УДК 004.832.22
DOI 10.26102/2310-6018/2025.50.3.030

Создание модуля для генерации набора данных для обучения задачи генерации исходного кода на основе изображения

И.В. Никитин✉

Российский экономический университет имени Г.В. Плеханова, Москва, Российская Федерация

Резюме. В рамках данного исследования предлагается новый механизм создания данных для обучения нейронной сети для задачи генерации кода на основе изображения. Для того, чтобы система могла выполнять поставленную перед ней задачу, ее необходимо обучить. Изначальный набор данных, который предоставляется с системой pix2code, позволяет обучить систему, однако он опирается на те данные, которые представлены в словаре предметно-ориентированного языка. Расширение или изменение слов в словаре никак не влияет на набор данных, что ограничивает гибкость в применении системы, не позволяя учесть правила, которые могут применяться на предприятии. В части исследований есть утверждения о том, что они создали свой набор данных, однако его отсутствие в открытом доступе не позволяет оценить сложность изображений, содержащихся в нем. Для решения этой проблемы, в рамках данного исследования разработан подмодуль, который позволяет на основе измененного словаря предметно-ориентированного языка создать свой набор данных для обучения, состоящий из пары изображение–исходный код, соответствующий этому изображению. Для проверки работоспособности созданного набора данных, доработанная система pix2code выполнила обучение, а после смогла предсказать код на тестовых примерах.

Ключевые слова: кодогенерация, изображение, машинное обучение, набор данных, исходный код.

Для цитирования: Никитин И.В. Создание модуля для генерации набора данных для обучения задачи генерации исходного кода на основе изображения. *Моделирование, оптимизация и информационные технологии*. 2025;13(3). URL: https://moitvivt.ru/ru/journal/pdf?id=1976 DOI: 10.26102/2310-6018/2025.50.3.030

Building a module to generate a dataset for training the image-based source code generation task

I.V. Nikitin✉

Plekhanov Russian University of Economics, Moscow, the Russian Federation

Abstract. In this study, a new mechanism for generating training data for a neural network for the task of image-based code generation is proposed. In order for a system to be able to perform the task assigned to it, it must be trained. The initial dataset that is provided with the pix2code system allows the system to be trained, but it relies on the data that is provided in the domain-specific dictionary. Expanding or changing words in the dictionary does not affect the data set in any way, which limits the flexibility of the system's application by not allowing for the rules that may apply to the enterprise to be taken into account. Some studies claim to have created their own dataset, but its lack of public access makes it difficult to assess the complexity of the images it contains. To solve this problem, within the framework of this study, a submodule was developed that allows, based on a modified dictionary of a domain-specific language, to create a custom training dataset consisting of an image-source code pair corresponding to this image. To test the functionality of the created dataset, the modified pix2code system performed training and was then able to predict the code on test examples.

Keywords: code generation, image, machine learning, dataset, source code.

For citation: Nikitin I.V. Building a module to generate a dataset for training the image-based source code generation task. *Modeling, Optimization and Information Technology*. 2025;13(3). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=1976 DOI: 10.26102/2310-6018/2025.50.3.030

Введение

Развитие систем машинного обучения может изменить, а в некоторых местах уже меняет, подходы к тому, как устроена наша повседневная жизнь, а также работа различных предприятий. Использование различных чат-ботов или систем, которые в своей основе содержат системы машинного обучения позволяют забрать или упростить часть рутинных задач за счет того, что их будет выполнять сама система. Так, например, чат-бот ChatGPT может быть использован как поисковая система, которая позволяет сформулировать свой запрос более точно и получить более точный ответ.

С другой стороны, бизнес и предприятия также заинтересованы в том, чтобы упростить или изменить свои процессы за счет внедрения машинного обучения. Один из способов это сделать – научиться автоматизировать генерацию исходного кода там, где это возможно. Например, с помощью изображения макета некой рекламной страницы, можно научиться создать исходный код верстки, который в среде исполнения браузера будет выглядеть также, как и на самом макете.

