
**SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI**

# PENGENALAN KOSA KATA BAHASA ISYARAT INDONESIA DENGAN MENGGUNAKAN METODE *LONG-SHORT TERM MEMORY*

**TUGAS AKHIR**

**MUTIYA ADIFA NURILMI**
**0110218034**

**PROGRAM STUDI TEKNIK INFORMATIKA**
**JAKARTA**
**AGUSTUS 2024**

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

# PENGENALAN KOSA KATA BAHASA ISYARAT INDONESIA DENGAN MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

**TUGAS AKHIR**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer**

**MUTIYA ADIFA NURILMI**

**0110218034**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**JAKARTA**

**AGUSTUS 2024**

## HALAMAN PERNYATAAN ORISINALITAS

**Skripsi/Tugas Akhir ini adalah hasil karya penulis, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.**

| | |
|---|---|
| **Nama** | **: Mutiya Adifa Nurilmi** |
| **NIM** | **: 0110218034** |



Jakarta, 06 Agustus 2024

Tanda Tangan

Mutiya Adifa Nurilmi

## HALAMAN PENGESAHAN

Skripsi/Tugas Akhir ini diajukan oleh :

Nama : Mutiya Adifa Nurilmi

NIM : 0110218034

Program Studi : Teknik Informatika

Judul Skripsi : Pengenalan Kosa Kata Bahasa Isyarat Indonesia dengan Menggunakan Metode Long Short Term Memory

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri**

**DEWAN PENGUJI**

| Pembimbing | Penguji |
| --- | --- |
| Ahmad Rio Adriansyah, S.Si., M.Si. | Tifanny Nabarian, S.Kom., M.T.I. |

Ditetapkan di : Depok

Tanggal : 22 Juli 2024

## KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi/Tugas Akhir ini. Penulisan skripsi/Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana komputer Program Studi Teknik Informatika pada Sekolah Tinggi Teknologi Terpadu Nurul Fikri Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi/tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT telah memberi petunjuk serta taufik hingga penulis bisa menyelesaikan tugas akhir ini
2. Ibu, Bapak, Kakak Syiffa, Mas Syaffa, dan Daffanti yang telah memberikan dorongan baik secara moril maupun materil dalam penyelesaian tugas ini.
3. Bapak Dr. Lukman Rosyidi selaku Ketua Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
4. Ibu Tifanny Nabarian, S.Kom, M.T.i selaku Ketua Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
5. Bapak Hilmy Abidzar Tawakal, S.T, M.Kom selaku Dosen Pembimbing Akademik yang telah membimbing penulis selama berkuliah di Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
6. Bapak Ahmad Rio Adriansyah, S.Si, M.Si. selaku Dosen Pembimbing Tugas Akhir penulis dalam menyelesaikan penulisan ilmiah ini.
7. Para Dosen di lingkungan Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang telah membimbing penulis dalam menuntut ilmu yang telah diberikan.
8. Nicholas Renotte yang telah membuat video mengenai pendeteksian bahasa isyarat menggunakan LSTM

Dalam penulisan ilmiah ini tentu saja masih banyak terdapat kekurangan-kekurangan yang mungkin disebabkan oleh keterbatasan kemampuan dan pengetahuan yang penulis miliki. Walaupun demikian, penulis telah berusaha menyelesaikan penulisan ilmiah ini sebaik mungkin. Oleh karena itu apabila terdapat kekurangan di dalam penulisan ilmiah ini, dengan rendah hati penulis menerima kritik dan saran dari pembaca.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Jakarta, 24 Juli 2024

Mutiya Adifa Nurilmi

## HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Sekolah Tinggi Teknologi Terpadu Nurul Fikri, saya yang bertanda tangan di bawah ini:

Nama : Mutiya Adifa Nurilmi

NIM : 0110218034

Program Studi : Teknik Informatika

Jenis karya : Skripsi / Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada STT-NF **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty - Free Right*)** atas karya ilmiah saya yang berjudul :

**Pengenalan Kosa Kata Bahasa Isyarat Indonesia dengan Menggunakan Metode *Long Short Term Memory***

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini STT-NF berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta

Pada tanggal : 06 Agustus 2024

Yang Menyatakan


( Mutiya Adifa Nurilmi )

# ABSTRAK

(300 kata )


Nama : Mutiya Adifa Nurilmi

NIM : 011218034

Program Studi : Teknik Informatika


Judul : Pengenalan Kosa Kata Bahasa Isyarat Indonesia dengan Menggunakan Metode *Long Short Term Memory*


BISINDO adalah bahasa isyarat yang banyak digunakan dan dipilih sebagai alat komunikasi oleh komunitas tuli. Berkembangnya teknologi pada masa ini banyak peneliti yang meneliti mengenai pengenalan bahasa isyarat menggunakan machine learning. Kebanyakan penelitian yang sudah ada mengenai pengenalan alfabet dan angka dalam bahasa isyarat. Tidak seperti alfabet dan angka dalam bahasa isyarat yang statis, kecuali alfabet J dan R yang memiliki sedikit pergerakan yang tidak diikut sertakan, kosa kata dalam bahasa isyarat memerlukan pergerakan tangan, ekspresi wajah dan anggota tubuh lainnya. *Long-Short Term Memory* (LSTM) dapat digunakan untuk data yang berjenis video, yang dimana pergerakan bahasa isyarat kosa kata dapat disimpan dalam bentuk video. MediaPipe Holistik digunakan pada penelitian ini. Data yang digunakan sebanyak 500 data, dibagi menjadi 95% data latih, 2.5% data validasi, dan 2.5% data uji. Menggunakan 2 model LSTM dengan aktivasi yang akan digunakan berbeda, yaitu *ReLu* dan *Tanh*. Dari hasil pengujian menggunakan data uji menghasilkan tingkat akurasi untuk model dengan *ReLu* sebesar 92% sedangkan dengan *Tanh* sebesar 84%.



Kata kunci : *Long Short Term Memory* (LSTM), MediaPipe Holistik, Bahasa Isyarat, BISINDO

## *ABSTRACT*


*Name* : Mutiya Adifa Nurilmi

NIM : 0110218034

*Study Program* : *Informatics*


*Title* : *Recognition Indonesian Sign Language using Long-Short Term Memory Method*


*BISINDO is a sign language that is widely used and chosen as a communication tool by the deaf community. With the development of technology at this time, many researchers are researched sign language recognition using machine learning. Many existing research concerns the recognition of the alphabet and numbers in sign languages. Unlike the alphabet and numbers in sign language which are static, except for the alphabet J and R which have little movement involved, vocabulary in sign language requires hand movements, facial expressions and other body parts. In this research LSTM is used for video data, to recognize the movement of sign language vocabulary. Holistic MediaPipe was used as a solution ti capture and trachk full-body pose. The data used was 500 data, divided into 95% training data, 2.5% validation data and 2.5% test data. Used 2 LSTM models with different activations, namely ReLu and Tanh. From the test results, the accuracy level for the model with ReLu is 92% while with Tanh it is 84%.*



*Key words* : *Long Short Term Memory* (LSTM), *MediaPipe Holistic*, *Sign Language, BISINDO*

**DAFTAR ISI**

## DAFTAR GAMBAR

**DAFTAR TABEL**

# BAB 1
# PENDAHULUAN

Pendahuluan merupakan bagian awal dari laporan penelitian yang berisi mengenai latar belakang dari penelitian yang akan dilakukan, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

## 1.1 Latar belakang

Manusia adalah makhluk sosial yang mengharuskan mereka untuk saling berinteraksi dan berkomunikasi. Agar dapat berkomunikasi dan memahami satu dengan yang lainnya memerlukan bahasa. Bentuk bahasa yang dapat digunakan oleh manusia ada dua jenis, yaitu bahasa verbal dan bahasa bukan verbal. Bahasa verbal adalah bahasa yang dapat diucapkan secara lisan. Sedangkan bahasa bukan verbal adalah bahasa yang bukan dalam bentuk lisan, seperti bahasa isyarat.

Bahasa isyarat adalah bentuk komunikasi dengan menggunakan gerak anggota tubuh, gerak mulut, dan ekspresi wajah. Bahasa isyarat digunakan oleh komunitas atau seseorang yang memiliki kekurangan dalam kemampuan berbicara dan mendengar, seperti tuli. Menurut Kementerian sosial pada tahun 2021 jumlah penyandang disabilitas di Indonesia tercatat sebanyak 30,38 juta, yang mana 7.03% adalah penyandang tuli [1].

Setiap negara memiliki standar sendiri dalam penggunaan bahasa isyarat. Di Indonesia terdapat dua bentuk bahasa isyarat, yaitu SIBI (Sistem Isyarat Bahasa Indonesia dan BISINDO (Bahasa Isyarat Indonesia). Pemerintah menetapkan Keputusan Mendikbud No. 0161/U/2994 tanggal 30 Juni 1994 tentang tentang Pembakuan Sistem Isyarat Bahasa Indonesia SIBI, dan menjadikan SIBI sebagai bahan pembelajaran di pendidikan sekolah luar biasa. Walaupun demikian BISINDO adalah bahasa isyarat yang banyak digunakan dan dipilih sebagai alat komunikasi oleh komunitas tuli [2].

BISINDO adalah bahasa isyarat sehari-hari yang digunakan oleh komunitas tuli di Indonesia. Perbedaan SIBI dan BISINDO adalah, SIBI diciptakan orang dengar dengan merujuk ASL, sedangkan BISINDO diciptakan oleh komunitas tuli yang berada dalam Gerakan untuk Kesejahteraan Tunarungu Indonesia

(GERKATIN). BISINDO selain digunakan oleh kelompok tuli, dapat dipelajari oleh orang yang tidak tuli melalui kelas yang disediakan oleh Pusat Bahasa Isyarat Indonesia (PUSBISINDO). Namun hanya sebagian orang Indonesia yang memahami BISINDO, hal ini mengakibatkan masih banyak orang yang tidak dapat memahami atau mengerti BISINDO.

Berkembangnya teknologi pada masa ini banyak peneliti yang meneliti mengenai pengenalan bahasa isyarat menggunakan *machine learning*. Kebanyakan penelitian yang sudah ada mengenai pengenalan alfabet dan angka dalam bahasa isyarat. Dalam penelitian tersebut salah satu metode machine learning yang digunakan adalah *Convolutional Neural Network* (CNN). Selain menggunakan CNN, deteksi alfabet bahasa isyarat dilakukan menggunakan pengolahan citra dengan menggunakan metode HOG dan SVM dengan hasil tingkat akurasinya sebesar 81,25% [3]

Namun penggunaan CNN dan pengolahan citra menggunakan HOG dan SVM hanya berlaku untuk citra statis atau diam seperti gambar. Penelitian yang dilakukan oleh Devina, dkk [4] yaitu pengenalan alfabet bahasa isyarat dengan menggunakan *Convolutional Neural Network* (CNN) dan dibantu oleh *Recurrent Neural Network* (RNN) untuk dapat memproses data video atau citra yang tidak statis. Diketahui bahwa CNN hanya berlaku untuk data berbentuk foto, maka dari itu untuk membantu CNN dalam mengklasifikasi data video diperlukan *time distributed layer* yang dimiliki oleh RNN.

Tidak seperti alfabet dan angka dalam bahasa isyarat yang statis, kecuali alfabet J dan R yang memiliki sedikit pergerakan yang tidak diikut sertakan [3], kosa kata dalam bahasa isyarat memerlukan pergerakan tangan, ekspresi wajah dan anggota tubuh lainnya. Di Indonesia penelitian mengenai pendeteksian atau pengenalan kosa kata bahasa isyarat belum banyak dilakukan, seperti menggunakan model LSTM yang dilakukan oleh Moetia Putri, dkk. [5] dengan menghasilkan nilai akurasi 92%. LSTM dapat digunakan untuk data yang berjenis video, yang dimana pergerakan bahasa isyarat kosa kata dapat disimpan dalam bentuk video.

Maka dari itu penelitian ini akan membuat sistem pengenalan bahasa isyarat yang dapat mendeteksi pergerakan atau dinamis. Diharapkan dengan adanya sistem

pengenalan kosa kata bahasa isyarat ini membantu orang tuli untuk dapat berkomunikasi dengan lingkungannya.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah penulis paparkan maka didapatkan rumusan masalah sebagai berikut :

1. Bagaimana mengenali kosa kata BISINDO dengan menggunakan 2 model LSTM?
2. Berapakah tingkat akurasi keberhasilan pengenalan kosa kata BISINDO?

## 1.3 Tujuan dan Manfaat Penelitian

Adapun tujuan dan manfaat yang dicapai dari penelitian ini adalah sebagai berikut.

### 1.3.1 Tujuan Penelitian

1. Untuk mengenali kosa kata BISINDO dengan menggunakan 2 model LSTM yang berbeda fungsi aktivasi
2. Menghasilkan nilai akurasi mendekati angka 1 dengan menggunakan model LSTM

### 1.3.2 Manfaat Penelitian

1. Dapat diterapkan di aplikasi penerjemah bahasa isyarat dalam mendeteksi bahasa isyarat yang dinamis
2. Membantu peneliti yang lain dalam membandingkan metode pendeteksian bahasa isyarat yang lain dengan penelitian ini.

## 1.4 Batasan Masalah

Pembatasan masalah pada penelitian ini adalah

1. Dalam pembuatan sistem pengenalan kosa kata bahasa isyarat menggunakan model LSTM
2. Data yang digunakan dalam bentuk frame-frame berurutan yang disimpan dalam bentuk array bukan bentuk gambar

3. Terdapat sepuluh kosa kata sehari-hari yang akan dijadikan contoh dalam penelitian ini, seperti : ‘terima kasih’, ‘tolong’, ‘maaf’, ‘nama’, ‘bagaimana’, ‘dimana’, ‘siapa’, ‘sama-sama’, ‘kapan’, dan ‘halo’
4. Menggunakan MediaPipe untuk menentukan *landmark* pada wajah, kedua tangan, dan pose
5. Eksperimen akan dilakukan dengan dua kombinasi fungsi aktivasi

## 1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini ditulis dengan sistematika penulisan sebagai berikut.

1. BAB I PENDAHULUAN berisikan latar belakang dari penelitian mengenai Pengenalan Kosa Kata Bahasa Isyarat dengan Menggunakan Metode LSTM, perumusan masalah, tujuan dan manfaat penelitian, batasan masalah, dan sistematika penulisan.
2. BAB II KAJIAN LITERATUR berisikan mengenai landasan teori mengenai Bahasa Isyarat, RNN, LSTM, TensorFlow, OpenCV, dan Mediapipe. Selain landasan teori, bab ini juga berisi table dan penjelasan dari penelitian terdahulu yang memiliki keterkaitan dengan penelitian ini.
3. BAB III METODOLOGI PENELITIAN berisikan mengenai tahapan penelitian dalam bentuk alur *flowchart* yang diperkuat dengan penjelasan setiap tahapan. Bab 3 juga berisi mengenai rancangan penelitian yang meliputi jenis penelitian, metode analisis, metode pengumpulan data, metode pengujian, metode implementasi dan evaluasi serta lingkungan pengembangan.
4. BAB IV HASIL DAN PEMBAHASAN berisikan penjelasan mengenai proses pengumpulan data, perancangan model, pelatihan dan pengujian model secara bertahap dan melakukan evaluasi terhadap model.
5. BAB V KESIMPULAN DAN SARAN berisikan kesimpulan dari hasil penelitian yang telah dilakukan serta saran untuk penelitian selanjutnya.

