УДК 004.89

ПОСТРОЕНИЕ МОДЕЛИ ОБУЧЕНИЯ ГЕНЕРАЦИИ ТЕСТОВЫХ ДАННЫХ ДЛЯ ТЕСТИРОВАНИЯ GUI

Сарычева Ю.Ю., Белов Ю.С.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», филиал, Калуга, e-mail: yulia.sarycheva99@mail.ru

Графический интерфейс является неотъемлемой частью программного обеспечения на большинстве основных платформ, включая Android, и представляет широкий интерес для многих исследователей. Конечные пользователи взаимодействуют с приложением в зависимости от того, что они хотят делать с приложением и что они видят в его графическом интерфейсе. Поскольку разные приложения часто используют общие шаблоны проектирования пользовательского интерфейса, интуитивно понятно, что способ взаимодействия людей с графическим интерфейсом можно обобщить для разных приложений. Цель модели взаимодействия состоит в том, чтобы зафиксировать такие обобщенные модели взаимодействия. В данном исследовании будет рассмотрен автоматизированный генератор входных данных с графическим интерфейсом, который может изучать, как люди взаимодействуют с мобильными приложениями, а затем использовать ее для управления генерацией входных данных, имитируя поведение человека. Причина, по которой эта стратегия может улучшить тестовое покрытие, заключается в том, что важные состояния, которые пользователи предпочитают посещать и с большим количеством возможных входных данных, изучаются чаще. При ограниченном времени и бюджете трата времени на эти важные состояния может эффективнее улучшить тестовое покрытие.

Ключевые слова: графический интерфейс пользователя, тестирование, модель генерации, автоматизированное тестирование

BUILDING A TRAINING MODEL TO GENERATE TEST DATA FOR GUI TESTING

Sarycheva Yu.Yu., Belov Yu.S.

Bauman Moscow State Technical University, Kaluga branch, Kaluga, e-mail: yulia.sarycheva99@mail.ru

The GUI is an integral part of software on most major platforms, including Android, and is of great interest to many researchers. End users interact with the application depending on what they want to do with the application and what they see in its GUI. Because different applications often share common user interface design patterns, it is intuitive that the way people interact with a GUI can be generalized across applications. The purpose of an interaction model is to capture such generalized interaction models. This study will explore an automated GUI input generator that can learn how people interact with mobile applications and then use it to control input generation by simulating human behavior. The reason this strategy can improve test coverage is because important states that users prefer to visit and with more possible inputs are explored more often. With limited time and budget, spending time on these important states can improve test coverage more effectively.

Keywords: graphical user interface, testing, generation model, automated testing

Анализ графического интерфейса приложения представляет большой интерес для многих исследователей и практиков, на данный момент существуют два направления исследований в этой области. Один из них – понять поведение приложений с точки зрения разработки программного обеспечения. Другой – с точки зрения взаимодействия человека с компьютером для анализа дизайна пользовательского интерфейса [1].

Цель исследования – построить модель обучения генерации тестовых данных для тестирования GUI.

Автоматизированный генератор. Конечные пользователи взаимодействуют с приложением в зависимости от того, что они хотят делать с приложением и что они видят в его графическом интерфейсе. Поскольку разные приложения часто используют общие шаблоны проектирования пользовательского интерфейса, интуитивно понятно, что способ взаимодействия людей с графическим интерфейсом можно обобщить для разных приложений. Цель модели взаимодействия состоит в том, чтобы зафиксировать такие обобщенные модели взаимодействия.

Представим модель, автоматизированный генератор входных данных с графическим интерфейсом, которая может изучать, как люди взаимодействуют с мобильными приложениями, а затем использовать ее для управления генерацией входных данных, имитируя поведение человека. Обладая знаниями и моделью, извлеченными из истории человеческого взаимодействия, модель может расставить приоритеты возможных взаимодействий с графическим интерфейсом в соответствии с их важностью с точки зрения пользователя, тем самым генерируя входные данные, которые могут достичь большего охвата.

