



MODEL PENERJEMAH BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

LAPORAN TUGAS AKHIR

Oleh:

Riestiya Zain Fadillah

105216003



**FAKULTAS SAINS DAN ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
UNIVERSITAS PERTAMINA
JANUARI 2020**

1. Dilarang mengutip karya tulis ini, kecuali:
 - a. menyebutkan sumber sesuai kaidah kecendekiaan;
 - b. pengutipan hanya untuk keperluan pendidikan, penulisan karya ilmiah atau penelitian;
 - c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

Dilindungi Undang-Undang

© Copyright of Universitas Pertamina



MODEL PENERJEMAH BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

LAPORAN TUGAS AKHIR

Oleh:

Riestiya Zain Fadillah

105216003



**FAKULTAS SAINS DAN ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
UNIVERSITAS PERTAMINA
JANUARI 2020**

1. Dilarang mengutip karya tulis ini, kecuali:
 - a. menyebutkan sumber sesuai kaidah kecendekiaan;
 - b. pengutipan hanya untuk keperluan pendidikan, penulisan karya ilmiah atau penelitian;
 - c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

Dilindungi Undang-Undang

© Copyright of Universitas Pertamina

LEMBAR PENGESAHAN

Judul Tugas Akhir : Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) menggunakan *Convolutional Neural Network*
Nama Mahasiswa : Riestiya Zain Fadillah
Nomor Induk Mahasiswa : 105216003
Program Studi : Ilmu Komputer
Fakultas : Sains dan Ilmu Komputer
Tanggal Lulus Sidang Tugas Akhir : 15 Januari 2020

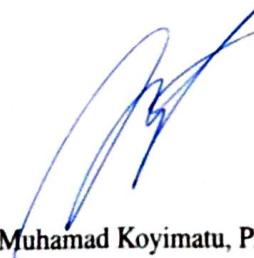
Jakarta, 29 Januari 2020

MENGESAHKAN

Pembimbing I : Nama : Ade Irawan, Ph.D.
NIP : 116130
Pembimbing II : Nama : Meredita Susanty, M.Sc.
NIP : 116020

MENGETAHUI,

Ketua Program Studi



Muhamad Koyimatu, Ph.D.

NIP. 116108

LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa Tugas Akhir berjudul Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) menggunakan *Convolutional Neural Network* ini adalah benar-benar merupakan hasil karya saya sendiri dan tidak mengandung materi yang ditulis oleh orang lain kecuali telah dikutip sebagai referensi yang sumbernya telah dituliskan secara jelas sesuai dengan kaidah penulisan karya ilmiah.

Apabila dikemudian hari ditemukan adanya kecurangan dalam karya ini, saya bersedia menerima sanksi dari Universitas Pertamina sesuai dengan peraturan yang berlaku.

Demi pengembangan ilmu pengetahuan, saya menyetujui untuk memberikan kepada Universitas Pertamina hak bebas royalti noneksklusif (*non-exclusive royalty-free right*) atas Tugas Akhir ini beserta perangkat yang ada. Dengan hak bebas royalti noneksklusif ini Universitas Pertamina berhak menyimpan, mengalih media/format-kan, mengelola dalam bentuk pangkatan 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

Jakarta, 29 Januari 2020

Yang membuat pernyataan,



Riestiya Zain Fadillah



ABSTRAK

Riestiya Zain Fadillah. 105216003. Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) menggunakan *Convolutional Neural Network*.

