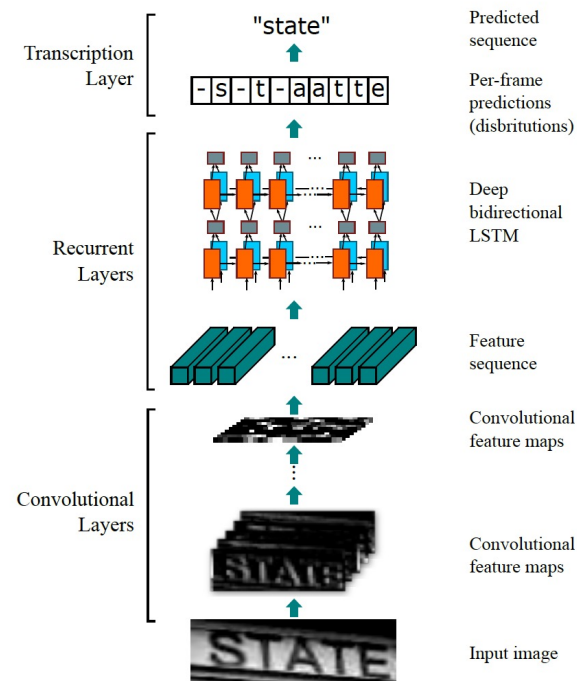


CRNN(Convolutional Recurrent Neural Networks)

- Shi, Baoguang, Xiang Bai, and Cong Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition." IEEE transactions on pattern analysis and machine intelligence 39.11 (2016): 2298-2304.



Abstract

- Compared with previous systems for scene text recognition, the proposed architecture possesses four distinctive properties:
 - ① It is **end-to-end trainable**, in contrast to most of the existing algorithms whose components are separately trained and tuned.
 - ② It **naturally handles** sequences in **arbitrary lengths**, involving no character segmentation or horizontal scale normalization.
 - ③ It is not confined to any predefined lexicon and **achieves remarkable performances** in both lexicon-free and lexicon-based scene text recognition tasks.
 - ④ It generates an **effective** yet much **smaller model**, which is more practical for real-world application scenarios.

Introduction

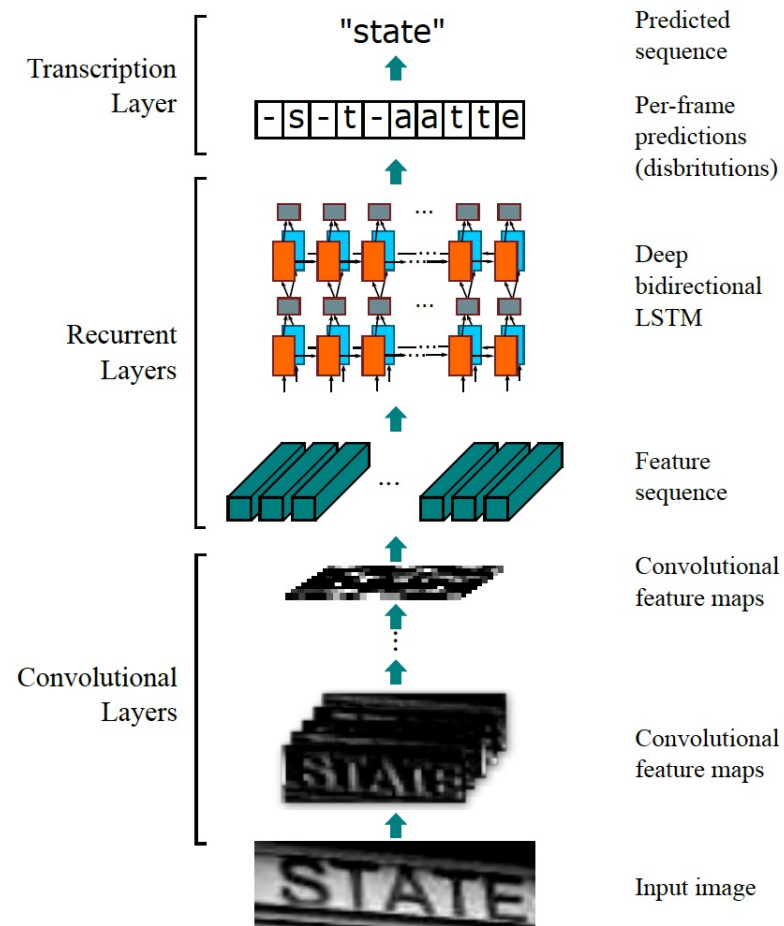
- DCNN models often operate on inputs and outputs with fixed dimensions, and thus are incapable of producing a variable-length label sequence.
- The proposed neural network model is named a Convolutional Recurrent Neural Network (CRNN), since it is a combination of DCNN and RNN.
- CRNN possesses several distinctive advantages over conventional neural network models:

