**УДК 004.032.26; 003.295**

# РАЗРАБОТКА НЕЙРОННОЙ СЕТИ НА БАЗЕ АЛГОРИТМА PIX2CODE

**Наталья Викторовна Картечина**

кандидат сельскохозяйственных наук, доцент

kartechnatali@mail.ru

**Анастасия Александровна Гущина**

студент

**Станислав Олегович Чиркин**

студент

stas.chirkin@bk.ru

**Алена Максимовна Дорохова**

студент

dorohovata@mail.ru

**Владислав Александрович Шацкий**

студент

shatskiy2000@list.ru

Мичуринский государственный аграрный университет

г. Мичуринск, Россия

**Аннотация**. В статье рассмотрена разработка нейронной сети на базе алгоритма pix2code.

**Ключевые слова:** алгоритм pix2code, HTML, CSS, нейронная сеть, CNN, LSTM, RNN.

Планируется разработка на базе алгоритма pix2code нейронной сети, которая способна создавать готовые сайты по картинке. Для этого требуется загружать скриншоты с конкретными HTML-тегами. Что бы получилась база с данными, которая позволит по текущим шаблонам создавать новые сайты.

Планируется что сеть будет постоянно обучаться и, соответственно, результат улучшаться.

Процесс создания сайта будет разбит на три этапа.

Вначале нейронная сеть будет просматривать загружаемые картинки-исходники в формате JPEG

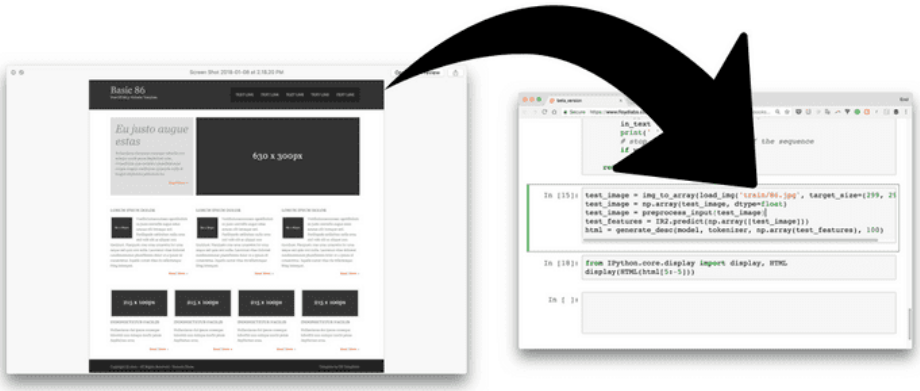


*Рисунок 1*– Процесс реализации проекта приложения

Далее сеть будет распознавать элементы на картинке и преобразовывать их в HTML и CSS [1].

После чего алгоритм будет выводить готовый сайт с написанным кодом.

