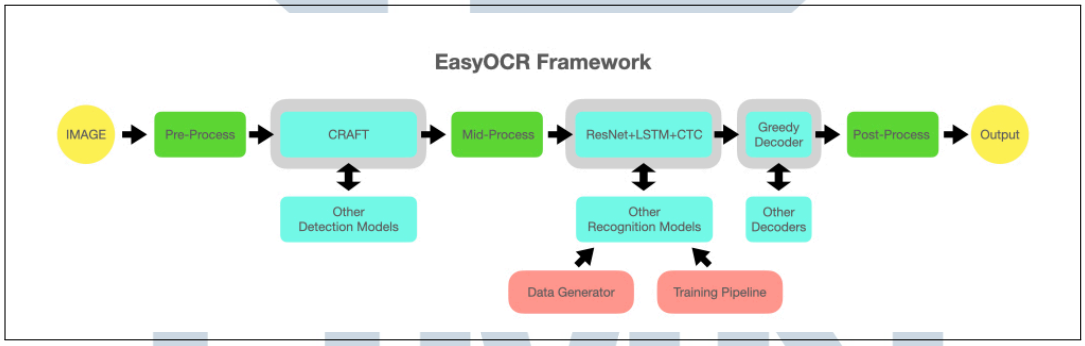
BAB 2
LANDASAN TEORI

2.1 EasyOCR

EasyOCR adalah Python package open-source yang dapat dipakai untuk melakukan OCR dengan mudah. EasyOCR dibangun dengan PyTorch sebagai pengendali *backend*-nya untuk memungkinkan *deep learning*. Terdapat beberapa langkah yang dilakukan oleh EasyOCR untuk mendeteksi tulisan di dalam sebuah gambar (lihat Gambar 2.1). Pertama akan dilakukan preprocessing seperti *gray scaling* untuk mengekstrak tulisan, lalu akan diterapkan algoritma Character Region Awareness for Text Detection (CRAFT) untuk mendeteksi tulisan tersebut. Model untuk *text recognition* akan menggunakan Convolutional Recurrent Neural Networks (CRNN) yang terdiri atas tiga komponen utama, yaitu *feature extraction* (ResNet), *sequence labeling* (LSTM), dan *decoding* (CTC) [9].



Gambar 2.1. Alur cara kerja EasyOCR [9]

Dari framework yang digambarkan oleh JaidedAI, yaitu pembuat EasyOCR, bisa dilihat bahwa gambar yang diterima akan dilakukan pre-processing dan dilanjutkan ke model deteksi tulisan bernama CRAFT yang mendeteksi daerah tulisan dengan menulusuri daerah setiap huruf dan jarak antara huruf. Setelah mid-processing, akan diteruskan ke model pengenalan tulisan yang menggunakan model CRNN yang berisi ResNet, LSTM, dan CTC.

```
1   import easyocr
2   reader = easyocr.Reader(['en'])
3   result = reader.readtext('file_path.jpg')
```

Gambar 2.2. Kode inisialisasi EasyOCR

Dari Gambar 2.2, bisa terlihat hanya diperlukan tiga baris kode untuk menjalankan EasyOCR. baris pertama adalah untuk meng-*import* library EasyOCR. Baris kedua hanya perlu dijalankan sekali untuk meng-*load* model dan baris terakhir adalah untuk menjalankan model tersebut. Hasil yang didapatkan berupa kotak pembatas, tulisan, dan *confidence level* yang disusun dalam bentuk list (lihat Gambar 2.3). Untuk kotak pembatas, diberikan sejumlah koordinat yang disusun dalam format [x,y] dan menunjukkan posisi sudut kiri bawah, kanan bawah, kanan atas, dan kiri atas kotak pembatas secara berurutan. Di Gambar 2.3, koordinat kotak pembatas adalah [[58, 372], [128, 372], [128, 428], [58, 428]], tulisan yang dideteksi adalah ”NO”, dan *confidence level*-nya adalah 0,94377.

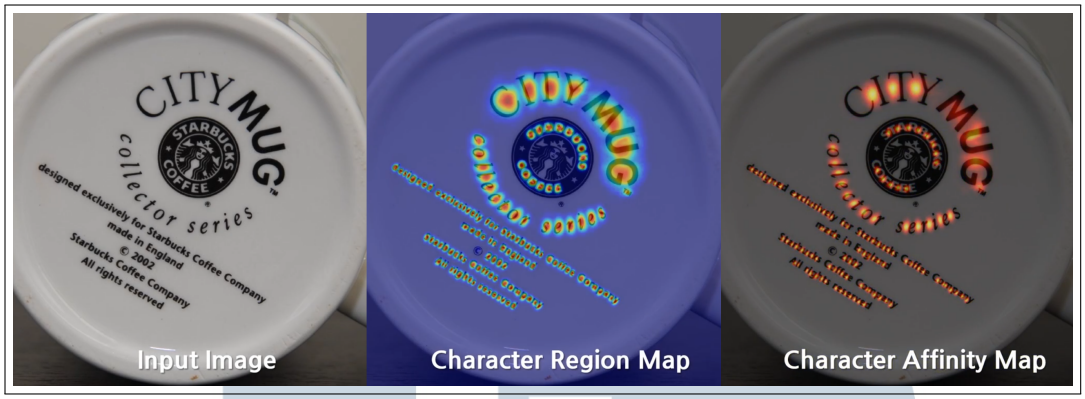
```
[([[58, 372], [128, 372], [128, 428], [58, 428]], 'NO', 0.9437757287087349)]
```