Penelitian ini mengusulkan pengembangan model penerjemah Bahasa Isyarat Indonesia (Bisindo) dengan memanfaatkan teknologi *machine learning* khususnya *Convolutional Neural Network* (CNN). Berbeda dengan penelitian yang sudah ada, Bisindo adalah bahasa isyarat yang relatif banyak digunakan Tuli namun tidak resmi, sehingga dataset yang diperlukan hampir tidak ada. Tujuan dari penelitian ini yaitu bertambahnya jumlah penerjemah Bisindo elektronik untuk meningkatkan aksesibilitas Tuli. Selain itu, penelitian ini juga bertujuan untuk mengevaluasi metode *parameter-transfer* untuk mengatasi keterbatasan jumlah data pada dataset dan implementasi pada sistem bahasa isyarat yang berbeda. Pengembangan model Bisindo dilakukan dengan memanfaatkan arsitektur Model American Sign Language (ASL) melalui dua pendekatan berdasarkan pemanfaatan parameter model tersebut, yang kemudian disebut Model A dan Model B. Model A dikembangkan tanpa menggunakan parameter dari Model ASL (**tanpa parameter-transfer**), sedangkan Model B dikembangkan dengan menggunakan parameter Model ASL serta *knowledge parameter* di dalamnya (**dengan parameter-transfer**). Metode yang digunakan dalam pengembangan adalah eksperimen dan analisis data untuk menentukan model terbaik secara kuantitatif dengan analisis variabel durasi training, akurasi saat *testing* dan *F1 Score* saat *testing*. Model ASL di penelitian memiliki akurasi *testing* sebesar 96.40%. Hasil penelitian menunjukkan bahwa Model A menghasilkan akurasi *testing* sebesar 94.38% sedangkan Model B sebesar 30%. Dapat disimpulkan bahwa *parameter-transfer* pada Model B memerlukan *waktu training* yang lebih sedikit dibandingkan Model A serta mampu mempelajari fitur Bisindo yang memiliki kemiripan dengan fitur ASL. Namun, untuk mempelajari keseluruhan alfabet Bisindo, pemanfaatan arsitektur tanpa *parameter-transfer* merupakan pendekatan yang lebih baik dibandingkan menggunakan *parameter-transfer*.

Kata kunci: Tuli, Bahasa isyarat, Bisindo, *Machine learning*, ASL, *Transfer learning*, *Parameter transfer*.



ABSTRACT

Riestiya Zain Fadillah. 105216003. Indonesian Sign Language Translator Model using Convolutional Neural Network.

This research proposed the development of the Indonesian Sign Language (Bisindo) translator model using machine learning technology especially the Convolutional Neural Network (CNN). In contrast to existing research, Bisindo is a sign language that is relatively wide-used by the Deaf but it is not the official sign language, so the required dataset is almost non-exist. The purpose of this study is to improve the accessibility of the Deaf by increasing the number of Bisindo electronic translators. Besides, this study also aimed to evaluate the parameter-transfer method to overcome the limitations on the amount of data in the dataset and its performance in the implementation of different sign language systems. The development of the Bisindo model is done by utilizing the architecture of the American Sign Language (ASL) Model through two approaches based on the usage of the model parameters, called Model A and Model B. Model A developed without using parameters from the ASL Model (**without parameter-transfer**), while the Model B developed using the ASL Model parameters and the knowledge parameters within (**with transfer parameters**). The method used in the development is experiment and data analysis to determine the best model was done quantitatively by analyzing the duration of the training, accuracy during testing and F1 score at testing. The ASL model in the study had a testing accuracy of 96.40%. The results showed that Model A produced a testing accuracy of 94.38% while Model B produced only 30% of testing accuracy. It can be concluded that the parameter-transfer in Model B requires less training time than Model A and makes the model be able to learn the Bisindo features that have similarities to the ASL features. However, to study the entire Bisindo alphabet, using architecture without parameter-transfer is a better approach than using parameter-transfer.

Keywords: Deaf, Sign language, Bisindo, Machine learning, ASL, Transfer Learning, Parameter transfer.



KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan penulis nikmat-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir ini dengan tepat waktu. Tanpa Ridho dan Nikmat-Nya tentu penulis tidak akan sanggup untuk menyelesaikan baik penelitian hingga penulisan laporan tugas akhir ini.

Pertama-tama, penulis hendak mengucapkan terimakasih atas bimbingan dan dukungan yang diberikan dari Pak Ade Irawan serta Ibu Meredita Susanty selaku pembimbing dalam mengerjakan tugas akhir ini. Selanjutnya, penulis ingin mengucapkan terima kasih kepada dosen Universitas Pertamina khususnya dosen program studi Ilmu Komputer atas bimbingan dan ilmu yang telah diberikan selama kuliah di Universitas Pertamina. Penulis juga ingin mengucapkan terima kasih kepada Laode Iman Syahrin, Megandi dan Aries Dwi Prasetiyo atas dukungan dan kontribusi dalam proses penelitian ini. Tak lupa penulis juga ucapan terima kasih kepada rekan – rekan Ilmu Komputer 2016 dan seluruh pihak yang telah berperan dalam penelitian tugas akhir ini yang tidak bisa penulis sebutkan satu persatu.

Akhir kata, penulis memohon maaf apabila penulisan laporan tugas akhir ini masih jauh dari kata sempurna, oleh karena itu penulis memohon saran untuk membantu penulis dapat menulis lebih baik lagi.