Первая успешная попытка создания такой системы была представлена в 2018 году в рамках работы [1] и называется pix2code. Эта система позволяет на основе входного изображения предсказать исходный код, который будет соответствовать этой странице. Система оказалась удачной и породила ряд исследований, как например, [2] или [3], которые пытались улучшить показатели качества системы. Также был проведен ряд доработок и экспериментов, позволивших как повысить качество полученных результатов в работе [4], так и предложить новый вариант для оценки качества результата в рамках работы [5]. Однако, все исследования, в том числе ранее упомянутые [2] или [3], обладают особенностью, связанной с набором данных, на которых обучаются описанные в них модели: набор везде одинаковый и берется из оригинальной работы pix2code. Этот набор данных, с одной стороны, показал свою эффективность, так как используется в научных исследованиях, связанных с системой pix2code. С другой стороны, данный набор не обладает гибкостью из-за того, что он фиксированный на уровне изображение-код, какие-либо изменения в языке DSL, на основе которого построены упомянутые системы генерации кода, не отражаются на процессе обучения. Из-за чего изменения набора данных на предприятии для его адаптации под задачи усложняется. Актуальность данной работы связана с повышением гибкости внедрения решения, предложенного в статье [4], с целью снижения издержек во время внедрения, связанных с адаптацией решения под конкретные правила, принятые на предприятии.

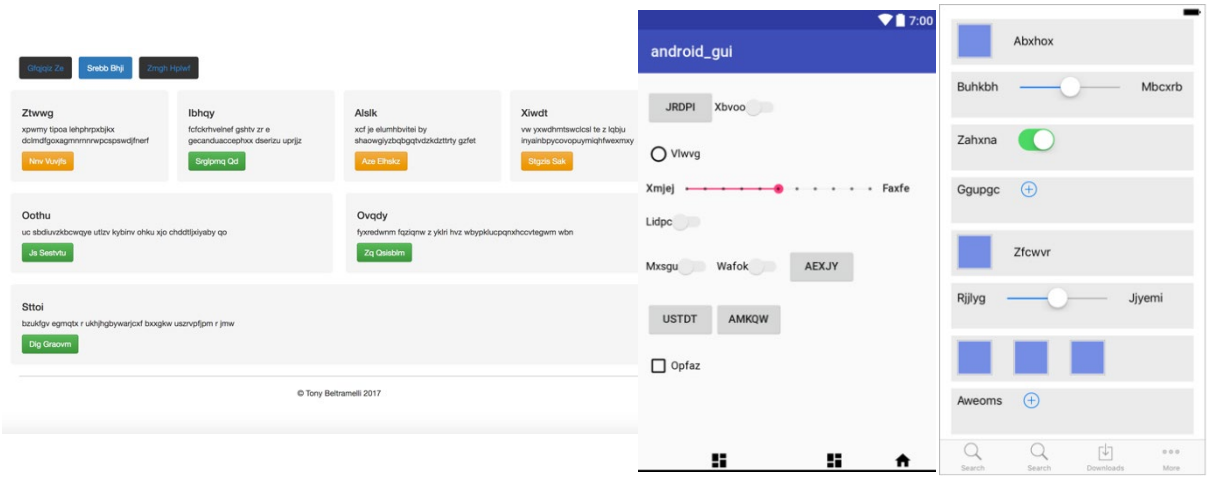
В рамках данного исследования реализован отдельный подмодуль, способный на основе описанных правил предоставить новый набор данных, состоящий из пар изображение–исходный код. Для проверки работоспособности данного модуля, проведен эксперимент, в ходе которого система из исследования [4] выполнила обучение на данном модуле, а после предсказала исходный код на основе изображения из исходного набора данных.

Материалы и методы

Как было отмечено во введении, большинство моделей из исследований, связанных с системой pix2code, проходили обучение на основе набора данных, который представлен в работе [1]. Однако, некоторые исследования все же отмечают минусы, которыми обладает оригинальный набор, поэтому пробовали также создать свой набор

данных. С целью выявления особенностей того, какими же именно свойствами должен обладать набор данных, чтобы помочь успешно выполнять задачу по генерации кода, более пристально были рассмотрены известные и упоминаемые наборы данных.

