BAB II

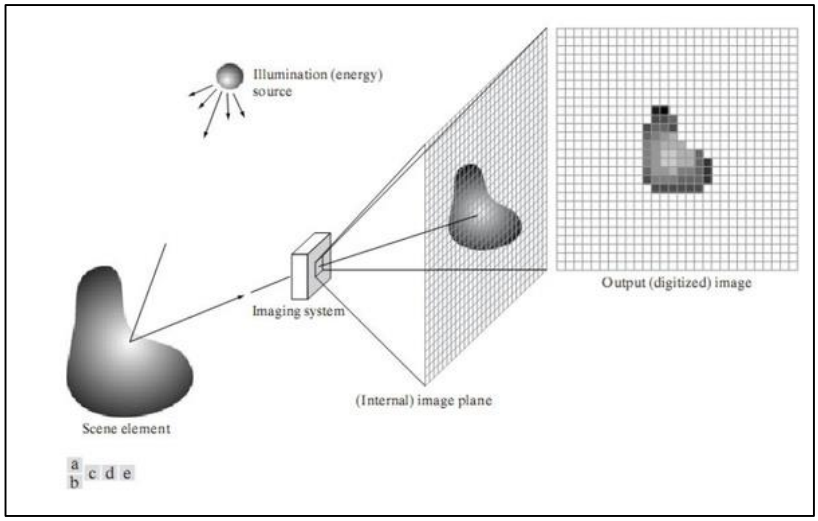
LANDASAN TEORI

2.1 Optical Character Recognition (OCR)

OCR adalah suatu solusi yang efektif untuk proses konversi dokumen cetak ke dokumen digital. OCR memiliki beberapa tahapan seperti image acquisition, preprocessing, feature extraction dan recognition (Hartanto, 2015). Berikut adalah tahapan OCR.

1. Image Acquisition

Image acquisition merupakan proses untuk mendapatkan representasi digital dari suatu *scene* seperti pada Gambar 2.1. Representasi tersebut dikenal sebagai gambar dan elemennya disebut piksel. Pada penelitian ini, *input* berupa *text image* yang didapatkan dari *dataset*.



Gambar 2.1. Tahap *Image Acquisition* (Dip, 2018)

2. Preprocessing

Preprocessing merupakan tahap penting untuk meningkatkan kualitas *image acquisition* agar dapat memberikan hasil yang lebih akurat dan membuat

gambar kita dapat diterima oleh model arsitektur. Tahap pada *preprocessing* adalah gambar diubah menjadi *gray-scale image,* membuat gambar menjadi *size* (256, 32), *expand* dimensi gambar menjadi (256, 32, 1) agar menjadi *compatible* ke dalam *input shape* pada arsitektur, dan normalisasi nilai *pixels* gambar yang dibagi dengan 255.

3. Feature Extraction

Feature extraction untuk *image processing* dan *computer vision* merupakan panduan penting untuk penerapan pemrosesan gambar dan teknik visi komputer (Nixon, 2019). *Texture analysis* digunakan dalam berbagai bidang dan aplikasi, dari *texture classification, segmentation,* dan *pattern recognition* yang merupakan *image processing* sangatlah penting untuk melakukan *extract* dari *raw images* (Humeau-Heurtier A., 2019).

4. Character Recognition

Setelah *image* dari *input* telah melewati tahap *preprocessing*, *feature extraction*, maka *image* akan diproses dalam tahap pengenalan karakter yang *output* berupa *printed text.*

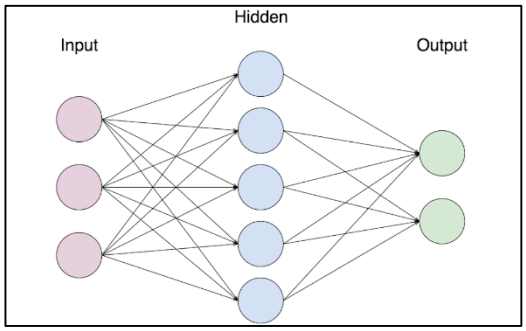
2.2 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) merupakan sebuah teknik atau pendekatan pengolahan informasi yang terinspirasi oleh cara kerja sistem saraf biologis, khususnya pada sel otak manusia dalam memproses informasi. ANN terdiri dari sejumlah besar elemen neuron/pemrosesan informasi yang saling terhubung dan bekerja sama untuk menyelesaikan masalah tertentu (Samuel, 2017).