Jakarta, 29 Januari 2020

Riestiya Zain Fadillah

1. Jika diminta, penulis dapat memberikan kembali, kecuali:
 - a. penyalinan atau penerbitan ulang; b. pengutipan tidak merugikan Universitas Pertamina;
 - c. pengutipan tidak merugikan Universitas Pertamina;
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.



ABSTRAK

ABSTRACT

KATA PENGANTAR

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I PENDAHULUAN

1.1	Latar Belakang	1
1.2	Rumusan Masalah	2
1.3	Hipotesis	2
1.4	Batasan Masalah	2
1.5	Tujuan Penelitian	2
1.6	Manfaat Penelitian	2
1.7	Notasi yang digunakan	3
BAB II	TINJAUAN PUSTAKA	
2.1	Bisindo	4
2.1.1	Sejarah Bahasa Isyarat di Indonesia	4
2.1.2	Penggunaan Bisindo	4
2.1.3	Isyarat Alfabet Bisindo	5
2.1.4	Sudut Pandang yang digunakan	6
2.2	CNN	6
2.2.1	<i>Feature Maps</i> dan <i>Weight Sharing</i>	6
2.2.2	<i>Sub-sampling</i> pada CNN	8
2.2.3	<i>Loss Function</i>	9

DAFTAR ISI

2.3	<i>Gradient-Based Parameter Optimization</i>	10
2.3.1	<i>Gradient Descent</i>	10
2.3.2	<i>Stochastic Gradient Descent</i>	10
2.4	<i>Regularization</i>	10
2.5	<i>Data Augmentation</i>	11
2.6	<i>Dropout</i>	11
2.7	Layer pada CNN	11
2.7.1	<i>Activation Layer</i>	11
2.7.2	<i>Pooling Layer</i>	12
2.7.3	<i>Fully Connected Layer</i>	13
2.7.4	<i>Convolutional Layer</i>	13
2.8	<i>Transfer Learning</i>	13
2.8.1	<i>Sejarah Transfer Learning</i>	13
2.8.2	<i>Definisi Transfer Learning</i>	14
2.8.3	<i>Teknik-teknik Transfer Learning</i>	16
2.9	Batasan Transfer dan <i>Negative Transfer</i>	17
2.10	Penelitian sebelumnya mengenai Model Penerjemah Bisindo	17
BAB III METODE PENELITIAN		18
3.1	Metodologi Penelitian	18
3.1.1	Pengembangan Model ASL	19
3.1.2	Pengembangan Model A	20
3.1.3	Pengembangan Model B	21
3.1.4	Pengujian dan Analisis	21
BAB IV HASIL DAN PEMBAHASAN		23
4.1	Pengembangan Model ASL	23
4.1.1	<i>Library</i> dan <i>Packages</i> yang digunakan	23
4.1.2	Dataset yang Digunakan	24
4.1.3	Strategi Pembagian Data	24
4.1.4	<i>Data Pre-processing</i>	25
4.1.5	Arsitektur Model	25

1. Dilarang mengutip karya tulis ini, kecuali:
 a. menyebutkan sumber sesuai izin;
 b. pengutipan hanya untuk keperluan pendakian, penulisan karya ilmiah atau penelitian;
 c. pengutipan tidak merugikan Universitas Pertamina.
 2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

4.1.6	Hasil <i>Training</i>	26
4.2	Pengembangan Model A	27
4.2.1	<i>Library</i> yang digunakan	27
4.2.2	Dataset yang digunakan	27
4.2.3	<i>Data Augmentation</i>	27
4.2.4	Hasil <i>Training</i>	28
4.3	Pengembangan Model B	31
4.3.1	Teknik <i>Transfer Learning</i>	31
4.3.2	Hasil Training	31
4.4	Pengujian dan Analisis	33
4.4.1	Testing Model	33
4.4.2	Analisis <i>Transfer Learning</i> pada Model A dan B	34
4.4.3	Perbandingan Performa Model A dan Model B	35
BAB V	KESIMPULAN DAN SARAN	38
5.1	Kesimpulan	38
5.2	Saran	38
LAMPIRAN A	Tautan menuju Kernel	39
DAFTAR PUSTAKA		40



- a. menjelaskan sumber suka-sah kependidikan;
- b. pengutipan hanya untuk keperluan penulisan karya ilmiah atau penelitian;
- c. pengutipan tidak merugikan Universitas Pertamina.

2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun

tanpa adanya izin dari Universitas Pertamina.