# BAB 2
# KAJIAN LITERATUR

Kajian literatur berisi mengenai pemaparan teori-teori yang dibutuhkan dalam penelitian ini dan pemaparan penelitian-penelitian sebelumnya yang memiliki keterkaitan topik dengan penelitian ini.

## 2.1 Landasan Teori

Landasan teori akan menjelaskan teori yang dijadikan sebagai acuan dari penelitian ini meliputi penjelasan tentang teori Bahasa Isyarat, RNN, LSTM, TensorFlow, OpenCV, dan Mediapipe.

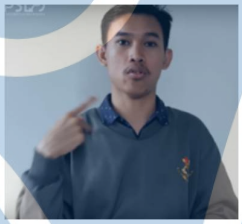
### 2.1.1 Bahasa Isyarat

Bahasa isyarat adalah alat komunikasi non-verbal dengan menggunakan gerakan tubuh, ekspresi wajah, dan gerak bibir [6]. Bahasa isyarat menjadi alat komunikasi yang digunakan oleh komunitas tuli (kehilangan fungsi mendengar). Setiap negara memiliki standar bahasa isyarat sendiri, seperti di Amerika ada ASL (American Sign Language), di Indonesia ada SIBI (Sistem Isyarat Bahasa Indonesia) dan BISINDO (Bahasa Isyarat Bahasa Indonesia) [6].

BISINDO adalah singkatan dari Bahasa Isyarat Indonesia, yaitu cara komunikasi yang muncul secara alamiah dari kalangan komunitas tuli [7]. BISINDO adalah hasil dari Kongres Nasional keenam Gerkatin (Gerakan untuk Kesejahteraan Tunarungu Indonesia) pada tahun 2002 yang diselenggarakan di Bali [7]. Pada tahun 2009 terbentuk PUSBISINDO (Pusat Bahasa Isyarat Indonesia) oleh Gerkatin, pusat ini berfungsi untuk memberikan kelas-kelas BISINDO untuk warga Indonesia yang ingin mempelajari BISINDO [7].

BISINDO juga diterapkan pada kosa kata. Berbeda dengan alfabet yang bahasa isyarat tidak ada pergerakan tangan (statis), pada kosakata BISINDO terdapat pergerakan tangan dan beberapa juga menggunakan ekspresi. Berikut adalah 10 kosa kata sehari-hari.

Tabel 2.1 Daftar 10 kosakata

| No | Kosa Kata | Gambar | Keterangan |
|---|---|---|---|
| 1 | Terima kasih |  Gambar 2.1 Bahasa Isyarat Terima kasih [8] | Tangan kanan terbuka menghadap kedalam sambil menyentuh mulut/dagu lalu diayunkan ke depan. |
| 2 | Tolong |  Gambar 2.2 Bahasa Isyarat Tolong [9] | Jari jempol, telunjuk dan tengah terbuka dan sisa jari tangan kanan(salah satu tangan) yang lain tertutup, lalu meletakkan di pundak pada sisi yang sama, sambil tangan kanan digerakan maju mundur |
| 3. | Maaf |  Gambar 2.3 Bahasa Isyarat Maaf [10] | Jari jempol dan telunjuk tangan kanan disatukan, menyentuh pipi sambil digerakan ke depan. |
| 4. | Nama |   Gambar 2.4 Bahasa Isyarat Nama [11] | telapak tangan kiri terbuka, lalu pada tangan kanan menyatukan antara jempol dan telunjuk, letekkan tangan kanan di ujung pergelangan tangan kiri lalu di tarik ke arah ujung jari kiri |

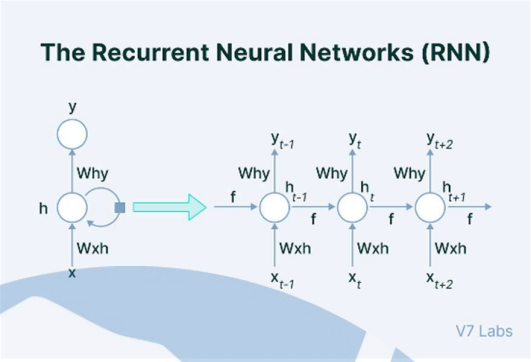
| No | Kosa Kata | Gambar | Keterangan |
|---|---|---|---|
| 5. | Bagaimana |  Gambar 2.5 Bahasa Isyarat Bagaimana [12] | Kedua tangan terbuka, telapak tangan menghadap ke atas, lalu digerakan dari dalam badan ke arah luar badan |
| 6. | Dimana |  Gambar 2.6 Bahasa Isyarat Dimana [12] | telunjuk membentuk seperti bilangan 1, lalu digerakan ke kanan dan kiri |
| 7. | Siapa |  Gambar 2.7 Bahasa Isyarat Siapa [13] | Menaruh jempol tangan kanan di bawah dagu, lalu digerakan ke arah depan atau bawah |
| 8. | Sama-sama |  Gambar 2.8 Bahasa Isyarat Sama-sama [14] | Tangan kanan sejajar dada dengan sudut lancip, tiga jari kecuali kelingking dan jempol ditekuk kedalam, seperti bentuk metal, lalu gerakan tangan maju mundur. |

| No | Kosa Kata | Gambar | Keterangan |
|---|---|---|---|
| 9. | Kapan |  Gambar 2.9 Bahasa Isyarat Kapan [15] | Tangan kanan sejajar bahu, lalu jari jari tangan kanan disatukan seperti bentuk paruh burung yang menghadap ke atas, tangan melakukan pergerakan buka tutup. |
| 10. | Halo |  Gambar 2.10 Bahasa Isyarat Halo [16] | Tangan kanan atau bisa juga tangan kiri, tangan terbuka, telapak tangan menghadap depan, lalu tangan dilambaikan ke kanan dan ke kiri. |

### 2.1.2 Recurrent Neural Network (RNN)

*Artificial Neural Network* (ANN) atau Jaringan Syaraf Tiruan adalah *deep learning* yang terinspirasi dari sistem saraf manusia. Salah satu arsitektur dari ANN adalah *Recurrent Neural Network* (RNN) atau jaringan syaraf berulang. RNN memiliki arsitektur dengan sedikit lapisan (*layer*) namun susunan arsitektur RNN rumit karena adanya *loop* [17]. RNN cocok digunakan untuk data *sequence* karena adanya aliran koneksi mundur (*loop*). Contoh data *sequence* seperti teks, video, suara, dan data *time series*. Koneksi *loop* digunakan untuk menyimpan informasi atau memori dari masa lalu [18].

Arsitektur RNN terdiri dari 3 *state*/*layer*, yaitu *input state* (x), *hidden state* (h), dan *output state* (y). Jika memasukan data pada x di segmen waktu tertentu t maka disebut $x_t$. Pada proses di *hidden state* terdapat fungsi aktivasi, yang biasanya berbentuk fungsi tangen hiperbolik atau dikenal dengan *tanh*. Selain *tanh* biasanya fungsi aktivasi yang digunakan adalah *rectified linear unit* (ReLU).




Gambar 2.11 Arsitektur RNN

[19]

*Hidden state/layer* bertugas sebagai memori yang menyimpan hasil dari pemrosesan. Nilai *hidden state/layer* saat ini ($h_t$) didapatkan dari proses penjumlah antara *hidden state* sebelumnya ($h_{t-1}$) yang dikali oleh *weight* dari *hidden layer* sebelumnya ($w_{hh}$), dengan *input state* sekarang ($x_t$) yang dikali oleh *weight* dari input ke hidden ($w_{xh}$), lalu dijumlahkah kan dengan bias hidden ($b_h$). Setelah itu hasilnya dikalikan menggunakan fungsi aktivasi *tanh*.

$$h_t = \tanh(w_{hh} \cdot h_{t-1} + w_{xh} \cdot x_t + b_h) \quad (1)$$

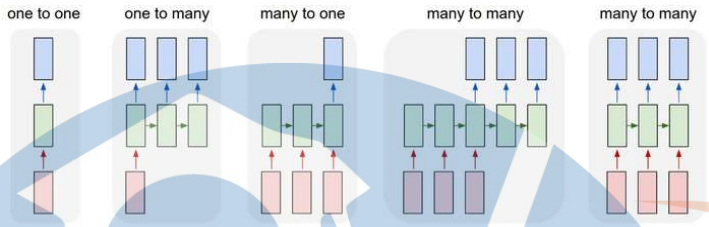
Output didapat setelah *hidden layer* sudah diketahui. Untuk dapat mendapatkan output dengan melakukan perkalian antara *weight* dari *output* ($w_{ho}$) dan *hidden layer* saat ini ($h_t$) lalu dijumlah dengan bias output ($b_h$). Lalu hasilnya dikalikan menggunakan sebuah fungsi aktivasi (f).

$$o_t = \mathrm{f}(w_{ho} \cdot h_t + b_o) \quad (2)$$

RNN memiliki beberapa variasi arsitektur sebagai berikut .

a. **arsitektur *one to many*** adalah ketika input satu, menghasilkan output yang banyak atau berupa *sequence*, contohnya adalah *image captioning*.
b. **arsitektur *many to one*** adalah ketika input berupa *sequence* menghasilkan satu output, contohnya adalah dalam menganalisis sebuah sentimen.
c. **arsitektur *many to many*** adalah ketika input dan outputnya sama sama berupa *sequence*, namun jumlahnya tidak sama, contohnya dalam mesin penerjemah.

d. **arsitektur *many to many*** adalah ketika input dan outputnya sama-sama berupa *sequence*, dan jumlah outputnya akan sama dengan yang diinput contohnya adalah klasifikasi video yang melabeli setia frame pada video

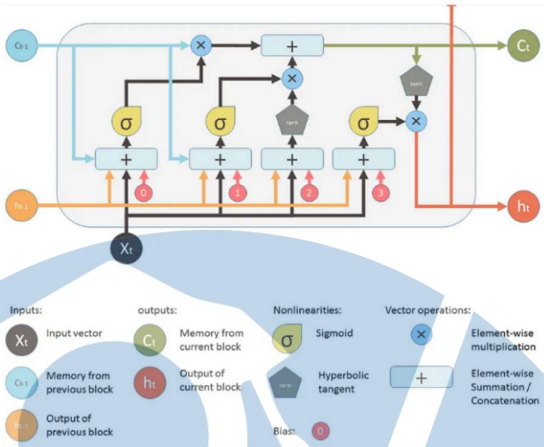



Gambar 2.12 Variasi arsitektur RNN [20]

RNN memiliki sebuah permasalah, yaitu *Gradient Vanishing Problem* [21]. Salah satu modifikasi RNN yang dapat memecahkan permasalahan tersebut adalah *Long Short Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) [21]. Selain LSTM dan GRU, RNN memiliki beberapa variasi lain, seperti *Bidirectional LSTM* (BiLSTM) [18].

### 2.1.3 *Long Short Term Memory* (LSTM)

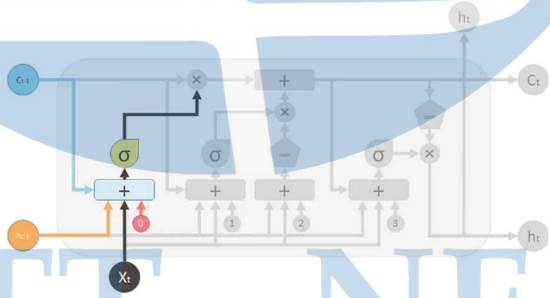
LTSM pertama kali ditemukan oleh Sepp Honchreiter dan Jurgen Schmidhuber pada tahun 1997. LSTM merupakan salah satu solusi mengenai *Gradien Vanishing Problem* yang dialami oleh RNN [21]. Pada RNN tidak mampu memproses *time step* yang panjang atau banyak, sedangkan di LSTM mampu memproses lebih dari 1000 *time step* [22]. Penggunaan LSTM dapat digunakan dalam data sekuens, salah satunya memproses video.




Gambar 2.13 Arsitektur LSTM

[23]

Arsitektur LSTM menggunakan topologi network, yang memiliki *input layer*, *hidden layer*, dan *output layer* [24]. Pada arsitektur LSTM terdapat dua komponen utama pada *hidden layer* yang membedakan LSTM dengan RNN, yaitu *memory cells* dan *gate unit* [24]. *Memory cells* digunakan sebagai wadah penyimpanan memori atau informasi [24]. *Gate unit* pada LSTM ada 3, yaitu *forget gate*, *input gate*, dan *output gate.* Masing-masing *gate* memiliki tugasnya masing-masing.



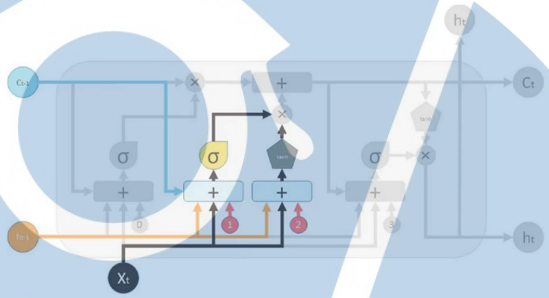
Gambar 2.14 Bagian yang berwarna adalah Forget Gate

[23]

*Gate* pertama yang disebut sebagai *forget gate*, berisi ketiga inputan,fungsi sigmoid sebagai aktivasi yang outputnya dari 0 sampai 1, dan *bias*. *Forget gate* berfungsi untuk mengontrol memori yang lama ($C_{t-1}$) akan disimpan atau dibuang. Hasil *forget gate* didapat *hidden state* sebelumnya ($h_{t-1}$) dengan input saat ini ($x_t$)

dan bias yang kemudian diproses menggunakan fungsi sigmoid. Jika hasil *forget gate* mendekati angka 1, maka sebagian besar atau seluruh memori yang lama akan disimpan, namun jika mendekati angka 0, maka sebagian besar atau seluruh memori yang lama akan dibuang. Berikut adalah formula matematika dari *forget gate*.

