BAB 2
TINJAUAN PUSTAKA

2.1 Tulisan Tangan

Tulisan tangan merupakan suatu hasil atau cara menulis dengan tangan seseorang [9]. Setiap tulisan tangan akan memiliki lebih banyak karakter karena tulisan setiap orang akan berbeda – beda. Dari beberapa penelitian menunjukkan tulisan tangan setiap orang unik dan mampu dikenali oleh suatu perangkat lunak khusus.

Tulisan tangan adalah keterampilan yang bersifat pribadi bagi individu [9]. Tulisan tangan telah berkembangkan sejak lama sebagai sarana untuk memperluas ingatan manusia dan untuk memfasilitasi komunikasi. Ada banyak cara untuk memperluas ingatan manusia serta memfasilitasi komunikasi seiring dengan berkembangnya jaman. Tulisan tangan berubah dari waktu ke waktu dan, sejauh ini, dorongan teknologi telah berkontribusi pada perluasannya [9].

2.2 Pengenalan Tulisan Tangan

Pengenalan tulisan tangan merupakan kemampuan komputer untuk menafsirkan inputan tulisan tangan yang dapat dimengerti dari suatu sumber seperti halnya dokumen kertas, foto dan perangkat lainnya [10]. Pengenalan tulisan tangan memerlukan sebuah perangkat lunak yang dapat mengubah teks dalam format citra ke dalam format teks yang bisa dibaca oleh komputer. Dimana teks dengan format citra didapat dengan cara memindai dengan scan, memfoto tulisan dipapan pengumuman, ataupun materi kuliah di papan tulis.

2.3 Penelitian Terkait

Pada bagian ini berisi penelitian yang pernah dilakukan dan literatur terkait dengan penelitian yang dilakukan:

Tabel 0.1 Penelitian Terkait

Tahun	Penulis	Isi
2018	Akhmad Ridhani, Fatma Indriani, Dodo T. Nugrahadi [1]	Pada penelitian tersebut menggunakan metode Kohonen Neural Network dalam pengenalan huruf tulisan tangan, yang memiliki tingkat akurasi sebesar 86,43% tetapi masih memiliki akurasi yang rendah terhadap pengenalan huruf B, N, R
2018	Alphien Andana, Ratna Widyati, Med Irzal [2]	Pengenalan citra tulisan tangan angka tersebut mendapatkan akurasi sebesar 96% menggunakan Backpropagation tetapi masih sulit dalam pengenalan angka tertentu seperti 4 dan 5 yang disebabkan dari tingkat ketajaman dan ketebalan sistem bisa juga dari ukuran dan bentuk tulisan dan juga dari presentasi keutuhan setiap digit citra.
2018	Yuyus Bahtiar Gustani [3]	Pengenalan tulisan tangan dalam kasus penilaian jawaban esai mendapatkan akurasi terbaik sebesar 77,79% hasilnya dipengaruhi oleh PCA Builder sebagai ektraksi ciri
2017	Sam'ani, M. Harris Qamaruzzaman[4]	Pengenalan huruf dan angka tulisan tangan menggunakan Convolutional Neural Network menghasilkan akurasi sebesar 82,72 % tanpa tahapan *preprocessing*.
2018	M. Ardi Firmansyah, Kurniawan Nur Ramadhani, Anditya Arifianto [5]	Pengenalan angka tulisan tangan menggunakan Diagonal Feature Ektraction dan klasifikasi Artificial Neural Network Multilayer Perception yang menghasilkan akurasi 92,30% pada dataset C1 dimana angka 5 dikenali sebagai angka 2 dan 92,60% pada dataset MNIST (dataset yang berisi citra

		angka tulisan tangan) angka 8 dikenali sebagai angka 3
2015	Ronaldo Messiana, Jerome Loulador [6]	pengenalan tulisan tangan cina tanpa segementasi dengan LSTM-RNN yang meghasilkan akurasi sebesar 83.5%

2.4 Pengolahan Citra

Pengolahan citra adalah suatu proses pengolahan citra dengan menggunakan komputer menjadi sebuah citra yang memiliki kualitas yang lebih baik. Pengolahan citra bertujuan untuk memperbaiki kualitas suatu citra sehingga dapat diinterpretasi dengan mudah oleh manusia atau sebuah mesin (komputer). Salah satu proses yang sering digunakan dalam pengolahan citra ialah pengenalan objek atau pengenalan pola. Pengenalan objek merupakan suatu proses untuk mengidentifikasi dan menemukan objek tertentu pada suatu citra atau video. Pengenalan objek atau citra pun berkaitan dengan klasifikasi citra, dimana data masukan berupa citra digital yang mengalami tahap *preprocessing*.