Gambar 2.3. Hasil EasyOCR

Untuk tidak menampilkan informasi seperti koordinat kotak pembatas dan *confidence level*, dapat diberikan parameter ”detail=0” di baris ketiga kode Gambar 2.2.

2.1.1 CRAFT

Character Region Awareness for Text Detection (CRAFT) adalah sebuah algoritma untuk mendeteksi tulisan dalam sebuah gambar. CRAFT memprediksi dua jenis penilaian untuk setiap huruf, yaitu Region Score dan Affinity Score [10]. Region Score adalah nilai daerah sebuah huruf, ini melokalisasi karakternya. Affinity Score adalah nilai sebuah daerah yang menghubungkan satu huruf dengan huruf berikutnya. CRAFT menghasilkan dua map sebagai output, yaitu Region Map dan Affinity Map (lihat Gambar 2.4). Setelah kedua map dihasilkan, CRAFT akan membuat map baru yang menggabungkan Region dan Affinity map yang menyoroti kata yang dideteksi seperti di Gambar 2.5.

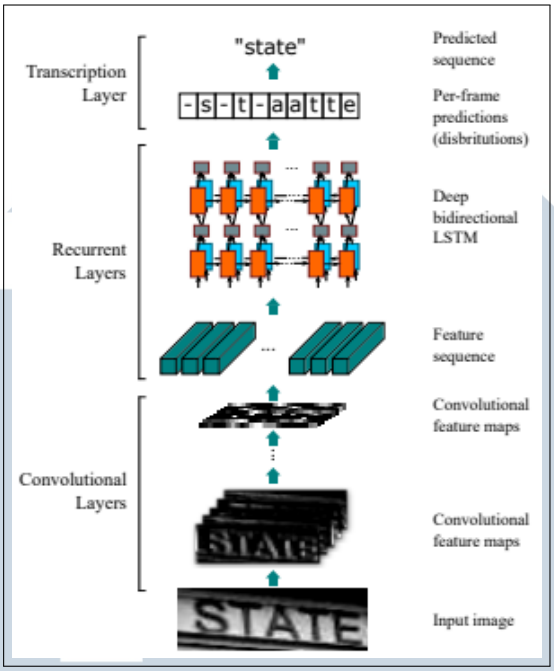


Gambar 2.4. Hasil output map dari CRAFT [11]

Gambar 2.5. Hasil deteksi tulisan dengan CRAFT [11]

2.1.2 CRNN

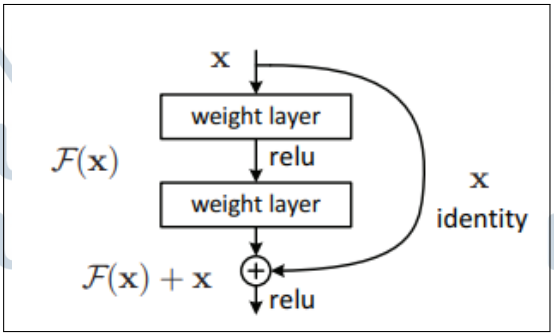
Convolutional Recurrent Neural Networks (CRNN) adalah model neural network yang merupakan gabungan dari dua *neural network*, yaitu Deep Convolutional Neural Network (DCNN) dan Recurrent Neural Network (RNN). CRNN terdiri dari tiga lapisan (lihat Gambar 2.6), yaitu Convolutional, Recurrent, dan Transcription. Di lapisan Convolutional, urutan fitur secara otomatis diekstrak dari setiap gambar input. Setelah lapisan Convolutional, terdapat lapisan Recurrent yang melakukan prediksi untuk setiap *frame* dari urutan fitur yang didapat dari lapisan Convolutional. Lapisan Transcription akan menerjemahkan prediksi yang dilaksanakan lapisan Recurrent menjadi urutan label yang akan dihasilkan sebagai output [12].