Контекст UI context$_i$ в модели состоит из текущего состояния UI s_i и трех последних переходов в UI (s_{i-1}, a_{i-1}), (s_{i-2}, a_{i-2}), (s_{i-3}, a_{i-3}). Текущее состояние пользовательского интерфейса представляет то, что видят пользователи, когда они выполняют действие, в то время как используются последние переходы пользовательского интерфейса для моделирования основного намерения пользователей во время текущего сеанса взаимодействия.

На рис. 1 показано, как в модели представляются состояния и действия пользовательского интерфейса. Каждое состояние пользовательского интерфейса представлено в виде двухканального скелетного изображения пользовательского интерфейса, в котором первый канал (красный канал) отображает области ограничивающей рамки текстовых элементов пользовательского интерфейса, а второй канал (зеленый канал) отображает области ограничивающей рамки нетекстового пользовательского интерфейса [2].

Каждое действие представлено своим типом действия и координатами целевого местоположения. Тип действия кодируется как семимерный вектор, в котором каждое измерение соответствует одному из семи типов действия. Местоположение цели действия кодируется в виде тепловой карты. Каждый пиксель на тепловой карте – это вероятность того, что пиксель является целевым местоположением действия [3].

В итоге контекст пользовательского интерфейса представляется в виде вектора 4x180x320x3.

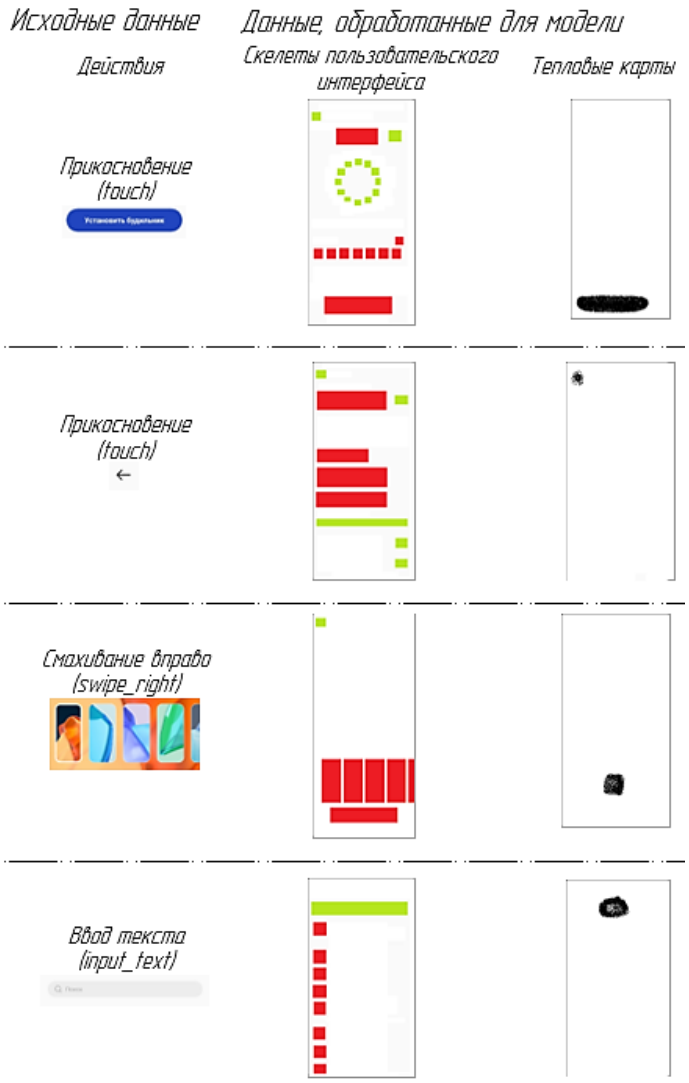

Рис. 1. Представление состояний и действий пользовательского интерфейса в модели взаимодействия