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$



Gambar 2.15 Bagian berwarna adalah input gate

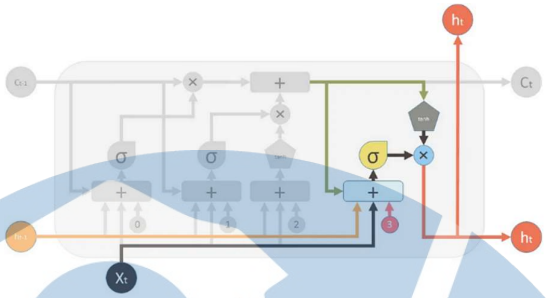
[23]

*Gate* kedua seperti gambar 2.15 adalah *input gate* yang berfungsi untuk menentukan informasi yang akan ditambahkan pada $C_t$ [17] . Pada *input gate* ada dua inputan, yaitu *hidden state* sebelumnya ($h_{t-1}$) dan input baru ($x_t$) Selain inputan, ada dua fungsi aktivasi, yaitu fungsi sigmoid dan fungsi tanh. Setiap fungsi tersebut melakukan proses aktivasi untuk $h_{t-1}$ dan $x_t$. *Hidden state* sebelumnya ($h_{t-1}$) dan input baru ($x_t$) diproses dengan fungsi sigmoid untuk menghasilkan nilai antara 0 sampai 1, jika hasilnya mendekati 0, maka informasi atau memori tidak akan diteruskan, namun jika hasilnya mendekati 1 maka memori akan diteruskan [17]. Fungsi *tanh* untuk menentukan akurasi informasi atau memori, yang hasilnya dimulai dari -1 sampai 1. Setelah hasil *input gate* dan hasil fungsi tanh diketahui, proses selanjutnya menggabungkan antara hasil *forget gate* dan $C_{t-1}$ serta *input gate* dan fungsi *tanh*. Hasil penggabungan tersebut akan menghasilkan memori baru $C_t$. Berikut adalah formulasi matematika pada *input gate* yang menggunakan fungsi sigmoid.

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\widetilde{C_t} = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \quad (6)$$



Gambar 2.16 Bagian bewarna adalah output gate [23]

*Gate* ketiga adalah *output gate*, yang berfungsi untuk menentukan seberapa besar nilai dari *cell state* pada *hidden state* yang baru ($h_t$) [17]. *Output gate* diperoleh melalui input saat ini ($x_t$) dan *hidden state* sebelumnya ($h_{t-1}$) yang diproses dengan sigmoid. Sedangkan untuk *hidden state* yang baru ($h_t$) didapat dari hasil *output gate* dan fungsi *tanh* yang memproses *Cell state* yang baru ($C_t$).

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * \tanh C_t \quad (8)$$

### 2.1.1 Tensorflow

TensorFlow adalah platform *end-to-end* yang dikembang oleh Google. TensorFlow dirancang untuk memudahkan dalam membuat sebuah model *machine learning* yang bisa digunakan di berbagai *device*, seperti *desktop*, *web*, *mobile*, dan *cloud*. Tensorflow menawarkan alat yang dapat digunakan pada data seperti untuk menggabungkan data, membersihkan data, dan memproses data [25].

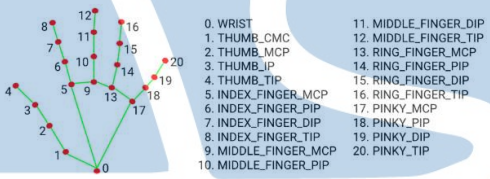
Ada berbagai macam *library* dan *extension* disediakan oleh TensorFlow. Terdapat API (*Application Programming Interface*) tingkat tinggi (*high-level*) yang biasa digunakan menggunakan Tensorflow, yaitu Keras. Penggunaan Keras lebih memfokuskan pada model-model *deep learning* [26].

### 2.1.4 OpenCV

OpenCV (Open Source Computer Vision Library) adalah produk berlisensi Apache 2 yang berisi *library* atau pustaka *open source* untuk *computer vision* dan *machine learning* [27]. *Library* pada OpenCV memiliki sekitar 2500 algoritma yang telah optimalisasi [27]. Penggunaan OpenCV bisa diimplementasikan dalam pengenalan dan pendeteksian waja, mengidentifikasi objek, mengklasifikasi pergerakan manusia melalui video, dan masih banyak lagi.

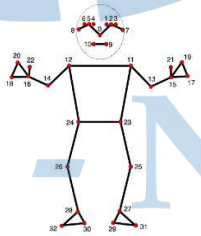
### 2.1.5 MediaPipe

MediaPipe adalah sebuah *framework* dan solusi yang digunakan untuk menentukan bentuk dan orientasi dengan mengekstraksi *keypoint* pada wajah, tangan, dan pose [18]. MediaPipe memiliki banyak solusi, salah satunya adalah MediaPipe Holistic yang menggabungkan model deteksi wajah, tangan, dan pose [28]. Pada *holistik landmark detection* terdapat 543 *landmarks* yang terdiri dari 33 *landmarks* pada pose, 468 *landmarks* pada wajah, dan 21 *landmarks* pada masing-masing tangan [29].




Gambar 2.17 Hand landmark

[30]



Gambar 2.18 Pose landmark

[31]

Gambar 2.19 Face landmark

[32]

### 2.1.6 Confusion Matrix

*Confusion Matrix* adalah metode kuantitatif yang biasa digunakan dalam *machine learning* [33]. Pada *Confusion Matrix* terdapat 4 indikator pengukuran yaitu *true negative* (TN), *false negative* (FN), *true positive* (TP), dan *false positive* (FP) [33]. Berikut adalah tabel *Confusion Matrix.*

Tabel 2.2 Confusion Matrix

| | | Prediksi | |
|---|---|---|---|
| | | Positif | Negatif |
| Aktual | Positif | TP | FN |
| | Negatif | FP | TN |

Dari tabel 2.2 dapat menghasilkan beberapa nilai, seperti nilai akurasi, nilai presisi, *recall*, dan skor F1 [15]. Berikut formula untk mendapatkan masing masing nilai.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

## 2.2 Penelitian Terkait

Penulisan penelitian ini tak lepas dari berbagai inspirasi lain dari penelitian-penelitian sebelumnya yang memiliki topik dan latar belakang yang sama.

Tabel 2.3 Penelitian Terkait

| No. | Nama Peneliti (Tahun) | Judul | Metode | Akurasi |
|---|---|---|---|---|
| 1. | Hamdan Ainul Atman Al Faruqi (2019) [3] | Pengenalan Bahasa Isyarat Bisindo Menggunakan Metode HOG Dan SVM Dalam Pengolahan Citra | HOG dan SVM | 81,25% |
| 2. | Husna Moetia Putri, Fadlisyah, Wahyu Fuadi (2022) [5] | Pendeteksian Bahasa Isyarat Indonesia Secara *Real-Time* Menggunakan *Long Short-Term Memory* (LSTM) | Bidirectional LSTM, 1 *layer* LSTM, dan 2 *layer* LSTM, Mediapipe Holistic | 92% |
| 3 | Alvaro A. Teran-Quezada, Victor Lopez-Cabrera, Jose Carlos Rangel, dan Javier E. Sanchez-Galan (2024) [33] | Sign-to-Text Translation from Panamanian Sign Language to Spanish in Continuous Capture Mode with Deep Neural Networks | 3 *layer* LSTM, Mediapipe Holistic | 98,8% |
| 4. | Devina Yolanda, Kartika Gunadi, Endang Setyati (2020) [4] | Pengenalan Alfabet Bahasa Isyarat Tangan Secara *Real-Time* dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network | CNN dan RNN | 60,58% |

| No. | Nama Peneliti (Tahun) | Judul | Metode | Akurasi |
|---|---|---|---|---|
| 5. | Gulpi Q. O. P., Fathorazi Nur. F., Puji Kurnia S. (2022) [34] | Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode *Deep Gated Recurrent Unit* (GRU) | RNN dan GRU | 88% |

Penelitian yang dilakukan oleh Hamdan A. A. Al Faruqi (2019) pada skripsinya yang berjudul "Pengenalan Bahasa Isyarat Bisindo Menggunakan Metode HOG Dan SVM Dalam Pengolahan Citra" [3]. Data citra yang berhasil dikumpulkan adalah 432, dengan rincian setiap huruf dari A sampai Z memiliki 18 citra, kecuali huruf J dan R tidak diikut sertakan. Data yang digunakan untuk *training* sebanyak 14 citra untuk setiap huruf, yang ditotalkan memiliki 336 citra. Sedangkan pada Data *testing* memiliki 4 citra untuk setiap huruf, yang ditotalkan menjadi 96 citra. *Testing* menghasilkan 48 citra yang dites benar, dan 48 lainnya yang dites salah. Dari hasil tersebut tingkat akurasinya sebesar 81.25%

Penelitian yang dilakukan oleh Husna Moetia Putri, Fadlisyah, dan Wahyu Fuadi (2022) "Pendeteksian Bahasa Isyarat Indonesia Secara *Real-Time* Menggunakan *Long Short-Term Memory* (LSTM)" [5]. Pada penelitian dilakukan dengan mengumpulkan 30 kosa kata, yang masing-masing kosakata memiliki 100 video dan setiap video memiliki 30 frame. Untuk menentukan model arsitektur LSTM yang lebih efektif dilakukan perbandingan dengan 3 jenis arsitektur, yaitu 1 *layer* LSTM, 2 *layer* LSTM, dan Bidirectional LSTM. Pada tahap pengujian pada dilakukan dengan dua jenis kelas yang berbeda , yaitu menggunakan 2 kelas (dua kosa kata) dan 30 kelas (10 kosa kata). Selain itu menggunakan 3 *batch size*, yaitu 32, 64, dan 128. Menggunakan 4 *hidden layer*, yaitu 32, 64, 128, dan 256. Menggunakan 3 epoch yang berbeda, yaitu 100, 500, dan 1000. Dari hasil pengujian pada 10 kelas yang memiliki tingkat akurasi paling tinggi dengan skor akurasinya

berada pada angka 92% adalah model yang menggunakan arsitektur Bidirectional LSTM dengan *batch size* 64, *hidden layer* 46, dan epoch 1000. Sedangkan pada pengujian 30 kelas yang memiliki tingkat akurasi paling tinggi dengan hasil akurasi berada pada angka 65 % adalah model yang menggunakan arsitektur Bidirectional LSTM dengan *batch size* 64, *hidden layer* 64, dan epoch 500.

Penelitian yang dilakukan oleh Alvaro A. Teran-Quezeda, dkk. (2024) " Sign-to-Text Translation from Panamanian Sign Language to Spanish in Continuous Capture Mode with Deep Neural Networks" [33]. Penelitian ini mengenali kosa kata dari Bahasa Isyarat Panamanian bentuk text Bahasa Spanyol. Kosa kata Bahasa Isyarat Panamanian merupakan bahasa isyarat yang dinamis atau memiliki pergerakan. Pada penelitian ini melakukan 2 percobaan dengan jumlah kosakata yang digunakan. Percobaan pertama, menggunakan 5 kosakata dalam bahasa spanyol, menggunakan 625 video dengan 30 frame setiap video, setiap kosakata ini memiliki 125 video, dan 20% dari total data ini digunakan sebagai *testing*. Percobaan kedua, menggunakan 3 kosa kata, menggunakan 375 video, dan 12 % digunakan untuk *testing*. Model LSTM yang dipakai menggunakan LSTM 3 layer, yaitu 64, 128, dan 64 neuron, dengan dense layer 64 dan 32 neuron, dan fungsi aktivasi yang digunakan adalah ReLu. Pada percobaan pertama menggunakan epoch 160 menghasilkan nilai akurasi sebesar 0,958 (95,8%). Sedangkan di percobaan kedua menggunakan epoch 46 menghasilkan nilai akurasi 0,988 (98,8%).

Penelitian yang dilakukan oleh Devina Yolanda, Kartika Gunadi, dan Endang Setyadi (2020) "Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real-Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural" [4]. Menggunakan data 2582 video alfabet bahasa isyarat. Video tersebut diubah kedalam bentuk *frame-frame*, yang berjumlah 10 *frame*. Pada penelitian ini memiliki 3 tahap pengujian, yaitu dengan menambahkan jumlah set layer pada CNN, *learning rate dan number of steps* (jumlah epoch), dan *penambahan dropout layer dan normalization layer*. Pengujian pertama ini melakukan perbandingan 3 model dengan jumlah set layer yang berbeda, yaitu 3 set layer, 4 set layer, dan 5 set layer. Satu set layer terdiri dari dua *convolutional layer*, *activation layer* ReLU, dan *max pooling*. Hasil pengujian pertama ketiga model set layer masih mengalami

*overfitting*. Pengujian kedua yaitu dengan *learning rate* dan jumlah epoch, nilai rate yang diuji adalah le-4 dan le-5 dan jumlah epoch dimulai dari 1 sampai 100. Hasilnya pengujian kedua adalah selama berjalan epoch nilai rate le-5 memiliki nilai loss yang lebih stabil bila dibandingkan le-4, namun semakin bertambah epoch, semakin tinggi nilai lossnya, dan diputuskan epoch hanya dimulai dari 1 sampai 50. Pengujian ketiga dilakukan dengan model yang sebelumnya dengan penambahan nilai *dropout* dan *batch*. Penambahan nilai dropout dilakukan dengan 2 model, yang pertama nilai *dropout* 0.5 dan kombinasi 0.5, 0.25, dan 0.25. Hasil pengujian ketiga ini adalah penggunaan nilai *dropout* kombinasi yang setiap penambah epochnya memiliki nilai loss yang semakin kecil dan nilai akurasi semakin naik. Dari 3 pengujian yang dilakukan, dilakukan pengujian menggunakan *testing* dengan model 4 set layer, nilai rate le-5, dan nilai *dropout* yang kombinasi menghasilkan nilai akurasi sebesar 60.58%.

Penelitian yang dilakukan oleh Gulpi Qorik O. P., dkk. (2022) "Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode Deep Gate Recurrent Unit (GRU)" [34]. Data diambil dengan pengambilan langsung menggunakan *smartphone*. menghasilkan satu video yang berisi 3 gerakan bahasa isyarat sehari-hari, dengan dengan resolusi 1280 x 720, berformat mp4, dengan durasi 39 menit. Dilakukan pemotongan pada video tersebut, menghasilkan 81 video dengan resolusi setiap video adalah 704 x 384, dan video-video tersebut diubah formatnya menjadi AVI. Dari 81 video dibagi untuk *training* 45 video dan *testing* 36 video. Pengujian dilakukan dengan epoch 10 dengan kombinasi pencarian model menggunakan batch_size 8, nb_frames 5, split_val 0.2, dan momentum 0.2 menghasilkan nilai akurasi 88%.