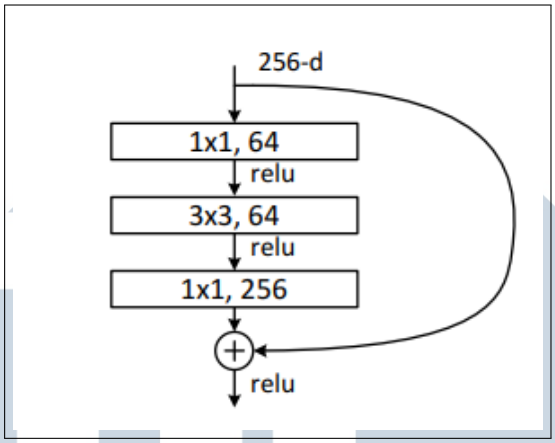
Gambar 2.6. Arsitektur CRNN [12]

2.1.3 ResNet

Residual Network (ResNet) adalah sebuah deep residual learning framework yang dirancang untuk menangkal masalah degradasi akurasi yang terjadi pada deep convolutional neural networks dengan jumlah lapisan yang banyak [13]. Degradasi akurasi adalah masalah yang muncul di deep networks yang memiliki banyak lapisan dimana akurasi pelatihan menurun dengan bertambahnya lapisan jaringan. ResNet menangkal masalah ini dengan mengenalkan "shortcut connections" yang melaksanakan identity mapping dimana hasilnya akan ditambahkan dengan hasil dari lapisan yang ditumpuk (lihat Gambar 2.7).



Gambar 2.7. Rancangan blok residual learning [13]



Gambar 2.8. Blok ResNet dengan 3 lapisan [13]

Rumus yang digunakan penelitian [13] untuk mendefinisikan perancangan blok residual learning adalah

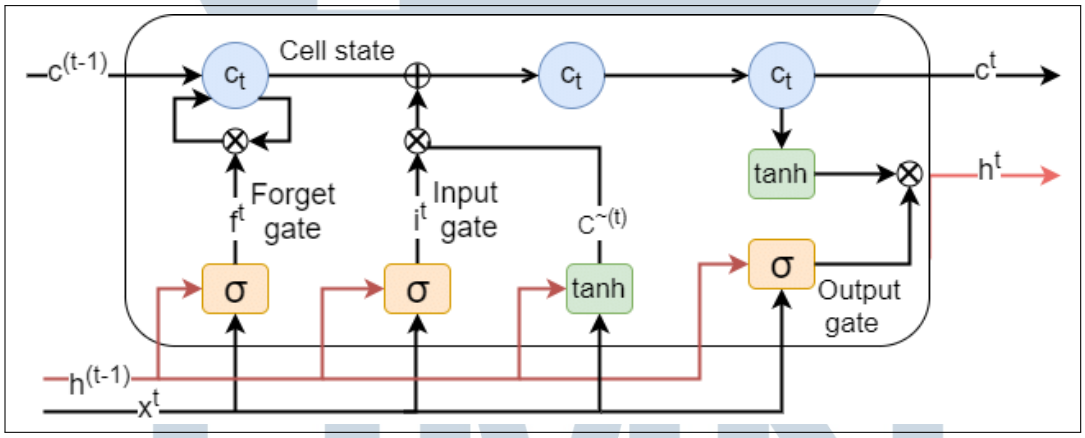
$$y = F(x, \{W_i\}) + x \tag{2.1}$$

Dimana x dan y adalah vektor input dan output dari lapisan. Fungsi $F(x, \{W_i\})$ mewakili *residual mapping* yang akan dipelajari. Untuk contoh di Gambar 2.7 yang memiliki dua lapisan, $F = W_2\sigma(W_1x)$ dimana σ menyatakan *rectified linear units* (ReLu). Untuk arsitektur yang memiliki 50/101/152 (ResNet50, ResNet101, ResNet152) lapisan dapat menggunakan blok ResNet yang terdiri atas tiga lapisan (lihat Gambar 2.8) untuk mengurangi waktu pelatihan [13]. Rumus yang tertulis mungkin tidak sama persis dengan rumus yang digunakan library EasyOCR, tetapi rumus tersebut bertujuan untuk memberikan gambaran terhadap cara kerja ResNet secara umum.