DAFTAR TABEL

Tabel 4.1	<i>Library</i> yang digunakan untuk membuat model ASL dan penggunaannya dalam model	23
Tabel 4.2	<i>Library</i> yang digunakan untuk <i>data augmentation</i> dataset Bisindo	27
Tabel 4.3	Rata-rata loss dan accuracy Model A untuk seluruh <i>round</i>	30
Tabel 4.4	Durasi <i>training</i> terhadap banyaknya <i>frozen layer</i>	32
Tabel 4.5	Jumlah Parameter dalam setiap Konfigurasi Model	35



Universitas

Pertamina

Hak Cipta milik Universitas Pertamina

Dilindungi Undang-Undang

© Copyright of Universitas Pertamina

1. Dilarang mengutip karya tulis ini, kecuali:

a. menyebutkan sumber sesuai kaidah kecendekiaan;

b. pengutipan hanya untuk keperluan pendidikan, penulisan karya ilmiah atau penelitian;

c. pengutipan tidak merugikan Universitas Pertamina.

2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

DAFTAR GAMBAR

Gambar 2.1	Alfabet Bisindo (Almuhamram, 2013)	5
Gambar 2.2	Arsitektur LeNet (Lecun et al., 1998).	6
Gambar 2.3	Arsitektur ConvNet (A. Krizhevsky, 2012).	7
Gambar 2.4	Konektivitas antara layer yang bertetangga (Fan, 2014).	7
Gambar 2.5	Contoh <i>weight sharing</i> dari sebuah <i>feature map</i> (Fan, 2014)).	8
Gambar 2.6	Perbedaan <i>learning process</i> antara <i>traditional machine learning</i> dan <i>transfer learning</i> (Sinno Jialin Pan, 2009)	14
Gambar 3.1	Diagram tahapan penelitian (kanan: detail proses dari gambar kiri)	18
Gambar 3.2	Arsitektur CNN yang digunakan (Modi, 2019)	20
Gambar 4.1	Sampel gambar data ASL	24
Gambar 4.2	layer pada model CNN diambil dari Model ASL referensi (Modi, 2019)	25
Gambar 4.3	Grafik akurasi dan loss dari Model ASL dengan 5 <i>epoch</i> saat <i>training</i> dan <i>testing</i>	26
Gambar 4.4	Sampel Dataset Bisindo	27
Gambar 4.5	Dataset Bisindo hasil augmentasi: <i>crop</i> , <i>rotate</i> dan <i>Gaussian Noise</i>	28
Gambar 4.6	Ilustrasi <i>training</i> menggunakan <i>k-fold cross validation</i>	29
Gambar 4.7	Grafik akurasi dan loss dari Model A saat <i>round</i> pertama	29
Gambar 4.8	Grafik akurasi dan loss dari Model A saat <i>round</i> kedua	29
Gambar 4.9	Grafik akurasi dan loss dari Model A saat <i>round</i> ketiga	30
Gambar 4.10	Grafik rata-rata akurasi dari konfigurasi <i>layer freezing</i> . (sumbu-y adalah akurasi, sumbu x adalah banyaknya <i>frozen layer</i> terhitung dari layer <i>input</i> model.)	32
Gambar 4.11	Hasil evaluasi Model A	33
Gambar 4.12	Hasil evaluasi Model B	33
Gambar 4.13	<i>F1 Score</i> Model A untuk setiap kelas dalam alfabet (kiri atas: <i>F1 Score</i> huruf A, tersusun alfabetis dari kiri ke kanan)	33
Gambar 4.14	<i>F1 Score</i> Model B untuk setiap kelas dalam alfabet (kiri atas: <i>F1 Score</i> huruf A, tersusun alfabetis dari kiri ke kanan)	34
Gambar 4.15	Contoh huruf yang diprediksi oleh Model B serta kemiripan gestur antara Bisindo (atas) dan ASL (bawah)	36



1.1 Latar Belakang

Sistem Bahasa Isyarat Indonesia (Sibi) adalah sistem bahasa isyarat yang ditetapkan oleh pemerintah Indonesia sebagai bahasa pengantar di Sekolah Luar Biasa (SLB) sejak tahun 1996. Namun, penggunaan Sibi bukan membantu Tuli¹ dalam berkomunikasi, melainkan membatasi hubungan sosial mereka (Utama, 2015). Hal ini dikarenakan Tuli terbiasa menggunakan Bahasa Isyarat Indonesia (Bisindo) sebagai bahasa ibu mereka (Utama, 2015). Pada tahun 2015, perwakilan Tuli melalui organisasi kemasyarakatan Gerakan untuk Kesejahteraan Tunarungu Indonesia (Gerkatin) telah meminta pemerintah untuk mengakui Bisindo sebagai bahasa pengantar resmi di SLB (Utama, 2015). Selain itu, Gerkatin juga berharap agar pemerintah dapat meningkatkan aksesibilitas Tuli dengan menambah jumlah penerjemah serta memperluas pemahaman Bisindo di masyarakat luas (Utama, 2015).

