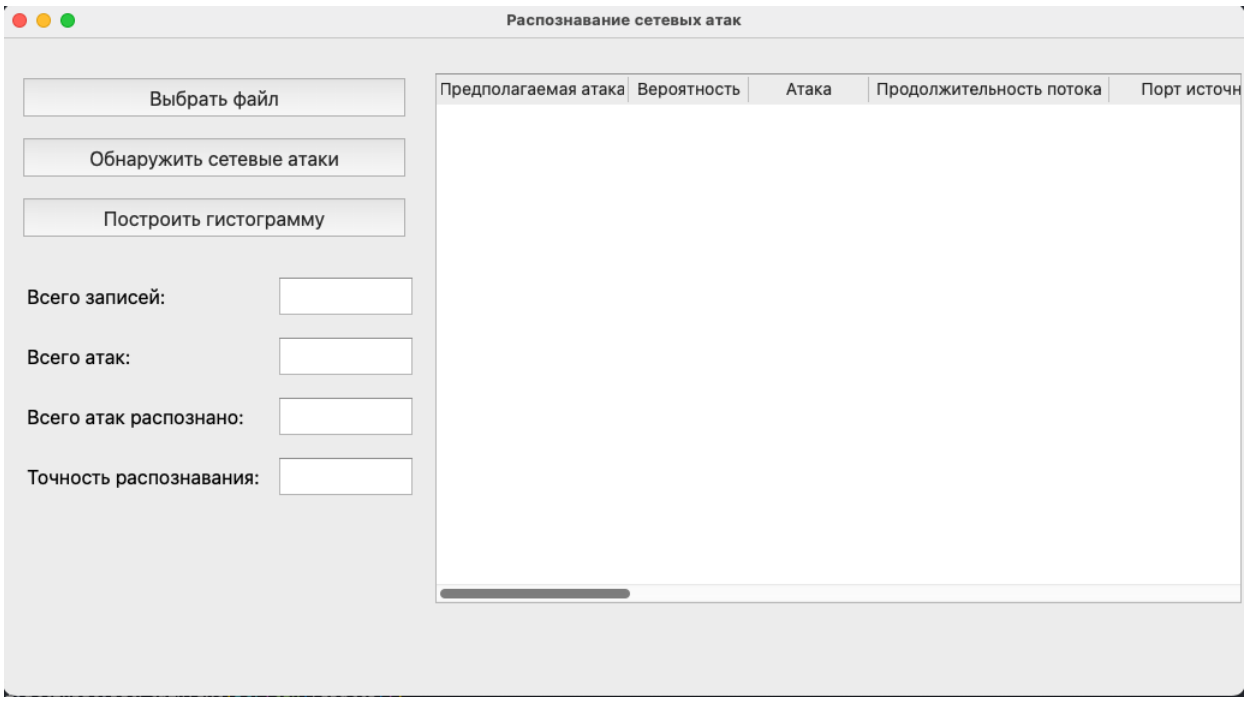

Рисунок 1.5 – Пользовательский интерфейс

В колонке «Предполагаемая атака» отображаются результаты обнаружения сетевых атак во входном CSV-файле. В колонке «Вероятность» отображается вероятнсть принадлежности атаки к соответствующему классу.

Если пользователь, не выбрав входной файл, нажмет на кнопку «Обнаружить сетевые атаки» или на кнопку «Построить гистограмму», то на экран выведется сообщение об ошибке, как показано на рисунке 1.6.