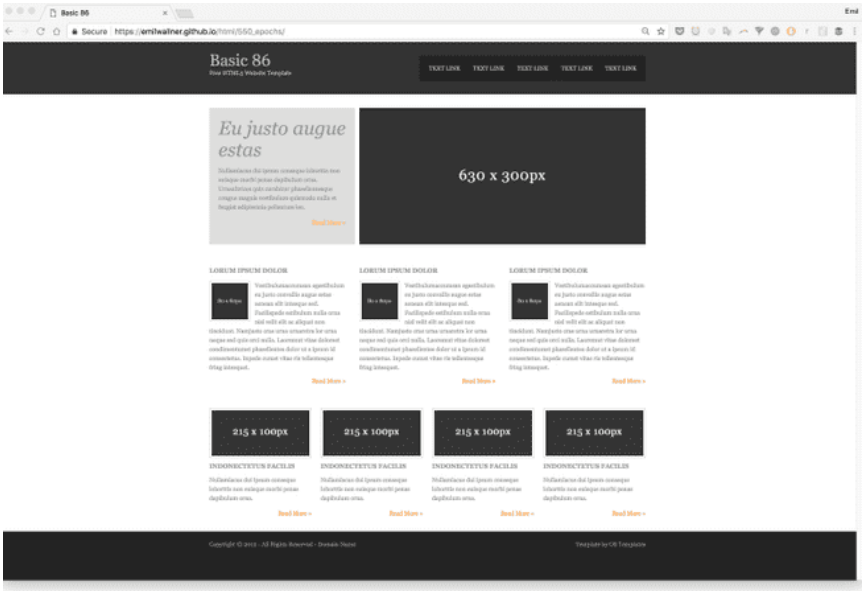


*Рисунок 2* – Предполагаемый итог реализации проекта

В настоящее время самым большим препятствием для автоматизации фронтенд-разработки являются вычислительные мощности. Тем не менее, мы можем использовать текущие алгоритмы глубокого обучения вместе с синтезированными обучающими данными, чтобы начать изучение искусственной клиентской автоматизации прямо сейчас [1, 2].

Мы разработали интерфейс клиентской части сайта и потратили на это несколько часов, однако используя систему генерации прототипов сайтов на основе нейронных сетей мы могли бы это сделать за считанные секунды.

Планируется научить нейронную сеть кодировать базовый веб-сайт HTML и CSS на основе изображения макета дизайна.

Планируется построить нейронную сеть за три итерации. В данной работе мы будем описывать данный проект.

В первой версии мы планируем сделать минимальную версию, чтобы получить представление о движущихся частях. Вторая версия, HTML, будет посвящена автоматизации всех шагов и объяснению слоев нейронной сети. В окончательной версии Bootstrap мы создадим модель, которая может обобщать и исследовать уровень LSTM.

Модели будут основаны на pix2code и руководствах по созданию подписей к изображениям Джейсона Браунли. Код будет написан на ython и Keras, фреймворке поверх TensorFlow.

Он учится, предсказывая все совпадающие теги разметки HTML один за другим. Когда он предсказывает следующий тег разметки, он получает снимок экрана, а также все правильные теги разметки до этого момента.

На сегодняшний день наиболее распространенным подходом является создание модели, которая предсказывает слово за словом. Есть и другие подходы [3].

Обратите внимание, что для каждого прогноза отображается один и тот же снимок экрана. Так что, если ему нужно предсказать 20 слов, он получит один и тот же макет дизайна двадцать раз.

Сфокусируемся на предыдущей разметке. Допустим, мы обучаем сеть предсказывать предложение «Я умею кодировать». Когда он получает «я», он предсказывает «могу». В следующий раз он получит «Я могу» и предскажет «код». Он получает все предыдущие слова и должен только предсказать следующее слово.

На основе данных нейронная сеть создает особенности. Нейронная сеть создает функции для связывания входных данных с выходными данными. Он должен создавать представления, чтобы понять, что находится на каждом снимке экрана, синтаксис HTML, который он предсказал [3, 4]. Это создает знания для предсказания следующего тега.

Когда вы хотите использовать обученную модель для реального использования, это похоже на то, как вы обучаете модель. Текст генерируется один за другим с одним и тем же снимком экрана каждый раз. Вместо того, чтобы кормить его правильными тегами HTML, он получает уже созданную разметку. Затем он предсказывает следующий тег разметки.

Прогнозирование начинается с «начального тега» и останавливается, когда предсказывается «конечный тег» или достигается максимальный предел.

Чтобы представить разметку так, как ее понимает нейронная сеть, можно использовать представление категориальных переменных в виде двоичных векторов.

Для этого сначала требуется, чтобы категориальные значения были сопоставлены с целочисленными значениями [1, 5, 6].

Затем каждое целочисленное значение представляется как двоичный вектор, который содержит все нулевые значения, за исключением индекса целого числа, который помечен 1.

Предположим, у нас есть последовательность меток со значениями «красный» и «зеленый».

Мы можем присвоить «красному» целочисленное значение 0, а «зеленому» - целочисленному значению 1. Пока мы всегда присваиваем эти числа этим меткам, это называется целочисленным кодированием.

Согласованность важна, чтобы мы могли позже инвертировать кодировку и получить метки обратно из целочисленных значений, например, в случае прогнозирования.

Затем мы можем создать двоичный вектор для представления каждого целочисленного значения. Вектор будет иметь длину 2 для 2 возможных целочисленных значений.