The Proposed Network Architecture

- ① At the bottom of CRNN, the **convolutional layers** automatically **extract a feature sequence** from **each input image**.
- ② On top of the convolutional network, a **recurrent network** is built for **making prediction for each frame** of the feature sequence, outputted by the convolutional layers.
- ③ The **transcription layer** at the top of CRNN is adopted to **translate the per-frame predictions** by the recurrent layers into a label sequence.

The Proposed Network Architecture

- CNN + RNN + Transcription Layer



Feature Sequence Extraction

- Before being fed into the network, all the images need to be scaled to the same height.
- Specifically, each feature vector of a feature sequence is generated from left to right on the feature maps by column.
- This means the i -th feature vector is the concatenation of the i -th columns of all the maps.

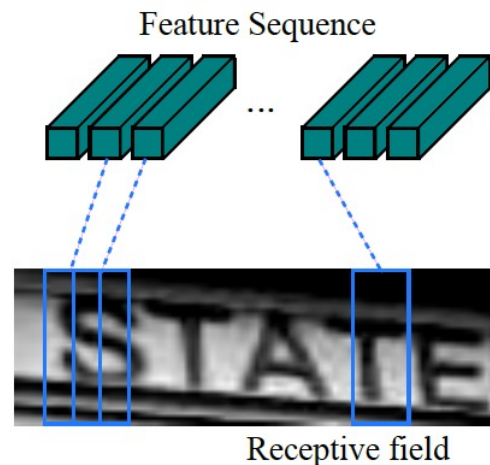


Figure 2. The receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field.

Sequence Labelling

- A deep **bidirectional Recurrent Neural Network** is built on the top of the convolutional layers, as the recurrent layers.
- The advantages of the recurrent layers are three-fold.
 - ① Firstly, RNN has a strong capability of **capturing contextual information** within a sequence. Besides, some ambiguous characters are easier to distinguish when observing their contexts, e.g. it is **easier to recognize “il”** by contrasting the character heights than by recognizing each of them separately.
 - ② the convolutional layer, allowing us to jointly train the recurrent layers and the convolutional layers in a **unified network**.
 - ③ Thirdly, RNN is able to operate on sequences of **arbitrary lengths**, traversing from starts to ends.

Sequence Labelling

- RNN의 필요성 – Context 정보 추가

i l vs i 1

Sequence Labelling

- resulting in a deep bidirectional LSTM as illustrated in Fig. 3.b.

