Распознавание текста на основе скелетного представления толстых линий и сверточных сетей

Соломатин А.А.

Московский физико-технический институт (Государственный университет)
solomatin.aa@phystech.edu

Аннотация Задача распознавания символа на картинке является чрезвычайно распространенной задачей машинного обучения. В связи с этим существует несколько основных подходов к решению. В первом методе мы используем сверточные нейросети небольшой глубины для анализа изображения. В другом методе мы представляем символ в виде графа. Из скелета этого графа строится вектор признаков, который затем обрабатывается нейросетью. Также приводится сравнение точности классификации этих алгоритмов на датасете MNIST.

Ключевые слова: сверточные нейронные сети, CNN, распознавание символов, скелетное представление, Graph embedding

1 Введение

Распознавание текста на изображении это классическая задача компьютерного зрения в машинном обучении. В данной работе мы имеем дело с изображениями символов. Регулярно появляются работы, призванные улучшить точность классификации. Основным подходом к решению является использование сверточных слоев в нейронных сетях, описанное в (1) (2). Для улучшения качества нам предложено использовать векторное представление изображения вместо растрового. Векторное представление получается из графого представления путем построения скелетов символов.

При скелетизации внутренности символов заполняются кругами, центры которых являются вершинами графа и соединяются его ребрами. На сегодняшний день существует много исследований на тему скелетизации. В работе (3) описывается математический аппарат для генерации рукописного текста с помощью жирных линий, а в работе (4) сравниваются формы растровых изображений при помощи скелетизации. В тоже время разработаны и альтернативные подходы, описанные в статье (5).

Граф, получаемый после скелетизации, не может быть подан на вход нейронной сети. Поэтому нам необходимо векторизовать граф с помощью его эмбеддинга. Для оценки качества работы алгоритма предлагается использовать датасет MNIST. Он состоит из рукописных цифр и использовался для валидации во многих работах, таких как (7), (8), (9).

2 Постановка задачи

Введем следующие обозначения:

- $\mathbb{I}$ - множество пар черно-белых бинаризованных изображений и сопоставленных им меток, символов из алфавита $\mathbb{A}$. На изображении содержится только один символ.
  Пусть функция $a : \mathbb{I} \to \mathbb{A}$ однозначно сопоставляет каждому изображению из $\mathbb{I}$ его метку класса.
- $\mathbb{G}$ - пространство графов скелетных представлений символов. Под скелетным представлением здесь подразумевается неориентированный граф, каждой вершине которого сопоставлено некоторое число, называемое радиусом.
  Функция $g : \mathbb{I} \to \mathbb{G}$ однозначно сопоставляет каждому изображению из $\mathbb{I}$ его скелетное представление. В данной работе в качестве такой функции используется алгоритм, описанный в работе (1).
- $\mathbb{V}$ - векторное пространство. Будем считать что граф представим в векторном виде сюрьективным преобразованием $v : \mathbb{G} \to \mathbb{V}$
- $\mathbb{F}$ - пространство признаков графов, описывающих скелетное представление символов.
  Функция $f : \mathbb{I} \to \mathbb{G}$ однозначно сопоставляет каждому элементу из $\mathbb{G}$ множество его признаков.
- Необходимо также ввести преобразование, которое вычисляет признаки графа. $t : \mathbb{G} \to \mathbb{F}$
- $\mathbb{Y}$ - пространство меток (номеров символов в алфавите).

Тогда, задачей данного исследования является нахождение функций:$y : \mathbb{I} \to \mathbb{Y}$, $h : \mathbb{F} \to \mathbb{Y}$, $d : \mathbb{V} \to \mathbb{Y}$,

Минимизируется функционал, который вычисляет функцию ошибки, L - cross entropy loss, для различных функций.

3 Решения

3.1 Неронная сеть

В качестве первого базового алгоритма было решено использовать сверточную нейронную сеть. Она написана с использованием библиотеки keras:

```
model = Sequential()
model.add(Conv2D(30, (5, 5),
    input_shape=(1, 28, 28),
    activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(15, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

3.2 Градиентный бустинг

В качестве второй базовой модели был использован LightGBM Classifier, который обучился на признаках из скелетного представления. Данный алгоритм основывается на градиентном бустинге над деревьями решений.

3.3 Смешанная модель

Смешанная модель, работающая с изображением и признаками, полученными с помощью градиентного бустинга. Признаки добав- ляются после прохождения картинкой свёрточных фильтров и преобразования в одномерный тензор. Нейронная сеть написана с помощью библиотеки keras:

```
model_cnn_input = Input((28, 28, 1))
cnn = Conv2D(30, (3, 3), activation='relu', input_shape = input_shape_cnn)(model_cnn_input)
cnn = Conv2D(60, (3, 3), activation='relu')(cnn)
cnn = MaxPooling2D(pool_size=(2, 2))(cnn)
cnn = BatchNormalization()(cnn)
cnn = Activation('relu')(cnn)
cnn = Dropout(0.2)(cnn)
cnn = MaxPooling2D(pool_size=(2, 2))(cnn)

cnn = Flatten()(cnn)
cnn = Dense(128, activation='relu')(cnn)

features_input = Input(input_dim_skelet)
features_layer1 = Dense(32, input_dim=input_dim_skelet)(features_input)

merged = Concatenate()([cnn, features_layer1])
layer1 = Dropout(0.5)(merged)
output_layer = Dense(num_classes, activation='softmax')(layer1)

model = Model(inputs=[model_cnn_input, features_input], outputs=output_layer)
model.compile(loss="categorical_crossentropy",
              optimizer="adam",
              metrics=['accuracy', f1])
```

4 Планирование эксперимента

4.1 Датасет

В исследовании используется набор изображений с рукописным текстом MNIST. К каждому изображению построен его скелетный граф, методом,

описанным в (3). Для скелетизации, картинка должна быть бинаризованной. В этих данных бинаризация была следующая: все пиксели, кроме черных(то есть > 0 в нотации цветов от 0 до 255), обращаются в белый. Вершины графа описываются четыремя числами: координатами по оси абсцисс, ординат, степень вершины (она может быть от 1 до 3, таков алгоритм построения скелета), радиальная функция в этой точке, или же, радиус вписанной в фигуру окружности в этой точке.

4.2 Неронная сеть

Модель на вход принимает черно-белые изображения размером 28 на 28. На выходе дает вероятности принадлежности к каждому из классов. Для обучения нейросети было произведено разбиение выборки на обучение, валидацию и тестирование в соотношении 600000:5000:5000. Веса оптимизировались алгоритмом AdaDelta.

4.3 Градиентный бустинг

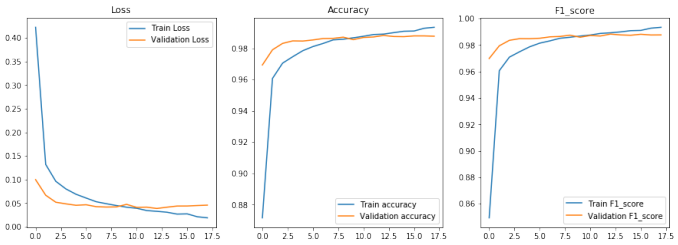
Для обучения и предсказания использовалась модель LGBMClassifier. На вход модель принимала различного рода статистики, посчитанные на скелетном представлении изображения. Используемые статистики:

- Количество вершин в скелетном представлении
- Среднее значение среди всех чисел, как представленных в описании графа, так и по каждой из 8 подряд идущих чисел
- Сумма как всех чисел, представленных в описании графа, так и по каждой из 8 подряд идущих чисел
- Дисперсия как среди всех чисел, представленных в описании графа, так и по каждой из 8 подряд идущих чисел
- Разность между максимальным и минимальным числом как среди всех чисел, представленных в описании графа, так и по каждой из 8 подряд идущих чисел
- Перцентили от 0 до 1 с шагом 0.1 как среди всех чисел, представленных в описании графа, так и по каждой из 8 подряд идущих чисел
- Для каждой из четырёх категорий самое частое значение и сколько раз оно встретилось

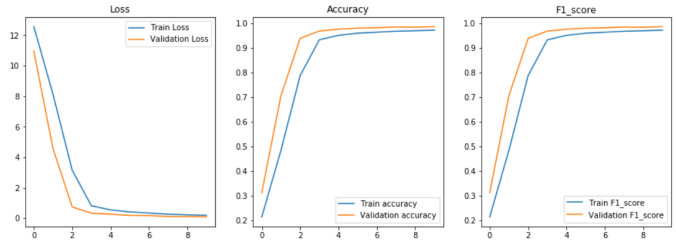
5 Анализ ошибок

Ниже приведены графики функции потери, а так же точности и F1 метрики на валидации Для первого эксперимента со сверточной нейросетью и третьего эксперимента со сверточной нейросетью и признаками из второго эксперимента.

Анализ функции ошибки и score обычной сверточной нейронной сети



Анализ функции ошибки и score в смешанном эксперименте



6 Анализ структуры модели

6.1 Неронная сеть

Путем перебора была подобрана финальная конфигурация фильтров свёрточных слоёв. Конфигурация слоев

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 24, 24)	780
max_pooling2d_1 (MaxPooling2	(None, 30, 12, 12)	0