Dalam perkembangan lebih lanjut, pemrosesan suatu gambar dan *computer vision* digunakan sebagai pengganti mata manusia yang berperan menjadi gambar gambar input seperti kamera, *scanner*, dan komputer sebagai otak yang mengolah informasi. Beberapa orang menggunakan *computer vision*, pengenalan pola, pengenalan manusia berdasarkan karakteristik manusia, konten berbasis gambar dan retrieval video (mengambil gambar atau video) dengan informasi tertentu), penyuntingan video dan banyak lagi [11].

2.4.1 Grayscale

Citra grayscale untuk menangani warna hitam dan putih, yang menghasilkan efek warna abu – abu. Warna gambar dinyatakan dengan intensitas yang dimana intensitas berkisar 0 sampai dengan 255 dan memiliki kedalaman warna 8 bit. Nilai 0 menyatakan hitam dan 255 menyatakan putih [12]. Berikut rumus konversi citra RGB menjadi grayscale.

$$I = (0.2989 * R) + (0.5870 * G) + (0.1141 * B) \quad (2.1)$$

Keterangan:

R = komponen nilai merah (Red) dari suatu titik piksel

G = komponen nilai hijau (Green) dari suatu titik piksel

B = komponen nilai biru (Blue) dari suatu titik piksel

Persamaan diatas merupakan rumus yang digunakan untuk mengkonversi citra berwarna menjadi citra grayscale. Karena mata manusia sensitif terhadap cahaya merah dan hijau maka persamaan 2.1 dipilih dalam penelitian ini. Warna-warna ini diberi bobot yang lebih tinggi untuk memastikan bahwa keseimbangan intensitas relatif dalam citra grayscale yang dihasilkan mirip dengan citra warna RGB [12].

2.4.2 Sauvola Thresholding

Sauvola Threshold adalah metode *threshold* yang mampu memberikan peningkatan dengan cara menghitung ambang menggunakan rentang nilai dinamis dari nilai standar deviasi citra *grayscale* [13]. *Sauvola* termasuk dalam *local threshold* di mana nilai ambang ditentukan oleh nilai piksel tetangga.

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} - 1)) \qquad (2.2)$$

Pada rumus (2.2) terdapat rumus untuk menghitung m(*x,y*) yang dapat dihitung menggunakan rumus (2.3) dan rumus untuk menghitung s(*x,y*) dapat dilihat pada rumus (2.4).

$$m(0,0) = \frac{\sum_{i=min}^{i\ max} \sum_{j=min}^{j\ max} img(i,j)}{i*j} \qquad (2.3)$$

$$s(x,y) = \sqrt{\frac{\sum_{i=min}^{i\ max} \sum_{j=min}^{j\ max} (img(i,j) - m(x,y))^2}{(i*j)-1}} \qquad (2.4)$$

Keterangan:

R = Nilai maksimum dari standar deviasi (128 untuk citra *grayscale*)

k = *Kernel* dengan nilai antara 0.2 – 0.5.

m = Fungsi yang menghasilkan nilai rata-rata dari sejumlah piksel citra.

s = Fungsi yang menghasilkan nilai standar deviasi dari sejumlah piksel citra

T = Fungsi yang menghasilkan nilai *threshold* (ambang)

x = Nilai koordinat lebar citra

y = Nilai koordinat tinggi citra

i = indek pada x

j = indek pada y

Setelah nilai ambang T(x,y) sudah didapatkan, selanjutnya masukkan ke persamaan (2.5) sebagai berikut:

$$f(x,y) = \begin{cases} 0, & img(x,y) < T(x,y) \\ 255, & img(x,y) \geq T(x,y) \end{cases} \quad (2.5)$$

Keterangan:

f = Fungsi yang menghasilkan nilai 0 atau 255

img = Nilai *grayscale* citra

2.4.3 Segmentasi

Pada umumnya, metode *Projection Profile* digunakan untuk menentukan baris pada dokumen yang mana spasi antar baris terkelompokkan dengan jelas. Namun, metode ini pun dapat digunakan pada tulisan tangan. Metode *projection profile/*profil proyeksi ini dapat dikelompokkan menjadi dua yaitu profil proyeksi secara vertikal atau horizontal. Profil proyeksi vertikal biasanya digunakan untuk segmentasi kata. Sedangkan profil proyeksi horizontal banyak digunakan untuk segmentasi baris.