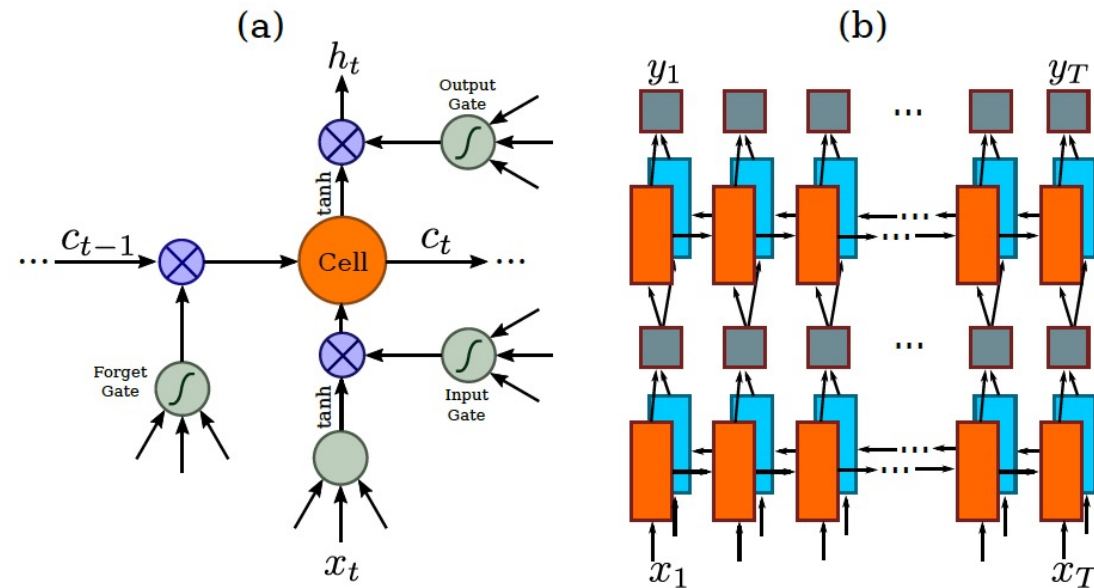


Figure 3. (a) The structure of a basic LSTM unit. An LSTM consists of a cell module and three gates, namely the input gate, the output gate and the forget gate. (b) The structure of deep bidirectional LSTM we use in our paper. Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.

Transcription

- Mathematically, transcription is to **find the label sequence** with the **highest probability conditioned on the per-frame predictions**.
- We adopt the conditional probability defined in the **Connectionist Temporal Classification (CTC) layer** proposed by Graves et al. [15].
- The formulation of the conditional probability is briefly described as follows:
The input is a sequence $y = y_1, \dots, y_T$ where T is the sequence length.
- Here, each $y_t \in \mathfrak{R}^{|L'|}$ is a probability distribution over the set $L' = L \cup \cdot$, where **L contains all labels in the task (e.g. all English characters)**, as well as a **'blank' label** denoted by \cdot .

Transcription

- A sequence-to-sequence mapping function β is defined on sequence $\pi \in L'^T$, where T is the length.
- β maps onto l by
 - ① firstly removing the repeated labels,
 - ② then removing the 'blank's.
- β maps onto l by For example, B maps “--hh-e-l-l-oo--” (‘-’ represents 'blank') onto “hello”.
- Then, the conditional probability is defined as the sum of probabilities of all that are mapped by β onto l :

$$p(l|y) = \sum_{\pi: \beta(\pi)=l} p(\pi|y)$$

Network Training

- The objective is to minimize the negative log-likelihood of conditional probability of ground truth:

$$O = \sum_{I_i, l_i \in \chi} \log p(l_i | y_i)$$

- For optimization, we use the ADADELTA [37] to automatically calculate per-dimension learning rates.

Datasets

- For all the experiments for scene text recognition, we use the **synthetic dataset (Synth)** released by Jaderberg et al. [20] as the training data.
- The dataset contains **8 millions training images** and their corresponding ground truth words.
- Even though the **CRNN model is purely trained with synthetic text data**, it **works well on real images** from standard text recognition benchmarks.

Implementation Details

- The architecture of the convolutional layers is based on the VGG-VeryDeep architectures [32].

Table 1. Network configuration summary. The first row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively

| Type | Configurations |
|--------------------|---------------------------------------|
| Transcription | - |
| Bidirectional-LSTM | #hidden units:256 |
| Bidirectional-LSTM | #hidden units:256 |
| Map-to-Sequence | - |
| Convolution | #maps:512, k: 2×2 , s:1, p:0 |
| MaxPooling | Window: 1×2 , s:2 |
| BatchNormalization | - |
| Convolution | #maps:512, k: 3×3 , s:1, p:1 |
| BatchNormalization | - |
| Convolution | #maps:512, k: 3×3 , s:1, p:1 |
| MaxPooling | Window: 1×2 , s:2 |
| Convolution | #maps:256, k: 3×3 , s:1, p:1 |
| Convolution | #maps:256, k: 3×3 , s:1, p:1 |
| MaxPooling | Window: 2×2 , s:2 |
| Convolution | #maps:128, k: 3×3 , s:1, p:1 |
| MaxPooling | Window: 2×2 , s:2 |
| Convolution | #maps:64, k: 3×3 , s:1, p:1 |
| Input | $W \times 32$ gray-scale image |

Implementation Details

- During training, all images are scaled to 100 x 32 in order to accelerate the training process.
- Testing images are scaled to have height 32.
- Widths are proportionally scaled with heights, but at least 100 pixels.