Penelitian ini mengusulkan pemanfaatan teknologi *machine learning* sebagai upaya untuk meningkatkan aksesibilitas Tuli melalui model pembelajaran gestur bahasa isyarat. *Machine learning* merupakan bagian dari riset *artificial intelligence* yang memungkinkan komputer untuk mempelajari suatu data tanpa intervensi manusia untuk membuat prediksi yang akurat dan seolah memiliki kecerdasan (Ehsan Othman, 2018). Algoritma *machine learning* yang digunakan untuk membuat model pembelajaran gestur bahasa isyarat adalah Convolutional Neural Network (CNN). Algoritma ini dipilih atas kesuksesannya dalam permasalahan *image recognition* dan klasifikasi khususnya pada implementasi untuk mengenali gestur manusia (Bheda and Radpour, 2017).

Model pembelajaran yang diusulkan di penelitian ini memerlukan data-data yang relevan yang kemudian digeneralisasi sebagai bahan dari proses pembelajaran (*learning process*) untuk dapat mengenali gestur bahasa isyarat. Dari dua sistem bahasa isyarat yang ada, Sibi memiliki ketersediaan *dataset* lebih banyak dibandingkan Bisindo. Hal ini dikarenakan Sibi mengadopsi sistem bahasa isyarat Amerika, American Sign Language, (ASL), yang ketersediaan *dataset*-nya relatif lebih tinggi dibandingkan Bisindo. Untuk mengatasi ketersediaan *dataset* Bisindo yang relatif sedikit, *data augmentation* dilakukan sebagai upaya untuk meningkatkan jumlah data Bisindo yang digunakan untuk proses pembelajaran. Selain itu, pada penelitian ini juga dikembangkan model dengan menggunakan metode *transfer learning*, yaitu dengan memanfaatkan model yang di-*training* dengan *dataset* ASL agar dapat digunakan kembali untuk melakukan pekerjaan yang sama pada Bisindo. *Transfer learning* adalah metode pembelajaran terhadap suatu tugas baru melalui pemindahan pengetahuan (*transfer of knowledge*) dari tugas serupa yang telah dipelajari sebelumnya (Lisa Torrey, 2009). Penelitian ini mengevaluasi tingkat akurasi pendekripsi Bisindo dengan dan tanpa menggunakan *transfer learning* dari Model ASL.

¹Istilah Tuli yang digunakan di laporan tugas akhir ini yaitu kata ganti orang yang tidak bisa mendengar dan menggunakan bahasa isyarat untuk berkomunikasi.

1.2 Rumusan Masalah

Dalam membuat model pembelajaran Bisindo, penelitian ini menggunakan metode *data augmentation* dan *transfer learning* untuk dapat mengatasi permasalahan ketersediaan *dataset* Bisindo yang terbatas. Penelitian ini menggunakan algoritma CNN dengan model ASL sebagai model yang digunakan sebagai *source domain* yang digunakan untuk *transfer learning* model Bisindo. Istilah yang digunakan dalam penelitian ini: Model ASL adalah model yang di-*training* menggunakan *dataset* ASL; Model A adalah model yang dihasilkan dari *training* menggunakan *dataset* Bisindo yang telah diaugmentasi dengan arsitektur yang sama dengan Model ASL tanpa menggunakan parameter dari Model ASL (**tanpa transfer learning**); Model B merupakan model yang di-*training* menggunakan *dataset* Bisindo yang sama serta menggunakan parameter dari Model ASL (**dengan transfer learning**). Rumusan masalah dari penelitian ini: Bagaimana performa akurasi dari Model B dibandingkan Model A?

1.3 Hipotesis

Model B membutuhkan waktu *training* yang lebih sedikit dibandingkan Model A karena jumlah *trainable parameter* pada Model B jauh lebih sedikit dibandingkan Model A, namun memiliki performa akurasi pendekripsi yang relatif lebih buruk dibandingkan Model A.