Первым набором данных, на который было обращено внимание, стал набор от Тони Белтрамелли в работе pix2cod, он называется PixCo. Как отмечается в статье [1], это первый набор данных, который содержит как набор изображений, так и кода, ассоциированного с этим изображением. Кроме того, важно отметить, что изображения генерируются для трех платформ: мобильных приложений на базе iOS и Android на основе Storyboard и XML соответственно, а также для Web на основе HTML/CSS. В открытом исходном коде проекта[1] был найден заархивированный набор этих данных. Примеры изображений для всех трех платформ, которые хранятся в архивах, можно увидеть на Рисунке 1. Как уже отмечалось ранее, важно, что этот набор используется как для обучения, так и для тестирования. Разбиение происходит на две группы с помощью отдельно запускаемой программы.



а) *б*) *в*)

Рисунок 1 – Примеры изображений из набора PixCo для разных платформ:
а – Web; *б* – Android; *в* – iOS
Figure 1 – Examples of images from the PixCo for different platforms:
а – Web; *б* – Android; *в* – iOS

В работе [2] используется два набора данных: первый – все тот же набор из работы pix2code, а второй набор реализован авторами самостоятельно и называется PixCo-e. Основная идея создания нового набора заключалась в том, что PixCo имеет довольно простые изображения, поэтому было принято решение создания набора с более комплексными изображениями. В частности, в PixCo-e изображения содержат большее количество элементов, а также определенные «фильтры», как, например, затемнение или высветление некоторых областей. К сожалению, набора данных в отрытом доступе нет, поэтому ориентиром остается только пример изображений из статьи [2].

Следующей работой, которая рассматривалась, является работа [6]. Однако в данном случае она не представляет большого интереса, так как обучалась на наборе PixCo. При этом важно отметить, что сравнение результатов в данной работе идет между предыдущими работами. В том числе по этой причине использовался один и тот же

[1] tonybeltramelli. pix2code: Generating Code from a Graphical User Interface Screenshot. GitHub. URL: https://github.com/tonybeltramelli/pix2code (дата обращения: 28.05.2025).

набор данных, чтобы на нем показать, насколько новая разработанная система стала лучше работать.

В других исследования, таких как, например, [3] или [7] также используется набор данных PixCo. С одной стороны, использование одного и того же набора данных можно использовать для оценки между собой эффективности обучения различных систем с доработками. С другой стороны, использование новых наборов данных, содержащих более сложные изображения, может позволить обучать систему более эффективно, позволяя ей лучше выполнять поставленную задачу.

Другой подход к организации данных был сделан в работе [8]. В ней авторы также создают систему по генерации исходного кода по изображению, однако для создания набора данных для обучения они брали изображения непосредственно из реально существующих приложений, что является главным плюсом такого подхода – данные максимально приближены к реальности. Однако в действительности это очень трудоемкий процесс, а также он может не содержать большого количества данных. Например, авторы упоминают, что из 100 лучших приложений на платформе iOS было взято всего 58 изображений, что очень сильно ограничивает набор данных.

Если же взять задачу кодогенерации шире (а генерация статичного кода разметки – это один из подвидов задачи кодогенерации), то можно найти в исследовании [9] упоминание различных наборов для обучения систем кодогенерации, однако данные из них не подходят для текущей задачи, так как требуют наличия изображения, из которого будут выделяться необходимые признаки.

Другой пример – это работа [10]. Данное исследование посвящено способам генерации кода, отличным от подхода с DSL языком, и на эту же работу есть ссылка в статье [1]. Для обучения автор использовал набор данных, состоящий из карточек для Коллекционной Карточной Игры (ККИ). На основе текста и признаков самой карточки создает класс, описывающий необходимые свойства. Несмотря на интересный подход, для данной задачи такой набор данных также не подходит, так как позволяет построить связь между изображением и исходным кодом.

Таким образом, можно заметить, что либо в исследованиях используется исходный набор данных PixCo, либо создан какой-то свой набор, но он не находится в открытом доступе, что не позволяет ни оценить его, ни использовать, либо набор данных просто не подходит для задачи генерации кода на основе изображения. Таким образом, мной предложен механизм по созданию своего набора данных.