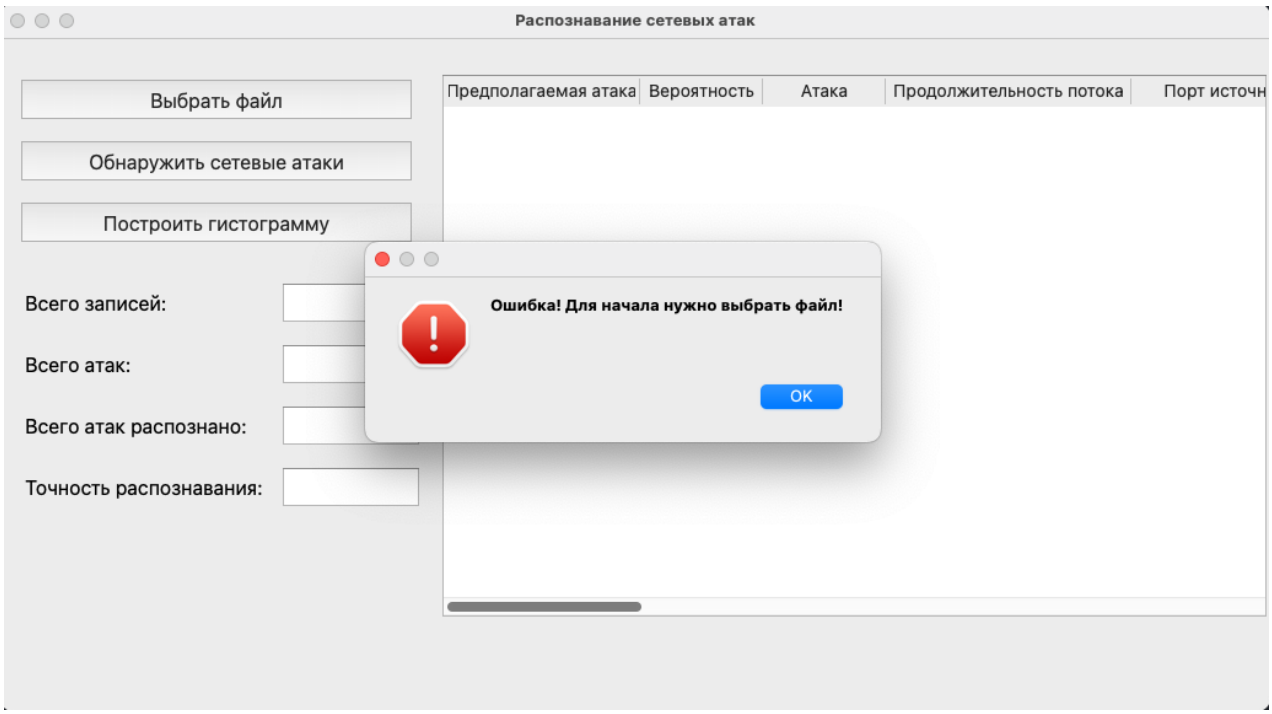

Рисунок 1.6 – Сообщение об ошибке

При нажатии на кнопку «Выбрать файл» открываются файловая система устройства, после чего необходимо выбрать интересующий CSV-файл. После выбора файла, сетевые параметры отобразятся в таблице «Исходные данные».

Если файл выбран, то при нажатии на кнопку «Обнаружить сетевые атаки», запустится алгоритм обнаружения сетевых атак для каждой строки из входного CSV-файла. Результат работы алгоритма представлен на рисунке 1.7.