Cara kerja dari metode *Projection Profile* adalah dengan menjadikan huruf atau objek kedalam bentuk garis-garis histogram vertikal dan horisontal. Metode *Projection Profile Histogram* ini berdasarkan pada penggunaan profil histogram yang telah tercipta dan diproporsikan oleh suatu citra. Profil proyeksi merupakan sebuah struktur data yang digunakan untuk menyimpan sejumlah piksel hitam yang merupakan objek ketika suatu citra diproyeksikan melalui sumbu x maupun sumbu y [14].

2.4.4 Resize

Resize citra artinya mengubah besarnya ukuran citra digital dalam piksel. Resize atau penskalaan adalah sebuah operasi geometri yang digunakan untuk memperbesar atau memperkecil ukuran dari sebuah citra sesuai dengan ukuran yang dibutuhkan [15]. Adakala ukurannya berubah menjadi lebih kecil dari ukuran aslinya dan ada kalanya sebaliknya. Proses *Resize* citra dapat mempengaruhi *size* citra lebih besar atau lebih kecil dan juga kualitas citra menjadi lebih baik atau lebih buruk. Untuk melakukan pensekalaan pada penelitian ini tidak digunakan metode

khusus. Untuk mendapatkan ukuran yang diinginkan maka akan dilakukan perbandingan ukuran antara citra segmentasi dengan target ukuran citra dengan persamaan berikut:

$$x = \frac{pb * pp}{pa} \qquad (2.6)$$

Keterangan:

x = Posisi x baru

pb = Ukuran panjang dari matriks baru.

pp = Posisi piksel x lama.

pa = Ukuran panjang dari matriks lama.

$$y = \frac{lb * pp}{la} \qquad (2.7)$$

Keterangan:

y = Posisi y baru

lb = Ukuran lebar dari matriks baru.

pp = Posisi piksel y lama.

la = Ukuran lebar dari matriks lama.

2.5 Deep Learning

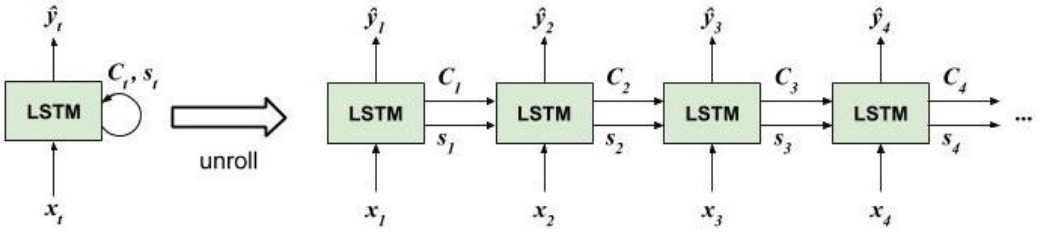
Deep Learning adalah representasi- metode pembelajaran dengan berbagai tingkat representasi, yang diperoleh dengan menyusun modul sederhana namun non-linear yang masing-masing mengubah representasi pada satu level (dimulai dengan input mentah) menjadi representasi tingkat yang lebih tinggi, yang mana tingkatnya sedikit lebih abstrak. Representasi pembelajaran adalah seperangkat metode yang memungkinkan mesin untuk diberi masukan dengan data mentah dan secara otomatis menemukan representasi yang diperlukan untuk mendeteksi atau klasifikasi. *Deep Learning* adalah suatu area baru dalam penelitian *Machine Learning*, yang bertujuan untuk menggerakkan *Machine Learning* lebih dekat dengan salah satu tujuan utamanya yaitu *Artificial Intelligence* [16].

Deep Learning adalah tentang bagaiman belajar beberapa tingkat representasi dan abstraksi yang membantu untuk memahami data seperti gambar, suara, dan teks. *Deep learning* akan memiliki lebih banyak keberhasilan dalam waktu dekat ini karena hanya membutuhkan sedikit teknik dengan tangan, oleh karena itu *deep learning* dapat

dengan mudah mengambil keuntungan dari peningkatan jumlah perhitungan yang tersedia dan data [17]. Algoritma dan arsitektur pembelajaran baru yang saat ini sedang dikembangkan *deep neural network* inipun akan mampu mempercepat kemajuan ini.

2.6 Long Short Term Memory (LSTM)

Long Short Term Memory network (LSTM) merupakan salah satu jenis pemrosesan lain dari RNN. LSTM diciptakan oleh Hochreiter dan Jurgen Schmidhuber pada tahun 1997 [18]. RNN tidak dapat belajar menghubungkan informasi karena memori lama yang tersimpan akan semakin tidak berguna dengan seiringnya waktu karena tertimpa atau tergantikan dengan memori baru. Berbeda dengan LSTM yang dapat mengatur memori pada setiap masukannya dengan menggunakan memory cells dan gate units.