Cara kerja ANN adalah *supervised learning*. ANN dapat digunakan untuk memperoleh pengetahuan dari data yang rumit atau tidak tepat, serta juga dapat

digunakan untuk mengekstrak pola dan mendeteksi *trend* yang terlalu kompleks bagi manusia maupun teknik komputer (Harya Damar Widiputra, 2016). ANN merupakan metode yang cocok untuk OCR karena dapat mendapatkan hasil akurasi yang baik dan ANN juga bisa mengatasi *fuzzy data* (Kurt, 2008).

ANN memiliki *node* yang memiliki bobot, *input layer, hidden layer,* dan *output layer* seperti pada Gambar 2.2. *Input layer* adalah *node* yang menerima *input* dari dunia luar. *Input* yang masuk merupakan penggambaran dari suatu masalah. *Hidden layer* adalah *node* yang tidak secara langsung dapat diamati. *Output layer* adalah *node* yang merupakan *output* ANN terhadap suatu permasalahan.



Gambar 2.2. Contoh gambar *Artificial Neural Network* (Ocktavia, 2018)

2.3 Convolutional Recurrent Neural Network (CRNN)

Convolutional Recurrent Neural Network (CRNN) merupakan penggabungan dari *Convolutional Neural Network* (CNN) dam *Recurrent Neural Network* (RNN). CRNN memiliki tiga *layer* seperti pada Gambar 2.3. (Baoguang, 2015). CRNN memiliki tiga *layer* yaitu:

1. Convolutional Layers

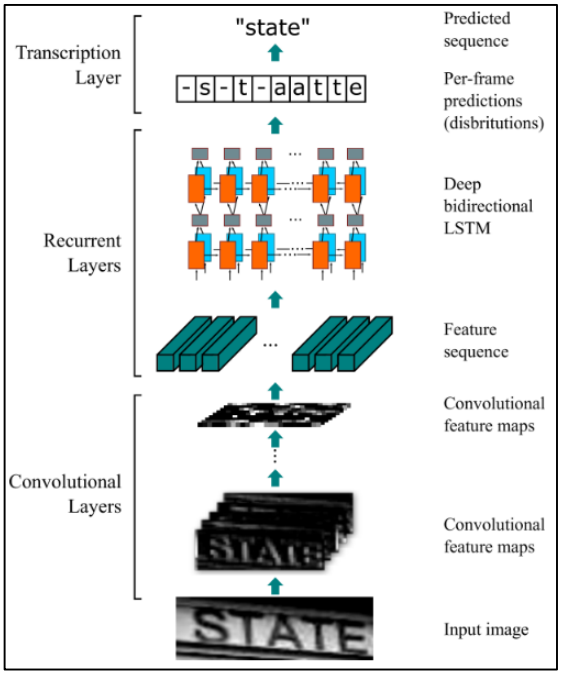
Convolutional layers diambil dari CNN untuk melakukan *feature extraction* secara otomatis pada setiap foto. *Max pooling* juga dilakukan di dalam arsitektur model pada tahap ini.

2. Recurrent Layers

Recurrent layers merupakan tahap membagi fitur menjadi ukuran tertentu dan dimasukkan ke dalam *input Long Short Term Memory* (LSTM). Tahap ini melakukan prediksi distribusi *label* untuk setiap *frame*.

3. Transcription Layers

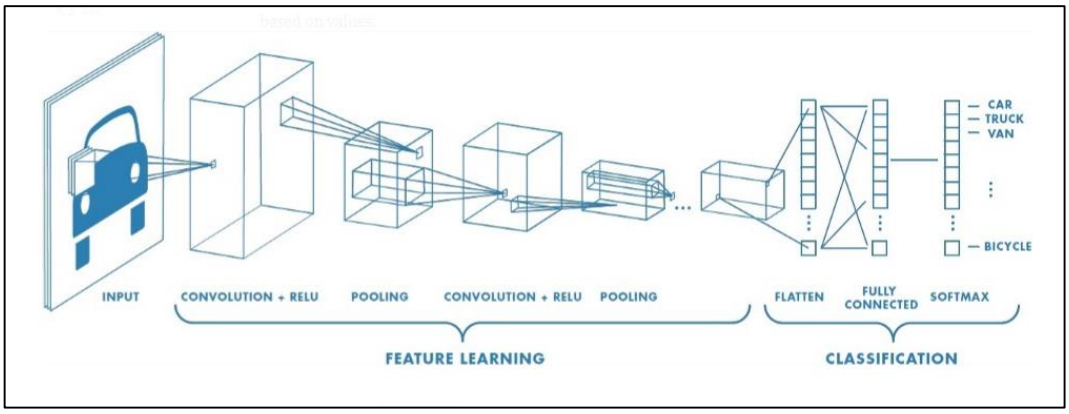
Transciption layers merupakan tahap menerjemahkan prediksi per-*frame* ke dalam *final label sequence* menggunakan CTC.