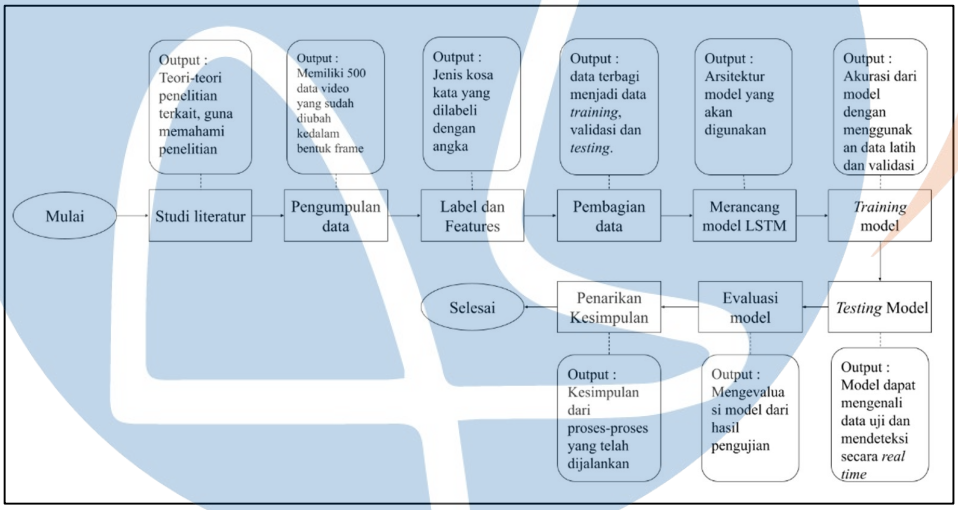
Pada penelitian yang akan dilakukan oleh peneliti memiliki persamaan topik dengan penelitian yang dilakukan oleh [3], [4], [5], [33], dan [34], yaitu penggunaan *machine learning* pada pengelan Bahasa Isyarat. Pada penelitian ini menggunakan kosa kata yang akan dideteksi, seperti dalam penelitian [5] dan [33]. Jenis data yang akan digunakan pada penelitian adalah video yang diubah menjadi dalam bentuk *frame* seperti yang dilakukan oleh [4], [5], dan [33]. Pada penelitian ini, peneliti akan menggunakan metode LSTM dan menggunakan MediaPipe, seperti yang dilakukan oleh [33] dan [5].

# BAB 3
# Metodologi Penelitian

Metodologi penelitian berisi mengenai tahapan yang akan dilakukan pada penelitian serta mengenai rancangan penelitian mulai dari jenis penelitian ini, metode analisis yang digunakan dalam penelitian ini, metode pengumpulan data, metode pengujian, metode implementasi dan evaluasi, dan lingkungan pengembangan.

## 3.1 Tahapan Penelitian

Penelitian ini dilakukan dalam beberapa tahapan. Berikut tahapan penelitian yang dibuat dalam bentuk diagram alir.

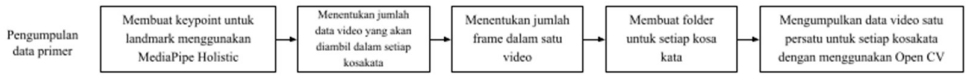



Gambar 3.1 Tahapan Penelitian

### 3.1.1 Studi Literatur

Kegiatan studi literatur pada penelitian ini dilakukan untuk mencari penelitian-penelitian sebelumnya yang serupa atau yang terkait sebagai rujukan dan landasan untuk melakukan penelitian tugas akhir ini. Dari kegiatan ini menjadikan penelitian dari Hamdan [3] menjadi landasan penelitian ini.

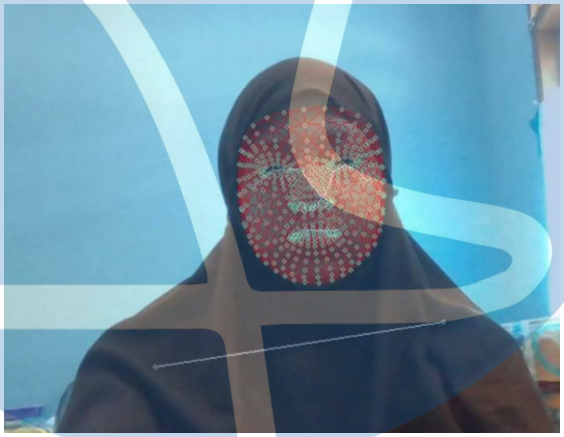
### 3.1.2 Pengumpulan Data




Gambar 3.2 Alur pengumpulan data

Data yang dikumpulkan pada penelitian ini adalah data video. Data yang akan digunakan adalah data primer. Jumlah data yang akan dikumpulkan dari data adalah 500 data. Proses pengumpulan data dilakukan menggunakan bahasa pemrogram Python dengan menggunakan *library* OpenCV. Berikut adalah proses pengumpulan data seperti pada gambar 3.2.

a. Membuat landmark pada tangan kanan, tangan kiri, wajah dan pose menggunakan MediaPipe Holistic



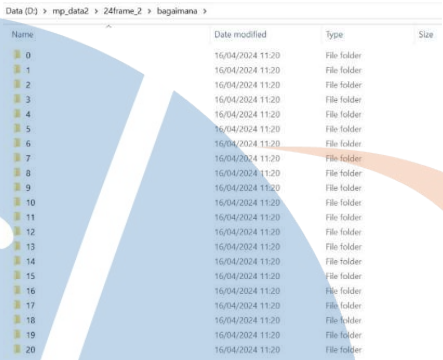
Gambar 3.3 Landmark pada bagian wajah, dan sedikit landmark pose

b. Menentukan kosa kata apa saja yang akan menjadi objek, jumlah video yang akan dikumpulkan pada setiap kosa kata, dan jumlah *frame* pada setiap video. Pada penelitian ini jumlah video yang dikumpulkan adalah 50 setiap kosa kata, dan setiap video memiliki 24 *frame*

c. Membuat folder untuk penyimpanan data menggunakan pemrograman python agar bisa otomatis buat sesuai jumlah yang

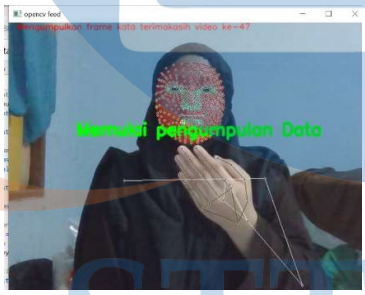
diinginkan. Pada gambar 23 adalah folder-folder untuk 10 kosa kata. Pada gambar 24 adalah contoh isi dari folder kosa kata yang sebelumnya, isinya ada 50 folder, yang nantinya akan berisi frame-frame.


Gambar 3.4 Folder untuk 10 kosa kata

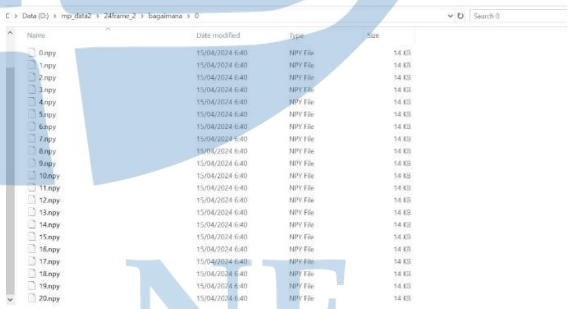



Gambar 3.5 Isi folder dari folder 'bagaimana'

d. Pada bagian pengumpulan data menggunakan OpenCV dan menerapkan landmark yang diawal. Video akan tersimpan pada folder yang sudah disiapkan. Datanya tersimpan dalam bentuk *frame*, format datanya adalah .npy




Gambar 3.6 Proses pengambilan data untuk kosa kata 'terima kasih'



Gambar 3.7 Ini adalah frame-frame, ada 24 frame yang telah disimpan

### 3.1.3 Label dan *Features*

Label menjadi target atau variabel terikat yang nanti diprediksi. Sedangkan *Features* adalah variabel bebas yang menjadi prediktor.

### 3.1.4 Membagi Data

Data yang telah dikumpulkan pada penelitian ini akan dibagi menjadi 3 bagian, yaitu 95% data *train* (latih), 2,5% data test, dan 2,5% data validasi. Data latih dan data validasi digunakan dalam proses *training* atau melatih model, yang berfungsi untuk mengukur kinerja model. Data test digunakan untuk menguji model yang sudah dilatih.

### 3.1.5 Merancang Model

Perancangan model adalah penentuan arsitektur dari model yang akan digunakan *training* dan *testing*. Arsitekturnya seperti model yang akan digunakan berjenis apa, berapa layer pada aristektur model tersebut, dan neuron yang akan digunakan di setiap layernya. Setelah model dirancang akan di *compile* dengan *metrics*, *loss*, dan *optimizer* yang akan ditentukan. Penelitian ini akan menggunakan 2 model dengan arsitektur yang sama namun menggunakan kombinasi fungsi aktivasi yang berbeda. Penelitian ini akan menggunakan *metrics* yang *categorical_accuracy*, *loss* yang *cetegorical_cossentropy*, dan *optimizer* menggunakan Adam.

### 3.1.6 *Training* Model

*Training* Model adalah proses melatih model yang sudah dirancang dengan menggunakan data *train* dan data validasi untuk mengukur kinerja model dan membandingkan akurasi model menggunakan data *train* dengan model menggunakan data validasi. Proses *training* model ini juga akan ditentukan epoch yang digunakan, penelitian ini akan menggunakan epoch 46 dalam proses *training* kedua model.

### 3.1.7 *Testing* Model

*Testing* model adalah melakukan pengujian terhadap model yang sudah dilatih dengan dua cara, yaitu menguji dengan menggunakan data test dan menguji secara *real time*. *Testing* dilakukan untuk melihat apakah model berhasil mendeteksi kosa kata dari data test dan dari *real time* dan akan dilakukan penghitungan akurasi.

### 3.1.8 Evaluasi

Mengevaluasi hasil *training* dan *testing* untuk melihat kekurangan dari kedua model, serta membandingkan kedua model tersebut dari segi keoptimalan model dalam bekerja.

### 3.1.9 Kesimpulan

Memberikan kesimpulan mulai dari studi pustaka sampai evaluasi serta akan menjawab rumusan masalah pada BAB 1.

## 3.2 Rancangan Penelitian

### 3.2.1 Jenis Penelitian

Penelitian ini bertujuan untuk mengetahui tingkat akurasi sebuah model LSTM untuk, untuk mengetahui model LSTM seperti apa yang dapat menghasilkan nilai akurasi mendekati atau mencapai angka 1 (satu), maka dilakukan uji coba atau eksperimen. Dari penjelasan tujuan dari penelitian ini, maka disimpulkan bahwa jenis penelitian ini adalah eksperimental. Penelitian eksperimental adalah salah satu dari metode penelitian kuantitatif yang dilakukan untuk membandingkan variabel independen (bebas) yang sudah diubah dan yang belum diubah pada sebuah penelitian[35].

### 3.2.2 Metode Analisis

Analisis data pada penelitian ini adalah analisis secara kuantitatif. Data pada penelitian ini adalah data sekuens, yaitu video. Video ini akan diubah bentuk menjadi *frame-frame*, lalu melalui *programming* akan diubah dalam bentuk *array* yang berisi angka-angka. Dalam model LSTM terdapat perhitungan matematika, yang hasil akhir akan menunjukkan tingkat akurasi dari model tersebut.

### 3.2.3 Metode Pengumpulan Data

Penelitian ini melakukan pengumpulan data untuk menghasilkan informasi atau latar belakang dengan cara sebagai berikut.

a. Studi Pustaka

   Studi pustaka dilakukan untuk menemukan informasi tambahan untuk penelitian ini dan penguat alasan penelitian ini dilakukan.

b. Dokumentasi

Penelitian ini adalah penelitian yang melakukan pengujian sebuah model LSTM. Data yang digunakan dalam pengujian LSTM ini adalah data video yang diubah menjadi *frame-frame*. Dokumentasi dilakukan dalam pengumpulan data tersebut.

### 3.2.4 Metode Pengujian

Metode pengujian pada penelitian ini ada dua yaitu dengan data test dan dengan mendeteksi secara real time. Menguji dengan data test akan menghitung akurasi model dari pengujian tersebut dan menghitung akurasi per kata untuk model. Pengujian secara *real time* dilakukan dengan 2 set pengujian menggunakan model yang terbaik, dari pengujian secara *real time* akan dihitung jumah bahasa isyarat yang berhasil terdeteksi benar sesuai dengan kosakatanya dan jumlah bahasa isyarat yang terdeteksi tidak sesuai dengan kosakatanya, lalu dihitung akurasi dari pengujian tersebut.

### 3.2.5 Metode Implementasi dan Evaluasi

Implementasi pada penelitian ini adalah pengujian model dengan mendeteksi pergerakan bahasa isyarat secara *real time*. Setelah dilakukan pendeteksian tersebut akan dihitung jumlah pergerakan bahasa isyarat yang terdeteksi benar dan yang terdeteksi salah, dari hasil itu akan menghasil skor akurasi terhadap pengujian secara langsung. Dilakukan evaluasi dari proses pengujian secara *real time*, dengan menilai dan menjelaskan hasil dari pengujia tersebut.