Gambar 0.1 Jaringan LSTM

Modul LSTM (satu kotak hijau) mempunyai pemrosesan yang berbeda dengan modul RNN biasa. Perbedaan lain adalah adanya tambahan sinyal yang diberikan dari satu langkah waktu ke langkah waktu berikutnya, yaitu konteks, direpresentasikan dengan simbol $\boldsymbol{C}_t$ [19]. Ide kunci dari LSTM adalah jalur yang menghubungkan konteks lama ($\boldsymbol{C}_{t-1}$) ke konteks baru ($\boldsymbol{C}_t$) dibagian atas modul LSTM. Konteks $\boldsymbol{C}_t$ disebut juga *cell state* atau *memory cell*. Dengan adanya jalur tersebut, suatu nilai dikonteks yang lama akan dengan mudah diteruskan ke konteks yang baru dengan sedikit sekali modifikasi, kalau diperlukan. Konteks adalah sebuah vektor, yang jumlah elemennya kita tentukan sebagai desainer jaringan LSTM. Intuisinya adalah, masing-masing elemen kita harapkan bisa merekam suatu fitur dari input. Terdapat empat proses fungsi aktivasi pada setiap masukan pada

neurons yang selanjutnya disebut sebagai *gates units. Gates units* tersebut ialah *forget gates, input gates, cell gates,* dan *output gates* [20].

Pada *forget gates* informasi pada setiap data masukan akan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada *memory cells.* dengan menggunakan gerbang sigmoid yang disebut "gerbang lupa" (*forget gate,* $\boldsymbol{f}_t$). Gerbang menghasilkan angka antara 0 dan 1 untuk setiap elemen dalam $\boldsymbol{C}_{t-1}$. Nilai 1 artinya "benar-benar jaga elemen ini" sementara 0 artinya "benar-benar singkirkan elemen ini." Dengan rumus sebagai berikut:

$$f_t = Sigmoid\left(W_{fx} . x_t + W_{fh} . h_{t-1} + b_f\right) \qquad (2.8)$$

Selanjutnya memutuskan informasi baru apa yang akan digunakan pada $\boldsymbol{C}_t$. Proses ini memiliki dua bagian. Pertama, gerbang sigmoid yang disebut "gerbang masukan" (*input gate,* $\boldsymbol{i}_t$) memutuskan nilai mana yang akan di perbarui. Lalu sebuah lapis *tanh* menghasilkan kandidat vektor konteks baru c_t. Lalu digabungkan keduanya untuk membuat pembaruan ke konteks nanti. Dengan rumus berikut:

$$i_t = Sigmoid(W_{ix} . x_t + W_{ih} . h_{t-1} + b_i) \qquad (2.9)$$

$$c_t = tanh(W_{cx} . x_t + W_{ch} . h_{t-1} + b_c) \qquad (2.10)$$

Pada *cell gates* akan mengganti nilai pada *memory cell* sebelumnya dengan nilai *memory cell* yang baru. Dimana nilai ini didapatkan dari menggabungkan nilai yang terdapat pada *forget gate* (f_t) dan *input gate* (c_t). Dengan rumus sebagai berikut:

$$C_t = (c_t \odot i_t) + (f_t \odot C_{t-1}) \qquad (2.11)$$

Tahap terakhir *output gates* memutuskan apa yang akan dihasilkan. Output ini akan didasarkan pada nilai dalam konteks dan dilewatkan ke suatu filter. Pertama, jalankan gerbang sigmoid yang dinamakan gerbang output (*output gate,* $\boldsymbol{o}_t$) untuk memutuskan bagian-bagian apa dari konteks yang akan kita hasilkan. Kemudian, lewatkan konteks melalui *tanh* untuk membuat nilainya menjadi antara −1 dan 1, dan kalikan dengan output gerbang sigmoid tadi sehingga kita hanya menghasilkan bagian yang kita putuskan. Dengan rumus sebagai berikut:

$$o_t = Sigmoid(W_{ox} . x_t + W_{oh} . h_{t-1} + b_o) \qquad (2.12)$$

$$h_t = \tanh(C_t) \odot o_t \qquad (2.13)$$

2.7 Fungsi Aktivasi

Dalam neural network fungsi aktivasi sangat umum digunakan. Fungsi aktivasi digunakan agar neural network dapat mengenali data yang non-linear, dikarenakan hasil output neural network jarang bersifat linear [21]. Selain itu fungsi aktivasi juga digunakan untuk mengaktif dan non-aktifkan neuron, bobot dan input-output fungsi aktivasi ditentukan sebagai karakteristik neural network. Fungsi aktivasi yang digunakan pada penelitian ini yaitu fungsi aktivasi *sigmoid*, *hyperbolic tangent* dan *softmax*.