1.4 Batasan Masalah

Batasan masalah dari penelitian ini yaitu model yang dihasilkan merupakan model pembelajaran Bisindo. Model ASL hanya digunakan sebagai *source domain* untuk *transfer learning* model Bisindo. Dalam penelitian ini, pembelajaran yang dibuat terbatas pada alfabet Bisindo dengan data yang diolah berupa gambar isyarat Bisindo dari tampak depan. Algoritma yang digunakan untuk mengembangkan model pembelajaran adalah CNN. Pendekatan transfer learning yang digunakan terbatas pada *transferring knowledge of parameter (parameter-transfer)*.

1.5 Tujuan Penelitian

Tujuan yang hendak dicapai yaitu:

1. Bertambahnya jumlah penerjemah Bisindo elektronik dan meningkatnya aksesibilitas Tuli melalui pemanfaatan teknologi *machine learning*.
2. Teridentifikasi dan terevaluasinya akurasi *parameter-transfer* dalam mengatasi keterbatasan jumlah *dataset* dan implementasi pada sistem bahasa isyarat yang berbeda.

1.6 Manfaat Penelitian

Interpreter digital dapat menurunkan ketergantungan terhadap penerjemah manusia. Selain itu, interpreter digital memiliki fleksibilitas yang tinggi sehingga pembelajaran dan penerjemahan bahasa isyarat dapat dilakukan kapanpun dan dimanapun. Kesenjangan dalam berkomunikasi antara pengguna Bisindo dengan masyarakat luas dapat dikurangi dengan adanya interpreter Bisindo digital. Dengan meningkatnya aksesibilitas Tuli, penelitian ini dapat menjadi langkah awal untuk tercapainya salah satu tujuan Sustainable Development Goals yaitu mengurangi kesenjangan (*reduce inequalities*).

1.7 Notasi yang digunakan

Terdapat dua istilah yang digunakan dalam penelitian tugas akhir ini yaitu *domain* dan *task*. *Domain* D terdiri dari dua komponen: *feature space* χ dan distribusi marjinal (*marginal probability distribution*) $P(X)$, dengan $X = \{x_1, x_2, \dots, x_n\} \in \chi$. *Domain* dinotasikan dengan $D = \{\chi, P(X)\}$.

Sebuah *task* terdiri dari dua komponen: *label space* Y dan *objective predictive function* $f(\cdot)$. *Task* dinotasikan dengan $T = \{Y, f(\cdot)\}$.

1. Dilarang mengutip karya tulis ini, kecuali:
 - a. menyebutkan sumber sesuai kaidah kecendekiaan;
 - b. pengutipan hanya untuk keperluan pendidikan, penulisan karya ilmiah atau penelitian;
 - c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.



2.1.1 Sejarah Bahasa Isyarat di Indonesia

Terdapat dua bahasa isyarat yang beredar di Indonesia, yaitu Sibi dan Bisindo. Sibi adalah bahasa isyarat yang diciptakan oleh Anton Widyatmoko selaku mantan Kepala Sekolah SLB/B Widya Bakti Semarang yang bekerjasama dengan mantan Kepala Sekolah SLB/B di Jakarta dan Surabaya (Gilang Gumelar, 2018). Pada tahun 1996, Sibi ditetapkan sebagai bahasa pengantar di seluruh Sekolah Luar Biasa/B (SLB/B), yaitu sekolah yang diperuntukkan bagi anak yang memiliki kekurangan pendengaran atau Tuli. Penyebaran Sibi juga didukung oleh pemerintah melalui penyebaran kamus ke seluruh SLB/B di Indonesia sejak tahun 2001 sehingga menjadikan Sibi sebagai bahasa isyarat yang populer di lingkungan SLB/B di Indonesia.

Dalam implementasinya, penggunaan Sibi tidak sepenuhnya diterima dan digunakan oleh para Tuli karena penggunaannya seringkali menyulitkan mereka disebabkan penerapan kosakata yang tidak sesuai dengan aspirasi dan nurani (Gilang Gumelar, 2018). Ada dua hal yang membuat Sibi sulit digunakan oleh Tuli untuk berkomunikasi. Pertama, Sibi menerapkan penggunaan bahasa baku sesuai dengan tata bahasa Indonesia. Kedua, beberapa kosakata di Sibi dipengaruhi oleh budaya dan isyarat Tuli dari luar negeri yang sulit dimengerti. Oleh karena itu, Sibi lebih sering digunakan sebagai bahasa isyarat di sekolah dan kurang dipergunakan sebagai bahasa isyarat komunikasi sehari-hari Tuli dalam berkomunikasi (Gilang Gumelar, 2018).