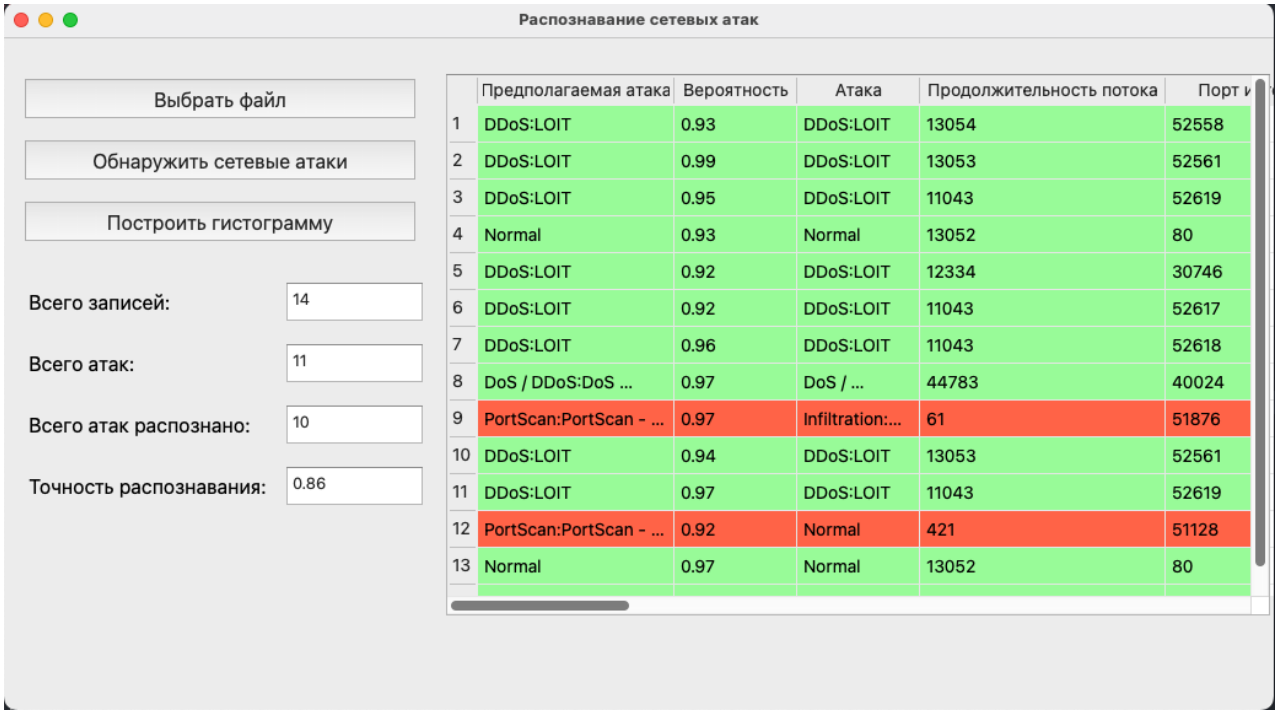

Рисунок 1.7 – Результат работы

При нажатии на кнопку «Построить гистограмму» на экран выведется гистограмма, отображающая количество атак принадлежащих к различным классам сетевых атак.

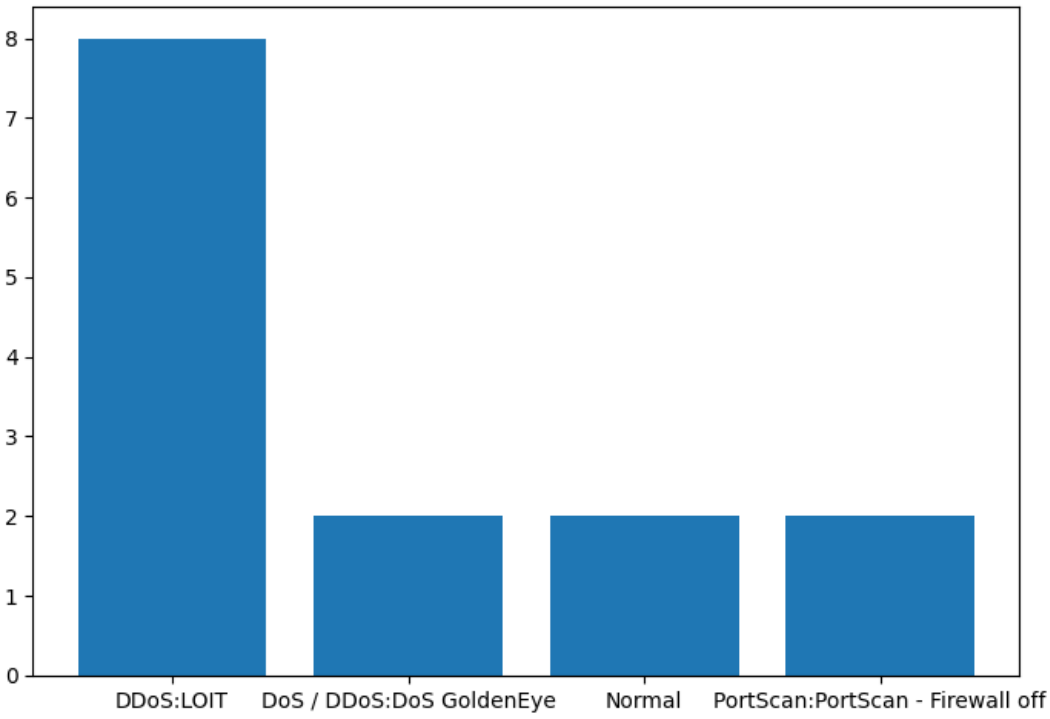

Рисунок 1.8 – Распределение сетевых атак

В листинге 1.1 приведены параметры для взаимодействия с модулем обучения модели.