2.7.1 Sigmoid

Fungsi aktivasi sigmoid digunakan untuk mengatur seberapa banyak informasi bisa lewat. Fungsi sigmoid menghasilkan output yang merupakan kisaran nilai antara 0 dan 1 [21]. Persamaan dari fungsi sigmoid dapat dilihat pada persamaan berikut:

$$Sigmoid(x_i) = \left(\frac{1}{1+e^{-x_i}}\right) \quad (2.14)$$

2.7.2 Tanh (Hyperbolic Tangent)

Fungsi aktivasi *hyperbolic tangent* yaitu tipe aktivasi fungsi dalam *deep learning* dan memiliki beberapa varian, selain itu fungsi aktivasi *hyperbolic tangent* dikenal juga sebagai fungsi *tanh* yang menghasilkan nilai -1 sampai 1 [22]. Dibandingkan dengan fungsi *sigmoid*, fungsi *tanh* memberikan kinerja pelatihan yang lebih baik untuk *multi layer neural network*. Persamaan dari fungsi *tanh* sebagai berikut:

$$tanh\ (x_i) = \frac{e^{2x_i}-1}{e^{2x_i}+1} \quad (2.15)$$

2.7.3 Softmax

Dalam permasalahan *multiclass classification*, output layer biasanya memiliki lebih dari 1 neuron. Untuk kasus ini, fungsi aktivasi yang biasa digunakan adalah fungsi *Softmax*. Persamaan dari fungsi *Softmax* sebagai berikut:

$$\sigma(z)j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (2.16)$$

2.8 Loss Function

Untuk menghitung BPTT maka perlu ditentukan terlebih dahulu fungsi *loss* atau *error*. Loss bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai yang sebenarnya [23]. Untuk penelitian ini, fungsi *loss* menggunakan persamaan berikut:

$$E = Y - label \tag{2.17}$$

Keterangan:

E = error dari output

Y = hasil softmax dari *time step* t (vektor).

$label$ = label sebenarnya pada *time step* t (vektor).

2.9 Backpropagation Through Time (BPTT)

Backpropagation Through Time, atau BPTT, merupakan algoritma pelatihan yang digunakan untuk memperbarui bobot dalam jaringan saraf berulang seperti LSTM. Prinsip dasar dari BPTT adalah *unfolding* [23]. Secara konseptual, BPTT bekerja dengan membuka gulungan (*unfolding*) setiap *input* layer pada setiap *timestep* (t), kemudian nilai *error* dihitung dan diakumulasikan untuk setiap *timestep* (t). Jaringan kemudian diputar kembali untuk memperbaharui bobot. Pada pelatihan LSTM terdapat 12 bobot yang akan diperbarui, agar memudahkan perhitungan dibuat pemisalan [24] sebagai berikut:

$$gates = \begin{bmatrix} c_t \\ i_t \\ f_t \\ o_t \end{bmatrix} \quad W_x = \begin{bmatrix} w_{cx} \\ w_{ix} \\ w_{fx} \\ w_{ox} \end{bmatrix} \quad W_h = \begin{bmatrix} w_{ch} \\ w_{ih} \\ w_{fh} \\ w_{oh} \end{bmatrix} \quad b = \begin{bmatrix} b_c \\ b_i \\ b_f \\ b_o \end{bmatrix}$$

Seperti yang ditunjukkan pemisalan diatas, maka untuk menghitung turunan masing-masing setiap *time step* (t) terhadap bobot yang terdapat pada *gates.* Persamaan yang digunakan untuk menghitung candidat c_t, dimana didalamnya terdapat cell memory C_t. Maka untuk menghitung cell memory dan candidat tersebut menggunakan persamaan berikut:

$$\partial C_t = \partial h_t \odot c_t \odot \left(1 - tanh^2(C_t)\right) + \partial C_{t+1} \odot f_{t+1} \tag{2.17}$$

$$\partial c_t = \partial C_t \odot i_t \odot (1 - c_t^2) \tag{2.18}$$

Untuk menghitung input i_t menggunakan persamaan berikut:

$$\partial i_t = \partial C_t \odot c_t \odot i_t \odot (1 - i_t) \qquad (2.19)$$

Dan untuk menghitung forget f_t menggunakan persamaan berikut:

$$\partial f_t = \partial C_t \odot C_{t-1} \odot f_t \odot (1 - f_t) \qquad (2.20)$$