Experiments

Table 2. Recognition accuracies (%) on four datasets. In the second row, “50”, “1k”, “50k” and “Full” denote the lexicon used, and “None” denotes recognition without a lexicon. (*[22] is not lexicon-free in the strict sense, as its outputs are constrained to a 90k dictionary).

| | IIT5k | | | SVT | | IC03 | | | | IC13 |
|--------------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------|--------------|
| | 50 | 1k | None | 50 | None | 50 | Full | 50k | None | None |
| ABBYY [34] | 24.3 | - | - | 35.0 | - | 56.0 | 55.0 | - | - | - |
| Wang <i>et al.</i> [34] | - | - | - | 57.0 | - | 76.0 | 62.0 | - | - | - |
| Mishra <i>et al.</i> [28] | 64.1 | 57.5 | - | 73.2 | - | 81.8 | 67.8 | - | - | - |
| Wang <i>et al.</i> [35] | - | - | - | 70.0 | - | 90.0 | 84.0 | - | - | - |
| Goel <i>et al.</i> [13] | - | - | - | 77.3 | - | 89.7 | - | - | - | - |
| Bissacco <i>et al.</i> [8] | - | - | - | 90.4 | 78.0 | - | - | - | - | 87.6 |
| Alsharif and Pineau [6] | - | - | - | 74.3 | - | 93.1 | 88.6 | 85.1 | - | - |
| Almazán <i>et al.</i> [5] | 91.2 | 82.1 | - | 89.2 | - | - | - | - | - | - |
| Yao <i>et al.</i> [36] | 80.2 | 69.3 | - | 75.9 | - | 88.5 | 80.3 | - | - | - |
| Rodríguez-Serrano <i>et al.</i> [30] | 76.1 | 57.4 | - | 70.0 | - | - | - | - | - | - |
| Jaderberg <i>et al.</i> [23] | - | - | - | 86.1 | - | 96.2 | 91.5 | - | - | - |
| Su and Lu [33] | - | - | - | 83.0 | - | 92.0 | 82.0 | - | - | - |
| Gordo [14] | 93.3 | 86.6 | - | 91.8 | - | - | - | - | - | - |
| Jaderberg <i>et al.</i> [22] | 97.1 | 92.7 | - | 95.4 | 80.7* | 98.7 | 98.6 | 93.3 | 93.1* | 90.8* |
| Jaderberg <i>et al.</i> [21] | 95.5 | 89.6 | - | 93.2 | 71.7 | 97.8 | 97.0 | 93.4 | 89.6 | 81.8 |
| CRNN | 97.6 | 94.4 | 78.2 | 96.4 | 80.8 | 98.7 | 97.6 | 95.5 | 89.4 | 86.7 |

OCR Open Dataset 1 - COCO-Text

- <https://bgshih.github.io/cocotext/>

Dataset Explorer

26187/53686 PREVIOUS NEXT ☒ Show Annotations



Image ID: 354626
Number of instances: 19

| | | |
|-----------------|-----------|---|
| Instance 102904 | "Central" | ▼ |
| Instance 102905 | "Ipswich" | ▼ |
| Instance 102906 | "Nacton" | ▼ |
| Instance 102907 | "Road" | ▼ |
| Instance 102908 | "John" | ▼ |
| Instance 102909 | "JN" | ▼ |
| Instance 102910 | "F" | ▼ |
| Instance 102911 | "Y294" | ▼ |
| Instance 102912 | - | ▼ |
| Instance 102913 | "BUSES" | ▼ |
| Instance 102914 | - | ▼ |

