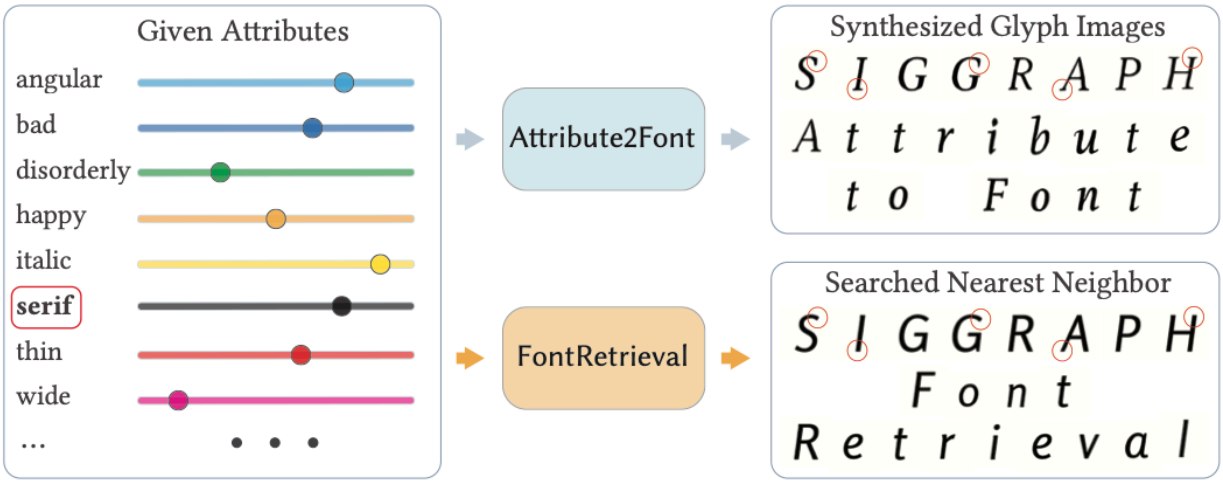
Attribute2Font: Creating Fonts You Want From Attributes

Статья: https://arxiv.org/abs/2005.07865

Идея

Идея статьи состоит в том, что мы хотим задавать атрибуты для генерации шрифта (например, сделать шрифт наклонным - italic, с засечками - serif, и т.п.). Для примера авторы приводят следующую иллюстрацию, на которой видно, что сгенерированный шрифт по заданным атрибутам лучше удовлетворяет запросу по сравнению с поиском среди существующих шрифтом с помощью ближайших соседей.



Датасет

Датасет (Exploratory font selection using crowdsourced attributes, O'Donovan et al. 2014) состоит из 148 размеченных данных (с атрибутами) и 968 неразмеченных (которые в дальнейшем обучаются). В качестве теста берется 28 шрифтов из размеченных данных. В каждом шрифте имеется по 62 изображения, но мы используем только символы (без цифр), итого получается для каждого шрифта имеется по 52 изображения с символами. Для каждого шрифта заданы 37 атрибутов со значениями от 0 до 1 (например, capital, italic, serif и тп).

