BAB 2
TINJAUAN PUSTAKA

2.1 Ekstraksi Informasi

Ekstraksi informasi adalah suatu proses untuk mengubah informasi tidak terstruktur yang terdapat dalam teks ke dalam data terstruktur[7]. Ekstraksi informasi dilakukan dengan cara menyeleksi teks pada dokumen dengan ketentuan-ketentuan tertentu. Ekstraksi informasi berperan sebagai sistem yang mengenali data yang tidak terstruktur yang memiliki informasi yang belum dikategorikan dan belum memiliki arti yang spesifik[8].

Ekstraksi informasi merupakan salah satu penerapan *text mining*. *Text mining* adalah proses menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisis keterhubungan antar dokumen tersebut. Text mining mengekstrak informasi dari dalam teks dokumen yang tidak terstruktur.

2.2 *Text Preprocessing*

Text Preprocessing merupakan suatu proses untuk menyiapkan data untuk diolah pada proses utama[8]. Pada penelitian ini tahapan preprocessing meliputi converting atau OCR, *cleansing*, *filtering* dan *tokenizing*, segmentasi baris, penyesuaian baris, pelabelan baris, serta pembobotan kata atau word representation yang akan dijadikan parameter untuk pemrosesan klasifikasi menggunakan algoritma

2.2.1 *Optical Character Recognation*

OCR adalah sebuah proses yang mengubah teks dalam format berkas citra atau gambar ke dalam format teks yang bisa dibaca dan disunting oleh aplikasi komputer atau perangkat lunak. Proses *OCR* adalah proses pengubahan format data menjadi format data yang diperlukan untuk menuju tahap proses preprocessing. Banyak cara yang dapat dilakukan dengan konversi, seperti menggunakan library, menggunakan software ataupun memakai kode API untuk mengakses pada situs

yang menyediakan layanan konversi. Tools yang digunakan pada penelitian ini berupa library menggunakan tesseract dari google untuk mengkonvert data *PDF* menjadi *CSV* untuk training dan *TXT* untuk testing.

2.2.2 Cleansing

Tahap *Cleansing* diperlukan untuk menghapus simbol-simbol yang tidak diperlukan. Pada penelitian ini simbol-simbol yang diperlukan hanyalah garis miring "/", koma ",", dan titik ".". Selain 3 simbol tersebut, maka pada proses ini akan dihilangkan atau dihapuskan agar tidak berpengaruh pada proses lain.

2.2.3 *Case Folding*

Tahap *Case Folding* ini diperulukan untuk mengubah simbol variabel menjadi simbol terminal. Semua huruf kapital diubah menjadi huruf kecil. *Case folding* ini perlu dilakukan untuk meragamkan kata dalam mempermudah dalam proses pengubahan kata menjadi vektor atau biasa disebut *word representation*.