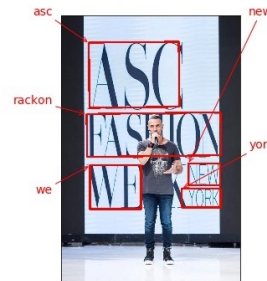
OCR Open Dataset 2 - FSNS

- <http://rrc.cvc.uab.es/?ch=6&com=introduction>



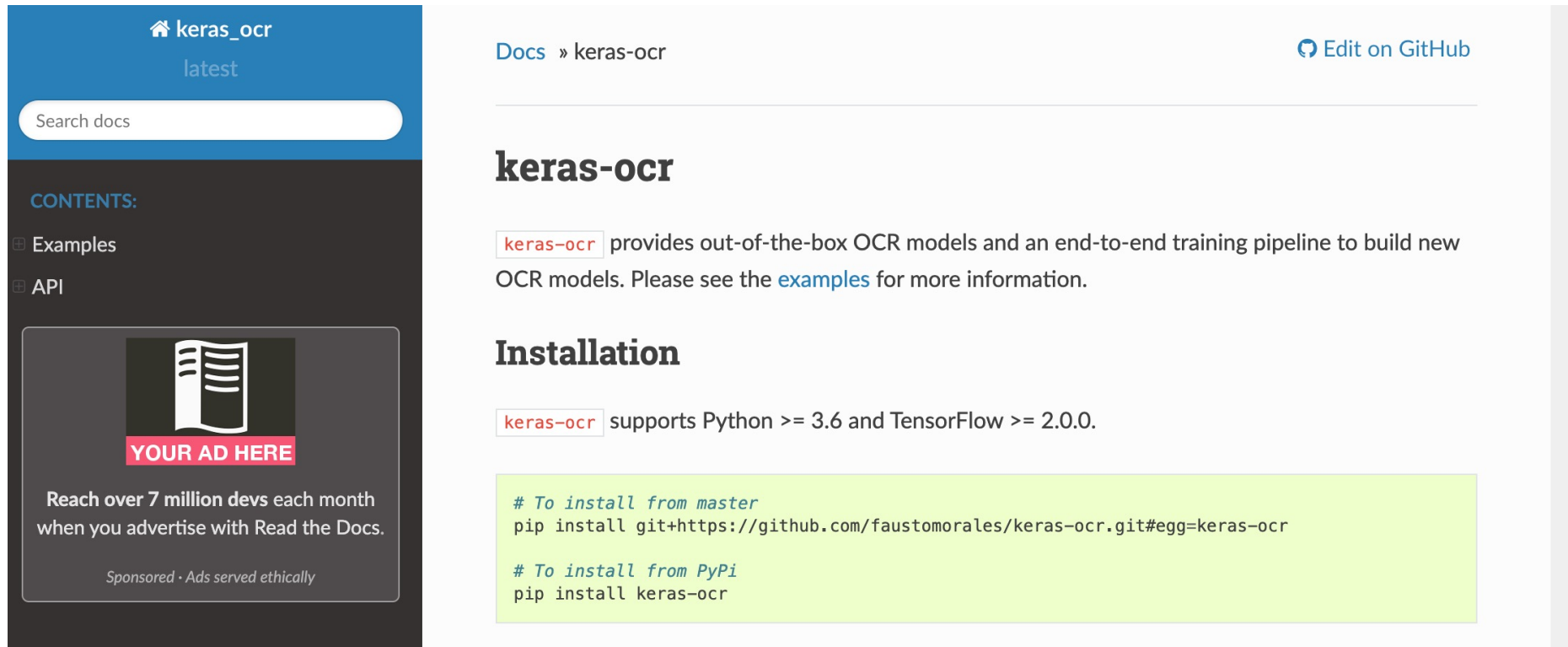
TensorFlow 2.0 CRNN Implementation

- <https://github.com/faustomorales/keras-ocr>



Keras-ocr documentation

- <https://keras-ocr.readthedocs.io/en/latest/>



The screenshot displays the Keras-OCR documentation page on Read the Docs. The left sidebar features a blue header with 'keras_ocr' and 'latest', a search bar, and a 'CONTENTS' section with links to 'Examples' and 'API'. Below this is an advertisement for Read the Docs, stating it reaches over 7 million developers monthly and is sponsored. The main content area has a breadcrumb 'Docs » keras-ocr', an 'Edit on GitHub' link, and a title 'keras-ocr'. The text describes the package as providing out-of-the-box OCR models and an end-to-end training pipeline, with a link to examples. An 'Installation' section follows, noting support for Python >= 3.6 and TensorFlow >= 2.0.0. A code block provides instructions for installing from the master branch or via PyPi.

keras_ocr
latest

Search docs

CONTENTS:

- Examples
- API

YOUR AD HERE

Reach over 7 million devs each month when you advertise with Read the Docs.