Распределение вероятностей. Учитывая вектор контекста пользовательского интерфейса, вывод модели взаимодействия – это «действие», которое будет выполняться людьми в текущем состоянии. Прогнозируемое «действие» не является фактически допустимым действием в текущем состоянии пользовательского интерфейса. Вместо этого это распределение вероятностей типов и мест ожидаемых действий, подобных человеческим. В частности, цель модели состоит в том, чтобы изучить два условных распределения вероятностей:

$$1)\ p_{type}\ (t \mid context_i)$$

где t {касание, долгое касание, пролистывание вверх, ...}, что означает распределение вероятности t, типа следующего действия a_i, с учетом текущего контекста пользовательского интерфейса.

$$2)\ p_{loc}(x, y \mid context_i)$$

где $0 < x <$ ширина экрана и $0 < y <$ высота экрана, что означает распределение вероятности целевого местоположения x, y следующего действия a_i с учетом текущего контекста пользовательского интерфейса.

Структура модели глубокой нейронной сети. На рис. 2 показана модель глубокой нейронной сети, используемая для изучения двух определенных выше распределений условной вероятности. Она принимает представление текущего контекста пользовательского интерфейса в качестве входных данных и выводит распределение местоположения и типа a_i. Модель состоит из пяти основных компонент: сверточные слои, остаточные модули LSTM, деконволюционные слои, полносвязный слой и функции потерь.

Сверточные слои. Структура сверточной сети стала популярным подходом для извлечения признаков изображения, поскольку она оказалась очень эффективной в задачах компьютерного зрения на больших наборах данных реального мира [3]. В модели используется пять сверточных слоев с активациями ReLU для извлечения функций из изображений скелета пользовательского интерфейса и тепловых карт действий. После каждого сверточного слоя следует слой max-pooling с шагом 2, который уменьшает ширину и высоту входных данных наполовину. Слои объединения также помогают модели идентифицировать элементы пользовательского интерфейса, имеющие одинаковую форму, но разное окружение.

Остаточные модули LSTM. Сети LSTM (Long-Short-Term Memory) широко используются в задачах моделирования последовательности, таких как машинный перевод, классификация видео [4] и т.д. В модели извлечение признаков из исторических переходов также является проблемой моделирования последовательности.