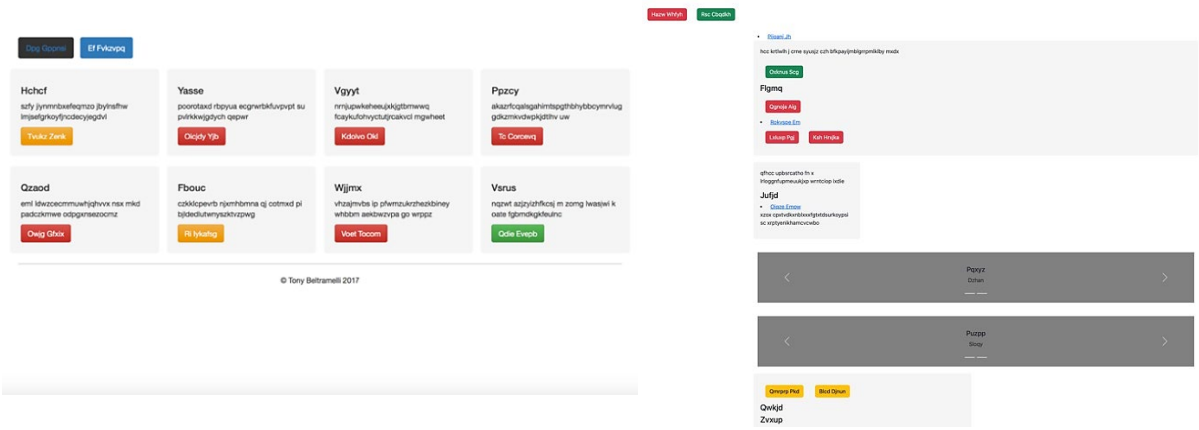
Результаты и обсуждение

Как можно отметить по итогам исследования на текущий момент нет подхода для создания своего набора данных для обучения. Большинство исследований используют набор из оригинальной pix2code (далее – оригинальный набор данных), а другие исследования создают свой, но эти наборы не находятся в открытом доступе. В результате, система теряет в своей гибкости на этапе внедрения на производство – из-за того, что набор данных очень общий, ситуаций, где система может быть использована, ограничен тем дизайном и набором компонентов, что присутствует в оригинальном наборе данных. Это также вызывает проблему в дообучении системы под свои требования и нужды: расширить список используемых компонентов можно, однако это никаким образом не скажется на наборе данных.

Многие компании используют свои собственные дизайн-системы, или определенные требования к дизайну, которых придерживаются в компании, и которые позволяют компании выделятся на фоне других узнаваемым стилем. Например,

компания Google использует дизайн систему Material UI[2], в которой описаны подходы, которыми компания придерживает по ходу создания своих интерфейсов. Или, например, компания Яндекс также имеет свою дизайн-систему Gravity UI[3] в открытом доступе.

Для того, чтобы решить эту проблему, в рамках данного исследования предложен модуль, который позволяет создавать свой набор данных на основе тех дизайн-систем, которые могут использоваться в компании. Этот модуль добавляет гибкости системе, благодаря чему ее внедрение упрощается, а сама система может обучаться на основе тех данных и примерах, которые считаются в компании эталонными. Кроме того, данный модуль позволяет создавать примеры более сложных сайтов, что позволяет обучать нейронную сеть на более сложных примерах. На Рисунке 2 представлено сравнение изображений из оригинального набора данных, а также изображения, полученного с помощью созданного модуля. Как можно заметить, новое изображение имеет больше компонентов, и выглядит несколько сложнее, чем изображение из оригинального набора. Исходный код данного модуля представлен в источнике[4].



а) *б*)

Рисунок 2 – Примеры изображений из наборов данных: *а* – изображений из оригинального набора данных; *б* – изображение, полученное с помощью созданного модуля
Figure 2 – Examples of images from datasets: *a* – images from the original dataset; *b* – image obtained using the created module

Работа данного модуля схожа с тем, как работают функциональные тесты в статье [5], однако имеет ряд отличий. Само по себе создание изображений происходит в цикле, позволяя таким образом создать достаточное количество изображений, необходимых для обучения. При этом создание происходит относительно быстро: на создание 4000 изображений потребовалось 70 минут. Для сравнения: обучение одной эпохе на 2349 изображениях занимает у системы из статьи [4] 6 часов. Одна итерация состоит из трех частей: создать gui-файл, затем преобразовать его в HTML-файл, а после создание снимка браузера, выполнившего HTML-файл. Преобразование gui в HTML происходит с помощью внутреннего скрипта. Так как для обучения важно, чтобы система смотрела на суть элемента, а не на его наполнение, текст в элементах должен быть случайно сгенерирован, в отличие от функциональных тестов, где случайно сгенерированный текст может оказать значительное влияние на результаты прохождения. Для получения