«Красная» метка, закодированная как 0, будет представлена двоичным вектором [1, 0], где нулевой индекс отмечен значением 1. В свою очередь, «зеленая» метка, закодированная как 1, будет представлена двоичный вектор [0, 1], где первый индекс отмечен значением 1.

Если бы у нас была последовательность: 'red', 'red', 'green'

Мы могли бы представить это в целочисленной кодировке: 0, 0, 1

И одна горячая кодировка:

[1, 0]

[1, 0]

[0, 1]

Одно горячее кодирование позволяет более выразительно представлять категориальные данные.

Многие алгоритмы машинного обучения не могут работать напрямую с категориальными данными. Категории необходимо преобразовать в числа. Это требуется как для входных, так и для выходных переменных, которые являются категориальными [3, 7].

Настраивая модель в документе pix2code, модель может предсказывать веб-компоненты с высокой точностью.

Извлечение функций из предварительно обученных моделей хорошо работает в моделях с субтитрами к изображениям. Но после нескольких экспериментов я понял, что сквозной подход pix2code лучше подходит для решения этой проблемы. Предварительно обученные модели не были обучены на веб-данных и настроены для классификации.

В этой модели мы заменяем предварительно обученные функции изображения легкой сверточной нейронной сетью. Вместо того, чтобы использовать max-pooling для увеличения плотности информации, мы увеличиваем шаги. При этом сохраняется положение и цвет элементов интерфейса.

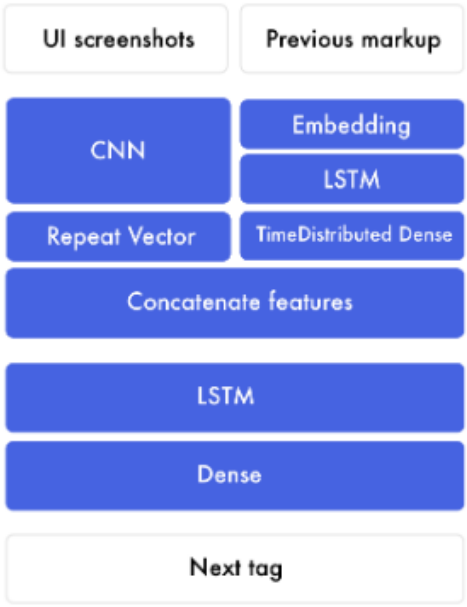


*Рисунок 3* – Комплексный подход

Есть две основные модели, которые позволяют это сделать: сверточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN). Наиболее распространенная рекуррентная нейронная сеть - это долговременная память (LSTM), поэтому мы будем обращаться именно к ней.

Одна из самых сложных вещей для понимания LSTM - это временные интервалы. Обычную нейронную сеть можно представить как два временных шага. Если вы ответите «Привет», он предсказывает «Мир». Но было бы сложно предсказать больше временных шагов. В приведенном ниже примере ввод имеет четыре временных шага, по одному для каждого слова.

LSTM предназначены для ввода с временными шагами. Это нейронная сеть, настроенная для получения информации по порядку. Если развернуть нашу модель, это выглядит так. Для каждого шага вниз вы сохраняете одинаковый вес. Вы применяете один набор весов к предыдущему выходу, а другой - к новому входу [8, 9].

Взвешенный ввод и вывод объединяются и складываются вместе с активацией. Это результат для этого временного шага. Поскольку мы повторно используем веса, они извлекают информацию из нескольких входных данных и формируют знания о последовательности.

Ниже представлена упрощенная версия процесса для каждого временного шага в LSTM.