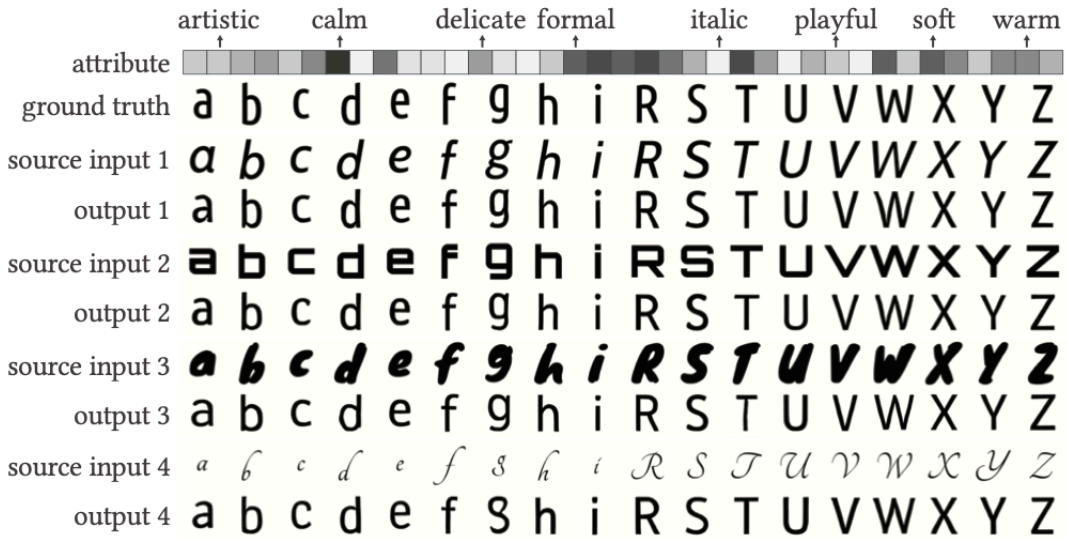
Мы хотим научиться из случайно выбранного шрифта, по заданным атрибутам генерировать новый шрифт. Для этого на вход модели мы подаем символ исходного шрифта (source content), m других символов из этого же шрифта (source style), атрибуты исходного шрифта (source attribute), такой же символ как и в source, но в другом шрифте, который мы хотим получить (grounf truth) и атрибуты этого шрифта (target attribute).

Dataloader выглядит следующим образом:

- на train мы выбираем source content из (120 + 968) * 52 = 56 756 изображений, затем выбираем m изображений символов из того же шрифта (source style) , что и source content. Затем случайным образом (с вероятностью 0.5) выбираем изображение для

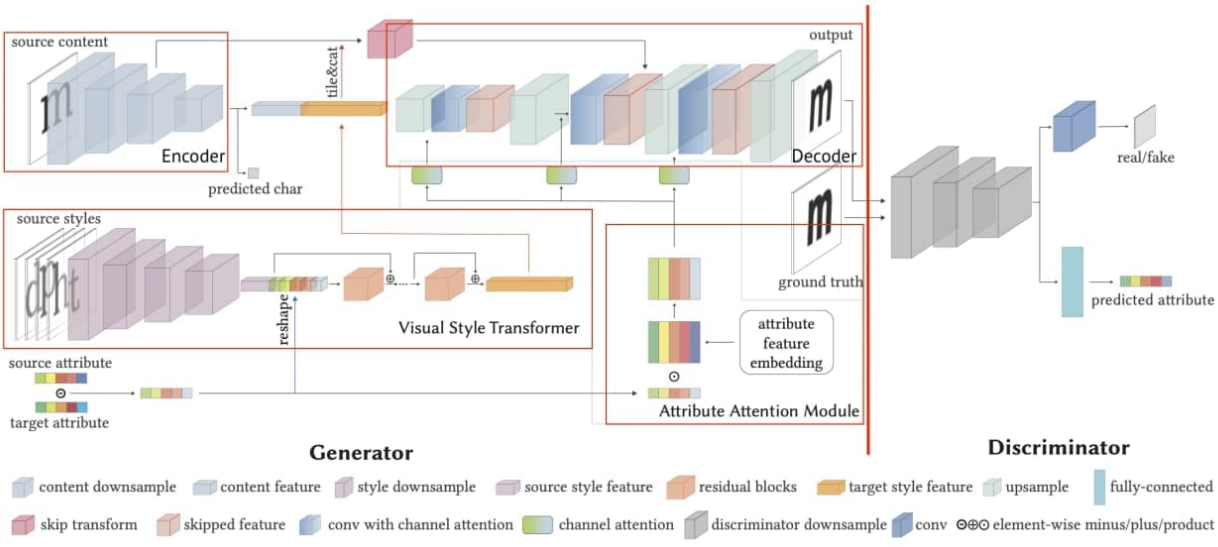
target: либо из размеченных данных, либо из неразмеченных данных.

- на test мы выбираем source content из неразмеченных данных, source style аналогично train, target выбираем из тестовых данных (28 шрифтов * 52 симлова = 1456 изображений).