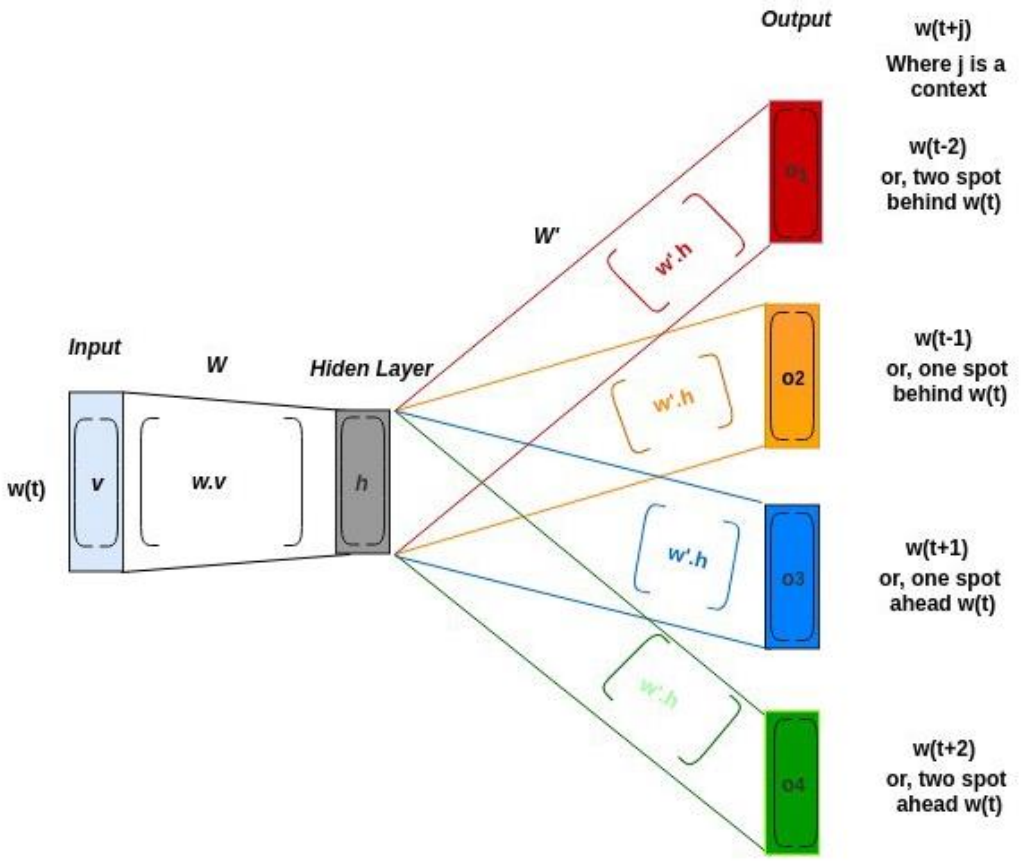
2.2.4 *Tokenizing*

Tokenizing adalah proses pemisahan kata-kata dalam barisan atau susunan kalimat. Pemisah antar kata akan dihapus atau dihilangkan, pemisah kata itu berupa spasi. Hasil tokenizing ini berupa array kata yang kemudian akan digunakan pada tahap word representation menggunakan word2vec untuk mengubah kata-kata menjadi bentuk vektor.

2.2.5 *Word Representation*

Untuk merepresentasikan pada sistem ini menggunakan metode *word2vec* yang dimana setiap kata direpresentasikan dalam konteks sebagai vektor dengan N demensi. Dalam mempresentasikan suatu kata, Word2Vec mengimplementasi neural network untuk menghitung *contextual and semantic similarity* (kesamaan kontekstual dan semantik) dari setiap kata (inputan) yang berbentuk *one-hot encoded vector*[9]. Pada penelitian ini, model yang digunakan adalah model skip gram neural network.

Skip-Gram merupakan model yang diperkenalkan oleh Mikolov, dkk[10]. Ilustrasi *feeding forward* Skip-Gram dengan *windows* (jarak antara kata-kata konteks dengan posisi kata yang menjadi inputan) = 2 dapat dilihat pada **Gambar 2.1 Arsitektur Skip-Gram.**



Gambar 2.1 Arsitektur Skip-Gram

Secara Arsitektur Skip-Gram menggunakan *current word* (sebagai input) untuk memprediksi konteks (sebagai target) disekitarnya, dimana Skip-Gram akan mempelajari distribusi probabilitas dari kata-kata didalam konteks dengan *windows* yang telah di tentukan. Misal konteks yang digunakan saat ini adalah "*balas dendam terbaik adalah kesuksesan besar*" dengan nilai *windows* = 2.

context						**Training Sample *(input, target)***
Balas	*dendam*	*terbaik*	*adalah*	*kesuksesan*	*Besar*	*(balas, dendam)* *(balas, terbaik)*
Balas	*dendam*	*terbaik*	*adalah*	*kesuksesan*	*Besar*	*(dendam, balas)* *(dendam, terbaik)* *(dendam, adalah)*
						(terbaik, balas) *(terbaik, dendam)* *(terbaik, adalah)* *(terbaik, kesuksesan)*
						(adalah, dendam) *(adalah, terbaik)* *(adalah, kesuksesan)*