Листинг 1.1 – Взаимодействие с модулем модели

```
python3 mlearn.py -h

options:
    -h, --help                show this help message and exit
    --dataroot DATAROOT       путь к датасету
    --batchSize BATCHSIZE     размер выборки данных на каждой итерации
    --nLayers NLAYERS         количество слоев
    --layerSize LAYERSIZE     размер слоя
    --niter NITER             количество эпох для обучения
    --net NET                 путь к сохраненной модели
    --manualSeed MANUALSEED   seed для перемешивания данных
    --stoppingWeightingMethod EAGERSTOPPINGWEIGHTINGMETHOD
                              функция весов
```

1.13 Зависимость точности и потерь модели от количества эпох

На рисунках 1.9 и 1.10 представлены результаты обучения модели. Точность модели составила 92.1 процента, а потери составили 4 процента. Стоит

отметить, что уже на десятой эпохе точность модели составляет более 88 процентов.

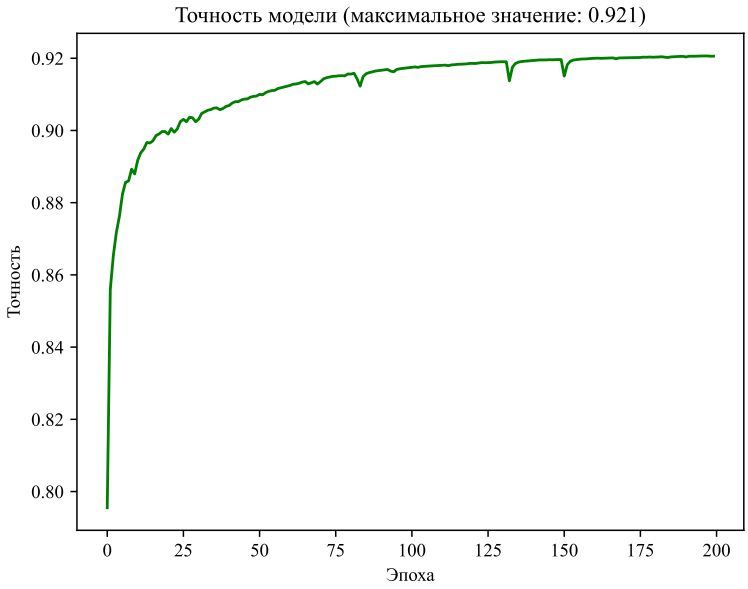

Рисунок 1.9 – Точность модели

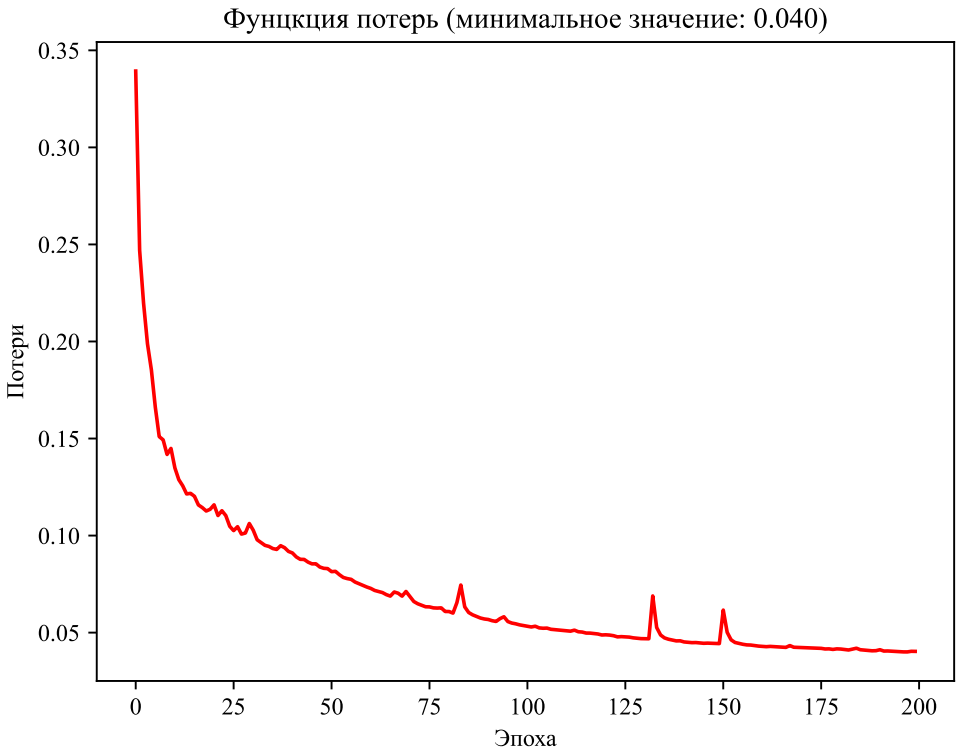

Рисунок 1.10 – Потери модели

1.14 Зависимость точности модели от класса сетевой атаки и количества слоев

На рисунках 1.11 и 1.12 продемонстрирована зависимось точности модели на каждом слое от класса сетевой атаки.