[2] Material UI. MUI. URL: https://mui.com/material-ui (дата обращения: 28.05.2025).
[3] Gravity UI. GitHub. URL: https://github.com/gravity-ui (дата обращения: 28.05.2025).
[4] IlyaNikk. html-code-generation: Using the AI model to generate HTML Code by giving a UI mockup image. GitHub. URL: https://github.com/IlyaNikk/html-code-generation (дата обращения: 28.05.2025).

снимка экрана браузера используется библиотека Playwright[5], которая выполняет переданную ей разметку. Небольшое отличие от функциональных тестов заключается в том, что изображение в данном случае размера 1280 на 2860 пикселей. Связано это с тем, что на изображениях с большим количеством элементов не всегда все помещается, поэтому итоговые изображения оказываются сильно вытянутыми по высоте.

Само же создание gui файла происходит по определенным правилам. Так как итоговая структура представляет собой дерево, генерация идет с верхнего элемента и вниз. Обычно HTML страницы имеют следующую структуру: заголовок, в котором присутствует текстовая информация, кнопки и ссылка; основной контент, который содержит больше всего информации; подвал, в котором находится служебная и юридическая информация. Как можно заметить, каждая из этих частей обладает своим набором элементов, которые могут присутствовать как во всех частях, так и в строго определенных. Для реализации этих правил был создан отдельный конфигурационный файл, содержащий следующую информацию: какие дочерние элементы могут быть у родителей; какая структура у документа в целом; какие есть ограничения у элемента (например, у элемента обязательно должны быть все его дети, или сам элемент может быть использован только один раз за все время); количество элементов на странице. Благодаря такой гибкой настройке, можно создать более специфичные под заданные данные изображения. Так, например, удалось добавить компонент Карусель, который можно увидеть на Рисунке 2*б* – серый блок с кнопками влево и вправо. Таким образом, можно расширить исходный набор данных новыми компонентами.

Для проверки «правильности» созданного набора данных и подтверждения того, что этот набор можно использовать в обучении, проведен эксперимент, в ходе которого система, представленная в статье [4], обучалась на созданном с помощью описанной подсистемы наборе данных, а затем в качестве входных данных передавались изображения из оригинального набора данных. Цель эксперимента – по результатам предсказания на основе старых данных убедиться, что обучение на новых данных работает. Результат в виде сравнения с изображением из оригинального набора данных связан с тем, что ранее по ходу работы над исследованиями [4] и [5] этот набор использовался как базовый, поэтому эффективность его работы уже подтверждена. Кроме того, это набор данных, который обладает тем же дизайном, поэтому результаты предсказания могут быть точными.

Однако в данном случае в процесс обучение было внесено небольшое изменение. Так как в оригинальном наборе было 2349 пар изображений-кодов, все эти пары использовались в процессе обучения на каждой эпохе. Так как количество новых пар составляет 4000 изображений, они были разбиты на два набора: 3000 – для обучения и 1000 – для валидации, а затем для обучения на каждой эпохе берется 2457 случайно выбранных изображений, а для валидации – 100 случайно выбранных изображений. Обучение проводилось на протяжении 50 эпох на процессоре M1.

На Рисунках 3 и 4 представлены результаты обучения на основе созданного набора данных со случайным выбором пар изображение-код. Самым главным отличием по сравнению с результатами, полученными в работе [4], является момент получения первых результатов: если раньше для этого требовалось 10 эпох, то теперь первые результаты были получены к 20 эпохе. Сами значения метрик также отличаются от того, что было получено ранее: они несколько ухудшились. Связано это может быть с ранее выдвинутыми утверждениями: во-первых, случайный выбор изображения сильно сказался на результатах, а, во-вторых, сложность самих изображений.

[5] Playwright for Python. URL: https://playwright.dev/python (дата обращения: 20.05.2025).