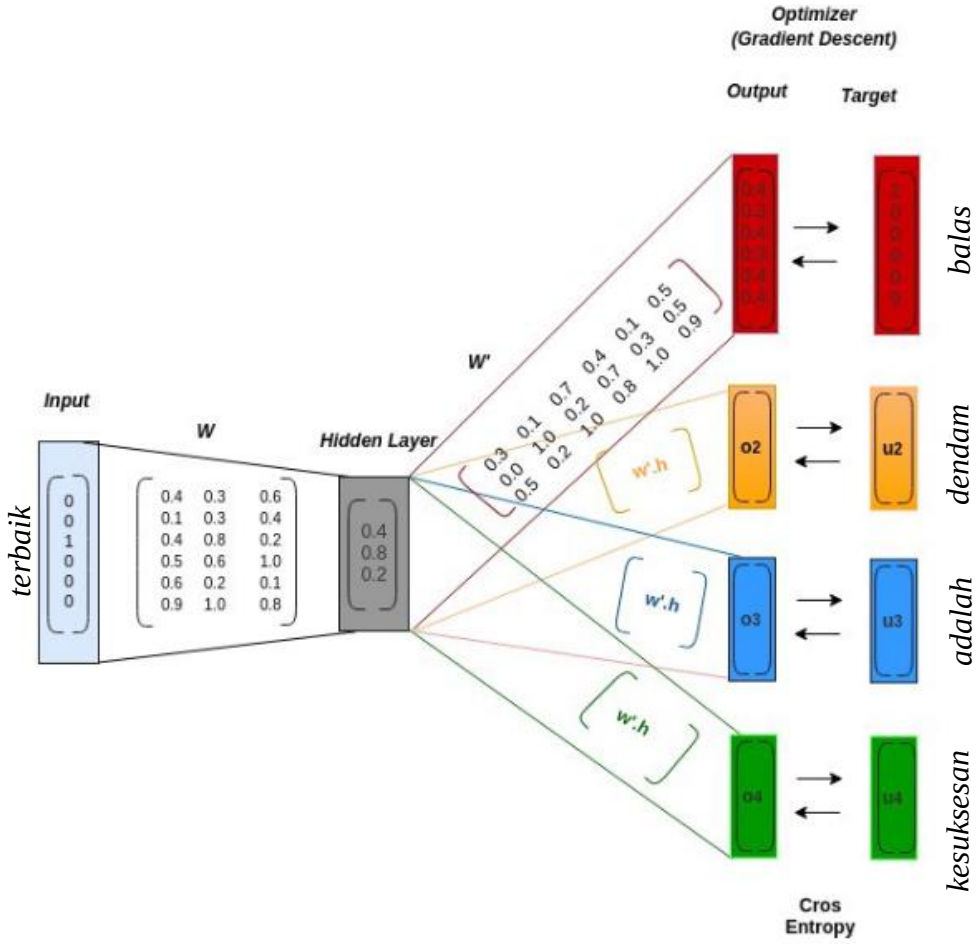
Balas	*dendam*	*Terbaik*	*adalah*	*kesuksesan*	*Besar*

Balas	*dendam*	*Terbaik*	*adalah*	*kesuksesan*	*Besar*

Untuk merepresentasikan konteks kedalam arsitektur Skip-Gram, maka kita harus merubah setiap kata menjadi *one-hot encoded vectors.*

balas = [1,0,0,0,0,0]
dendam = [0,1,0,0,0,0]
terbaik = [0,0,1,0,0,0]
adalah = [0,0,0,1,0,0]
kesuksesan = [0,0,0,0,1,0]
besar = [0,0,0,0,0,1]

Ilustrasi *forward-backward training* Skip-Gram dengan nilai random pada *weight (W* dan *W`),* dengan *w(t)* = "*terbaik*" **sebagai input**, dan *w(t - 2)* = "*balas*", *w(t - 1)* = "*dendam*", *w(t + 1)* = "*adalah*", *w(t + 2)* = "*kesuksesan*" **sebagai target** dapat dilihat pada **Gambar 2.2 Ilustrasi Forward-Backward *Training Skip-Gram***.