Gambar 2.3. Tahapan CRNN (Baoguang, 2015)

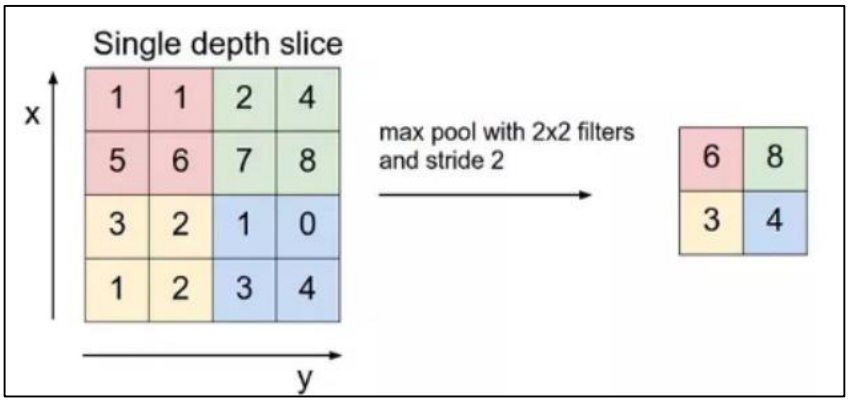
2.4 Convolutional Neural Network (CNN)

Secara teknis, *deep learning* CNN digunakan untuk *training* dan *testing*, yang setiap gambar *input* akan melewati serangkaian *convolution layer* dengan *filter* (*Kernels*), *Pooling*, *Fully Connected layers* (FC) dan menerapkan fungsi *Softmax* untuk mengklasifikasikan objek dengan nilai probabilitas antara 0 dan 1 seperti pada Gambar 2.4.



Gambar 2.4. Contoh CNN (Saha, 2018)

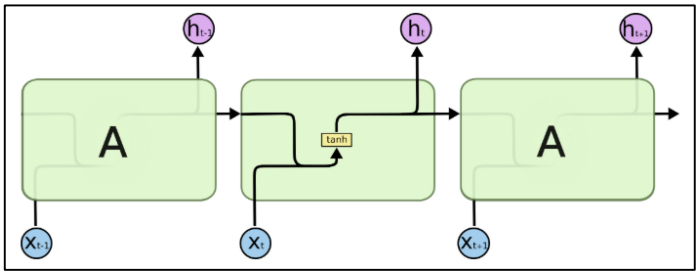
Pada penelitian ini, *convolution layer* dipakai karena keunggulannya yaitu dapat melakukan *feature extraction* terhadap foto. *Convolution layer* terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (*pixels*). *Pooling layer* dipakai untuk mengurangi jumlah parameter ketika gambar terlalu besar dengan menggunakan *max pooling* seperti pada Gambar 2.5. *Max pooling* mengambil elemen terbesar dari *feature map* yang diperbaiki, lalu mengambil elemen terbesar yang juga bisa mengambil rata-rata *pooling,* dan menjumlah semua elemen dalam panggilan *feature map* sebagai *sum pooling*.