2.1.4 LSTM

Long-Short Term Memory (LSTM) adalah turunan RNN yang diperkenalkan sebagai solusi untuk sinyal error yang meledak (exploding) atau menghilang (vanishing) yang dikarenakan oleh beban yang dihitung berulang kali sehingga nilai dari beban tersebut menjadi lebih kecil (vanishing) atau lebih besar (exploding) [14]. LSTM menggunakan algoritma berbasis gradien untuk arsitektur yang menegakkan alur error konstan (tidak mengecil atau membesar) melalui keadaan internal unit khusus sehingga dapat menjembatani interval waktu yang besar tanpa menyebabkan masalah vanishing dan exploding gradient [15]. Sel LSTM terdiri dari empat

komponen, yaitu forget gate, input gate, cell state, dan output gate (lihat Gambar 7). Forget gate menentukan apakah suatu informasi akan dibuang dari cell state dengan melihat h_{t-1} dan x_t, dan menghasilkan angka nol atau satu untuk setiap angka di dalam cell state C_{t-1} dimana nol berarti ”hapus” dan satu berarti ”simpan”. Input gate memutuskan informasi baru apa yang akan disimpan di dalam cell state, terdapat dua bagian disini, yaitu lapisan sigmoid yang menentukan nilai mana yang akan diperbarui dan lapisan tanh yang membuat vector nilai kandidat baru, $\tilde{C}_t$, yang dapat dimasukkan ke dalam state. Cell state akan memperbarui nilai C_{t-1} dengan nilai baru, $\tilde{C}_t$, dan disimpan di C_t. Output gate akan mengeluarkan informasi berdasarkan cell state, pertama dijalankan lapisan sigmoid yang menentukan bagian dari cell state yang akan diteruskan, hasilnya akan dikalikan dengan tanh yang berisi cell state supaya hanya perlu mengeluarkan bagian yang diperlukan [16]. Rumus untuk setiap bagian adalah sebagai berikut.



Gambar 2.9. Susunan sel LSTM [17]

2.1.5 CTC

Connectionist Temporal Classification (CTC) adalah sebuah neural network output yang digunakan untuk menangani masalah urutan seperti tulisan tangan dan pengenalan suara dimana panjang/durasi waktunya bervariasi. CTC dapat menyelesaikan masalah dimana sebuah huruf memakan lebih dari satu time-step di dalam gambar dengan cara menggabungkan semua huruf yang berulang menjadi satu karakter. Untuk kata yang memiliki huruf yang berulang seperti “see” dimana dua huruf ”e” ditempatkan bersampingan, CRNN dilatih untuk mengeluarkan tulisan yang encoded dan menghitung kemungkinan huruf di setiap time-step [18]. Tugas CTC di EasyOCR adalah untuk men-decode hasil output LSTM dan disusun men-

jadi sebuah kata yang memiliki probabilitas benar tertinggi.

2.2 Character Error Rate

Terdapat dua metode untuk mengevaluasi akurasi model OCR, yaitu Character Error Rate (CER) dan Word Error Rate (WER). CER akan mnghitung kesalahan yang terjadi dalam tingkat huruf dimana WER akan menghitung dalam tingkat kata. CER akan menghasilkan nilai yang lebih akurat karena WER akan menandai sebuah kata sebagai salah waluapun hanya satu huruf dalam kata tersebut yang salah. Untuk menghitung akurasi sebuah OCR terdapat 3 jenis kesalahan yang perlu dipertimbangkan yaitu *Substitution*, *Deletion*, dan *Insertion*. *Substitution* adalah kesalahan dimana sebuah huruf diganti menjadi huruf yang berbeda. *Deletion* adalah ketika sebuah huruf tidak dideteksi OCR sehingga jumlah huruf dalam suatu kata berkurang. *Insertion* adalah ketika sebuah huruf ditambah ke dalam suatu kata walaupun awalnya tidak ada huruf tersebut. Rumus untuk menghitung nilai CER adalah sebagai berikut.

$$CER = \frac{S+D+I}{N} \tag{2.2}$$

Rumus tersebut didasarkan pada konsep *Levenshtein distance* dimana kita menghitung jumlah operasi tingkat huruf untuk mengubah tulisan *ground-truth* (tulisan asli) menjadi output yang dihasilkan model OCR. Keterangan istilah yang terdapat di Persamaan 2.2 adalah S sebagai *Substitution*, D sebagai *Deletion*, I sebagai *Insertion*, dan N sebagai jumlah huruf. Persamaan 2.2 dapat dinormalisasi untuk mencegah nilai yang didapat melebihi 100 dengan mengubah N menjadi $S+D+I+C$ dimana C adalah jumlah huruf yang benar [19]. Nilai yang didapat dari Persamaan 2.2 mewakilkan persentase karakter yang salah diprediksi, lebih rendah nilai yang dihasilkan (nol berarti 100% benar), lebih bagus performa model OCR.

2.3 LibreTranslate

LibreTranslate adalah sebuah *Machine Translation* API yang *open-source*. Mesin terjemahan yang digunakan LibreTranslate adalah *open-source* library yang bernama Argos Translate [20]. Website yang dirancang akan menggunakan LibreTranslate untuk fitur penerjemahannya. Dikarenakan fokus utama penelitian ini

adalah OCR, tidak akan dilakukan evaluasi terhadap LibreTranslate tetapi library ini dipilih sebagai nomor empat dalam ranking yang ditentukan oleh penelitian [21].