МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ ПО ДИСЦИПЛИНЕ
«Модели решения задач в интеллектуальных системах»
Тема: «Классификация текстов программного кода с использованием сверточных нейронных сетей»

КП.ИИ-21.210560-40 03-01

Листов: 14

Выполнил:
студент 4-го курса,
ФЭИС,
группы ИИ-21
Кирилович А.А
Проверил:
Головко В.А.

Брест 2024

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

В современном мире стремительного развития технологий обработки и анализа данных задача автоматической классификации текстовых данных, включая языки программирования, становится особенно актуальной. Она открывает новые возможности для обучения нейронных сетей, повышения безопасности информационных систем и улучшения пользовательского опыта в мессенджерах.

Классификация программного кода играет ключевую роль в создании структурированных данных для обучения крупных языковых моделей. Эти модели, такие как трансформеры, требуют чистых и четко классифицированных данных, чтобы эффективно различать текст на разных языках и работать со смешанными типами информации. Применение методов классификации позволяет не только улучшить качество таких моделей, но и значительно ускорить подготовку данных.

Кроме того, растет значение анализа кода, передаваемого через текстовые сообщения в мессенджерах, таких как Telegram. Увеличение объема такого рода информации создает как новые возможности для сотрудничества разработчиков, так и риски, связанные с вредоносными скриптами и фишингом. Автоматическая классификация помогает защитить пользователей, идентифицируя и блокируя потенциально опасный контент.

Разработка технологий, способных отличать программный код от обычного текста, также важна для повышения уровня кибербезопасности. Такие системы могут применяться для фильтрации вредоносных сообщений, предотвращения атак с использованием кода и защиты информационных систем.

Таким образом, создание методов классификации языков программирования с использованием сверточных нейронных сетей не только повышает эффективность анализа данных, но и способствует решению задач безопасности и оптимизации процессов в сфере информационных технологий.

2 ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является разработка системы классификации текстовых данных с целью определения языка программирования, основанной на методах глубокого обучения. В качестве базовой модели будет использоваться предварительно обученная сверточная нейронная сеть, которая будет дообучаться (файнтюниться) на специализированном датасете, содержащем фрагменты кода на различных языках программирования.

Задачи, которые необходимо решить в рамках проекта:

1. **подготовка набора данных;**
   Необходимо собрать и подготовить набор текстов, содержащих примеры программного кода на различных языках, а также отфильтровать их от нерелевантной информации. Важно учесть разнообразие языков программирования, а также сбалансированность данных по количеству примеров для каждого языка. Также требуется подготовить текстовые данные для последующего подачи в модель.

2. **выбор архитектуры базовой модели;**
   Проанализировать сущесвтвующие модели свертночных нейронных сетей для классификации изображений и выбрать подходящую для данной работы.

3. **обучение (файнтюнинг) предобученной модели;**
   Дообучить предобученную модель на подготовленных данных с примерами программного кода. Это позволит использовать уже усвоенные моделью особенности обработки изображений и адаптировать её к специфике классификации языков программирования.

4. **оценка качества работы модели;**
   После обучения модели необходимо провести её тестирование на независимом наборе данных используя метрики, такие как accuracy, F1-score, precision и recall.

3 ВЫБОР И ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ИНСТРУМЕНТОВ

Для решения задачи классификации языков программирования с использованием нейронных сетей были выбраны следующие инструменты и библиотеки:

- **requests:** это библиотека для работы с HTTP-запросами, которая позволяет удобно скачивать данные с веб-ресурсов. Она является полезной для сбора данных из открытых источников, таких как GitHub, где можно найти репозитории с примерами кода на различных языках программирования. Requests предоставляет простой и интуитивно понятный интерфейс для работы с HTTP-API, что делает процесс извлечения данных быстрым и эффективным;
- **Pillow:** библиотека для обработки изображений, которая позволяет работать с графическими файлами в Python. Она полезна для преобразования текстовых данных в изображения, что является важным этапом при использовании сверточных нейронных сетей, которые эффективно работают с визуальной информацией. Pillow поддерживает множество форматов изображений и предоставляет богатый функционал для обработки, что делает её удобным инструментом для работы с изображениями;
- **PyTorch:** один из самых популярных фреймворков для разработки и обучения нейронных сетей. PyTorch предоставляет широкие возможности для работы с различными типами архитектур, включая сверточные нейронные сети, и поддерживает эффективное использование GPU. Его гибкость и высокопроизводительные вычисления делают его идеальным инструментом для разработки сложных моделей, включая те, которые требуют тонкой настройки и оптимизации;
- **Kaggle:** платформа для работы с данными и машинным обучением. Kaggle предоставляет удобные средства для хранения и совместного использования датасетов, а также для обучения и тестирования моделей. Платформа поддерживает работу с вычислительными мощностями в облаке, что позволяет эффективно обучать модели на больших объемах данных без необходимости иметь собственные ресурсы. Kaggle также предоставляет инструменты для сохранения результатов и обмена ими с сообществом;

Эти инструменты обеспечивают гибкость, эффективность и производительность на различных этапах работы с данными и моделями, что делает их оптимальным выбором для решения задачи классификации языков программирования.