Gambar 2.2 Ilustrasi *Forward-Backward Training* Skip-Gram

Setelah *output* didapatkan, perlu dilakukan perhitungan nilai *error* dengan metode *cross entropy (target - output)*. Tahap setelah perhitungan nilai *error* adalah *backprogation*, dengan menghitung *gradien lost function* terhadap semua parameter yang ada dengan cara mencari *partial derivative*. Pada tahap *backprogation* terjadi proses update parameter, yaitu mengurangi atau menambahkan *weight* (*W* dan *W`*) lama dengan nilai gradient yang sudah didapatkan (Gradient Descent) hingga tercapai nilai *minimum error* pada *cross entropy*.

2.3 *Convolutional Neural Network*

Deep learning adalah bagian dari *Artificial Neural Network* yang pertama kali diperkenalkan oleh Rina Dechter pada tahun 1986. *Convolutional Neural Network* (CNN) adalah salah satu bentuk *feed forward artificial neural network* yang sering

digunakan untuk model dengan input berupa data gambar atau video[11]. *CNN* termasuk di dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. CNN terinspirasi oleh proses-proses biologi dimana pola konektivitas antar *neuron* menyerupai organisasi *visual cortex* pada binatang[12].

Dalam penelitian ini akan diterapkan algoritma *Convolutional Neural Network* (CNN) untuk melakukan ekstraksi informasi. CNN adalah kategori *Neural Network* yang menggunakan multilayer variasi persepsi yang dirancang untuk preprocessing minimal[2]. CNN biasanya diterapkan untuk pengenalan gambar, namun banyak studi yang mempelajari CNN untuk pengenalan text. Sebelumnya ada yang sudah melakukan penelitian dengan menerapkan CNN pada bidang *Natural Language Processing*(NLP). Penerapan CNN pada NLP, masukan yang digunakan adalah kalimat yang berada di dalam dokumen yang direpresentasikan sebagai matriks[3] Representasi kata diubah dari kalimat menjadi vektor bilangan *real* dengan menggunakan metode *word2vec.* Vektor inilah yang akan dijadikan sebagai masukan untuk pemproses teks dengan menggunakan CNN.

Nilai input matriks yang telah di dapat akan melalui proses *convolution* pada *convolution layer.*

2.3.1 Convolution Layer

Sebelum proses Convolution dimulai, dilakukan proses pemilahan setiap dari input layer lalu kemudian di *filter* setiap input *layer* nya. *Stride* merupakan jumlah langkah pergeseran *filter*. Semakin kecil *stride* maka semakin detail informasi yang didapat, namun membutuhkan komputasi lebih berat dibanding jumlah *stride* yang lebih besar. Sedangkan *Padding* merupakan jumlah vektor yang akan ditambahkan pada setiap sisi dari matriks masukan. Biasanya *padding* berisi vektor yang nilainya 0. Tujuan dari *padding* adalah untuk mengurangi informasi yang terbuang sehingga ukuran matriks *input* dan *output* tetap sama.

Untuk menghitung jumlah filter maka digunakanlah rumus dibawah ini:

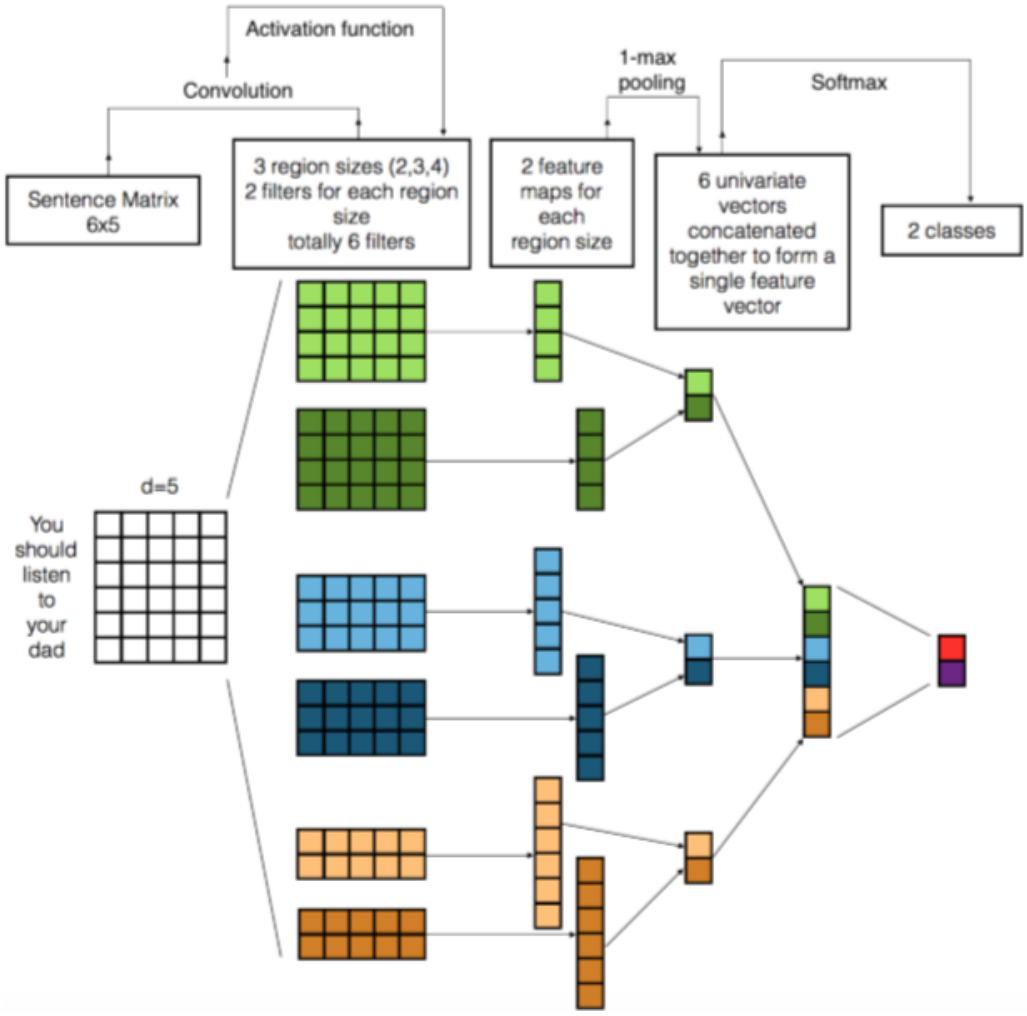
$$F = f.s \qquad \textbf{2.1}$$

F : Jumlah filter

f : filter size

s : region size

Convolution Layer pada NLP[13] dapat dilihat pada **Gambar 2.3 Convolution Layer pada NLP**.



Gambar 2.3 Convolution Layer pada NLP

(Sumber[13] : Lopez, M. M., & Kalita, J. (2017). Deep Learning applied to NLP. *arXiv preprint arXiv:1703.03091*)

Pada lapisan ini akan dilakukan operasi konvolusi dengan metode *sliding window*. Dimana *sliding window* dilakukan dengan cara mengalikan nilai *input*

layer terhadap nilai dari masing masing element tiap filter. Kemudian dihitung rata-rata dari jumlah hasil perhitungan sliding window. Hasil dari konvolusi kemudian akan masuk ke *layer feature maps*.