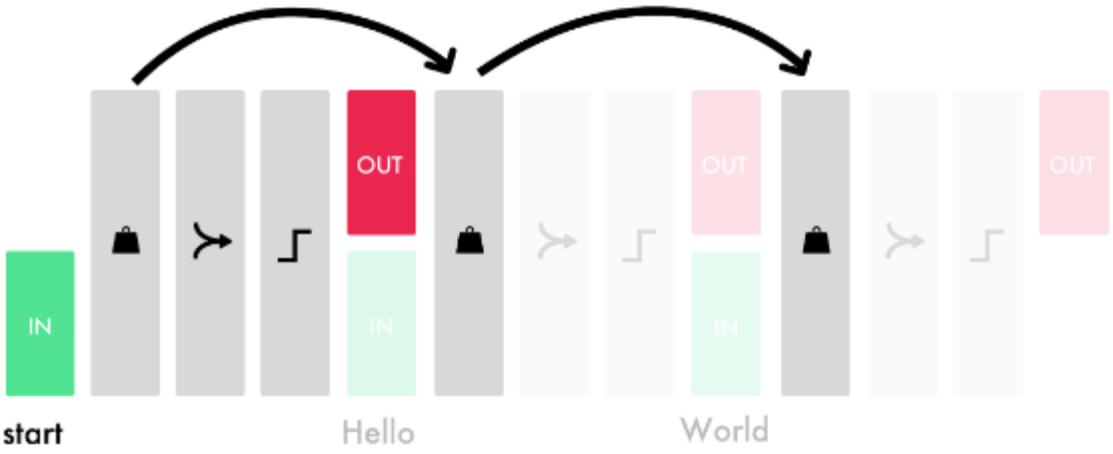


*Рисунок 4* - Упрощенная версия процесса для каждого временного шага в LSTM

Количество единиц в каждом слое LSTM определяет его способность запоминать. Это также соответствует размеру каждого выходного объекта. Опять же, функция - это длинный список номеров, используемых для передачи информации между слоями.

Каждый модуль на уровне LSTM учится отслеживать различные аспекты синтаксиса. Ниже представлена визуализация модуля, отслеживающего информацию в строке div. Это упрощенная разметка, которую мы планируем использовать для обучения модели начальной загрузки.

```
<START> header { btn-inactive , btn-active } row { single { small-title , text , btn-red } } row
{ double { small-title , text , btn-red } double { small-title , text , btn-red } } row { quadruple
{ small-title , text , btn-red } quadruple { small-title , text , btn-green } quadruple { small-
title , text , btn-green } quadruple { small-title , text , btn-orange } } <END>
```

*Рисунок 5* - Упрощенная разметка

Каждый модуль LSTM поддерживает состояние ячейки. Думайте о состоянии клетки как о памяти. Веса и активации используются для изменения

состояния по-разному. Это позволяет слоям LSTM точно настраивать, какую информацию сохранять и отбрасывать для каждого входа.

Помимо прохождения функции вывода для каждого входа, он также пересылает состояния ячеек, по одному значению для каждой единицы в LSTM.

Сложно найти способ измерить точность. Скажем, вы сравниваете слово в слово. Если ваш прогноз на одно слово не синхронизирован, у вас может быть 0% точности. Если вы удалите одно слово, которое синхронизирует предсказание, вы можете получить 99/100.

Лучше всего использовать оценку BLEU, лучшую практику в моделях машинного перевода и субтитров к изображениям. Он разбивает предложение на четыре n-граммы, состоящие из 1–4 последовательностей слов [10].

Например, в приведенном ниже предсказании «кот» должен быть «кодом».

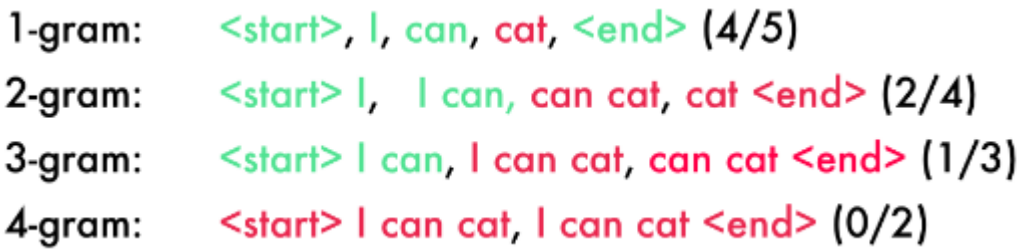


*Рисунок 6* – Пример тестирования

Чтобы получить окончательную оценку, вы умножаете каждую оценку на 25%, (4/5) * 0,25 + (2/4) * 0,25 + (1/3) * 0,25 + (0/2) * 0,25 = 0,2 + 0,125 + 0,083 + 0 = 0,408. Затем сумма умножается на штраф за длину предложения. Поскольку длина в нашем примере правильная, она становится нашим окончательным результатом.