Gambar 2.5. Contoh *Max Pooling* (Prabhu, 2018)

2.5 Long Short Term Memory (LSTM)

LSTM dirancang secara eksplisit untuk menghindari masalah ketergantungan jangka panjang. Mengingat informasi dalam jangka waktu yang lama merupakan perilaku standar mereka, bukan sesuatu yang mereka perjuangkan untuk pelajari. LSTM-RNN unggul dalam mengerjakan tugas urutan dan dapat menyelaraskan data yang tidak teregmentasi dengan kebenaran dasar yang sesuai saat digabungkan dengan kerugian *Connectionist Temporal Classification* (CTC) (Graves, 2006). LSTM memiliki sel ingatan dan tiga *multiplicative gates,* yaitu *input (i)*, *output (o)*, dan *forget gates (f)*. Secara konseptual, sel memori menyimpan konteks masa lalu dan gerbang *input* dan *output* yang memungkinkan sel untuk menyimpan konteks untuk jangka waktu yang lama. Sementara itu, memori di dalam sel bisa dibersihkan oleh *forget gate* (Baoguang Shi, Xiang Bai and Cong Yao, 2015). Fungsi dari setiap *gate,* dapat diartikan sebagai operasi *write, reset,* dan *read,* yang masing-masing sehubung dengan *internal state cell* (*c*). Semua RNN memiliki bentuk rangkaian modul berulang *neural network.* Pada Gambar 2.6., gambar tersebut merupakan contoh struktur sederhana dari modul berulang RNN yang memiliki satu lapisan *tanh*.



Gambar 2.6. Modul *standard* berulang RNN (Olah C., 2015)

Berikut adalah tahapan pada LSTM.

1. Menentukan informasi yang akan dibuang

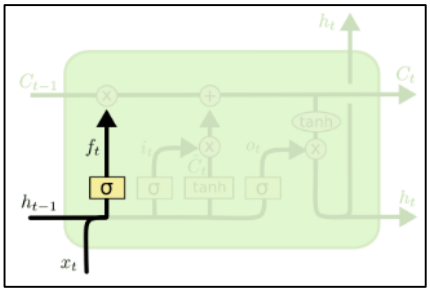
Tahap pertama LSTM merupakan tahap untuk menentukan informasi apa yang akan dibuang dari *cell state* oleh *sigmoid layer* yaitu *forget gate layer*. Gambar 2.7. merupakan struktur modul berulang RNN tahap pertama dengan rumus:

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right) \quad \text{... (2.1)}$$

$$\sigma = \frac{1}{1+\exp(-x)} \quad \text{... (2.2)}$$

dimana f_t merupakan *forget gate layer*, σ merupakan fungsi logistik *sigmoid* untuk menghitung probabilitas untuk semua *gates*, W dan b merupakan *weight* dan *bias* untuk *transform* dua vektor menjadi satu *common space.*

σ digunakan untuk memprediksi probabilitas sebagai *output* ketika *range* dari probabilitas antara 0 sampai 1. Jika *output* berupa 0, maka menandakan informasi tersebut akan dibuang. Jika *output* berupa 1, maka menandakan informasi tersebut akan di simpan.



Gambar 2.7. Struktur tahap pertama LSTM (Olah C., 2015)

2. Menentukan dalam menyimpan informasi baru ke dalam *cell state*

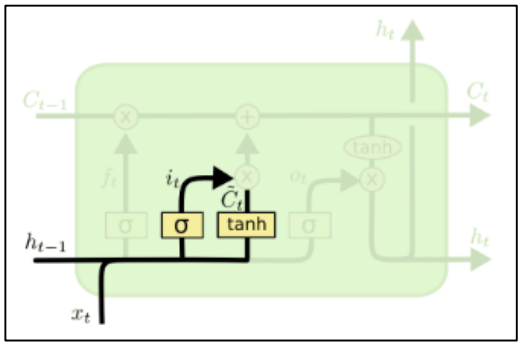
Tahap kedua LSTM merupakan tahap untuk menentukan informasi baru apa yang akan disimpan ke dalam *cell state.* Gambar 2.8. merupakan struktur modul berulang RNN tahap kedua dengan rumus:

$$i_t = \sigma\left(W_i . [h_{t-1}, x_t] + b_i\right) \quad \text{... (2.3)}$$

$$\tilde{C}_t = tanh\left(W_c . [h_{t-1}, x_t] + b_c\right) \quad \text{... (2.4)}$$