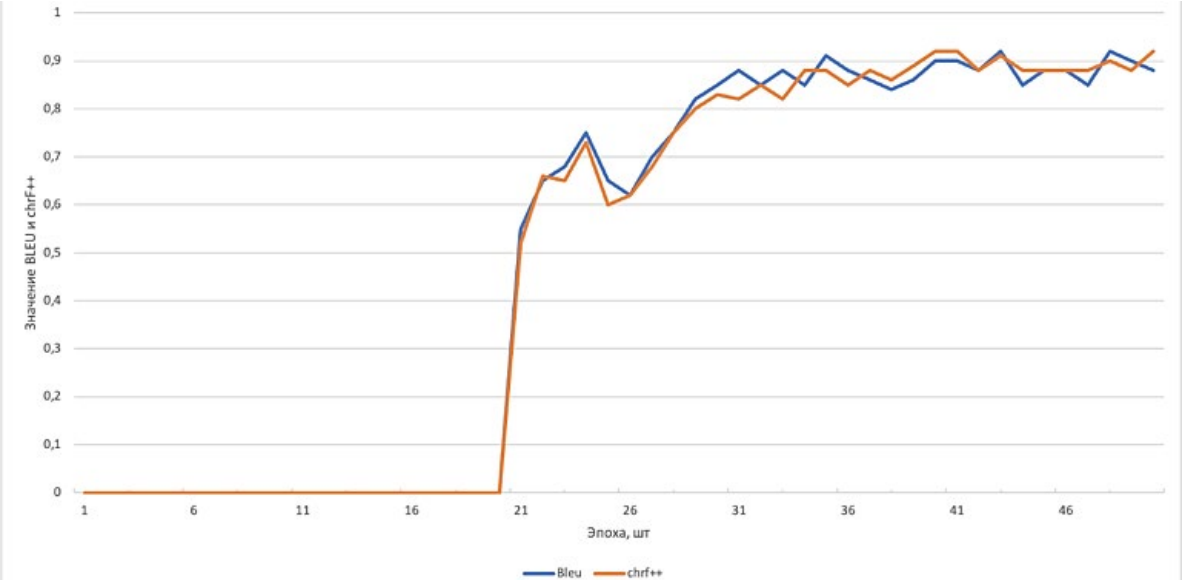


Рисунок 3 – Изменения значения метрики BLEU и chrF++ в процессе обучения с использованием архитектур ResNet и Трансформеров и на созданном наборе данных
Figure 3 – Changes in BLEU and chrF++ metric values during training using ResNet and Transformers architectures and on the generated dataset