Sponsored · Ads served ethically

Docs » keras-ocr [Edit on GitHub](#)

keras-ocr

`keras-ocr` provides out-of-the-box OCR models and an end-to-end training pipeline to build new OCR models. Please see the [examples](#) for more information.

Installation

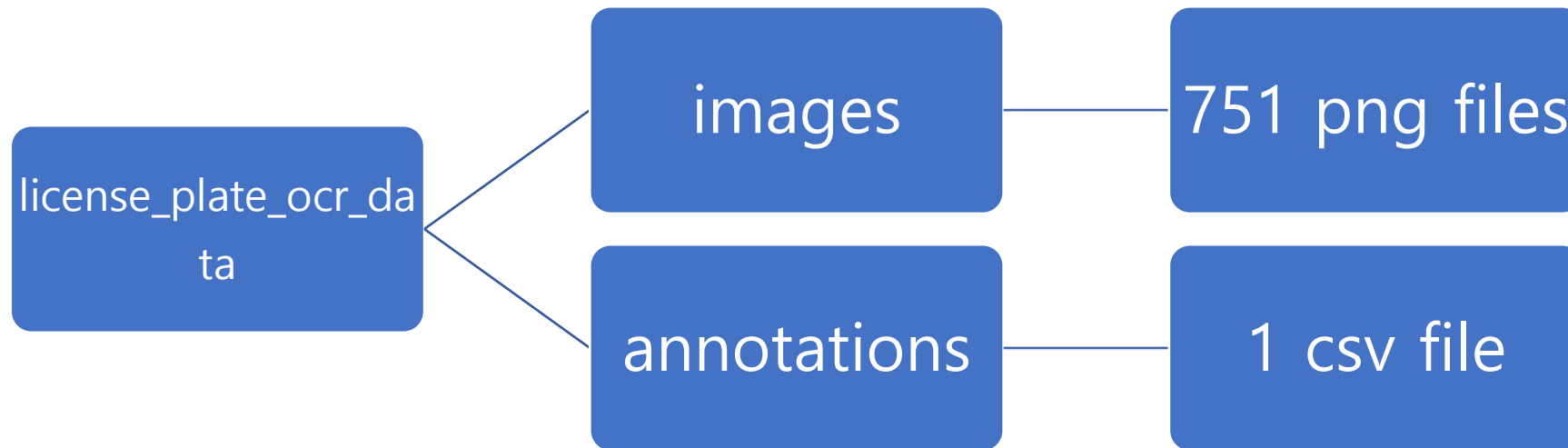
`keras-ocr` supports Python >= 3.6 and TensorFlow >= 2.0.0.

```
# To install from master
pip install git+https://github.com/faustomorales/keras-ocr.git#egg=keras-ocr

# To install from PyPi
pip install keras-ocr
```

License Plate Dataset

- <https://drive.google.com/file/d/16INTGvRGFooolxFX2IR3n-g24NkqDEX6/view>
- 751 license plate images



License Plate Data Format

① Image:



② Annotation:

ak399.png,ak,FGJ235

filename, state alias, groundtruth label

실습 - License Plate OCR Dataset에 대해 CRNN Recognizer 학습

- 기존의 CRNN 모델을 License Plate OCR Dataset에 적합한 파라미터로 Fine-Tuning 해봅시다.



ak399.png
77.9 kB



ak721.png
86.3 kB



ak848.png
82.9 kB



ak1165.png
90.7 kB



al47.png
94.6 kB



al145.png
87.9 kB



al1156.png
88.5 kB



al1181.png
95.7 kB



al1204.png
64.1 kB



al1247.png
84.1 kB



al1259.png
85.6 kB



al1528.png
90.6 kB



al1662.png
93.6 kB



ar127.png
81.8 kB



ar285.png
89.2 kB



ar477.png
66.5 kB



ar480.png
84.0 kB



ar624.png
73.9 kB



ar726.png
60.4 kB



ar785.png
85.7 kB

Thank you!