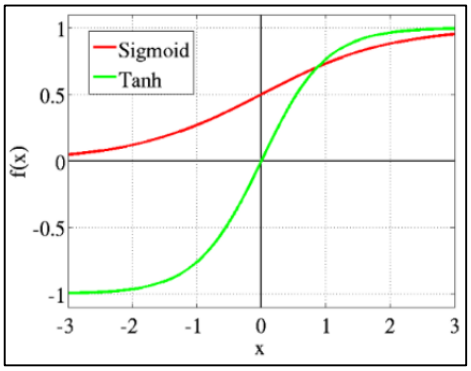
$$tanh = \frac{e^{2x}-1}{e^{2x}+1} \quad \text{... (2.5)}$$

dimana tahap ini memiliki 2 bagian, yaitu i_t *yang* merupakan *input gate layer* menentukan *value* mana yang akan di *update*, dan *tanh layer* akan menciptakan $\tilde{C}_t$ yang merupakan vektor nilai kandidat baru yang bisa ditambahkan ke dalam *state.* Lalu kedua bagian tersebut akan di gabung untuk menciptakan sebuah *update* ke dalam *state.*



Gambar 2.8. Struktur tahap kedua LSTM (Olah C., 2015)

Tanh berfungsi seperti *sigmoid* (σ), dengan *range* antara -1 sampai -1. Perbedaan *shape* antara *sigmoid* dan *tanh* ditampilkan dalam Gambar 2.9.



Gambar 2.9. Perbedaan *shape tanh* dan *sigmoid* (Olah C., 2015)

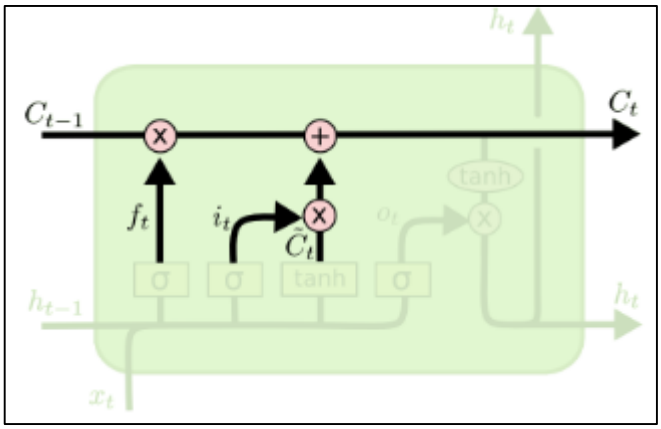
3. Memperbarui *cell state* yang lama

Tahap ketiga LSTM merupakan tahap memperbarui *cell state* yang lama (C_{t-1}) menjadi *cell state* yang baru (C_t) dengan mengalikan *cell state* yang lama

dengan f_t, yaitu melupakan yang sudah ditentukan untuk dilupakan sebelumnya. Lalu tambahkan dengan $i_t * \tilde{C}_t$. Ini adalah nilai kandidat baru, yang diskalakan dengan seberapa banyak kita menentukan untuk memperbarui setiap nilai *state*. Gambar 2.10. merupakan struktur modul berulang RNN tahap ketiga dengan rumus:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{... (2.6)}$$

dimana C_t merupakan *cell state* baru, f_t merupakan *forget gate layer*, i_t merupakan *input gate layer,* dan $\tilde{C}_t$ yang merupakan nilai kandidat baru.



Gambar 2.10. Struktur tahap ketiga LSTM (Olah C., 2015)

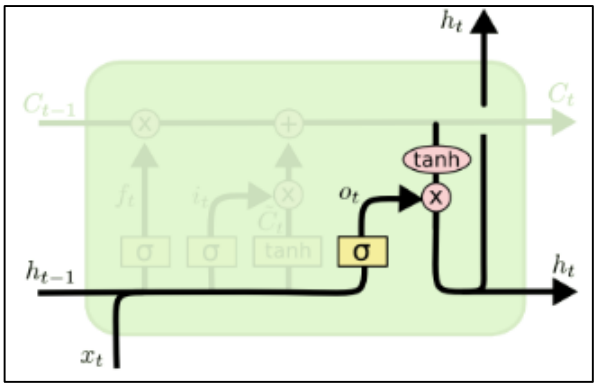
4. Menentukan apa yang akan menjadi *output*