Архитектура Attr2Font

Рассмотрим подробнее архитектуру предложенную авторами:



- Generator
  - Encoder. Состоит из downsample блоков, которые применяются к source content (изображение буквы из шрифта). Выход из encoder - вектор, который далее конкатенируется с выходом из Visual Style Transformer. Кроме того, этот вектор используется для предсказания буквы, которая была пропущена через encoder.

    - Decoder. Состоит из upsample блоков. На вход каждому блоку подается выход предыдущего блока, выход из Attribute Attention Module и выход соответствующего блока encoder, сконкатенированный с размноженным выходом из Visual Style Transformer(skip-connection).
    - Visual Style Transformer. На вход получает изображения из того же шрифта, что и вход для encoder, и разницу между source attribute и target attribute. С помощью residual блоков получает вектор - оценку исходного стиля и его отличие от целевого стиля, который далее используется в decoder.
    - Attribute Attention Module. На вход получает разницу source attribute и target attribute. Каждый атрибут подается в embedding слой и на выходе для каждого примера получается матрица. Это подается в channel attention слой. В нем сначала делается global average pooling, к нему применяется upsample - и так получаются веса, на которые домножается исходный тензор. Выход из Attribute Attention Module подается в каждый блок decoder.
- Discriminator - делает предсказание для полученного изображение real/fake и предсказание атрибутов полученного изображения

Обучаем мы все это с помощью следующих лоссов:

- Generator
    - $L_G = \lambda_1 l_G + \lambda_2 l_{pixel} + \lambda_3 l_{char} + \lambda_4 l_{CX} + \lambda_4 l_{attr}$
    - $l_G = -\log p(y_d = 1|\hat{x}(b,k))$ - стандартный loss генератора ($\hat{x}(b,k)$ - сгенерированный символ k в шрифте b)
    - $l_{pixel} = ||\hat{x}(b,k) - x(b,k)||$ - L1 loss между сгенерированным символом $\hat{x}(b,k)$ и настоящим target $x(b,k)$
    - $l_{char} = -\log p(y_c = k|x(a,k))$ - loss для предсказания символа на исходном изображении (символ k в шрифте a) после downsapling'a в encoder
    - $l_{CX} = CX(\hat{x}(b,k), x(b,k))$ - contextual loss, который сравнивает фичи разных слоев VGG
    - $l_{attr} = smooth_{L_1}(\hat{\alpha}(\hat{x}(b,k)) - \alpha(b))$ - разница между предсказаниями дискриминатора на атрибуты ($\hat{\alpha}$) для сгенерированного изображения ($\hat{x}(b,k)$) и настоящими атрибутами target ($\alpha(b)$)
- Discriminator
    - $L_D = l_D + l'_{attr}$
    - $l_D = -\log p(y_d = 1|x(b,k)) - \log p(y_d = 0|\hat{x}(b,k))$ - стандартный loss дискриминатора
    - $l'_{attr} = smooth_{L_1}(\hat{\alpha}(x(b,k)) - \alpha(b))$ - разница между предсказаниями дискриминатора на атрибуты ($\hat{\alpha}$) для target изображения ($x(b,k)$) и его реальными атрибутами ($\alpha(b)$)

Baseline

Авторы сравнивают результаты Attr2Font с несколькими моделями:

Model	IS	FID	LPIPS	SSIM	pix-acc	Hausdorff	Chamfer
AttGAN	3.4008	200.1708	0.24039	0.6198	0.5287	11.2044	330.238
StarGAN	**3.6179**	91.1436	0.12172	0.7024	0.7146	8.8748	317.818
RelGAN	3.1412	183.0307	0.23220	0.6216	0.5380	11.1048	339.822
STGAN	3.6178	83.3167	0.11779	0.7150	0.7444	8.7815	286.509
Attr2Font	3.0740	**26.8779**	**0.08742**	**0.7580**	**0.8153**	**7.1954**	**241.670**

Мы в качестве baseline брали StarGAN и подавали ему на вход source content (изображение символа в каком-нибудь шрифте) и его аттрибуты (так как аттрибуты заданы от 0 до 1, то для StarGAN мы трансформировали их в бинарные).