### 3.2.6 Lingkungan Pengembangan

Pada penelitian ini perangkat yang digunakan di rumah peneliti dengan menggunakan Laptop Asus VivoBook 14 model A412F dengan spesifikasi sebagai berikut.

a. Processor : Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
b. RAM : 8,00 GB
c. System Type : 64-bit operating system
   OS : Windows 10

Aplikasi yang menunjang penelitian ini adalah Jupyter Notebook yang digunakan untuk menulis program yang akan dibuat. Program akan dibuat menggunakan bahasa Python versi 3.11.7. Menggunakan pustaka Medipipe versi

0.10.11, Tensorflow versi 2.16.1, Matplotlib versi 3.0.8, OpenCV (cv2) versi 4.9.0, dan numpy versi 1.26.4

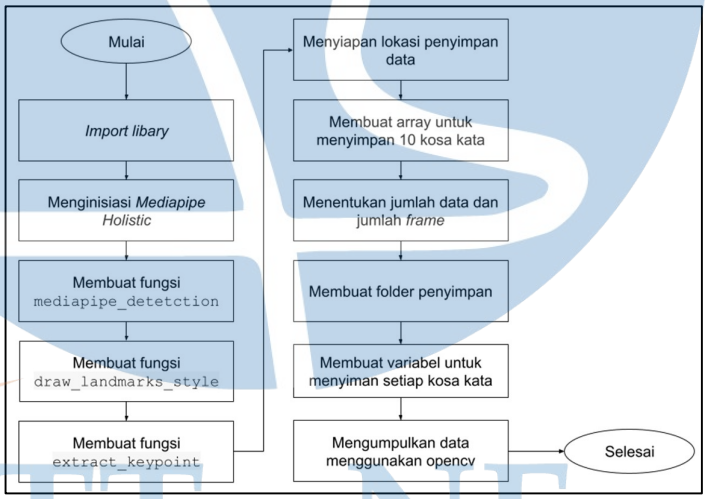
# BAB 4
# Hasil dan Pembahasan

Bab empat memaparkan mengenai pengumpulan dataset, perancangan model, *training* model, *testing* model, dan tahap terakhir adalah evaluasi. Penelitian ini menggunakan tiga file *python*, yaitu file untuk pengumpulan data, file untuk model *ReLu*, dan file untuk model *Tanh*. File untuk model dimulai dari pemanggilan data path, lalu label dan features, pembagian data, *training*, dan *testing*

## 4.1 Dataset

Data dikumpulkan dengan satu cara, yaitu merekam 10 gerakan kosakata BISINDO secara langsung menggunakan kamera Laptop Asus Vivobook 14 model A412F. Kamera atau *webcam* ini merupakan USB2.0 HD UVC WebCam. Pengumpulan data dilakukan oleh satu orang yang bukan penutur asli di dalam ruang yang memiliki penerangan yang cukup dan latar belakang yang tidak ramai. Bagian tubuh yang tertangkap kamera adalah setengah badan.

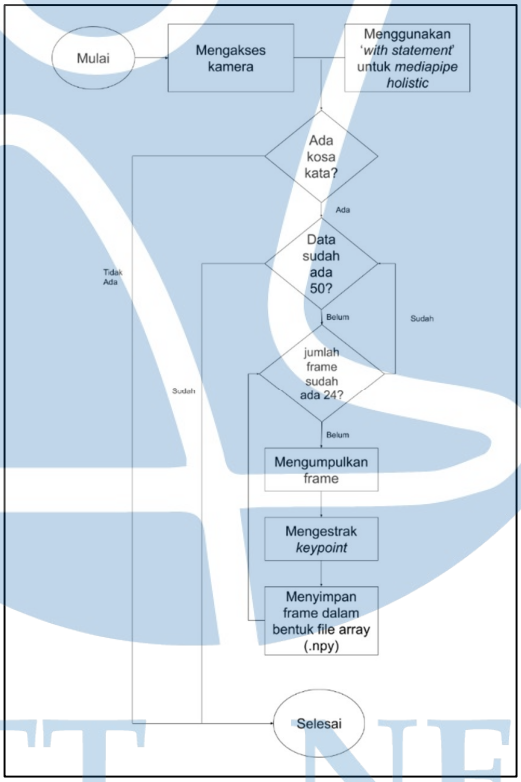
### 4.1.1 Pengumpulan Data




Gambar 4.1 Alur sebelum pengumpulan data

Pengumpulan data menggunakan bahasa pemrograman *python*. *Library* yang digunakan adalah *MediaPipe*, *cv2*, dan *numpy*, os. Melakukan inisiasi variable untuk *MediaPipe holistic* dan *drawing utilities* dengan menggunakan *library MediaPipe*. Membuat fungsi `mediapipe_detection` yang digunakan untuk pendeteksian dan mengubah warna gambar yang awalnya BGR menjadi RGB.

Fungsi `draw_landmarks_style` untuk menyimpan style untuk *landmark* pada wajah, pose, dan tangan kiri serta kanan. Membuat fungsi `extract_keypoint`. Menyiapkan lokasi penyimpanan data dengan variabel `DATA_PATH` menggunakan *library* os. Membuat variabel `actions` untuk menyimpan 10 kosa kata dalam bentuk *array*. Membuat variabel `no_sequence` untuk menyimpan jumlah data per kata, yaitu 50 data. Membuat variabel `sequence_lenght` untuk menyimpan jumlah *frame* per data, yaitu 24 *frame*. Membuat folder untuk setiap data, 1 folder kosa kata berisi 50 folder, dan di dalam folder tersebut diisi dengan 24 file *frame*.




Gambar 4.2 Alur d dalam proses pengumpulan data

Pengambilan data dilakukan perkata, dalam alur di gambar 4.2 ada bagian membuat variabel untuk setiap kosa kata, misal untuk kata ‘terimakasih’ maka variabel untuk kata tersebut adalah `action_terimakasih`. Variabel ini menyinpan ‘terimakasih’ dalam bentuk array. Masuk dalam proses pengambilan data menggunakan *library* opencv, dimulai dengan mengakases kamera laptop dan

menambahkan mediapipe holistic. Program akan memanggil variabel kosa kata yang akan dikumpulkan datanya, misal mengumpulkan kata ‘terimakasih’. Jika ada variabel ‘terimakasih’, selanjutnya program melihat apakah data yang dimiliki sudah berjumlah 50 atau belum, jika belum maka proses dilanjutkan, yaitu melihat *frame*. Program melihat apakah jumlah frame dalam 1 data sudah 24 atau belum, jika belum, program memulai mengumpulkan data pertama untuk kosa kata ‘terimakasih’.

*Frame-frame* pada data pertama tersebut langsung disimpan dalam bentuk array (.npy). Setelah data pertama terkumpul dan tersimpan program melanjutkan mengumpulkan data selanjutnya. Ada jeda setelah mengambil data pertama ke data kedua, yaitu 2,5 detik, hal ini dilakukan sampai dengan data ke 50.

Tabel 4.1 Jumlah data perkosakata yang dikumpulkan

| Kosa kata | Jumlah |
|---|---|
| Terimakasih | 50 |
| Tolong | 50 |
| Maaf | 50 |
| Nama | 50 |
| Bagaimana | 50 |
| Dimana | 50 |
| Siapa | 50 |
| Samasama | 50 |
| Kapan | 50 |
| Halo | 50 |

### 4.1.2 Pembagian Data

Data yang dikumpulkan akan dipecah menggunakan *library* sklearn split menjadi 5% (25 data) data test dan 95% (475) data latih. Setelah dibagi menjadi data test dan data latih, data test akan dibagi lagi secara acak menjadi 13 data test untuk pengujian dan 12 data validasi yang akan digunaka ketika melatih model. Setiap model akan memiliki data latih, data validasi, dan data uji yang berbeda.

## 4.2 Merancang Model LSTM

Model yang dipakai pada penelitian ini adalah model jenis *sequential*. Pada penelitian ini ada dua model dengan parameter kombinasi fungsi aktivasi yang dibuat berbeda, selain itu dibuat sama seperti pada tabel 4.2. Setelah merancangan model, model akan di *compile* dengan menggunakan *metrics* yang *categorical_accuracy*, *loss* yang *cetegorical_cossentropy*, dan *optimizer* menggunakan Adam.

Tabel 4.2 Rincian model yang akan digunakan

| Parameter | Model 1 | Model 2 |
|---|---|---|
| *Hidden layer* | 5 layer | |
| *Neuron* | 64, 64, 64, 32, 10 | |
| *Optimizer* | Adam | |
| *loss* | categorical_crossentropy | |
| *Metrics* | categorical_accuracy | |
| Fungsi aktivasi | Relu, sigmoid, dan softmax | Tanh, sigmoid dan softmax |
| *Epoch* | 46 | 46 |

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm (LSTM) | (None, 24, 64) | 442,112 |
| lstm_1 (LSTM) | (None, 64) | 33,024 |
| dense (Dense) | (None, 64) | 4,160 |
| dense_1 (Dense) | (None, 32) | 2,080 |
| dense_2 (Dense) | (None, 10) | 330 |

**Total params:** 1,445,120 (5.51 MB)
**Trainable params:** 481,706 (1.84 MB)
**Non-trainable params:** 0 (0.00 B)
**Optimizer params:** 963,414 (3.68 MB)

Gambar 4.3 *Summary* arsitektur model

Pada gambar 4.4 adalah susunan layer yang digunakan pada model yang akan di training. Hidden layer pada model ini memiliki 5 layer dengan rincian 2 layer lstm dan 3 layer dense. Pada layer pertama ditambah pada input shape (24, 1662) dan unit output pada layer 1 adalah 64 karena itu output shape (none, 24, 64) maka dari itu layer 1 memiliki 3 dimensi dan pada layer pertama menghasil parameter sebanyak 442.112. Pada layer kedua sampai kelima adalah 2 dimensi. Layer kedua unit outputnya 64 dan menghasilkan parameter sebanyak 33.024. Pada layer ketiga unit outputnya adalah 64 dan menghasilkan parameter 4.160. Pada layer keempat

unit output adalah 32 menghasilkan parameter 2.080. Pada layer kelima, untu output adalah sejumlah kelas/kosa kata yang akan digunakan yaitu 10, menghasilkan paramter 330. Total seluruh parameter model ini adalah 1.445.120.

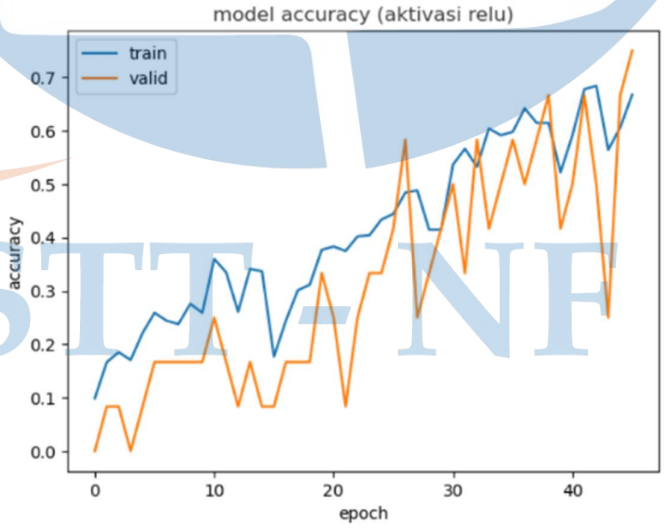
Model 1 aktivasi *ReLu* pada layer 1 sampai 4 bersama dengan aktivasi *sigmoid*, dan di layer 5 terdapat aktivasi *softmax*. Hal yang sama pada Model 2, yaitu layer 1 sampai 4 yang menggunakan aktivasi *tanh* dan sigmoid, pada layer ke 5 menggunakan *softmax*.

## 4.3 *Training* Model

*Training* model adalah proses untuk melatih sebuah model menggunakan data *train* atau data latih. Proses *training* juga menggunakan data validasi untuk melihat kinerja model yang sedang dilatih dengan data latih.

### 4.3.1 Training Model dengan Fungsi Aktivasi *ReLu*

Model pertama yang dilatih adalah model dengan fungsi aktivasi *ReLu*. Model ini dilatih dengan menggunakan epoch 46. Data latih yang digunakan sebanyak 475 data yang dipilih secara acak, dan 12 data validasi untuk mengetes kinerja model dalam proses *training*. Dari hasil *training* terlihat pada gambar 4.2 yang menampilkan nilai akurasi kategori dari data *train* dan data valid.




Gambar 4.4 Grafik akurasi dari model yang menggunakan aktivasi *ReLu*

Pada grafik tersebut terlihat bahwa kedua grafik tersebut terlihat tidak stabil karena adanya perubahan *time per step*. Posisi awal grafik data *train* berada di atas grafik data valid dan posisi akhir grafik data *train* berada di bawah data valid. Pada masa *training* grafik data *valid* hampir selalu di bawah garfik data *train* dan terdapat *gap* yang sedang sampai besar, hanya pada *epoch* ke-27, 30, 33, dan 46 grafik data valid berada di atas seperti pada tabel 4.3. Selain itu ada kalanya garis grafik data *train* dan data valid saling beririsan, nilai akurasinya hampir sama dan hanya ada *gap* kecil sekali, yaitu pada *epoch* ke 26, 30, 31, 36, 39, 41, dan 42 seperti pada tabel 4.3. Pada baris terakhir tabel 4.3 nilai akurasi dari data train adalah 0.6519 dan pada data validasi nilai akurasinya adalah 0.75, dari kedua nilai terlihat bahwa nilai akurasi validasi lebih tinggi, dan memiliki *gap* sebesar 0,1.