conv2d_2 (Conv2D)	(None, 15, 10, 10)	4065
max_pooling2d_2 (MaxPooling2	(None, 15, 5, 5)	0
dropout_1 (Dropout)	(None, 15, 5, 5)	0
flatten_1 (Flatten)	(None, 375)	0
dense_1 (Dense)	(None, 128)	48128
dense_2 (Dense)	(None, 50)	6450
dense_3 (Dense)	(None, 10)	510

Total params: 59,933
Trainable params: 59,933
Non-trainable params: 0

6.2 Градиентный бустинг

После построения признаков были проведенны эксперименты по изменению параметров модели. Количество листьев было увеличено до 110. learning$_rate, n_estimators$7.

6.3 Смешанная модель

Итоговая структура комбинированной нейронной сети: Конфигурация слоев

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 1)	0	
conv2d_1 (Conv2D)	(None, 26, 26, 30)	300	input_1[0][0]
conv2d_2 (Conv2D)	(None, 24, 24, 60)	16260	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 60)	0	conv2d_2[0][0]
batch_normalization_1 (BatchNor	(None, 12, 12, 60)	240	max_pooling2d_1[0][0]
activation_1 (Activation)	(None, 12, 12, 60)	0	batch_normalization_1[0][0]
dropout_1 (Dropout)	(None, 12, 12, 60)	0	activation_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 60)	0	dropout_1[0][0]
flatten_1 (Flatten)	(None, 2160)	0	max_pooling2d_2[0][0]
input_2 (InputLayer)	(None, 88)	0	
dense_1 (Dense)	(None, 128)	276608	flatten_1[0][0]
dense_2 (Dense)	(None, 32)	2848	input_2[0][0]
concatenate_1 (Concatenate)	(None, 160)	0	dense_1[0][0] dense_2[0][0]
dropout_2 (Dropout)	(None, 160)	0	concatenate_1[0][0]
dense_3 (Dense)	(None, 10)	1610	dropout_2[0][0]

Total params: 297,866
Trainable params: 297,746

7 Выбор модели

Модель	Accuracy	F1-score
Нейросеть:	**0.9911**	**0.9897**
LightGBM:	**0.9311**	**0.9309**
Смешанная модель	**0.9849**	**0.9851**

8 Результат работы

В результате приведенного исследования было рассмотренно несколько подходов к решению задачи распознования символов на изображении. Один из методов использовал сверточные сети для классификации изображений. Другой метод заключался в анализе графовых струтур с помощью скелетного представления, полученных по изображению. Так же были приведены сравнения точности этих подходов и архитектур на датасетах MNIST.

Литература

[1] LeCun Y. et al. Convolutional networks for images, speech, and time series //The handbook of brain theory and neural networks. – 1995. – Т. 3361. – №. 10. – С. 1995.

[2] Ciresan D. C. et al. Convolutional neural network committees for handwritten character classification //Document Analysis and Recognition (ICDAR), 2011 International Conference on. – IEEE, 2011. – С. 1135-1139.

[3] Клименко С. В., Местецкий Л. М., Семенов А. Б. Моделирование рукописного шрифта с помощью жирных линий //Труды. – 2006. – Т. 16.

[4] Кушнир О. и др. Сравнение формы бинарных растровых изображений на основе скелетизации //Машинное обучение и анализ данных. – 2012. – Т. 1. – №. 3. – С. 255-263.

[5] Масалович А., Местецкий Л. Распрямление текстовых строк на основе непрерывного гранично-скелетного представления изображений //Труды Международной конференции «Графикон», Новосибирск.–2006.–4 с.

[6] LeCun Y., Cortes C., Burges C. J. MNIST handwritten digit database // Available: http://yann. lecun. com/exdb/mnist. – 2010. – Т. 2.

[7] Zhu D. et al. Negative Log Likelihood Ratio Loss for Deep Neural Network Classification //arXiv preprint arXiv:1804.10690. – 2018.

[8] Nair P., Doshi R., Keselj S. Pushing the limits of capsule networks //Technical note. – 2018.

[9] Hsieh P. C., Chen C. P. Multi-task Learning on MNIST Image Datasets. – 2018.