Вы можете увеличить количество граммов, чтобы усложнить задачу. Модель с четырьмя н-граммами - это модель, которая лучше всего соответствует человеческим переводам.

Поймите слабые места моделей вместо тестирования случайных моделей. Посмотрите на тестовые данные и убедитесь что они могут предсказать цвет и положение с высокой точностью.

Используйте предварительно обученные модели, только если они актуальны. Предварительно обученная модель изображения улучшит производительность. Сквозная модель медленнее обучается и требует больше памяти, но является точнее.

Планируйте небольшое отклонение при запуске модели на удаленном сервере.Убедитесь, что вы понимаете функции библиотеки. Включите в свой словарь место для пустого токена.  Для экспериментов используйте более легкие модели.

**Список литературы:**


1. Использование возможностей языка R для реализации алгоритмов машинного обучения в среде MS SQL SERVER 2019 / А.А. Крумкаченко, Д.В. Косенков, В.В. Гавриков, Р.Н. Абалуев // Наука и Образование. 2020. Т. 3. № 2. С. 2.
2. Давид Марка, Клемент Мак Гоуэн. Методология программного анализа и проектирования. М.: Мета Технология, 2018. - 240 с.
3. Practical application of variance analysis of four-factor experience data as a technology of scientific research / N.V. Kartechina, L.V. Bobrovich, L.I. Nikonorova, N.V. Pchelinceva, R.N. Abaluev // В сборнике: IOP Conference Series: Materials Science and Engineering. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. Krasnoyarsk, Russia, 2020. С. 52030.
4. Разработка диаграммы прецедентов web-сайта / Н.В. Картечина, Р.Н. Абалуев, В.А. Шацкий, А.М. Дорохова // Наука и Образование. 2021. Т. 4. № 1.
5. Автоматизированная система управления технологическим процессом / В.И. Долженко, А.А. Автомонов, Н.В. Картечина, Н.В. Пчелинцева // Наука и Образование. 2020. Т. 3. № 2. С. 25.
6. Structure of software package for bioenergy assessment of agricultural production / Abaluev R.N., Kartechina N.V., Bobrovich L.V., Kartechina O.S.,

Chirkin S.O., Shatsky V.A. // В сборнике: Journal of Physics: Conference Series. Krasnoyarsk, Russian Federation, 2020. С. 32059.

7. "Web/ISO". International Organization for Standardization. Archived from the original on 15 December 2019.

8. Андрейчиков А.В., Андрейчикова О.Н. Анализ, синтез, планирование решений в IT - М.: Программирование и статистика, 2020. - 368 с.

9. Хатунцев В.В., Манаенков К.А., Криволапов И.П. Перспективы использования цифровизации при формировании профессиональных компетенций обучающихся технических направлений аграрного высшего образования // Наука и Образование. 2020. Т. 3. № 1. С. 41.

10. Абалуев Р.Н., Чиркин С.О., Картечина О.С. Проектирование и реализация информационно-справочной системы «Программное и аппаратное обеспечение аддитивных технологий» // Наука и Образование. 2020. Т. 3. № 4. С. 3.

**UDC 004.032.26; 003.295**

# DEVELOPMENT OF A NEPHRONIC NETWORK BASED ON THE PIX2CODE ALGORITHM

**Natalya V. Kartechina**
Candidate of Agricultural Sciences, Associate Professor
kartechnatali@mail.ru
**Anastasia A. Gushchina**
student
**Natalya V. Pchelintseva**
Senior lecturer
**Stanislav O. Chirkin**
student
stas.chirkin@bk.ru
**Alena M. Dorokhova**
student
dorohovata@mail.ru
**Vladislav A. Shatskiy**
student
shatskiy2000@list.ru
Michurinsk State Agrarian University
Michurinsk, Russia

**Annotation**. The article discusses the development of a neural network based on the pix2code algorithm.

**Key words**: pix2code algorithm, HTML, CSS, neural network, CNN, LSTM, RNN.