Параметры модели

Авторы проводили анализ для параметра m (количество изображений в качестве source style): чем выше значение, те лучше результаты. Но при увеличении параметра m, также растет вычислительная сложность модели, поэтому в качестве оптимального параметра авторы статьи предлагают $m = 4$. Также проводилась оценка параметра N_{rb} (количество residual blocks в Visual Style Transformer) при фиксированном параметре $m = 4$. И в качестве баланса между результатами и вычислительной сложностью модели авторы предлагают $N_{rb} = 16$. Но мы чтобы снизить вычислительную сложность выбрали значение $N_{rb} = 8$. Остальные параметры мы брали как в оригинальной статье: $batch\ size = 16$, $learning\ rate = 0.0002$, $image\ size = 64 \times 64$, $embedding\ dim = 64$, $attribute\ num = 37$ (деленные на 100, чтобы значения были в интервале [0, 1]) и коэфициенты для generator loss: $\lambda_1 = 5$, $\lambda_2 = 50$, $\lambda_3 = 5$, $\lambda_4 = 5$, $\lambda_5 = 20$.

Результаты

Мы обучали Attr2Font и StarGAN. StarGAN обучается гораздо быстрее, поэтому результаты мы сравнивали по истечению одинакового количества эпох: 107(в статье - 500). Возникла проблема со StarGAN - он выдает идентичные входным данным изображения. Метрики, которые мы выбрали - FID и LPIPS. По ним сравниваются распределение сгенерированных изображений и ground truth изображений с target атрибутами. Результаты Attr2Font лучше, чем StarGAN.

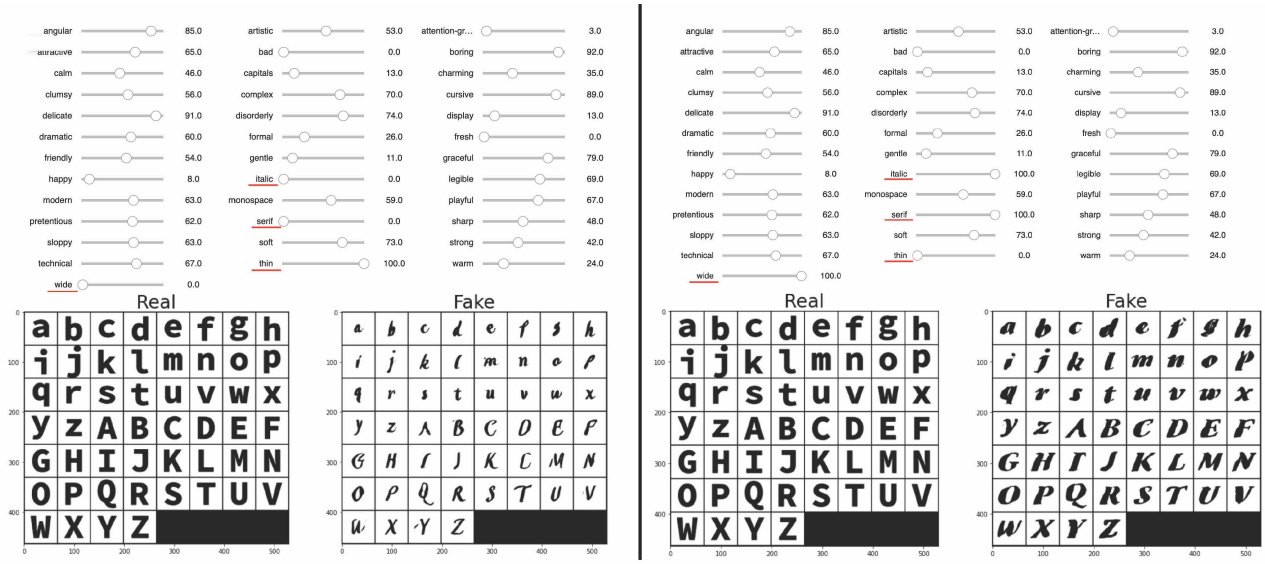
	FID	LPIPS
Attr2Font	85.25	8.79
StarGAN	157.93	19.35