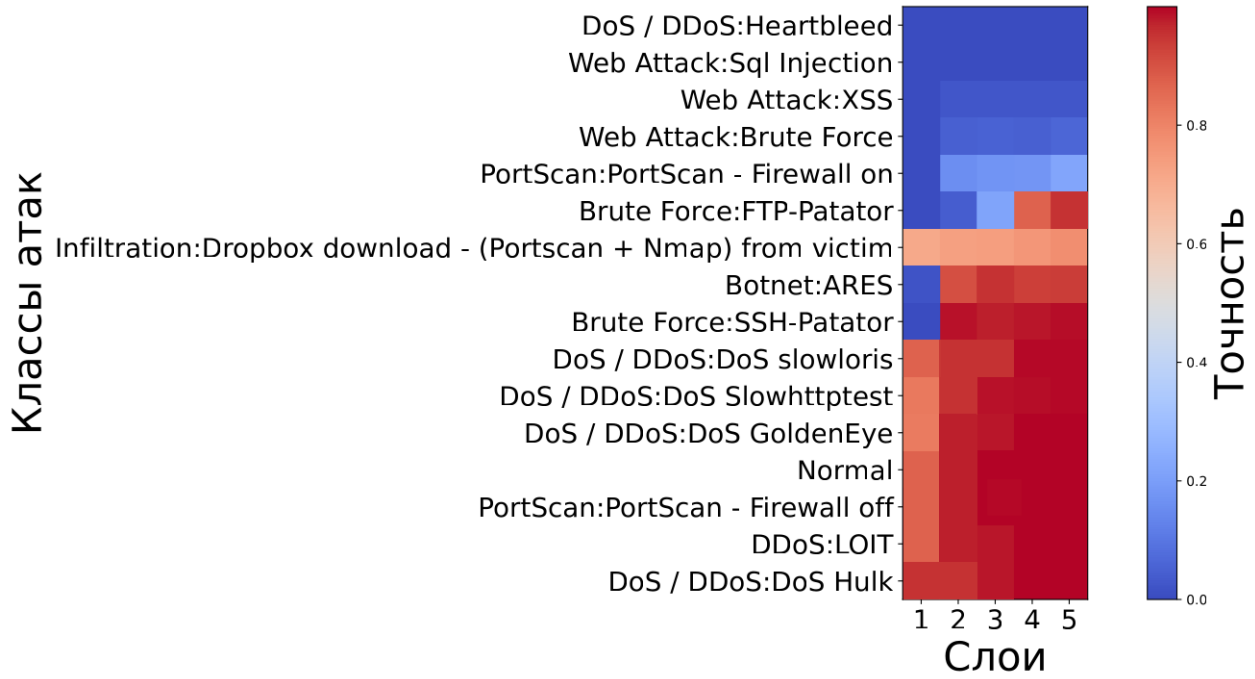

Рисунок 1.11 – 5 слоев × 128 нейронов

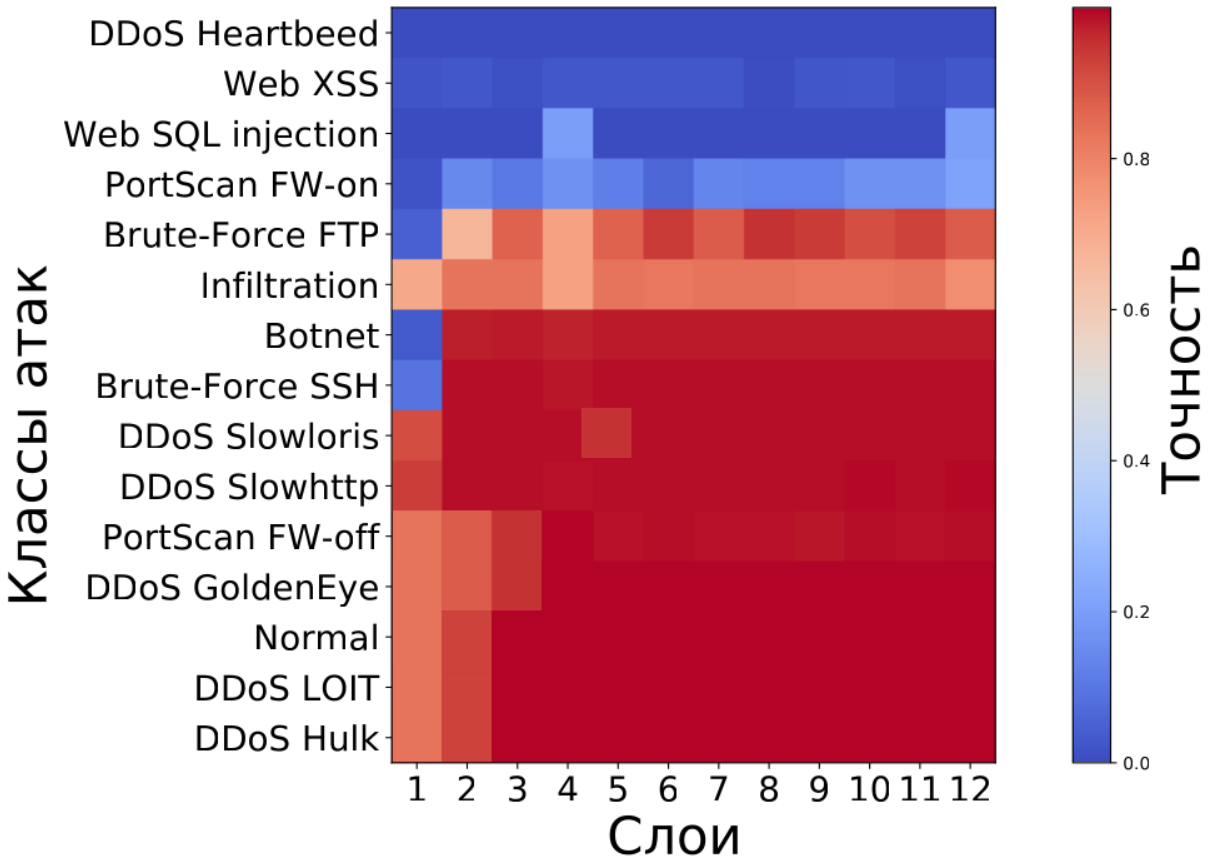

Рисунок 1.12 – 12 слоев × 64 нейрона

На основе полученных данных можно сделать следующие выводы:

— в среднем точность прогнозов возрастает по мере того, как увеличивается количество слоев. Это соответствует действительности, поскольку большее количество слоев обеспечивает большую абстракцию входного пространства и, следовательно, лучшее разделение экземпляров. Однако для получения максимальной точности для некоторых классов атак требуется всего несколько уровней. Это означает, что примененная модификация архитектуры многослойной нейронной сети позволяет значительно сокращать ресурсы при обучении, сохраняя максимальную точность;

— некоторые семейства атак (DDoS Heartbeed, Web-XSS, Web SQL injection, PortScan FW-on) демонстрируют крайне низкую точность. Это происходит по двум причинам:

1. Слишком мало выборок данных классов в обучающем множестве.
2. Потери, оптимизированные для обнаружения таких классов на определенном уровне перезаписываются потерями, оптимизированными для изучения большинства классов что приводит к стиранию информации об таких классах.