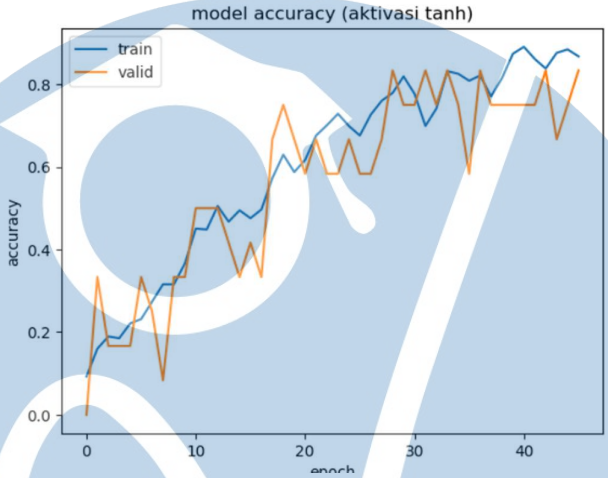
Tabel 4.3 Hasil *training* model dengan aktivasi *ReLu*

| Epoch | | time/step | train categorical accuracy | valid categorical accuracy | train loss | valid loss |
|---|---|---|---|---|---|---|
| Epoch 1/46 | 15/15 | 5s 62ms/step | 0.0799 | 0.00E+00 | 2.369 | 2.3126 |
| Epoch 2/46 | 15/15 | 0s 28ms/step | 0.1434 | 0.0833 | 2.2653 | 2.2205 |
| Epoch 3/46 | 15/15 | 0s 29ms/step | 0.1833 | 0.0833 | 2.264 | 2.3177 |
| Epoch 4/46 | 15/15 | 0s 27ms/step | 0.182 | 0.00E+00 | 2.2461 | 2.2513 |
| Epoch 5/46 | 15/15 | 0s 27ms/step | 0.1721 | 0.0833 | 2.0901 | 2.1649 |
| Epoch 6/46 | 15/15 | 0s 26ms/step | 0.2527 | 0.1667 | 1.9588 | 2.1469 |
| Epoch 7/46 | 15/15 | 0s 26ms/step | 0.2463 | 0.1667 | 1.9419 | 2.133 |
| Epoch 8/46 | 15/15 | 0s 27ms/step | 0.2353 | 0.1667 | 1.8662 | 2.0952 |
| Epoch 9/46 | 15/15 | 0s 29ms/step | 0.2845 | 0.1667 | 1.8097 | 2.039 |
| Epoch 10/46 | 15/15 | 0s 30ms/step | 0.2609 | 0.1667 | 1.7758 | 2.0727 |
| Epoch 11/46 | 15/15 | 0s 26ms/step | 0.3559 | 0.25 | 1.7004 | 2.0227 |
| Epoch 12/46 | 15/15 | 0s 27ms/step | 0.389 | 0.1667 | 1.6342 | 1.9554 |
| Epoch 13/46 | 15/15 | 0s 27ms/step | 0.2604 | 0.0833 | 1.8358 | 2.0542 |
| Epoch 14/46 | 15/15 | 0s 27ms/step | 0.3191 | 0.1667 | 1.7317 | 1.9575 |
| Epoch 15/46 | 15/15 | 0s 27ms/step | 0.2954 | 0.0833 | 1.6689 | 2.1598 |
| Epoch 16/46 | 15/15 | 0s 30ms/step | 0.1744 | 0.0833 | 2.3357 | 2.1793 |
| Epoch 17/46 | 15/15 | 0s 28ms/step | 0.2195 | 0.1667 | 2.0231 | 2.0962 |
| Epoch 18/46 | 15/15 | 0s 28ms/step | 0.2782 | 0.1667 | 1.7819 | 2.0682 |
| Epoch 19/46 | 15/15 | 0s 27ms/step | 0.3179 | 0.1667 | 1.7406 | 2.1483 |

| Epoch | | time/step | train categorical accuracy | valid categorical accuracy | train loss | valid loss |
|---|---|---|---|---|---|---|
| Epoch 20/46 | 15/15 | 0s 26ms/step | 0.3952 | 0.3333 | 1.6333 | 1.8046 |
| Epoch 21/46 | 15/15 | 0s 27ms/step | 0.3748 | 0.25 | 1.6095 | 1.7573 |
| Epoch 22/46 | 15/15 | 0s 27ms/step | 0.3947 | 0.0833 | 1.5132 | 2.0677 |
| Epoch 23/46 | 15/15 | 0s 27ms/step | 0.4379 | 0.25 | 1.5422 | 1.9227 |
| Epoch 24/46 | 15/15 | 0s 27ms/step | 0.4329 | 0.3333 | 1.5428 | 1.7225 |
| Epoch 25/46 | 15/15 | 0s 27ms/step | 0.4083 | 0.3333 | 1.539 | 1.7423 |
| Epoch 26/46 | 15/15 | 0s 28ms/step | 0.4615 | 0.4167 | 1.415 | 1.4394 |
| Epoch 27/46 | 15/15 | 0s 30ms/step | 0.435 | 0.5833 | 1.5145 | 1.5747 |
| Epoch 28/46 | 15/15 | 0s 28ms/step | 0.5217 | 0.25 | 1.3299 | 1.8711 |
| Epoch 29/46 | 15/15 | 0s 28ms/step | 0.4534 | 0.3333 | 1.4781 | 1.5191 |
| Epoch 30/46 | 15/15 | 0s 29ms/step | 0.3904 | 0.4167 | 1.5582 | 1.6629 |
| Epoch 31/46 | 15/15 | 0s 30ms/step | 0.5333 | 0.5 | 1.1938 | 1.4089 |
| Epoch 32/46 | 15/15 | 0s 28ms/step | 0.551 | 0.3333 | 1.2171 | 2.0298 |
| Epoch 33/46 | 15/15 | 1s 32ms/step | 0.494 | 0.5833 | 1.3705 | 1.4051 |
| Epoch 34/46 | 15/15 | 1s 32ms/step | 0.6039 | 0.4167 | 1.1051 | 1.5706 |
| Epoch 35/46 | 15/15 | 1s 32ms/step | 0.5706 | 0.5 | 1.0311 | 1.2332 |
| Epoch 36/46 | 15/15 | 1s 31ms/step | 0.5923 | 0.5833 | 1.0434 | 0.9678 |
| Epoch 37/46 | 15/15 | 1s 33ms/step | 0.626 | 0.5 | 0.9657 | 1.4925 |
| Epoch 38/46 | 15/15 | 1s 32ms/step | 0.6498 | 0.5833 | 0.8683 | 0.9494 |
| Epoch 39/46 | 15/15 | 1s 31ms/step | 0.6829 | 0.6667 | 0.8853 | 1.1319 |
| Epoch 40/46 | 15/15 | 1s 31ms/step | 0.5197 | 0.4167 | 1.2036 | 1.4632 |
| Epoch 41/46 | 15/15 | 1s 33ms/step | 0.5575 | 0.5 | 1.167 | 1.1945 |
| Epoch 42/46 | 15/15 | 1s 32ms/step | 0.692 | 0.6667 | 0.8612 | 0.9797 |
| Epoch 43/46 | 15/15 | 1s 31ms/step | 0.6671 | 0.5 | 0.891 | 0.868 |
| Epoch 44/46 | 15/15 | 1s 32ms/step | 0.6131 | 0.25 | 0.9988 | 1.3782 |
| Epoch 45/46 | 15/15 | 1s 32ms/step | 0.5685 | 0.6667 | 1.3493 | 1.144 |
| Epoch 46/46 | 15/15 | 1s 33ms/step | 0.6519 | 0.75 | 0.8708 | 0.9848 |

### 4.3.2 Training Model dengan Fungsi Aktivasi *Tanh*

Proses *training* model kedua dengan model pertama tidak banyak berbeda kecuali pada fungsi aktivasi yang digunakan. Pada model kedua fungsi aktivasi yang digunakan adalah *tanh*. Fungsi aktivasi *tanh* dan sigmoid ada pada layer satu sampai empat, dan pada layer kelima ada fungsi *softmax*.




Gambar 4.5 Grafik akurasi dari model yang menggunakan aktivasi *Tanh*

Hasil *training* model dengang aktivasi *tanh* ditampilkan dalam bentuk grafik akurasi. Pada gambar 4.5 terdapat dua garis grafik, yaitu grafik data *train* dan grafik data valid. Grafik model ini cenderng stabil karena *time per step* ada direntang 1 detik 37 milidetik sampai 1 detik 65 milidetik, kecuali pada *epoch* pertama. Posisi grafik *train* lebih banyak berada di atas grafik valid dan memiliki *gap* yang sedang dan besar. Grafik valid kadang ada di posisi atas, yang dapat dilihat dari tabel 4.4 epoch ke-2, 6, 11, 18, 19, 29, dan 34. Selain itu ada kalanya garis grafik data *train* dan data valid saling beririsan, nilai akurasinya hampir sama dan hanya ada *gap* kecil sekali, yaitu pada *epoch* ke 9, 10, 13, 16, 43, dan 37 seperti pada tabel 4.4. Pada baris terakhir tabel 4.4 nilai akurasi dari data train adalah 0.8452 dan pada data validasi nilai akurasinya adalah 0.8333, dari kedua nilai terlihat bahwa nilai akurasi *train* lebih tinggi, namun perbedaan antara keduanya hanya 0.01.

Tabel 4.4 Hasil *training* model dengan aktivasi *Tanh*

| Epoch | | time/step | train categorical accuracy | valid categorical accuracy | train loss | valid loss |
|---|---|---|---|---|---|---|
| Epoch 1/46 | 15/15 | 7s 89ms/step | 0.0833 | 0.00E+00 | 2.3651 | 2.3078 |

| Epoch | | time/step | train categorical accuracy | valid categorical accuracy | train loss | valid loss |
|---|---|---|---|---|---|---|
| Epoch 2/46 | 15/15 | 1s 43ms/step | 0.1481 | 0.3333 | 2.2735 | 2.1138 |
| Epoch 3/46 | 15/15 | 1s 37ms/step | 0.2313 | 0.1667 | 2.0974 | 2.1315 |
| Epoch 4/46 | 15/15 | 1s 38ms/step | 0.2334 | 0.1667 | 2.0923 | 1.9727 |
| Epoch 5/46 | 15/15 | 1s 38ms/step | 0.2361 | 0.1667 | 1.977 | 2.009 |
| Epoch 6/46 | 15/15 | 1s 37ms/step | 0.2091 | 0.3333 | 1.9584 | 1.8681 |
| Epoch 7/46 | 15/15 | 1s 38ms/step | 0.2535 | 0.25 | 1.8531 | 1.9117 |
| Epoch 8/46 | 15/15 | 1s 38ms/step | 0.3392 | 0.0833 | 1.7563 | 1.9969 |
| Epoch 9/46 | 15/15 | 1s 44ms/step | 0.303 | 0.3333 | 1.7112 | 1.7695 |
| Epoch 10/46 | 15/15 | 1s 47ms/step | 0.3253 | 0.3333 | 1.7238 | 1.7087 |
| Epoch 11/46 | 15/15 | 1s 47ms/step | 0.4174 | 0.5 | 1.5479 | 1.665 |
| Epoch 12/46 | 15/15 | 1s 46ms/step | 0.428 | 0.5 | 1.4758 | 1.4559 |
| Epoch 13/46 | 15/15 | 1s 46ms/step | 0.5043 | 0.5 | 1.3648 | 1.4818 |
| Epoch 14/46 | 15/15 | 1s 47ms/step | 0.4877 | 0.4167 | 1.355 | 1.5946 |
| Epoch 15/46 | 15/15 | 1s 57ms/step | 0.5019 | 0.3333 | 1.3626 | 2.0373 |
| Epoch 16/46 | 15/15 | 1s 56ms/step | 0.453 | 0.4167 | 1.4818 | 1.268 |
| Epoch 17/46 | 15/15 | 1s 45ms/step | 0.4927 | 0.3333 | 1.3222 | 1.8631 |
| Epoch 18/46 | 15/15 | 1s 49ms/step | 0.5541 | 0.6667 | 1.2236 | 1.2631 |
| Epoch 19/46 | 15/15 | 1s 52ms/step | 0.6298 | 0.75 | 1.0557 | 1.0758 |
| Epoch 20/46 | 15/15 | 1s 47ms/step | 0.6016 | 0.6667 | 1.0239 | 1.0682 |
| Epoch 21/46 | 15/15 | 1s 47ms/step | 0.6321 | 0.5833 | 1.0357 | 1.4679 |
| Epoch 22/46 | 15/15 | 1s 49ms/step | 0.6931 | 0.6667 | 0.9034 | 0.9123 |
| Epoch 23/46 | 15/15 | 1s 49ms/step | 0.6787 | 0.5833 | 0.9018 | 0.9598 |
| Epoch 24/46 | 15/15 | 1s 50ms/step | 0.7605 | 0.5833 | 0.7555 | 0.9957 |
| Epoch 25/46 | 15/15 | 1s 49ms/step | 0.7193 | 0.6667 | 0.8527 | 0.8842 |
| Epoch 26/46 | 15/15 | 1s 46ms/step | 0.6836 | 0.5833 | 0.9378 | 0.9862 |
| Epoch 27/46 | 15/15 | 1s 47ms/step | 0.6995 | 0.5833 | 0.8186 | 0.849 |
| Epoch 28/46 | 15/15 | 1s 65ms/step | 0.7403 | 0.6667 | 0.7525 | 0.789 |
| Epoch 29/46 | 15/15 | 1s 61ms/step | 0.7609 | 0.8333 | 0.6491 | 0.7037 |
| Epoch 30/46 | 15/15 | 1s 65ms/step | 0.8226 | 0.75 | 0.5706 | 0.7617 |
| Epoch 31/46 | 15/15 | 1s 57ms/step | 0.7857 | 0.75 | 0.5885 | 0.725 |
| Epoch 32/46 | 15/15 | 1s 46ms/step | 0.7183 | 0.8333 | 0.8264 | 0.5575 |
| Epoch 33/46 | 15/15 | 1s 44ms/step | 0.7275 | 0.75 | 0.7813 | 0.8909 |
| Epoch 34/46 | 15/15 | 1s 43ms/step | 0.7874 | 0.8333 | 0.6539 | 0.5999 |

| Epoch | | time/step | train categorical accuracy | valid categorical accuracy | train loss | valid loss |
|---|---|---|---|---|---|---|
| Epoch 35/46 | 15/15 | 1s 45ms/step | 0.83 | 0.75 | 0.5072 | 0.7064 |
| Epoch 36/46 | 15/15 | 1s 50ms/step | 0.8203 | 0.5833 | 0.5779 | 0.9041 |
| Epoch 37/46 | 15/15 | 1s 46ms/step | 0.8279 | 0.8333 | 0.5313 | 0.6464 |
| Epoch 38/46 | 15/15 | 1s 45ms/step | 0.7814 | 0.75 | 0.5759 | 0.8975 |
| Epoch 39/46 | 15/15 | 1s 44ms/step | 0.7875 | 0.75 | 0.6914 | 0.6193 |
| Epoch 40/46 | 15/15 | 1s 45ms/step | 0.8794 | 0.75 | 0.4192 | 0.5529 |
| Epoch 41/46 | 15/15 | 1s 48ms/step | 0.8904 | 0.75 | 0.4153 | 0.64 |
| Epoch 42/46 | 15/15 | 1s 45ms/step | 0.8586 | 0.75 | 0.4746 | 0.6775 |
| Epoch 43/46 | 15/15 | 1s 46ms/step | 0.8467 | 0.8333 | 0.434 | 0.3789 |
| Epoch 44/46 | 15/15 | 1s 45ms/step | 0.8727 | 0.6667 | 0.3733 | 0.6657 |
| Epoch 45/46 | 15/15 | 1s 46ms/step | 0.9042 | 0.75 | 0.3152 | 0.751 |
| Epoch 46/46 | 15/15 | 1s 48ms/step | 0.8452 | 0.8333 | 0.4312 | 0.6058 |

## 4.4 Pengujian Model

Pengujian model pada penelitian ini dilakukan dengan dua cara, yaitu menguji model menggunkan data uji dan menguji model secara *real time*.

