

Rapid Frontend Prototyping with Deep Learning

Simon Deussen

23. August 2018

Generierung von HTML/CSS Code anhand von pixelbasierten Screenshots und -designs. Aufbauend auf dem Pix2Code Paper [4], wird diese Arbeit eine eigene Implementation mit einem anderen Framework sowie einer ausführlicheren DSL erstellen. Durch automatisches Erstellen von Frontend-Code kann diese Anwendung rapide Entwicklungszyklen realisieren.

Inhaltsverzeichnis

1	Einleitung	2
2	Ähnliche Arbeiten	2
3	Motivation	3
4	Benutzte Technologien	3
4.1	Neuronale Netzwerke	3
4.2	CNN	5
4.3	RNN	5
4.4	Image Captioning Techniken	5
4.4.1	CNN-Part	5
4.4.2	RNN-Part	5
4.5	DSL	5
4.6	Training	5
4.7	Sampling	5
4.8	Keras	5
4.9	Colab	5
5	Überblick	5
6	Vision Model	5

7	Language Model	5
8	Decoder	5
9	Daten Synthese	5
10	Vergleich zu dem Original	5
11	Experimente	5
12	Zusammenfassung	5
13	Fazit	5

1 Einleitung

Viele client-basierte Anwendungen brauchen ein Frontend. Um dieses zu realisieren benötigt man eine enge Kooperation von Designern und Entwicklern. In der klassischen Entwicklung sieht diese Zusammenarbeit folgendermaßen aus:

Ein Designer macht einen grafischen Entwurf, dieser wird vom Kunde abgenommen, dann geht er zu dem Entwickler, der nun zu aller erst Markup für den Content und anschließend das Design und die richtige Darstellung nachbauen muss. Für jede grafische Veränderung muss dieser Prozess wieder ausgeführt werden. Für die meisten Entwickler ist die Markup- und CSS-Erstellung der widrigste Part der ganzen Arbeit, da er recht zeit- und aufwendig, repetitiv und langweilig ist. Es gab bisher viele Ansätze diese Arbeit zu automatisieren, z.B. durch Tools, in denen man gleichzeitig Designen und den Markup exportieren kann. Leider sind diese Tools entweder nicht besonders gut die Designs zu erstellen oder darin den Markup zu exportieren.

Eine Abhilfe soll diese Arbeit liefern: Sie ermöglicht es dem Designer mit seinem bevorzugten Tools das Design zu erstellen und dem Entwickler den fertigen Markup zu bekommen und sich vollends auf die Realisierung des Verhaltens und der Logik der Anwendung konzentrieren kann.

2 Ähnliche Arbeiten

Diese Arbeit basiert auf dem Pix2Code Paper von Tony Beltramelli [4]. Er war der erste, der Code anhand von visuellen Input generieren kann. Anderen Ansätze wie DeepCoder [3] benötigen komplizierte DSL als Input und schränken so die Benutzbarkeit stark ein. Visuelle Versuche mit Android-GUIs von Nguyen [5] benötigen ebenfalls unpraktische von Experten erstellte Heuristiken. Pix2Code ist so das erste Paper, das einen allgemeinen Input hat, und daraus momentan drei verschiedene Targetsprachen hat. Zum einen kann

es HTML/CSS Code erstellen, zum anderen aber auch Android- und iOS-Markup. Siehe Original Code auf Github [2]

3 Motivation

Computergenierte Programme werden die Zukunft der Software Entwicklung sein und diesen Bereich auch grundsätzlich verändern.

Ich denke das sich die Webtechnologien auch in der Desktop Umgebung durchsetzen. Da Plattform unabhängig und sehr stark optimiert. Sehr einfach zu lernen, weit verbreitet. Zum Beispiel Electron [1] ermöglicht den einfachen Einsatz durch einen eingebetteten Browser.

4 Benutzte Technologien

4.1 Neuronale Netzwerke

Neuronale Netzwerke sind einfach zu benutzende Modelle, welche nicht-lineare Abhängigkeiten mit vielen latenten Variablen stochastisch abbilden können. Im einfachen Sinne, sind sie gerichtete Graphen, deren Knoten oder Nodes aus ihren Inputs Werte errechnen und die an die folgenden Nodes weitergeben. Hierbei werden zwischen 3 verschiedenen Arten von Nodes unterschieden:

Input Nodes Über diese Nodes bekommt das Netzwerke die Input Parameter.

Hidden Nodes Nodes, welche das netzwerke-interne Modell repräsentieren.

Output Nodes Diese Nodes bilden die Repräsentation des Ergebnisses ab.

Jede der Nodes hat eine bestimmte Anzahl an Verbindungen, diese hängt von der Art der Nodes und deren Zweck ab. Wichtig ist jedoch, das jede Node mit mehreren anderen Nodes verbunden ist, dies soll heißen, den Output mehrerer Nodes als Input zu bekommen und den eigenen Output als Input für die folgenden Nodes weiterzuleiten. Die Stärke der Abhängigkeit zwischen zwei Nodes wird als Gewicht ausgedrückt. Jede Verbindung in einem Neuronalen Netzwerk hat ein Gewicht, welches mit dem Output der vorangegangenen Node multipliziert wird bevor es als Input weiter verwendet wird.

Nachdem die Node aus den Inputs einen Wert errechnet hat, geht dieser durch eine Aktivierungsfunktion. Diese Funktion gibt das Aktivitätslevel der Node wieder.

Das netzwerk-interne Modell wird in diesen Gewichten abgespeichert. Es repräsentiert also das Wissen, dass durch das Training entstanden ist.

Das Training eines Netzwerkes, ist das schrittweise Anpassen der Gewichte bis es ein gutes Modell des Problems gelernt hat.

4.2 CNN

4.3 RNN

4.4 Image Captioning Techniken

4.4.1 CNN-Part

4.4.2 RNN-Part

4.5 DSL

4.6 Training

4.7 Sampling

4.8 Keras

4.9 Colab

5 Überblick

6 Vision Model

7 Language Model

8 Decoder

9 Daten Synthese

10 Vergleich zu dem Original

11 Experimente

12 Zusammenfassung

13 Fazit

Literatur

[1] Electron. <https://www.electronjs.org/>.

- [2] pix2code. <https://www.github.com/tonybeltramelli/pix2code>.
- [3] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *CoRR*, abs/1611.01989, 2016.
- [4] Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot. *CoRR*, abs/1705.07962, 2017.
- [5] Tuan Anh Nguyen and Christoph Csallner. Reverse engineering mobile application user interfaces with remaui (t). *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 248–259, 2015.