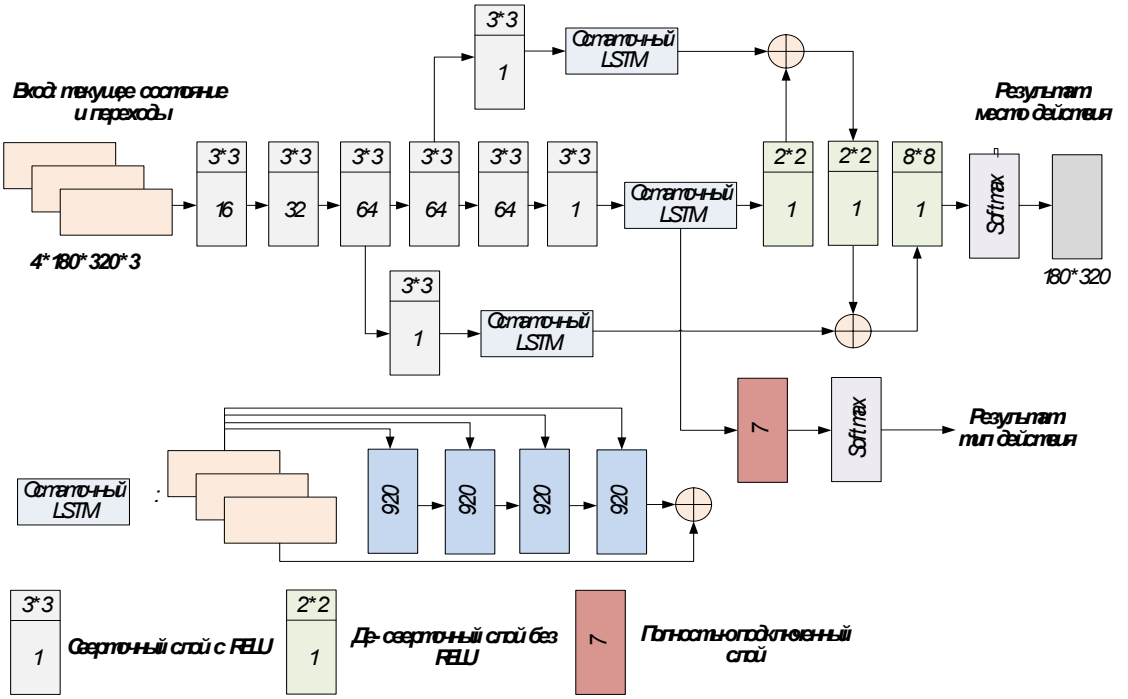

Рис. 2. Архитектура модели глубокого обучения

Остаточные LSTM-модули вставляются после каждого из последних трех инволюционных слоев, чтобы зафиксировать функции последовательности перехода пользовательского интерфейса на разных уровнях разрешения. В остаточном модуле LSTM последнее измерение ввода и вывода обычного LSTM напрямую добавляется через остаточный путь [4].

Такая остаточная структура облегчает оптимизацию нейронной сети [5] и дает намек на то, что расположение действия должно находиться внутри элемента пользовательского интерфейса. Чтобы уменьшить сложность модели, добавляется сверточный слой 1x1 перед каждым остаточным модулем LSTM, чтобы уменьшить размер объекта.

Деконволюционные слои. Этот компонент используется для создания распределений вероятностей с высоким разрешением из выходных данных с низким разрешением остаточных модулей LSTM [4]. Для этого существует несколько вариантов, таких как билинейная интерполяция, деконволюция и т.д. В модели используются слои деконволюции, поскольку их легче интегрировать с глубокими нейронными сетями, и они более общие, чем методы интерполяции [5]. Функции на разных уровнях разрешения объединяются для улучшения качества сгенерированной тепловой карты [6]. Затем следует слой softmax, чтобы нормализовать сгенерированную тепловую карту, чтобы сумма всех пикселей на тепловой карте равнялась 1, что является вероятностным распределением местоположений действий.

Полностью связанный слой. Один полносвязный слой с softmax используется для генерации вероятностного распределения типов действий.

Функции потерь. Модель предсказывает как место действия, так и тип действия как распределения вероятностей. Таким образом, их кросс-энтропийные потери по сравнению с основной правдой (действия, выполняемые людьми) подходят для оптимизации модели [7].

Процесс обучения модели. В рассматриваемой модели используется сумма этих двух потерь и регуляризатор веса слоя (чтобы избежать переобучения) в качестве окончательной функции потерь в процессе обучения.

Во время обучения каждое действие a_i в потоке взаимодействия s_1, s_2, s_3, …, s_n, a_1, a_2, a_3, ..., a_n преобразуется в следующие распределения вероятностей:

$$p_{type}(t) = \begin{cases} 1, & t = a_i \cdot type \\ 0, & otherwise \end{cases}$$

и

$$p_{loc}(x, y) = f(x - a_i \cdot x, y - a_i \cdot y),$$

где f – функция плотности распределения Гаусса с дисперсией = 20 для аппроксимации распределения вероятностей фактических координат экрана, распознаваемых устройством, когда один и тот же элемент пользовательского интерфейса взаимодействует со многими людьми много раз. Точно так же при применении модели передается представление текущего состояния пользовательского интерфейса для прогнозирования вероятностных распределений $p_{type}(t)$ и $p_{loc}(x, y)$ для следующего действия.

Поскольку предсказанные распределения нельзя напрямую использовать для управления генерацией тестов, необходимо дополнительно преобразовать их в вероятности действий, которые могут быть выполнены в текущем состоянии. Для этого сначала необходимо пройти по дереву пользовательского интерфейса [8], чтобы найти все возможные действия в текущем состоянии, причем каждое действие содержит тип действия (обозначается как action.type) и целевой элемент действия (обозначается как action.element).