### 4.4.1 Pengujian dengan Data *Testing*

Proses *training* telah dilakukan, setelah itu dilakukan pengujian dengan menggunakan data *testing* sebanyak 13 data yang dipilih secara acal melalui library sklearn seperti pada tabel 4.5

Tabel 4.5 Data test

| Kosa kata | Jumlah | |
|---|---|---|
| | *Tanh* | *ReLu* |
| Terimakasih | 0 | 0 |
| Tolong | 1 | 2 |
| Maaf | 1 | 1 |
| Nama | 1 | 6 |
| Bagaimana | 2 | 2 |
| Dimana | 2 | 1 |
| Siapa | 1 | 0 |
| Samasama | 3 | 1 |
| Kapan | 2 | 0 |
| Halo | 0 | 0 |

Data tersebut diuji menggunakan model yang sudah ditraining akurasi pada model dengan aktivasi tanh adalah 84% sedangkan akurasi dari model dengan aktivasi relu adalah 92%.

Tabel 4.6 Akurasi perkata dari model yang menggunakan aktivasi tanh

| Kosa kata | True Positive | True Negative | False Negative | False Positive | Akurasi (%) |
|---|---|---|---|---|---|
| Terima Kasih | 0 | 0 | 0 | 0 | 0 |
| Tolong | 1 | 11 | 1 | 0 | 92% |
| Sama-sama | 2 | 10 | 0 | 2 | 92% |
| Halo | 0 | 0 | 0 | 0 | 0 |
| Siapa | 1 | 12 | 0 | 0 | 100% |
| Kapan | 1 | 11 | 0 | 1 | 92% |
| Dimana | 2 | 10 | 1 | 0 | 92% |
| Bagaimana | 2 | 11 | 0 | 0 | 100% |
| Nama | 1 | 12 | 0 | 0 | 100% |
| Maaf | 1 | 12 | 0 | 0 | 100% |

Tabel 4.7 Akurasi perkata dari model yang menggunakan aktivasi ReLu

| Kosa kata | True Positive | True Negative | False Negative | False Positive | Akurasi (%) |
|---|---|---|---|---|---|
| Terima Kasih | 0 | 0 | 0 | 0 | 0 |
| Tolong | 2 | 11 | 0 | 0 | 100% |
| Sama-sama | 1 | 12 | 0 | 0 | 100% |
| Halo | 0 | 12 | 1 | 10 | 92% |
| Siapa | 0 | 0 | 0 | 0 | 0 |
| Kapan | 0 | 0 | 0 | 0 | 0 |
| Dimana | 1 | 12 | 0 | 0 | 100% |

| | | | | | |
|---|---|---|---|---|---|
| Bagaimana | 2 | 11 | 0 | 0 | 100% |
| Nama | 6 | 7 | 0 | 0 | 100% |
| Maaf | 0 | 12 | 1 | 1 | 92% |

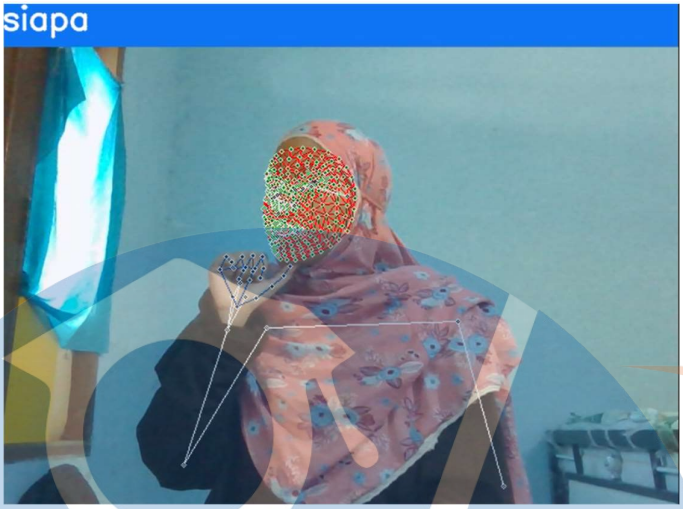
### 4.4.2 Pengujian secara *Real time*

Implementasi model dilakukan dengan mencoba model dengan fungsi aktivasi *tanh* dalam mendeteksi kosa kata bahasa isyarat dengan secara *real time* , karena dari hasil *training* model ini lebih stabil dibandingkan dengan model yang menggunakan aktivasi *relu*. Untuk menguji model secara *real* time akan dilakukan dua kali pengujian oleh penulis. *Device* yang digunakan adalah kamera USB2.0 HD UVC WebCam yang ada pada laptop Asus Vivobook 14 model A412F, dengan posisi kamera dapat menangkap gambar setengah badan atau seperempat badan. Tempat ketika pengujian secara *real time* adalah ruangan yang terang dan *background* tidak ramai

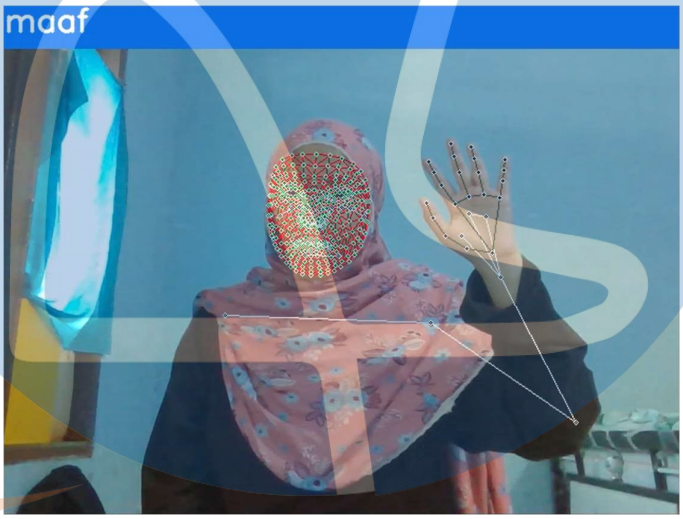
Tabel 4.8 adalah hasil dari pengujian pendeteksian secara *real time* sebanyak dua kali. Dari hasil pengujian tersebut ada kosa yang tidak terdeteksi dengan benar, seperti contoh pada tabel 4.8 gerakan bahasa isyarat dari kata “terima kasih” pada pengujian pertama berhasil terdeteksi sebagai kata “terima kasih”, namun pada pengujian kedua gerakan tersebut terdeteksi sebagai kata “siapa”. Hal ini mungkin saja terjadi karena gerakan kata “terima kasih” dengan kata “siapa” memiliki kemiripin.

Tabel 4.8 Hasil pengujian LSTM dengan aktivasi *tanh* secara *real-time*

| Kosa Kata | Uji-1 | Uji-2 |
|---|---|---|
| Terima Kasih | Terima kasih | Siapa |
| Tolong | Tolong | Siapa |
| Sama-Sama | Siapa | Siapa |
| Halo | Halo | Maaf |
| Siapa | Siapa | Siapa |
| Kapan | Dimana | Dimana |
| Dimana | Dimana | Dimana |
| Bagaimana | Bagaimana | Bagaimana |
| Nama | Nama | Nama |
| Maaf | Maaf | Maaf |




Gambar 4.6 Gerakan isyarat "**Siapa**" yang terdeteksi "**Siapa**"




Gambar 4.7 Gerakan isyarat "**Halo**" yang terdeteksi "**Maaf**"



Hasil dari menguji secara *real-time* telah dilakukan 20 kali pengujian, dari 20 yang benar hanya 13 pengujian. Contoh pada gambar 4.4 yang bahasa isyarat berhasil terdeteksi sesuai dengan kosa katanya, sedangkan pada gambar 4.5 adalah bahasa isyarat yang terdeteksinya berbeda dengan kosa kata dari pergeran bahasa isyarat. Akurasi dari pengujian secara *real-time* ini adalah 65%, dihitung dari rumus akurasi.

$$Akurasi = \frac{Jumlah\ yang\ terdeteksi\ benar}{Jumlah\ total\ yang\ deteksi} \times 100\%$$

$$= \frac{13}{20} \times 100\%$$

$$= 65\%$$

Pengujian secara *real time* dilakukan satu kali lagi dengan orang yang berbeda yang bukan penutur asli dengan menggunakan *device* yang sama namun dalam ruangan yang berbeda, yang dimana ruang tersebut *background* lebih ramai. Dari hasil pengujian ini tidak ada satu katapun yang berhasil terdeteksi. Maka dari itu tidak dimasukan ke dalam tabel 4.8.

Peneliti juga melakukan pengujian terhadap model dengan aktivasi *ReLu*, berbeda dengan model *Tanh* yang dapat mendeteksi benar beberapa, pada model *ReLu* ketika dilakukan pengujian berkali-kali tidak berhasil mendeteksi dengan benar pergerakan kosa kata bahasa isyarat sesuai dengan kosa katanya.

## 4.5 Evaluasi

Kedua model ini dijalankan pada file python yang berbeda. Setiap file akan dimulai dari pembagian data, hal ini menyebabkan data yang digunakan untuk data *train*, data validasi dan data uji berbeda pada setiap model. Data yang dikumpulkan adalah yang dilakukan oleh satu orang yang bukan penutur asli, yaitu peneliti sendiri. Lokasi pengumpulan data adalah tempat yang di bagian belakangnya tidak ramai atau polosan.

Pada proses *training*, skor akurasi pada epoch terakhir untuk model *tanh* lebih tinggi daripada model *ReLu*. Skor akurasi pada hasil *training* data *train* dan data uji pada model dengan aktivasi *tanh*, yaitu 0,84 dan 0,83. Sedangkan skor akurasi data *train* dan data uji pada *training* untuk model dengan aktivasi *ReLu*, yaitu 0,65 dan 0,75. Fungsi aktivasi *tanh* mengubah nilai inputan dari rentang -1 sampai 1, sedangkan fungsi aktivasi *ReLu* mengubah nilai inputan dari 0 sampai x, yang dimana x ini adalah nilai inputan, yang apabila nilai inputan tersebut positif. Hasil *training* ini bisa jadi tidak akurat bila untuk membandingkan dua model ini, karena data train dan data validasi yang digunakan pada kedua model tersebut tidak sama.

Pada proses pengujian menggunakan data uji untuk model dengan aktivasi *tanh* skor akurasinya adalah 0,84 dan model dengan aktivasi *ReLu* skor akurasinya

adalah 0,92. Berbeda dengan akurasi pada proses *training*, pada hasil pengujian dengan data uji skor akurasi yang paling tinggi adalah model dengan akurasi *ReLu*. Hal ini disebabkan karena *overitting* pada tahap *training*, selain itu data yang digunakan untuk data uji pada setiap model berbeda, jadi hasil ini kurang akurat.

Pada proses *training* hasil akurasi pada model *tanh* lebih tinggi, maka peneliti melakukan 2 set pengujian secara *real time* dengan menggunakan model *tanh*. Pengujian ini diperagakan oleh peneliti juga dengan lokasi yang sama ketika pengumpulan. Hasil dari pengujian secara *real time* 65%. Pada pengujian ini peneliti juga menggunakan orang lain yang bukan penutur asli untuk menguji model dengan *tanh*. Namun, hasil dari pengujian dengan orang lain ini tidak ada pergerakan bahasa isyarat yang berhasil terdeteksi benar sesuai kosa katanya.

Untuk mendapatkan hasil yang akurat ketika mengumpulkan data dilakukan oleh lebih dari satu orang dan menggunakan seorang penutur asli, selain itu penggunakan data pada proses *training* dan proses *testing* di setiap model dibuat sama. Pengumpulan data dengan menggunakan penutur asli membuat gerakan kosa kata bahasa isyarat tidak kaku dan sesuai. Menggunakan lebih dari satu orang untuk pengumpulan data juga diperlukan agar dapat mendeteksi menggunakan orang banyak juga.

# BAB 5
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Sesuai dengan rumusan masalah pada penelitian ini dapat disimpulkan sebagai berikut.

1. Penelitian ini melakukan pengenalan kosakata bahasa isyarat dengan menggunakan dua model LSTM yang arsitekturnya sama namun fungsi aktivasi yang berebeda. Model pertama menggunakan fungsi aktivasi *ReLu*, sedangkan model kedua menggunakan fungsi aktivasi *Tanh*. Kedua model tersebut dilatih dengan menggunakan 475 data latih dan 12 data validasi. Dari hasil melatih kedua model tersebut bahwa model dengan aktivasi tanh lebih optimal bila dibandingkan model dengan aktivasi ReLu. Dari hasil pengujian dengan data uji kedua model tersebut dapat dilakukan, walau tidak semua data berhasil terdeteksi benar. Disimpulkan bahwa kedua model ini dapat digunakan untuk melakukan pengenalan bahasa isyarat.
2. Pada proses pengujian dengan data uji Model dengan aktivasi *tanh* menghasilkan akurasi 84% sedangkan model dengan aktivasi *ReLu* menghasilkan akurasi 92%.

## 5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran yang dapat diberikan peneliti untuk penelitian yang akan mendatang sebagai berikut.

1. Data yang dikumpulkan menggunakan beberapa orang
2. Menggunakan yang penutur asli ketika pengambilan data.
3. Jika penelitian yang dilakukan menggunakan beberapa model, menggunakan data latih, data validasi dan data pengujian yang sama untuk setiap model.

## DAFTAR REFERENSI


[1] KEMDIKBUD, "Kemendikbudristek Gelar Kelas Akhir Pekan: Belajar Bahasa Isyarat," Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi. Diakses: 2 Maret 2024. [Daring]. Tersedia pada: https://www.kemdikbud.go.id/main/blog/2023/12/kemendikbudristek-gelar-kelas-akhir-pekan-belajar-bahasa-isyarat

[2] R. A. Mursita, "RESPON TUNARUNGU TERHADAP PENGGUNAAN SISTEM BAHASA ISYARAT INDONESA (SIBI) DAN BAHASA ISYARAT INDONESIA (BISINDO) DALAM KOMUNIKASI," *INKLUSI*, vol. 2, no. 2, hlm. 221, Des 2015, doi: 10.14421/ijds.2202.

[3] H. A. A. A. Faruqi, "Pengenalan Bahasa Isyarat Bisindo Menggunakan Metode Hog dan Svm Dalam Pengolahan Citra," Skripsi, STT Terpadu Nurul Fikri, Jakarta, 2019.

[4] D. Yolanda, K. Gunadi, dan E. Setyati, "Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real-Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network," *INFRA*, vol. 8, no. 1, hlm. 203–208, 2020.

[5] H. Moetia Putri dan W. Fuadi, "PENDETEKSIAN BAHASA ISYARAT INDONESIA SECARA REAL-TIME MENGGUNAKAN LONG SHORT-TERM MEMORY (LSTM)", doi: doi.org/10.29103/tts.v3i1.6853.