Berbeda dengan Sibi, Bisindo dianggap lebih mewakili budaya Tuli Indonesia karena Bisindo terbentuk secara alami dari interaksi Tuli dengan lingkungannya sejak kecil (Gilang Gumelar, 2018). Bisindo juga memiliki keunikan layaknya bahasa daerah karena isyarat pada Bisindo dipengaruhi oleh interaksi nilai-nilai tiap daerah. Oleh karena itu, Bisindo terus diperjuangkan oleh Gerkatin sebagai budaya Tuli agar dapat diakui secara nasional dan dapat digunakan sebagai bahasa pengantar di seluruh SLB/B di Indonesia (Utama, 2015).

2.1.2 Penggunaan Bisindo

Bisindo digunakan sebagai bahasa isyarat dalam komunikasi sehari-hari bagi Tuli dengan menggunakan gerakan kedua tangan dan ekspresi wajah. Seperti bahasa daerah, ungkapan Bisindo memiliki keberagaman dalam penggunaan di setiap daerah. Berbeda dengan Sibi yang menggunakan pengejaan hingga tingkat imbuhan dan susunan kalimat sesuai dengan Ejaan Bahasa Indonesia (EBI), Bisindo tidak mengikuti kaidah EBI dan lebih mengedepankan kesederhanaan serta ekspresi yang menunjukkan kejadian yang sedang berlangsung. Ekspresi inilah yang menyebabkan isyarat berbeda-beda di setiap daerah karena isyarat terbentuk atas interaksi nilai-nilai dari tiap daerah.

Meski beragam, terdapat isyarat-isyarat yang ditetapkan untuk memudahkan komunikasi Tuli seperti: alfabet. Alfabet merupakan dasar dari pembelajaran bahasa baik bahasa isyarat maupun bahasa verbal. Oleh karena itu, penelitian ini memilih alfabet Bisindo sebagai langkah awal untuk mengembangkan model pembelajaran yang dapat meningkatkan aksesibilitas Tuli pengguna Bisindo.

2.1.3 Isyarat Alfabet Bisindo

Alfabet Bisindo tersusun dari kombinasi gerakan kedua tangan, berikut adalah gambar dari gerakan tangan dari alfabet Bisindo.



Gambar 2.1. Alfabet Bisindo (Almuhammad, 2013)

- 2.
- c. pengutipan tidak merugikan Universitas Pertamina.
- Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

ya ilmiah atau penelitian;

Copyright of Universitas Pertamina

2.1.4 Sudut Pandang yang digunakan

Pada Gambar 2.1, sudut pandang umum yang digunakan dalam berkomunikasi menggunakan bahasa isyarat adalah tampak depan sehingga data gambar yang digunakan di penelitian ini juga memuat gestur Bisindo dari tampak depan.

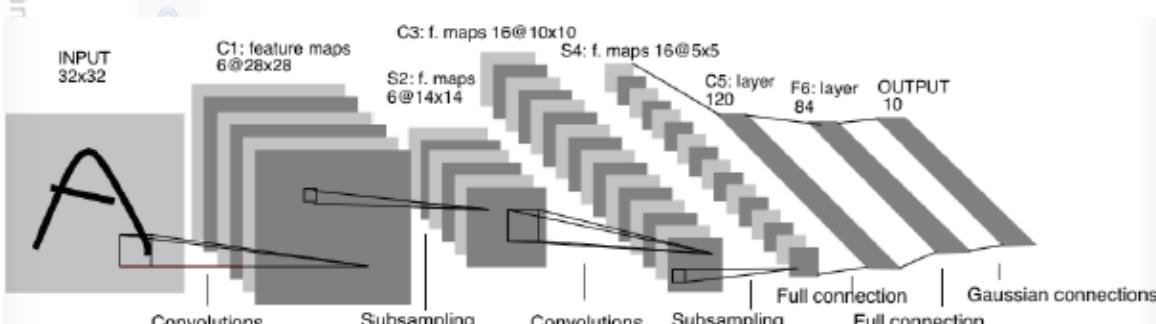
2.2 CNN

CNN merupakan salah satu teknik dari *neural network*. Konsep *neural network* sudah ada semenjak tahun 1950 saat Frank Rosenblatt membuat Perceptron (Fan, 2014). CNN pertama kali digunakan pada tahun 1989 untuk mengenali ZIP *code* dalam bentuk tulisan tangan yang kemudian berkembang hingga saat ini untuk melakukan *task* seperti mengenali dan melakukan klasifikasi berbagai objek seperti tulisan tangan untuk angka (Fan, 2014). Beberapa komponen utama dari CNN dijelaskan pada sub-bab berikut ini.