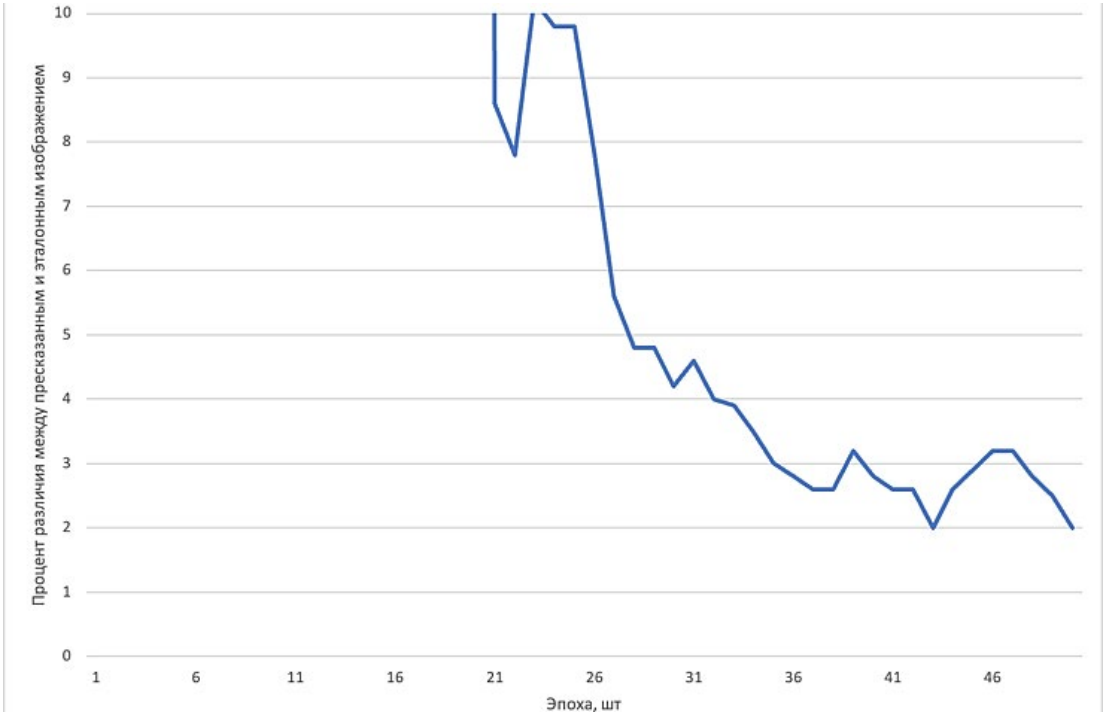
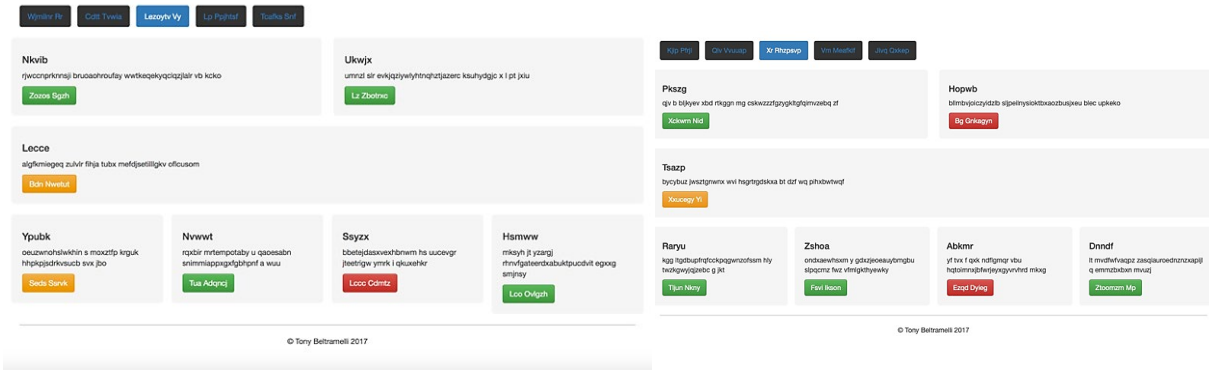


Рисунок 4 – Изменения результата выполнения функциональных в процессе обучения с использованием архитектуры ResNet и Трансформеров и на созданном наборе данных
Figure 4 – Changes in the result of the functional execution during the training process using the ResNet and Transformers architecture and on the created dataset

Кроме того, на Рисунке 5 представлен результат предсказания обученной системы на основе изображения из оригинального набора данных.



а) *б*)

Рисунок 5 – Результат предсказания системы, обученной на основе созданного набора данных: *а* – изображений из оригинального набора данных; *б* – изображение, предсказанное после обучения

Figure 5 – The prediction result of the system trained on the basis of the created dataset: *a* – images from the original dataset; *b* – image predicted after training

Как можно увидеть, результаты предсказания различаются между собой на 1 % (различие в цвете двух кнопок), что говорит о том, что система способна обучаться на данных, созданных с помощью подмодуля для генерации набора данных.

Заключение

В ходе исследования было предложено создать новый модуль, который позволяет на основе словаря предметно-ориентированного языка и правил, описывающих взаимосвязи элементов между этими элементами, создавать новый набор данных. Анализ научных публикаций, связанных с системой pix2code показал, что проблема создания набора тестовых данных актуальна, однако в открытом доступе этих решений нет. Именно последний факт побудил к созданию подмодуля, который способен создавать свой набор тестовых данных.

В рамках экспериментов, во-первых, был создан новый набор данных из 4000 пар изображений исходного кода, который содержит новые элементы, учитывая расширение словаря. Во-вторых, полученный набор данных использовался в обучении системы pix2code из статьи [4]. После обучения системе было предложено несколько тестовых изображений из изначального набора pix2code. Обученная система успешно справилась с предсказанием исходного кода, что говорит о том, что созданный набор данных позволяет обучить систему для выполнения поставленной задачи.