Tahap keempat LSTM merupakan tahap untuk menentukan apa yang akan menjadi *output* yang didasarkan oleh *state cell,* tetapi menjadi versi yang di-*filter*. Pertama, *sigmoid layer* dijalankan yang akan menentukan bagian sel apa yang ingin dihasilkan. Lalu, menempatkan *state cell* melalui *tanh* untuk mendorong nilai menjadi antara -1 dan 1, setelah itu dikalikan dengan *output sigmoid gate* sehingga dapat menampilkan *output* dari bagian yang diputuskan. Gambar 2.11. merupakan struktur modul berulang RNN tahap keempat dengan rumus:

$$o_t = \sigma(\, W_o\, [h_{t-1}\, , x_t] + b_o \qquad \text{... (2.7)}$$

$$h_t = \; o_t * \tanh(C_t) \qquad \text{... (2.8)}$$

dimana o_t merupakan output layer dan h_t merupakan *hidden unit.*



Gambar 2.11. Struktur tahap keempat LSTM (Olah C., 2015)

Menyimpan informasi melalui beberapa langkah waktu memungkinkan LSTM untuk mempelajari hubungan temporal dalam data. Dalam hal OCR, berarti secara inheren membangun *model* bahasa statistik berdasarkan hubungan *temporal* karakter dan kata-kata.

2.6 Connectionist Temporal Classification (CTC)

CTC Loss memungkinkan RNN untuk dilatih dengan data yang tidak tersegmentasi (Graves et al., 2006). Dalam CTC, perlu menyelaraskan setiap karakter dengan urutan dalam *network output*, sambil menjaga *temporal order* tetap konsisten. Karena RNN beroperasi pada *frame level*, panjang urutan *input x* dan urutan *output y* harus sama. Hal ini dapat dipahami dengan memperpanjang sesuai dengan panjang *input sequence* dengan menambahkan karakter "*blank*" yang mengacu kepada tidak ada *output*, atau menduplikasi karakter yang sama beberapa kali yang ekuivalen dengan input berturut-turut yang mengacu kepada karakter yang sama dalam transkripsi. Contoh "…aa.b." dan "-aabb---", dengan "*blank*" menggunakan *label* "." atau "-", yang merupakan dua kemungkinan ekstensi dalam

transkripsi “ab” dengan panjang urutan 8. Karena itu, kemungkinan *labeling (l)* merupakan penjumlahan probabilitas dari semua kemungkinan ekstensi *T(l)* ke panjang *network output,* yaitu dengan rumus:

$$p(l|x) = \sum_{\pi \epsilon T(l)} p(\pi|x) \quad \text{... (2.9)}$$

Lalu menggunakan algoritma *forward-backward* untuk menghitung *p(l|x),* yaitu dengan rumus:

$$p(l|x) = \sum_{n} \alpha_t(l_n)\beta_t(l_n) \quad \text{... (2.10)}$$

dengan $\alpha_t(l_n)$ dan $\beta_t(l_n)$ yang merupakan variabel *forward* dan *backward* untuk *n*-th simbol l_n pada label l dalam *network output t.* Kemudian LSTM-RNN dapat dilatih dengan memaksimalkan kesemua kemungkinan ekstensi.

2.7 Character Error Rate

Hasil dari OCR sering mengandung sejumlah kesalah seperti kata yang salah eja atau mengandung karakter yang salah. *Character Error Rate* (CER) merupakan salah satu tingkat dalam menghitung *error rate.* CER dihitung dengan melihat jumlah minimum operasi yang dibutuhkan untuk mengubah teks sebenarkan ke teks yang menjadi *output* dalam OCR, jumlah ini disebut *Levenshtein Distance.* Rumus CER adalah sebagai berikut.

$$CER = \frac{(i+s+d)}{n} \quad \text{... (2.12)}$$

dimana n merupakan jumlah karakter, i adalah *insertion* atau jumlah penambahan karakter, s adalah *substituted* atau jumlah karakter yang sama dengan karakter yang berbeda, dan d adalah *deletion* atau jumlah karakter berkurang.

2.8 Akurasi

Akurasi pada *string* adalah mengukur jarak kalimat uji dan kalimat terbaik. Perbandingan *string* yang menjadi dasar akurasi kata ditentukan oleh jumlah minimal *substitution, deletion* dan *insertion* kata-kata yang diperlukan untuk mengubah kalimat terbaik menjadi kalimat uji yang biasa dikenal sebagai *Levenshtein Distance* (Zanten, 1999). Perhitungan akurasi berhubungan dengan CER. Akurasi didefinisikan sebagai:

$$Accuracy = 1 - \frac{ld}{n} \quad \text{... (2.13)}$$

dimana *n* merupakan panjang kata sebenarnya dan *ld* merupakan minimal yang didapatkan menggunakan *Levenshtein Distance.*