4 ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

В данном разделе описаны этапы обучения модели для классификации текстов программного кода с использованием сверточных нейронный сетей, а также процесс подготовки данных.

4.1 Подготовка данных

Подготовка данных — один из ключевых этапов в процессе обучения нейронной сети. Для эффективной классификации языков программирования, на основе данных, собранных с GitHub, были выполнены следующие шаги:

- **скачивание данных:** для сбора данных использовалась библиотека requests, с помощью которой были скачаны файлы с кодом на 28 языках программирования, включая такие популярные языки, как Python, Java, C++, JavaScript и другие. Эти языки были выбраны на основе их распространенности и представительности в различных областях программирования. В дополнение к этому, был создан специальный класс для языков, не вошедших в основной список, а также для текстовых данных, не относящихся к программированию;
- **объединение данных в файлы:** каждый язык программирования был помещен в отдельный файл, который содержал код на соответствующем языке. Все файлы были объединены в один большой файл для каждого класса языкпрограммированияилитекст. Это объединение помогло организовать данные и упростить дальнейшую обработку;
- **разделение данных:** для подготовки данных к обучению, все файлы были случайным образом разделены на фрагменты длиной от 64 до 1024 символов. Это важно для того, чтобы модель могла обрабатывать тексты разной длины, а также для увеличения разнообразия входных данных. В случаях, когда файл содержал меньше 1024 символов, его содержимое было дополнено путем дублирования, чтобы добиться нужной длины и обеспечить постоянство входных данных для нейронной сети;
- **преобразование в изображения :** следующим шагом была обработка данных с использованием библиотеки Pillow. Каждый файл, представляющий код на определенном языке программирования или текст, был преобразован в изображение. Это преобразование позволило представить текстовую информацию в визуальной форме, что является необходимым для работы сверточных нейронных сетей.

4.2 Выбор архитектуры нейронной сети

Для задачи классификации языков программирования, представленных в виде изображений, была выбрана модель **EfficientNetV2-S**, оригинальная архитектура которой представлена в таблице 4.1. Это небольшая версия модели **EfficientNetV2**, которая предобучена на датасете ImageNet-21k и затем дополнительно дообучалась на ImageNet-1k для задач классификации изображений. Данная модель показывает хороший баланс между временем обработки данных и достигаемой точностью. Выбор данной модели обусловлен ее небольшим размером, неплохими результатами классификации по сравнению с другими моделями в State of Art, а также использованием механизмов внимания, что может положительно сказаться на классификации языков, где важны мелкие детали. Для доообучения последний слой модели был заменен на полносвязный слой, соответствующий количеству классов в задаче классификации языков программирования. Этот подход позволяет адаптировать предобученную архитектуру EfficientNetV2-S под специфику новой задачи, сохранив преимущества предварительного обучения на обширных датасетах упомянутых выше.

Этап	**Оператор**	**Каналы**	**Слои**
0	Conv3x3	24	1
1	Fused-MBConv1, k3x3	24	2
2	Fused-MBConv4, k3x3	48	4
3	Fused-MBConv4, k3x3	64	4
4	MBConv4, k3x3, SE0.25	128	6
5	MBConv6, k3x3, SE0.25	160	9
6	MBConv6, k3x3, SE0.25	256	15
7	Conv1x1 & Pooling & FC	1280	1

Таблица 4.1 – Архитектура EfficientNetV2-S

Как видно из таблицы 4.1, основными блоками в архитектуре EfficientNetV2-S я вляются **Fused-MBConv** и **MBConv**.

Fused-MBConv представляет собой упрощённый мобильный блок свёртки, который объединяет две операции (Depthwise и Pointwise свёртки) в одну стандартную свёртку. Это позволяет уменьшить количество вычислений и увеличить производительность, что особенно важно для задач, выполняемых на устройствах с ограниченными ресурсами. Блок состоит из двух основных этапов. Сначала применяется свёртка 3x3 с нормализацией BatchNorm и активацией SiLU, которая извлекает пространственные признаки. Затем используется свёртка 1x1 с нормализацией BatchNorm, предназначенная для уменьшения размерности каналов и снижения вычислительной сложности.