В будущих исследованиях планируется расширить количество операционных систем, поддерживаемых системой pix2code. Оригинальная система содержит изображения в том числе для платформ Android и iOS. Возможность по созданию набора данных для этих платформ позволит расширить область применения системы, предложенной в рамках работы [4].

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Beltramelli T. pix2code: Generating Code from a Graphical User Interface Screenshot. In: *EICS '18: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 19–22 June 2018, Paris, France.* New York: Association for Computing Machinery; 2018. https://doi.org/10.1145/3220134.3220135
2. Zhu Zh., Xue Zh., Yuan Z. Automatic Graphics Program Generation Using Attention-Based Hierarchical Decoder. In: *Computer Vision – ACCV 2018: 14th Asian Conference on Computer Vision: Revised Selected Papers: Part VI, 02–06 December 2018, Perth,*

Australia. Cham: Springer; 2019. P. 181–196. https://doi.org/10.1007/978-3-030-20876-9_12

3. Liu Ya., Hu Q., Shu K. Improving pix2code Based BI-directional LSTM. In: *2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), 16–18 November 2018, Shenyang, China.* IEEE; 2019. P. 220–223. https://doi.org/10.1109/AUTEEE.2018.8720784
4. Никитин И.В. Использование архитектур ResNet и Трансформеров в задаче генерации исходного кода на основе изображения. *Моделирование, оптимизация и информационные технологии.* 2025;13(2). https://doi.org/10.26102/2310-6018/2025.49.2.002
   Nikitin I.V. Using ResNet and Transformers Architectures in the Problem of Source Code Generation from an Image. *Modeling, Optimization and Information Technology.* 2025;13(2). (In Russ.). https://doi.org/10.26102/2310-6018/2025.49.2.002
5. Никитин И.В. Оценка качества полученного результата в задаче генерации исходного кода по изображению. *Моделирование, оптимизация и информационные технологии*. 2025;13(1). https://doi.org/10.26102/2310-6018/2025.48.1.030
   Nikitin I.V. Assessing the Quality of the Result in the Problem of Source Code Generation from an Image. *Modeling, Optimization and Information Technology*. 2025;13(1). (In Russ.). https://doi.org/10.26102/2310-6018/2025.48.1.030
6. Chen W.-Yi., Podstreleny P., Cheng W.-H., Chen Yu.-Ya., Hua K.-L. Code Generation from a Graphical User Interface Via Attention-Based Encoder-Decoder Model. *Multimedia Systems.* 2022;28(1):121–130. https://doi.org/10.1007/s00530-021-00804-7
7. Zou D., Wu G. Automatic Code Generation for Android Applications Based on Improved Pix2code. *Journal of Artificial Intelligence and Technology.* 2024;4(4):325–331. https://doi.org/10.37965/jait.2024.0515
8. Nguyen T.A., Csallner Ch. Reverse Engineering Mobile Application User Interfaces with REMAUI (T). In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 09–13 November 2015, Lincoln, NE, USA*. IEEE; 2016. P. 248–259. URL: https://doi.org/10.1109/ASE.2015.32
9. Paul D.Gh., Zhu H., Bayley I. Benchmarks and Metrics for Evaluations of Code Generation: A Critical Review. In: *2024 IEEE International Conference on Artificial Intelligence Testing (AITest), 15–18 July 2024, Shanghai, China.* IEEE; 2024. P. 87–94. https://doi.org/10.1109/AITest62860.2024.00019
10. Ling W., Blunsom Ph., Grefenstette E., et al. Latent Predictor Networks for Code Generation. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016: Volume 1: Long Papers, 07–12 August 2016, Berlin, Germany.* The Association for Computer Linguistics; 2016. P. 599–609. https://doi.org/10.18653/v1/P16-1057

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Никитин Илья Владимирович, аспирант Российский экономический университет имени Г.В. Плеханова, Москва, Российская Федерация
e-mail: vic096@yandex.ru

Ilya V. Nikitin, Postgraduate, Plekhanov Russian University of Economics, Moscow, the Russian Federation.

Статья поступила в редакцию 28.05.2025; одобрена после рецензирования 07.07.2025; принята к публикации 31.07.2025.

The article was submitted 28.05.2025; approved after reviewing 07.07.2025; accepted for publication 31.07.2025.