[6] S. T. Isma, "MENELITI BAHASA ISYARAT DALAM PERSPEKTIF VARIASI BAHASA," *Kongres Bahasa Indonesia*, hlm. 1, 2018. Diakses: 2 Maret 2024. [Daring]. Tersedia pada: https://kbi.kemdikbud.go.id/kbi_back/file/dokumen_makalah/dokumen_makalah_1540468871.pdf

[7] PUSBISINDO, "Asal Usul BISINDO dan PUSBISINDO," PUSBISINDO. Diakses: 2 Maret 2024. [Daring]. Tersedia pada: https://pusbisindo.org/

[8] studiokece app development, "Bisindo Terima Kasih," YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=S-2Lj8OzPqQ

[9] CDS UB, “Indonesia Sign Language: Salam, Pronouns, Place, and Family Members,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=hBuoIRtB9Xw

[10] studiokece app development, “Bisindo Maaf,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=yiNQZ4qP-gQ

[11] CDS UB, “Short Conversations in Indonesian Sign Language,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=5fKXIJrzyqQ

[12] O. H. Kahfi, “Belajar bahasa Isyarat BISINDO #2 KATA SAPA,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=xnxydJPDD1M

[13] thnksgvng, “BAHASA ISYARAT BISINDO,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=mukNGgweHSI

[14] studiokece app development, “Bisindo Sama-Sama,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=MvyeZK6hHPg

[15] studiokece app development, “Bisindo Kapan,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://www.youtube.com/watch?v=0kym9mtmJHY

[16] studiokece app development, “Bisindo Halo,” YouTube. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://youtu.be/GCFfwXFi6hA?si=2r8KxxdfHy6ppW98

[17] Suyanto, K. N. Ramadhani, dan S. Mandala, *Deep Learning : Modernisasi Machine Learning Untuk Big Data*. Bandung: INFORMATIKA, 2019.

[18] G. H. Samaan *dkk.*, “MediaPipe’s Landmarks with RNN for Dynamic Sign Language Recognition,” *Electronics (Switzerland)*, vol. 11, no. 19, Okt 2022, doi: 10.3390/electronics11193228.

[19] M. Nabil, “The Architecture of a Basic RNN,” Medium. Diakses: 11 Maret 2024. [Daring]. Tersedia pada: https://medium.com/@navarai/the-architecture-of-a-basic-rnn-eb5ffe7f571e

[20] R. Variawa, “Neural Networks And Deep Learning: CNN vs. RNN,” Medium. Diakses: 21 Mei 2024. [Daring]. Tersedia pada: https://becominghuman.ai/neural-networks-and-deep-learning-cnn-vs-rnn-7710d69feebf

[21] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A. B. Gil-González, dan J. M. Corchado, “Deepsign: Sign Language Detection and Recognition Using Deep Learning,” *Electronics (Switzerland)*, vol. 11, no. 11, Jun 2022, doi: 10.3390/electronics11111780.

[22] R. C. Staudemeyer dan E. R. Morris, “Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks,” Sep 2019, [Daring]. Tersedia pada: http://arxiv.org/abs/1909.09586

[23] S. Yan, “Understanding LSTM and its diagrams,” Medium. Diakses: 21 Mei 2024. [Daring]. Tersedia pada: https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714

[24] S. Hochreiter dan J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, hlm. 1735–1780, Nov 1997, doi: 10.1162/neco.1997.9.8.1735.

[25] Tim Tensorflow, “Introduction to TensorFlow.” Diakses: 21 Maret 2024. [Daring]. Tersedia pada: https://www.tensorflow.org/learn

[26] Tim Tensorflow, “Keras: The high-level API for TensorFlow | TensorFlow Core.” Diakses: 21 Maret 2024. [Daring]. Tersedia pada: https://www.tensorflow.org/guide/keras

[27] Tim OpenCV, “About OpenCV.” Diakses: 23 April 2024. [Daring]. Tersedia pada: https://opencv.org/about/

[28] Tim Mediapipe, “Holistic landmarks detection task guide | MediaPipe | Google for Developers.” Diakses: 25 Maret 2024. [Daring]. Tersedia pada: https://developers.google.com/mediapipe/solutions/vision/holistic_landmarker

[29] Tim Mediapipe, “Mediapipe · GitHub.” Diakses: 8 Maret 2024. [Daring]. Tersedia pada:

https://github.com/google/mediapipe/blob/master/docs/solutions/holistic.md

[30] Tim Mediapipe, “Hand landmarks detection guide.” Diakses: 21 Mei 2024. [Daring]. Tersedia pada: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

[31] Tim Mediapipe, “Pose landmark detection guide.” Diakses: 21 Mei 2024. [Daring]. Tersedia pada: https://developers.google.com/mediapipe/solutions/vision/pose_landmarke

[32] Tim Mediapipe, “Face landmark detection guide”, Diakses: 21 Mei 2024. [Daring]. Tersedia pada: https://developers.google.com/mediapipe/solutions/vision/face_landmarke

[33] A. A. Teran-Quezada, V. Lopez-Cabrera, J. C. Rangel, dan J. E. Sanchez-Galan, “Sign-to-Text Translation from Panamanian Sign Language to Spanish in Continuous Capture Mode with Deep Neural Networks,” *Big Data and Cognitive Computing*, vol. 8, no. 3, hlm. 25, Feb 2024, doi: 10.3390/bdcc8030025.

[34] P. Kurnia Sari, G. Qorik Oktagalu Pratamasunu, dan F. Nur Fajri, “Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode Deep Gated Recurrent Unit (GRU),” *Jurnal Komputer Terapan*, vol. 8, no. 1, hlm. 186–193, Jun 2022, doi: 10.35143/jkt.v8i1.4901.

[35] I. P. A. A. Payadnya dan I. G. A. N. T. Jayantika, *Panduan Penelitian Eksperimen Beserta Analisis Statistik dengan SPSS*. DIY Yogyakarta: Deepublish, 2018.

# LAMPIRAN

## Import Library

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import tensorflow

from sklearn.metrics import multilabel_confusion_matrix, confusion_matrix, accur
acy_score
```

## MediaPipe Holistik

```
mp_holistic = mp.solutions.holistic # holistic model
mp_drawing = mp.solutions.drawing_utils # drawing utilities

# fungsi deteksi menggunakan mediapipe (contohnya menggunakan image)
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # mengubah warna dari BGR ke
RGB
    image.flags.writeable = False # image is no longer writeable (tidak ada imag
e yang dideteksi)
    results = model.process(image) # membuat prediksi
    image.flags.writeable = True # image is now writeable (ada image yang terdet
eksi)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # mengubah warna dari BGR ke
RGB
    return image, results

# fungsi untuk membuat point point penghubung landmark (telah disesuaikan)
def draw_landmarks_style(image, results):
    # menggambar point penghubung pada wajah
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMES
H_TESSELATION,
                              mp_drawing.DrawingSpec(color = (0, 128, 0), thickne
ss = 1, circle_radius = 1), # ini bagian mulut, hidung, dan mata
```

```
                            mp_drawing.DrawingSpec(color = (0, 0, 255), thickne
ss = 1, circle_radius = 1) # seluruh wajah kecuali 3 diatas
                            )
    # menggambar point penghubung pada pose seluruh tubuh
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CO
NNECTIONS,
                            mp_drawing.DrawingSpec(color = (0, 0, 0), thickness
 = 1, circle_radius = 1),
                            mp_drawing.DrawingSpec(color = (224, 224, 224), thi
ckness = 1, circle_radius = 1) #untuk garis pose
                            )
    # menggambar point penghubung pada tangan kiri
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HA
ND_CONNECTIONS,
                            mp_drawing.DrawingSpec(color = (0, 0, 0), thickness
 = 1, circle_radius = 1),
                            mp_drawing.DrawingSpec(color = (47, 79, 79), thickn
ess = 1, circle_radius = 1) # warna garis gestur pada tangan kiri
                            )
    # menggambar point penghubung pada tangan kanan
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.H
AND_CONNECTIONS,
                            mp_drawing.DrawingSpec(color = (0, 0, 0), thickness
 = 1, circle_radius = 1),
                            mp_drawing.DrawingSpec(color = (139, 69, 19), thick
ness = 1, circle_radius = 1) # warna garis gestur pada tangan kiri
                            )
```

### Exktrak Keypoint

```
def extract_keypoints(result):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.po
se_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33*4)
    face =np.array([[res.x, res.y, res.z] for res in results.face_landmarks.land
mark]).flatten() if results.face_landmarks else np.zeros(468*3)
    r_hand =np.array([[res.x, res.y, res.z] for res in results.right_hand_landm
arks.landmark]).flatten() if results.right_hand_landmarks else np.zeros(
21*3)
    l_hand =np.array([[res.x, res.y, res.z] for res in results.left_hand
_landmarks.landmark]).flatten() if results.left_hand_landmarks else np.z
eros(21*3)
    return np.concatenate([pose, face, l_hand, r_hand])
```

### Penyimpanan

```
DATA_PATH = os.path.join('D:/mp_data1/24frame')
```

### Kosa kata

```
# kosakata yang akan dideteksi
actions = np.array(['terimakasih', 'tolong', 'samasama', 'halo', 'siapa', 'kapan
', 'dimana', 'bagaimana', 'nama', 'maaf']) # 10 kosa kata yang akan digunakan. D
an ini akan disimpan untuk nama folder untuk masing-masing kosakata

# jumlah video yang digunakan
no_sequences = 50 # jumlah video untuk setiap kosakata

# pembagian video dalam bentuk frame
sequence_length = 24 # jumlah framenya adalah 24
```

### Buat folder

```
# untuk membuat folder pada folder action yang udah dibuat. masing masing folder
 berisi 30 folder
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

### Pengumpulan Data

```
# akses kamera menggunakan webcam
cap = cv2.VideoCapture(0) # akses webcam
# menambahkan model mediapipe
with mp_holistic.Holistic(min_detection_confidence = 0.5, min_tracking_confidenc
e = 0.5) as holistic:

    # untuk mengumpulkan data video dalam bentuk frame terdapat looping
    for action in actions_maaf:

        # pengulangan dalam video
        for sequence in range(no_sequences):

            # pengulangan untuk frame dalam satu video
            for frame_num in range(sequence_length):

                # read feed
                # membaca frame dari webcame. frame adalah image dari video yang
 ditankap menggunakan webcame
                ret, frame = cap.read()
```

```
                frame = cv2.flip(frame, 1) # me-
mirror kan video realtime (awalnya tidak mirror)

                # mendeteksi
                image, results = mediapipe_detection(frame, holistic)
        #         print(results)

                # draw point landmark
                draw_landmarks_style(image, results)

                # mengumpulkan data video dengan menggunakan if else
                if frame_num == 0:
                    cv2.putText(image, 'Memulai pengumpulan Data', (120, 200),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 4, cv2.
LINE_AA)
                    cv2.putText(image, 'Mengumpulkan frame kata {} video ke-
{}'.format(action, sequence), (15, 12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv
2.LINE_AA)
                    # menampilkan
                    cv2.imshow('opencv feed', image)
                    cv2.waitKey(2500)
                else:
                    cv2.putText(image, 'Mengumpulkan frame kata {} video ke-
{}'.format(action, sequence), (15, 12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv
2.LINE_AA)
                    cv2.imshow('opencv feed', image)

                # new export keypoints
                keypoints = extract_keypoints(results)
                npy_path = os.path.join(DATA_PATH, action, str(sequence), str(fr
ame_num))
                np.save(npy_path, keypoints)

                # menghentikan camera = mengambil gambar ketika di klik
                if cv2.waitKey(10) & 0xFF == ord('q'): # menginisiasi jika menek
an salah satu tombol keyboard (contoh disini adalah q) maka proses open webcame
terhenti dan foto akan terambil
                    break

    cap.release()
    cv2.destroyAllWindows()
    #nanti tampilan penangkapan videonya seperti menggunakan kamera belakang
```

### Labeling

```
label_map = {label:num for num, label in enumerate(actions)}

sequences, labels = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy
".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])

X = np.array(sequences)
y = to_categorical(labels).astype(int)
```

### Split Data Test, Validasi, dan Train

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05)
X_valid, X_testing, y_valid, y_testing = train_test_split(X_test, y_test, test_s
ize=0.5)
```

### Model Aktivasi tanh

```
model = Sequential()
model.add(LSTM(64, activation='tanh', recurrent_activation='sigmoid', return_seq
uences=True, input_shape = (24, 1662)))
model.add(LSTM(64, activation='tanh', recurrent_activation='sigmoid', return_seq
uences=False))
model.add(Dense(64, activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(actions.shape[0], activation='softmax'))
```



### Compile Model Tanh

```
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categ
orical_accuracy'])
```

### Train Model Tanh

```
history = model.fit(X_train, y_train, validation_data = (X_valid, y_valid), epoc
hs = 160)
```

### Model LSTM aktivasi Relu

```
model = Sequential()
model.add(LSTM(64, activation='relu', recurrent_activation='sigmoid', return_seq
uences=True, input_shape = (24, 1662)))
model.add(LSTM(64, activation='relu', recurrent_activation='sigmoid', return_seq
uences=False))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation=''))
model.add(Dense(actions.shape[0], activation='softmax'))
```

### Compile Model Relu

```
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categ
orical_accuracy'])
```

### Train Model Relu

```
history = model.fit(X_train, y_train, validation_data = (X_valid, y_valid), epoc
hs = 46)
```

### Save Model Tanh

```
model.save('bisindo_tanh.h5')
```

### Save Model Relu

```
model.save('bisindo_relu.h5')
```

### Confusion Matrix

```
res = model.predict(X_testing)

yhat = model.predict(X_testing)

ytrue = np.argmax(y_testing, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()

multilabel_confusion_matrix(ytrue, yhat)
```

### Skor Akurasi

```
accuracy_score(ytrue, yhat)
```

### Pengujian secara *Real Time*

```
sequence = []
sentence = []
threshold = 0.8

cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=
0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()
        frame = cv2.flip(frame, 1)

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_landmarks_style(image, results)

        # 2. Prediction logic
        keypoints = extract_keypoints(results)
        # sequence.insert(0,keypoints)
        # sequence = sequence[:24]
        sequence.append(keypoints)
        sequence = sequence[-24:]

        if len(sequence) == 24:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])


        #3. Viz logic
            if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                    if actions[np.argmax(res)] != sentence[-1]:
                        sentence.append(actions[np.argmax(res)])
                else:
                    sentence.append(actions[np.argmax(res)])

            if len(sentence) > 1:
                sentence = sentence[-1:]

            # Viz probabilities
            # image = prob_viz(res, actions, image, colors)
```

```
            cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
            cv2.putText(image, ' '.join(sentence), (3,24),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE
_AA)

            # Show to screen
            cv2.imshow('OpenCV Feed', image)

            # Break gracefully
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break
        cap.release()
        cv2.destroyAllWindows()
```