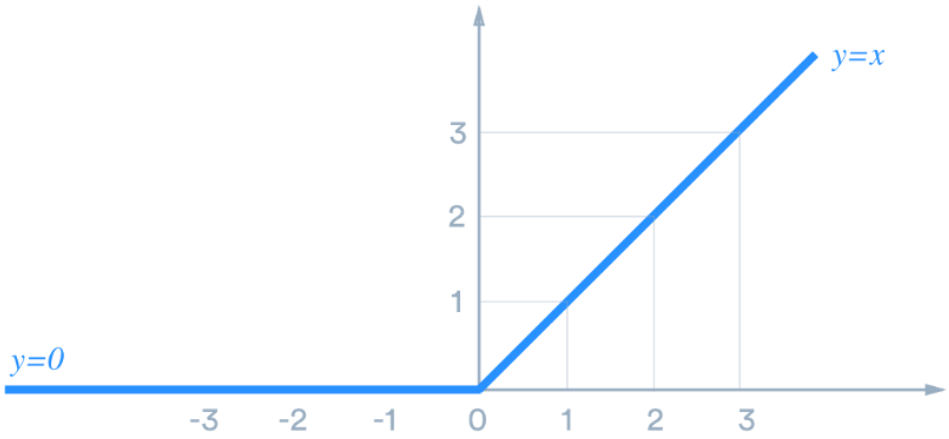
2.3.2 Activation Function dan Feature Map

Fungsi aktivasi adalah fungsi non linear yang memungkinkan sebuah JST untuk dapat mentransformasikan data input menjadi dimensi yang lebih tinggi sehingga dapat dilakukan pemotongan *hyperlane* sederhana yang memungkinkan dilakukan klasifikas[14]. Pada penelitian ini, fungsi aktivasi yang digunakan yaitu fungsi *Rectified Linear Unit* (ReLU). Fungsi ini meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang repesitif pada *convolution layer*. Fungsi ReLU mempunyai persamaan sebagai berikut.

$$f(x) = \max(0, x) \qquad \textbf{2.2}$$

Keterangan:

f(x) : fungsi ReLU
max(0,x) : fungsi maximum
x : nilai input



Gambar 2.4 Grafik Fungsi ReLu

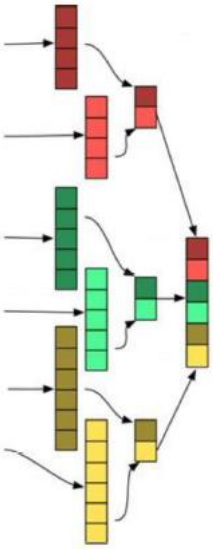
Secara sederhana, cara kerja fungsi ReLU ialah dengan melihat nilai x yang masuk padanya. Ketika nilai x kurang dari atau sama dengan 0 maka nilai x = 0 sedangkan apabila nilai x lebih dari 0 maka nilai x = x. Bila digambarkan dengan persamaan maka akan persamaanya adalah sebagai berikut.

$$\max(0, x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases} \quad \textbf{2.3}$$

2.3.3 Pooling Layer

Setelah dilakukan aktivasi fungsi dan feature maps dengan menggunakan ReLU, selanjutkan dilakukan *max pooling* dengan cara mencarsi nilai tertinggi hasil dari proses konvolusi yang telah diaktivasi dengan *ReLU* sebelumnya[15] dengan persamaan sebagai berikut :

$$c = \max\{c\} \quad \textbf{2.4}$$



Gambar 2.5 *Max Pooling Layer* dan *Flatten*

Hasil *max pooling* kemudian digabungkan dan menjadi masukan untuk *fully connected layer*.

2.3.4 Fully Connected Layer

Fully Connected Layer adalah layer yang biasanya digunakan dalam penerapan NLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya.

Perbedaan antara *Fully Connected Layer* dengan *Convolution Layer* adalah neuron pada *Convolution Layer* terhubung hanya ke daerah tertentu pada input, sedangkan *Fully Connected Layer* memiliki neuron yang secara keseluruhan terhubung. *Fully Connected Layer* berperan untuk mengklasifikasi data masukan.

Sebelum masuk ke tahap *Fully Connected Layer*, ouput dari layer sebelumnya terlebih dahulu ditransformasikan menjadi bentuk vektor satu dimensi atau biasa disebut proses flatten. Hasil dari tersebut kemudian diklasifikasikan kedalam *output layer*. Berikut bentuk persamaan dari output *layer*:

$$\text{Fully}_i = \sum_{i=1}^{length(flatten)} W_{i,j} * flatten_j + b_i \quad \textbf{2.5}$$

dimana:

Fully_i : Hasil dari perhitungan pada *fully-connected layer*

$Flatten_j$: Nilai *Flatten* dari vektor ke-j

b_i : Nilai bias yang digunakan

i : Kelas ke-i (i = 1,2,3,4,5,6,7).