В отличие от Fused-MBConv, блок MBConv реализует более сложный подход, включающий последовательное применение Pointwise и Depthwise свёрток. Сначала происходит расширение каналов с помощью свёртки 1x1, что позволяет извлекать сложные признаки из входных данных. Далее используется Depthwise свёртка 3x3, которая обрабатывает каждый канал независимо, значительно сокращая количество параметров при сохранении пространственной информации. После этого в работу включается блок Squeeze-and-Excitation (SE), который усиливает наиболее важные признаки, выделяя значимые каналы с помощью глобального контекста. Это достигается через адаптивное усреднение, два полносвязных слоя и активацию Sigmoid. Завершающим этапом является свёртка 1x1 с нормализацией BatchNorm, возвращающая количество каналов к исходной размерности.

Как было уже сказано выше, в качестве функции активации была выбрана **SiLU** (Sigmoid Linear Unit), которая вычисляется как произведение входного значения и сигмоида от этого значения:

$$\text{SiLU}(x) = x \cdot \text{sigmoid}(x) = x \cdot \frac{1}{1 + e^{-x}}$$

Главное преимущество SiLU заключается в том, что она сочетает свойства ReLU и сигмоида, обеспечивая нелинейность, необходимую для обучения сложных моделей, при этом избегая резких обнулений градиентов, характерных для ReLU. SiLU помогает модели лучше обучаться, так как она сохраняет больше информации даже для отрицательных значений входов, что снижает вероятность возникновения проблемы "мертвых"нейронов, характерной для ReLU.

4.3 Обучение нейронной сети

Для обучения нейронной сети была использована функция потерь CrossEntropyLoss, которая хорошо подходит для задач многоклассовой классификации:

$$H(y, \hat{y}) = -\sum_{i=1} y_i \log(\hat{y}_i),$$

где:

- H: функция потерь (кросс-энтропия), измеряющая расхождение между истинным распределением меток и прогнозируемыми вероятностями;
- y_i**:** истинное значение для класса i, которое принимает значение 1, если данный класс является правильным, и 0 в противном случае (в случае one-hot encoding меток);
- $\hat{y}_i$**:** прогнозируемая вероятность принадлежности к классу i (выход модели после применения softmax).

В качестве оптимизатора применялся SGD (Stochastic Gradient Descent), обеспечивающий стабильное обучение за счёт использования метода стохастического градиентного спуска. Для повышения эффективности оптимизации использовался моментум со значением 0.9, а скорость обучения была установлена на уровне 0.001.

Выбор данных параметров позволил достичь сбалансированной производительности модели и обеспечить её стабильную сходимость в процессе обучения. На рисунке 4.1 представлен график изменения точности нейронной сети на обучающей и валидационной выборках, иллюстрирующий динамику улучшения качества модели в ходе её тренировки.