Sedangkan untuk menghitung output o_t yang dimana didalamnya terdapat hidden h_t. Maka untuk menghitung output dan hidden tersebut menggunakan persamaan berikut:

$$\partial o_t = \partial h_t \odot tanh(C_t) \odot o_t \odot (1 - o_t) \qquad (2.21)$$

Dan untuk menghitung hidden h_t dengan menggunakan persamaan berikut jika t=28:

$$\Delta h_t = E \qquad (2.22)$$

Dimana E adalah *loss* atau *error*. Tetepi jika t < 28 maka menggunakan persamaan berikut:

$$\Delta h_t = \partial h_c + \partial h_i + \partial h_f + \partial h_o \qquad (2.23)$$

Dalam menghitung proses setiap bobot ∂h menggunakan persamaan berikut:

$$\partial h_c = Wc_{h_{t+1}} . \partial c_{t+1}$$

$$\partial h_i = Wi_{h_{t+1}} . \partial i_{t+1}$$

$$\partial h_f = Wf_{h_{t+1}} . \partial f_{t+1}$$

$$\partial h_o = Wo_{h_{t+1}} . \partial o_{t+1}$$

Selanjutnya menghitung turunan masing-masing setiap *time step t* terhadap bobot yang terdapat pada W_x. Maka perhitungan menggunakan persamaan berikut:

$$\partial W_x = \partial gates \odot X_t^T \qquad (2.24)$$

$$\begin{bmatrix} \partial c_t \\ \partial i_t \\ \partial f_t \\ \partial o_t \end{bmatrix} \odot X_t^T$$

Jika kita jabarkan maka persamaan diatas menjadi:

$$\partial W_{cx} = \partial c_t \odot X_t^T$$

$$\partial W_{ix} = \partial i_t \odot X_t^T$$

$$\partial W_{fx} = \partial f_t \odot X_t^T$$

$$\partial W_{ox} = \partial o_t \odot X_t^T$$

Dan menghitung turunan masing-masing setiap *time step t* terhadap bobot yang terdapat pada W_h. Maka perhitungan menggunakan persamaan berikut:

$$\partial W_h = \partial gates \odot h_t \qquad (2.25)$$

$$\begin{bmatrix} \partial c_t \\ \partial i_t \\ \partial f_t \\ \partial o_t \end{bmatrix} \odot h_t$$

Jika kita jabarkan maka persamaan diatas menjadi:

$$\partial W_{ch} = \partial c_t \odot h_t$$
$$\partial W_{ih} = \partial i_t \odot h_t$$
$$\partial W_{fh} = \partial f_t \odot h_t$$
$$\partial W_{oh} = \partial o_t \odot h_t$$

Terakhir menghitung turunan masing-masing setiap *time step t* terhadap bobot yang terdapat pada b. Maka perhitungan menggunakan persamaan berikut:

$$\partial b = \partial gates \qquad (2.26)$$

$$\begin{bmatrix} \partial c_t \\ \partial i_t \\ \partial f_t \\ \partial o_t \end{bmatrix}$$

2.10 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent adalah salah satu metode dalam dengan meminimalkan nilai *error* untuk mengukur keakuratan jaringan dalam melakukan tugas yang diberikan [22]. SGD bertujuan untuk mengatasi nilai *error* (*loss*) yang tinggi dengan cara memperbarui parameter. Algoritma *machine learning* dikatakan algoritma yang bagus apabila memberikan sedikit kesalahan (*error*). Biasanya, apabila angka *loss function* ini kecil atau mendekati nol maka performa dikatakan bagus. Besar kecilnya nilai *loss* bergantung pada bobot—tentunya algoritma itu sendiri juga berperan. Maka pada setiap algoritma *machine learning* terdapat yang namanya fase pelatihan (*trainning*) yang bertujuan mencari bobot optimal, yaitu bobot yang memberikan *error* minimal. Adapaun perhitungan mencari bobot baru sebagai berikut:

$$W_{baru} = W_{lama} + \alpha \,.\, \partial W_{lama} \qquad (2.27)$$

Keterangan :

W_{lama} = Nilai bobot yang lama

∂W_{lama} = Nilai bobot yang telah dilakukan perhitungan sebelumnya

α = *learning reat*

2.11 Akurasi

Perhitungan akurasi yang digunakan pada penelitian untuk mengetahui tingkat akurasi pengenalan tulisan tangan [23]. Untuk menghitung akurasi dari LSTM dilakukan perhitungan dengan rumus berikut.

$$Akurasi = \frac{Jumlah\ Yang\ Benar}{Jumlah\ Data}\ X100\% \qquad (2.28)$$

2.12 Unified Modeling Language (UML)

Sebuah teknik pembuatan model untuk pengembangan atau pembangunan sistem. Secara umum modeling language dapat juga dibuat dengan menggunakan pseudo-code, actual code, gambar, maupun diagram. Beberapa model yang digunakan dalam UML:

2.12.1 *Use Case* Diagram

Use case merupakan situasi sistem yang digunakan untuk memenuhi kebutuhan user. *Use case* merupakan bagian terpenting dalam pembangunan sistem. *Use case* ini suatu interaksi antara *actor* dengan sistem. Relasi antara use case dengan use case lainnya ada tiga jenis, yaitu:

1. Relasi *Include*

   Relasi antar use case dimana jika use case A meng-include use case B, maka use case B akan diimplementasikan setiap kali use case A diimplementasikan. Digambarkan dengan garis putus-putus bertuliskan <<include>> ke arah use case yang akan di-include.

2. Relasi *Extend*

   Relasi antar use case dimana jika use case A di extend oleh use case B maka use case B akan bisa saja diimplementasikan atau tidak setiap kali use case A diimplementasikan. Digambarkan dengan garis putus-putus bertuliskan <<extend>> ke arah use case yang akan di-extend.

3. Relasi *Genealization*

Digunakan untuk membuat use case lebih spesifik.

Setelah pembangunan *use case* akan dibangun pula sebuah skenario dari masing-masing *use case* yang ada. Skenario *use case* digunakan untuk memperjelas proses yang ada di dalam masing-masing use case tersebut secara tekstual.

2.12.2 Activity Diagram

Activity diagram yang menjelaskan alur dari sistem, yang mengendalikan satu aktivitas ke aktivitas lainnya. *Activity* diagram menjelaskan alurnya dengan cara *actions chained* atau aksi-aksi yang saling berhubungan pada sistem tersebut. *Activity diagram* berupa operasi-operasi dan aktivitas-aktivitas di *use case*. *Activity diagram* dapat digunakan untuk :

1. Pandangan dalam yang dilakukan di operasi.
2. Pandangan dalam bagaimana objek-objek bekerja.
3. Pandangan dalam di aksi-aksi dan pengaruhnya pada objek-objek
4. Pandangan dalam dari suatu use case.
5. Logik dari proses bisnis.

2.12.3 *Class* Diagram

Class diagram yang menjelaskan bagian objek-objek yang dibutuhkan pada sistem. *Class* diagram menunjukkan kelas-kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi - operasi di setiap kelas. Diagram ini menggambarkan aspek statis sistem terutama untuk mendukung kebutuhan fungsional sistem. Elemen-elemen yang penting dalam class diagram adalah sebagai berikut.

1. Kelas

   Elemen terpenting yang berorientasi objek pada sistem. Kelas menggambarkan satu blok pembangunan sistem. Bagian-bagian dari kelas adalah sebagai berikut:

   a. Nama, nama kelas pada program harus unik karena akan menjadi identifier pada program
   b. Atribut, properti bernama dikelas yang menggambarkan range nilai yang dimiliki oleh kelas.

c. Operasi, Implementasi layanan yang dapat memperbaiki objek kelas apa pun untuk mempengaruhi perilaku sistem.
d. Access Modifier di kelas digunakan untuk membatasi akses dari kelas lain yang ingin mengakses atribut atau operasi kelas. Ada tiga Access Modifiers yang digunakan yaitu publik (+), protected (#) dan private (-).

2. Antarmuka

   Antarmuka (Interface) adalah korelasi operasi yang menentukan layanan kelas atau komponen. Antarmuka menggambarkan perilaku elemen yang terlihat secara eksternal.

3. Kolaborasi

   Kolaborasi adalah pendefinisian suatu interaksi dan sekelompok peran elemen-elemen lain yang bekerja sama untuk menyediakan suatu perilaku kooperatif yang lebih besar dari penjumlahan seluruh elemen.

4. Hubungan

   Hubungan antar kelas di diagram kelas terdiri dari Asosiasi dan Generalisasi.

2.12.4 *Sequence* Diagram

Sequence diagram untuk menjelaskan urutan hubungan atau interaksi terhadap bagian-bagian dari sistem. Sequence diagram menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horizontal dari objek pengirim ke objek pengirim ke objek penerima.

2.13 Python

Python adalah bahasa pemrograman interpretatif multiguna [25]. Bahasa pemrograman python merupakan bahasa pemrograman tingkat tinggi yang dapat melakukan eksekusi sejumlah instruksi multi guna secara langsung (interpretatif) dengan metode orientasi objek (Object Oriented Programming) serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax. Sebagai bahasa pemrograman tingat tinggi, python dapat dipelajari dengan mudah karena sudah dilengkapi dengan manajemen memori otomatis (pointer). Python ini diklaim sebagai bahasa yang mampu menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas serta dilengkapi dengan fungsionalitas pustaka

standar yang besar dan komprehensif. Python memiliki banyak fitur salah satunya adalah fitur python sebagai bahasa pemrograman yang dinamis serta dilengkapi dengan manajemen memori otomatis.