2.3.5 Softmax Classifier

Softmax Classifier merupakan standar fungsi yang digunakan ketika proses klasifikasi melibatkan lebih dari dua kelas[16]. Bentuk persamaan *Softmax Classifier* adalah sebagai berikut.

$$Softmax_i = \frac{e^{Fully_i}}{(\sum_{i=1}^{kelas} e^{Fully_i})} \quad \textbf{2.6}$$

Dimana:

$Softmax_i$: Hasil dari aktifasi fungsi softmax untuk *output layer* ke-i

$Fully_i$: Hasil dari perhitungan pada fully-connected layer ke-i

kelas : jumlah kelas keluaran

$\sum_{i=1}^{kelas} e^{Fully_i}$: semua masukan untuk *output layer* sejumlah *m* buah

e : nilai 2.7182…

2.4 *Loss Function*

Loss Function merupakan fungsi untuk menghitung kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh suatu model. *Loss function* bekerja ketika model pembelajaran memberikan kesalahan yang harus diperhatikan.

Loss Function yang baik adalah fungsi yang menghasilkan error seminimal mungkin. Pada penelitian ini *loss function* yang digunakan yaitu *Cross Entrophy Function*. Adapun persamaannya adalah sebagai berikut.

$$L = - \sum_{i}^{kelas} t_i \log(Softmax_i) \quad \textbf{2.7}$$

Dimana:

L : nilai *loss function*

t_i : nilai vektor yang diharapkan / ground truth

kelas : jumlah kelas keluaran

$Softmax_i$: Hasil dari aktifasi fungsi softmax untuk *output layer* ke-i

2.5 *Backpropagation*

Backpropagation adalah salah satu algoritma supervised learning yang digunakan dalam artificial neural networks. Backpropagation mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar[16]. Adapun tahapan Backpropagation adalah sebagai berikut:

1. Turunan gradien error terhadap softmax

   Tahapan turunan gradien error terhadap softmax berdasarkan rumus yang ditulis oleh Peter Sadowski.

   $$\Delta Softmax_i = Softmax_i - t_i \quad \textbf{2.8}$$

   Keterangan:

   $\Delta Softmax_i$ = Nilai turunan dari fungsi *softmax*.

   t_i = Nilai dari target atau Ground truth, bernilai 1 apabila target adalah kelas yang dituju, dan bernilai 0 apabila nilai target bukanlah kelas yang dituju.

   i = Kelas ke-i (i = 1,2,3,4,5,6,7).

2. Turunan gradien error terhadap bobot

   Tahapan turunan gradien error terhadap bobot berdasarkan rumus berikut:

$$\Delta W_{i,j} = \Delta Softmax_i * Flatten_j \quad \textbf{2.9}$$

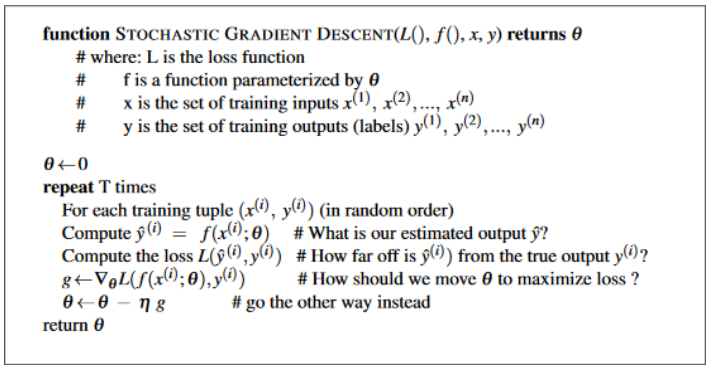
3. Turunan gradien error terhadap bias

   Tahapan turunan gradien *error* terhadap bias berdasarkan rumus berikut:

$$\Delta Bias_i = \Delta Softmax_i \quad \textbf{2.10}$$

2.6 *Stocastic Gradient Descent*

Stochastic Gradient Descent adalah sebuah algoritma untuk menemukan nilai minimum lokal dari sebuah fungsi, Algoritma dari *Stochastic Gradient Descent* dapat dilihat pada **Gambar 2.6 Algoritma *Stochastic Gradient Descent***.