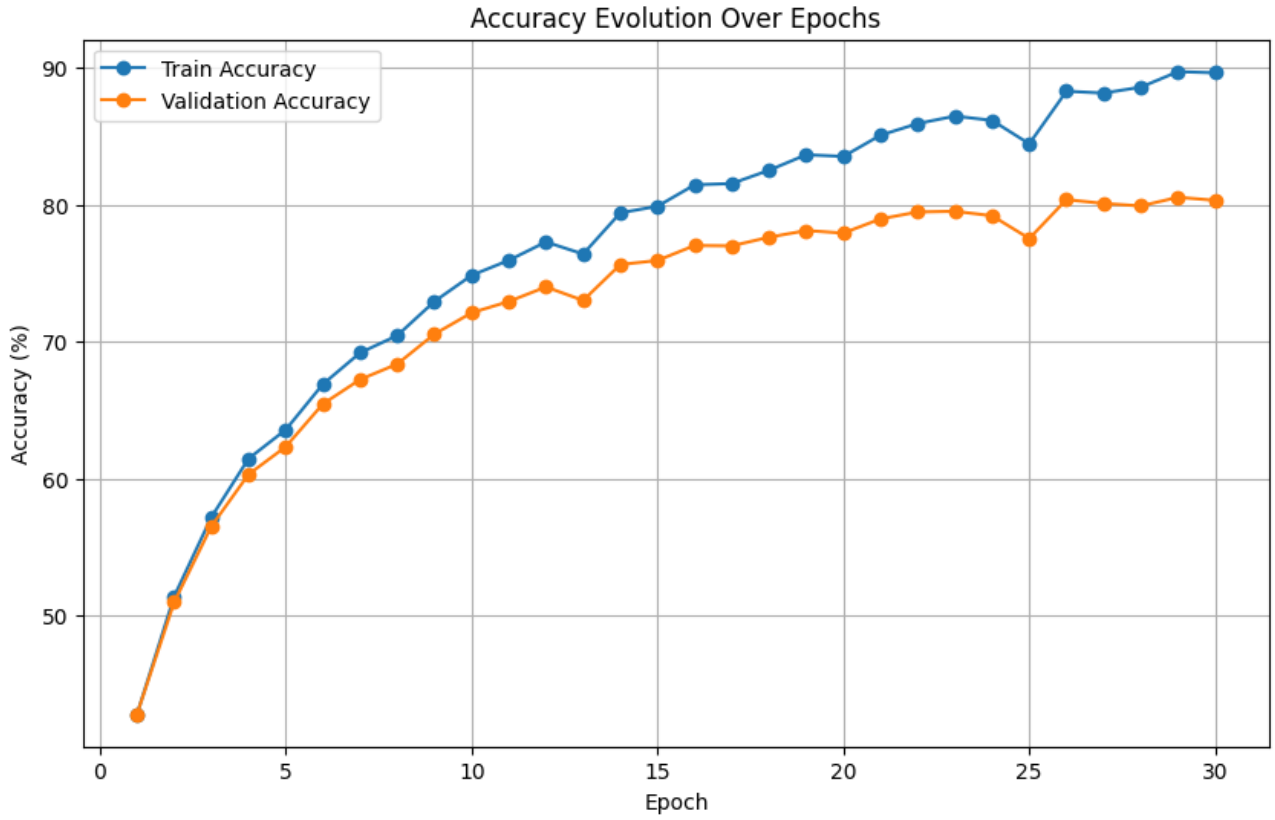

Рисунок 4.1 – График ошибки при обучении

5 ТЕСТИРОВАНИЕ НЕЙРОННОЙ СЕТИ

В данном разделе описаны этапы тестирования обученной нейронной сети для классификации текстов программного кода. Основной целью тестирования является проверка корректности работы модели, её точности и эффективности. Тестирование включало следующие этапы:

- **функциональное тестирование:** проверка работы модели на тестовых данных для анализа её способности различать положительный и отрицательный классы;
- **оценка качества модели:** после обучения модель тестируется на тестовой выборке. Оценка качества производится с использованием следующих метрик, представленных в таблице 5.1 и построение матрицы ошибок (нормализованной в целях упрощения визуального анализа и повышения интерпретируемости), представленной на рисунке 5.1:
  - **Accuracy (точность):** процент правильных предсказаний среди всех;
  - **F-мера:** среднее гармоническое между precision и recall;
  - **Recall (полнота):** доля правильных положительных предсказаний;
  - **Precision (точность):** доля правильных положительных предсказаний среди всех предсказанных положительных.
- **тестирование производительности:** проверка времени выполнения модели на тестовом наборе данных, а также анализ её эффективности при обработке больших объёмов текстовой информации.

Метрика	**Значение**
Precision	0.8071
Recall	0.8054
F1 Score	0.8052
Accuracy	0.8054