Sebuah komputer hanya dapat mengeksekusi program yang ditulis dalam bentuk bahasa mesin. Oleh karena itu, jika suatu program ditulis dalam bentuk bahasa tingkat tinggi, maka program tersebut harus diproses dulu sebelum bisa dijalankan dalam komputer. Hal ini merupakan salah satu kekurangan bahasa tingkat tinggi yang memerlukan waktu untuk memproses suatu program sebelum program tersebut dijalankan. Akan tetapi, bahasa tingkat tinggi mempunyai banyak sekali keuntungan. Bahasa tingkat tinggi juga mudah diubah portabel untuk disesuaikan dengan mesin yang menjalankannya. Hal ini berbeda dengan bahasa mesin yang hanya dapat digunakan untuk mesin tersebut. Dengan berbagai kelebihan ini, maka banyak aplikasi ditulis menggunakan bahasa tingkat tinggi. Proses mengubah dad bentuk bahasa tingkat tinggi ke tingkat rendah datam bahasa pemrograman ada dua tipe, yakni interpreter dan compiler.

Python memiliki banyak library yang dapat dimanfaatkan sesuai dengan kebutuhan, beberapa dari library yang digunakan pada penelitian ini:

2.13.1 Tensorflow

Tensorflow merupakan sebuah perpustakaan software yang digunakan machine learning dalam berbagai macam tugas pemahaman persepsi dan bahasa. Tensorflow dikembangkan oleh Brain Google Team untuk penelitian dan produksi google, kemudian dirilis dibawah lisensi Apache 2.0 pada November 2015. Tensorflow merupakan API generasi kedua yang menggantikan DistBelief yang digunakan oleh 50 tim yang berbeda untuk penelitian dan pengembangan puluhan produk komersial milik google.

Tensorflow juga merupakan library yang ditulis dengan bahasa C++ dan biasanya digunakan dengan bahasa pemrograman Python. Dengan adanya tensorflow, sekarang sudah bisa menggunakan beberapa fitur tensorflow di sisi web browser tanpa harus dibebani oleh instalasi yang cukup ‘menantang’. Dengan tensorflow kita tinggal melakukan instalasi dengan `npm install @tensorflow/tfjs` ataupun juga dengan menggunakan CDN. TensorFlow

kemudian menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan. Fitur utamanya meliputi:

a. Mendefinisikan, mengoptimalkan, dan menghitung secara efisien ekspresi matematis yang melibatkan array multi dimensi (tensors).
b. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin
c. Penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. Tensorflow bisa menulis kode yang sama dan menjalankannya baik di CPU atau GPU. Lebih khusus lagi, TensorFlow akan mengetahui bagian perhitungan mana yang harus dipindahkan ke GPU.

Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

2.13.2 Numpy

Numpy merupakan library pada Python yang digunakan untuk melakukan perhitungan scientific. Di dalam numpy ini terdapat paket-paket untuk melakukan operasi terhadap objek array yang berupa matriks dengan N-dimensi, aljabar linear, dan banyak lagi (https://www.numpy.org/ diakses pada tanggal 11 November 2019).

2.13.3 OpenCV

OpenCV merupakan library yang digunakan khusus untuk pembangunan computer vision dan machine learning. OpenCV memiliki algoritma yang teroptimasi dengan jumlah lebih dari 2500. Library OpenCV tidak hanya ada untuk bahasa pemrograman Python saja, melainkan ada pula untuk bahasa lain seperti C++, Java, dan MATLAB Interfaces dan sudah sangat support penggunaannya pada operasi sistem Windows, Linux, Mac OS dan Android (https://opencv.org/about/ diakses pada tanggal 11 November 2019).

2.13.4 PyQt

PyQt merupakan library pada python untuk membangun Graphic User Interface (GUI). Di dalam PyQt terdapat paket bernama QtDesigner untuk

mempermudah pembangunan GUI. Dengan cara drag & drop desain saja. PyQt memiliki lebih dari 35 ekstensi modul yang siap digunakan dan mampu membuat Python sebagai bahasa alternatif dari C++ (https://wiki.python.org/moin/PyQt diakses pada tanggal 11 November 2019).


UNIVERSITAS KOMPUTER INDONESIA
UNIKOM