function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ
    # where: L is the loss function
    # f is a function parameterized by θ
    # x is the set of training inputs $x^{(1)}$, $x^{(2)}$, ..., $x^{(n)}$
    # y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$, ..., $y^{(n)}$

$\theta \leftarrow 0$
repeat T times
  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
  Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
  Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
  $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss ?
  $\theta \leftarrow \theta - \eta\, g$ # go the other way instead
return θ

Gambar 2.6 Algoritma *Stochastic Gradient Descent*

Dimana dari algoritma tersebut, bisa juga dirumuskan nilai perbaikan bobot dan bias baru seperti pada persamaan berikut:

1. Perbaikan nilai pada bobot

$$\theta W = W - \alpha(\Delta W) \quad \textbf{2.11}$$

Keterangan:

ΔW = Nilai turunan dari bobot.

W = Nilai bobot.

α = Nilai *learning rate.*

θW = Nilai bobot baru.

2. Perbaikan nilai pada bias

$$\theta b = b - \alpha(\Delta b) \quad \textbf{2.12}$$

Keterangan:

Δb = Nilai turunan dari bias.

b = Nilai bias.

α = Nilai *learning rate.*

θb =Nilai bias baru

2.7 *Python*

Python adalah bahasa pemrograman yang interpreter dan berorientasi pada objek[17]. Python memiliki fitur-fitur seperti *module*, *exception*, *dynamic typing*, *very high level dinamic data type*, dan *class*. Python tersedia dalam banyak sistem operasi, termasuk sistem operasi Windows.

Proram Python pada Windows terdapat dua pilihan program yang dapat digunakan, yaitu Python dan IDLE. Python bekerja dalam bentuk teks, sedangan IDLE dalam bentuk grafis. Python yang bekerja dalam bentuk teks lebih unggul dalam hal kecepatan, namun IDLE memiliki lingkungan kerja yang lebih lengkap dan terintegrasi[17].

2.8 *Tesseract*

Tesseract merupakan *free engine Optical Character Recognition* (OCR) yang dirilis dibawah lisensi *Apache* dan pengembangannya disponsori oleh *Google*. *Tesseract* saat ini merupakan salah satu *engine* OCR *open source* yang paling akurat dibanding dengan *engine* yang lain. *Tesseract* dapat membaca berbagai format gambar dan mengkonversinya ke teks. Selain gambar, *Tesseract* juga dapat membaca file PDF[18].

2.9 *Confussion Matrix*

Pengujian dilakukan dengan menggunakan confussion matriks untuk mengetahui nilai akuras, presisi, dan recall algoritma CNN dalam mengekstraksi

informasi. Confusion matrix umumnya dievaluasi dengan menggunakan data dalam matriks. Parameter yang ada dalam pengujian *confussion matrix* dapat dilihat pada **Tabel 2.1 Definisi Parameter TP, FP, FN, TN**.

Tabel 2.1 Definisi Parameter TP, FP, FN, TN

True Label	*Prediction Label*	
	True Positives	*False Negatives*
	False Positives	*True Negatives*

Pada **Tabel 2.1** menunjukan *confussion matrix* untuk klasifikasi dua kelas yang terdiri dari TP (*True Positives*), FP (*False Positive*), FN (*False Negatives*) dan TN (*True Negatives*). Setelah didapat semua nilai tersebut, selanjutnya dihitung nilai akurasi.

$$\text{Accuracy} = \frac{TP\ +\ TN}{TP + FP + TN + FN} * 100\% \quad \textbf{2.13}$$

$$Precision = \frac{TP}{TP + FP} * 100\% \quad \textbf{2.14}$$

$$Recall = \frac{TP}{TP + FN} * 100\% \quad \textbf{2.15}$$