Таблица 5.1 – Точность нейронной сети на тестовом наборе

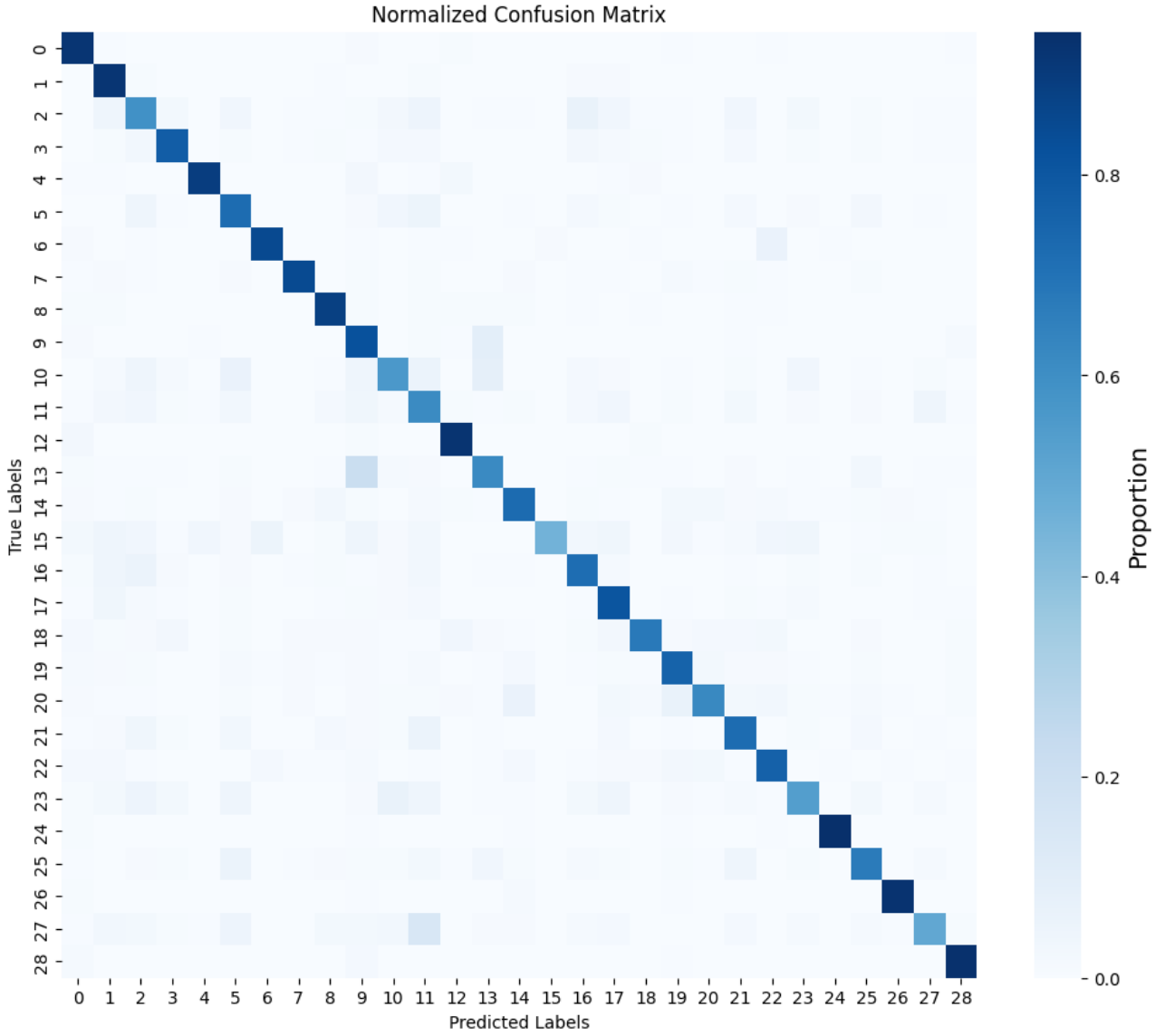

Рисунок 5.1 – Матрица ошибок

В результате тестирования модель показала хорошую общую точность, особенно при классификации популярных языков программирования. Основные ошибки были связаны с трудно различимыми языками и текстами, не относящимися к программированию. Для повышения точности в таких случаях планируется дополнительно расширить и улучшить данные.

6 ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была достигнута основная цель — разработана система классификации языков программирования с использованием сверточных нейронных сетей. Основное внимание было уделено подготовке специализированного датасета, обучению и тестированию модели на базе архитектуры EfficientNetV2-S.

В процессе работы была реализована модель, которая эффективно классифицирует фрагменты программного кода по языкам программирования. Обучение проводилось на подготовленных данных с использованием метрик точности, F1-score, precision и recall, что позволило всесторонне оценить её эффективность. Проведённая тонкая настройка гиперпараметров и архитектуры модели позволила достичь высокой точности классификации - порядка 80% на тестовой выборке.

Использованные методы и технологии продемонстрировали потенциал сверточных нейронных сетей для решения задач классификации текстовых данных. Разработанная система может быть применена для автоматизации анализа программного кода, повышения безопасности информационных систем и оптимизации процессов обработки данных.

Результаты работы показывают перспективность применения машинного обучения для классификации языков программирования. Данный проект открывает новые возможности для развития технологий анализа кода, защиты от вредоносных скриптов и улучшения взаимодействия разработчиков в информационной среде.

7 Список использованной литературы

1. Tan M., Le Q.V. EfficientNetV2: Smaller Models and Faster Training Электронныйресурс [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/2104.00298 . – Дата доступа: 19.11.2024.
2. Sandler M., et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1801.04381. – Дата доступа: 19.11.2024.
3. Большое сравнение 400 нейронных сетей для задачи классификации [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/666314/. – Дата доступа: 19.11.2024.
4. PyTorch Documentation [Электронный ресурс]. – Режим доступа: https://pytorch.org/docs/stable/index.html. – Дата доступа: 17.11.2024.
5. Kaggle Documentation [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/docs. – Дата доступа: 19.11.2024.