Визуализация

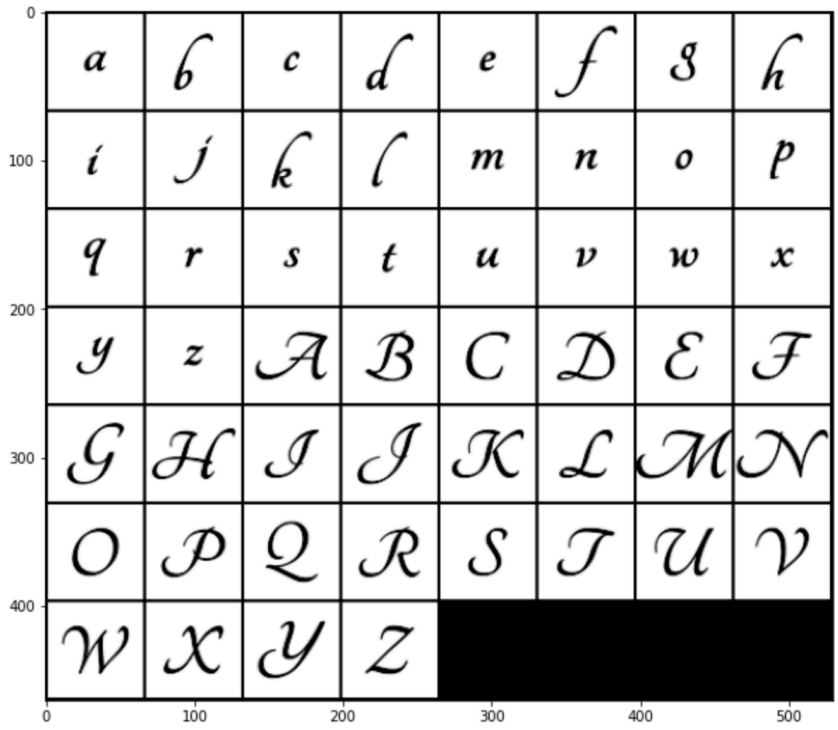
Google Colab: https://colab.research.google.com/drive/1_3_HXNhasK5qZR3nvziMixiOvfPJKd4Z#scrollTo=oNg99iWJuAOP

По ссылке на Google Colab можно запустить модель и потестировать, позадавать различные атрибуты и посмотреть на сегенированные шрифты.

Ниже приведены примеры: генерация шрифта на основе одного и того же шрифта, но с разными атрибутами (те что различаются подчеркнуты красным). Можно заметить, что при измении атрибутов действительно меняется шрифта (шрифт стал наклоннее - italic, толще - wide и thin и есть намеки на засечки - serif).



Также, как уже упоминалось, в датасете было 968 неразмеченных шрифтов, и модель выучила их атрибуты. Ниже приведен пример выученных атрибутов для неразмеченного шрифта:



Атрибуты:

```
angular: 95.99, artistic: 97.95, attention-grabbing: 40.54, attractive: 85.49, bad: 31.43, boring: 19.24, calm: 69.51, capitals: 28.74, charming: 94.07, clumsy: 84.37, complex: 95.80, cursive: 87.77, delicate: 89.75, disorderly: 80.99, display: 68.98, dramatic: 83.25, formal: 45.60, fresh: 74.59, friendly: 76.44, gentle: 77.88, graceful: 95.88, happy: 62.64, italic: 78.76, legible: 27.86, modern: 81.05, monospace: 9.09, playful: 95.36, pretentious: 57.84, serif: 53.15, sharp: 55.34, sloppy: 99.43, soft: 82.81, strong: 27.32, technical: 83.93, thin: 95.89, warm: 54.15, wide: 13.05
```

Похоже на правду, шрифт наклонный, а значение для italic достаточно высокое (78.76), текст тонкий и значения wide и thin этому соответсвуют.