1.15 Оценка разработанного программного обеспечения

У разработанного программного обеспечения были выявлены следующие достоинства и недостатки.

Достоинства:

— высокая точность;

— возможность сокращения времени обучения модели за счет уменьшения количества слоев.

Недостатки:

— низкая точность предсказания следующих атак: DDoS Heartbeed, Web XSS, Web SQL injection, PortScan FW-on.

Вывод

Было разработано программное обеспечение, демонстрирующее практическую осуществимость спроектированного в ходе выполнения выпускной квалификационной работы метода обнаружения сетевых атак с использованием многослойной нейронной сети.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Welcome to Python.org [Электронный ресурс]. — Режим доступа: https://www.python.org/ (дата обращения 18.05.2023).
2. PyTorch [Электронный ресурс]. — Режим доступа: https://pytorch.org/ (дата обращения 18.05.2023).
3. PyQt5 [Электронный ресурс]. — Режим доступа: https://pypi.org/project/PyQt5/ (дата обращения 18.05.2023).
4. Scikit-learn [Электронный ресурс]. — Режим доступа: https://scikit-learn.org/stable/ (дата обращения 18.05.2023).
5. NumPy [Электронный ресурс]. — Режим доступа: https://numpy.org (дата обращения 18.05.2023).
6. Matplotlib [Электронный ресурс]. — Режим доступа: https://matplotlib.org (дата обращения 18.05.2023).
7. scikit-learn 1.1.0 documentation [Электронный ресурс]. — Режим доступа: https://scikit-learn.org/stable/modules/svm.html/ (дата обращения 18.05.2023).
8. pandas.DataFrame [Электронный ресурс]. — Режим доступа: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (дата обращения 18.05.2023).

ПРИЛОЖЕНИЕ А

Листинг 1.2 – Модель нейронной сети

```
    class EagerNet(torch.nn.Module):
    def __init__(self, n_input, n_output, n_layers, layer_size, device):
        super(EagerNet, self).__init__()
        self.n_output = n_output
        self.n_layers = n_layers
        self.beginning = torch.nn.Linear(n_input, layer_size+n_output).to(device)
        self.middle = torch.nn.Sequential(*[torch.nn.Linear(layer_size, layer_size+n_output).to(device) for _ in range(n_layers)])
        self.end = torch.nn.Linear(layer_size, n_output).to(device)

    def forward(self, x):
        all_outputs = []
        all_xs = []
        output_beginning = self.beginning(x)
        all_xs.append(x)

        x = torch.nn.functional.leaky_relu(output_beginning[:,:-self.n_output])

        all_outputs.append(output_beginning[:,-self.n_output:])

        for current_layer in self.middle:
            current_output = current_layer(x)
            all_xs.append(x)
            x = torch.nn.functional.leaky_relu(current_output[:,:-self.n_output])
            all_outputs.append(current_output[:,-self.n_output:])

        all_xs.append(x)
        output_end = self.end(x)
        all_outputs.append(output_end)

        return all_outputs, all_xs
```

Листинг 1.3 – Функция классификации

```
def multiclass_predict(net, dataset, device, n_outputs):
    batchSize = 1
    loader = torch.utils.data.DataLoader(dataset, batch_size=batchSize, shuffle=False)

    y_pred_list = [[] for _ in range(n_outputs)]
    y_list = []
    with torch.no_grad():
        net.eval()
        for data, labels in tqdm(loader):
            X_batch = data.to(device)
            outputs, _ = net(X_batch)

            for output_index, y_test_pred in enumerate(outputs):
```

```
                y_pred_softmax = torch.log_softmax(y_test_pred, dim
    = 1)
                _, y_pred_tags = torch.max(y_pred_softmax, dim = 1)
                y_pred_list[output_index].append(y_pred_tags.cpu().
    numpy())

            _, labels = torch.max(labels, dim = 1)
            y_list.append(labels.cpu().numpy())


     y_list = [a.squeeze().tolist() for a in y_list]
     for i, output in enumerate(y_pred_list):
            y_pred_list[i] = [a.squeeze().tolist() for a in output]

     return y_pred_list[-1]
```