Затем вычисляется вероятность каждого действия на основе распределения, предсказанного моделью

$$p(action) = p_{type}(action.type) * x$$

$$x, y \ в \ action.element$$

$$location(x, y)$$

Наконец, вероятности действий можно использовать для управления генерацией тестовых входных данных на следующем этапе.

Результаты исследования и их обсуждение

Представление контекста пользовательского интерфейса, то есть входной признак для модели взаимодействия представляет собой стек изображений, включая одно двухканальное изображение для текущего состояния пользовательского интерфейса и три трехканальных изображения для трех последних переходов пользовательского интерфейса (каждый переход включает одно двухканальное изображение для состояния пользовательского интерфейса и одно одноканальное изображение для действия). Все изображения масштабируются до размера 180x320 пикселей. Для простоты обучения также добавляется один канал заполнения нулями для текущего состояния пользовательского интерфейса.

Модель генерирует два типа тестовых входных данных, включая исследования и навигацию. Входные данные исследования используются для обнаружения невидимого поведения в приложении, а входные данные навигации переводят приложение в известные состояния, содержащие неисследованные действия. При выборе из входных данных исследования генератор тестов не знает о последствиях каждого входного теста, и решение принимается на основе руководства модели человеческого. При генерации навигационных входов генератор тестов знает целевые состояния входа, так как он сохранил память переходов.

Заключение

Графический интерфейс является неотъемлемой частью программного обеспечения на большинстве основных платформ, включая Android, и представляет широкий интерес для многих исследователей.

Была рассмотрена модель, которая представляет собой автоматизированный генератор входных данных с графическим интерфейсом.

По сравнению с существующими инструментами тестирования, главная особенность модели (и главное отличие между различными генераторами тестов на основе моделей) заключается в том, как выбираются входные данные исследования. Модель отдает приоритет более ценным действиям в исследовании на основе модели взаимодействия, которая была обучена на основе следов человеческого взаимодействия. Эта функция ускоряет обнаружение правильных входных последовательностей, что, в свою очередь, переводит приложение в важные состояния пользовательского интерфейса, что приводит к более широкому охвату тестами.

Список литературы

1. Винокуров А.В., Лавлинская О.Ю. Уровни организации автоматизированного тестирования мобильных приложений для операционной системы Android // Вестник Воронежского института высоких технологий. 2020. № 3 (34). С. 22–26.

2. Naja F., Mansur S., Wibawanto A. Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform. 20th International Conference on Advances in ICT for Emerging Regions. 2020. P. 4–6.

3. Михалевская К.А., Сергачева М.А. Сравнение инструментов для автоматизации тестирования мобильных приложений на ОС Android // Chronos: естественные и технические науки. 2020. № 2 (30). С. 45–49.

4. Воробьев Н.А., Бурмин Л.Н., Степанов Ю.А. Сравнительный анализ средств тестирования мобильных приложений // Евразийский союз ученых. 2020. № 6–1 (75). С. 36–38.

5. King T.M., Arbon J., Santiago D. AI for Testing Today and Tomorrow: Industry Perspectives. IEEE International Conference On Artificial Intelligence Testing (AITest). 2019. P. 81–88.

6. Сарычева Ю.Ю., Белов Ю.С. Применение искусственного интеллекта в автоматизированном тестировании GUI // Научные исследования в современном мире. Теория и практика: сборник избранных статей Всероссийской (национальной) научно-практической конференции. 2022. С. 55–56.

7. Pan M., Xu To., Pei Yu. GUI-Guided Test Script Repair for Mobile Apps. IEEE Transactions on Software Engineering. 2022. Vol. 48. No. 3. P. 3–5.

8. Плодухин Д.М. Реализация модели автоматизированного тестирования // Огарёв-Online. 2020. № 13 (150).