2.2.1 Feature Maps dan Weight Sharing

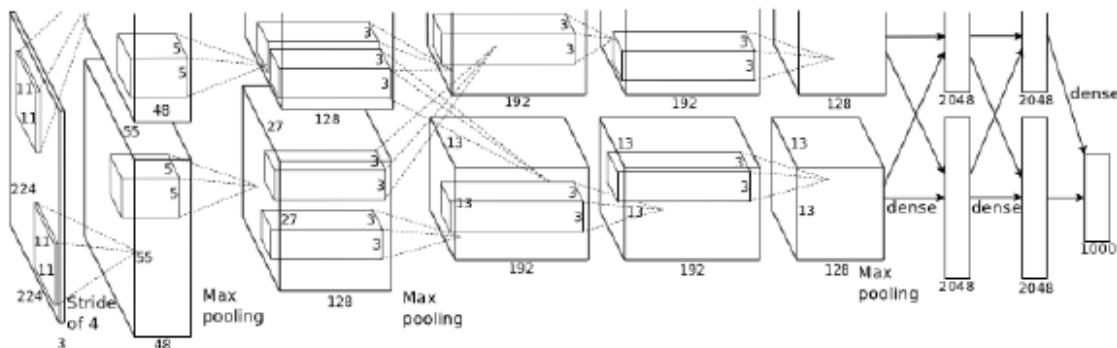
Pada *neural network*, *hidden units* yang dikenal sebagai *neuron* disusun sebagai vektor satu dimensi. Berbeda dengan *neural network*, *hidden unit* pada CNN disusun dalam bidang dua dimensi (*2D planes*) yang disebut sebagai *feature maps* untuk mengikuti bentuk dasar dari masukannya, yaitu gambar (*image*). Ukuran *feature maps* pada suatu layer konvolusi (*convolutional layer*) beragam dan bergantung kepada lebar (*width*) dari layer tersebut.

Feature maps diperoleh melalui operasi konvolusi pada *input image* atau hasil fitur pada tahap sebelumnya menggunakan *linear filter*, lalu menambahkan bias kemudian dihitung menggunakan fungsi non-linier. Setiap unit pada *feature maps* menerima input dari kombinasi dari subset atau keseluruhan *feature maps* dari layer sebelumnya. Area hasil kombinasi tersebut disebut sebagai *receptive fields* dari unit tersebut. Berikut adalah ilustrasi untuk menjelaskan keterhubungan antar unit tersebut.



Gambar 2.2. Arsitektur LeNet (Lecun et al., 1998).

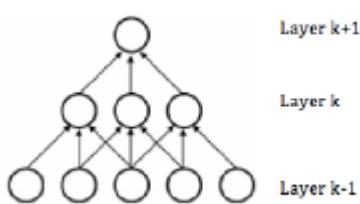
Terdapat dua *convolutional layer* dan dua *sub-sampling layer* pada keempat layer pertama dari arsitektur LeNet. *Activation layer* terdapat pada setiap layer (tidak ditampilkan dalam gambar). Pada layer *sub-sampling* terakhir ditambahkan dua *fully connected layer* untuk vektorisasi (membentuk vektor) dari hasil representasi gambar. Layer terakhir dan *output layer* terdiri dari *Euclidean Radial Basis Function (RBF)*, unit yang akan menghasilkan *Euclidean distance* antara *network* dan *output*, serta hasil klasifikasi berupa *truth table* untuk sepuluh kelas (Fan, 2014).



Gambar 2.3. Arsitektur ConvNet (A. Krizhevsky, 2012).

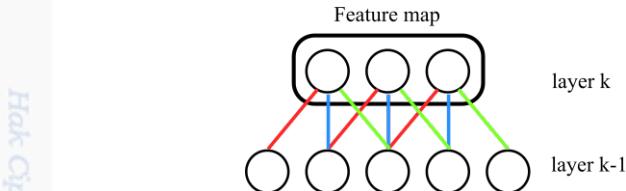
Arsitektur ConvNet terdiri dari lima *convolution layer* dan tiga *sub-sampling layer (max-pooling layer)* pada setiap layer dalam model. *Activation layer* terdapat setelah *convolutional layer* (tidak ditampilkan dalam gambar). Pada layer terakhir dari *sub-sampling* ditambahkan dua *fully connected layer* untuk vektorisasi hasil representasi gambar layaknya pada arsitektur LeNet namun dengan dimensi yang lebih tinggi. Layer terakhir dan *output layer* menghasilkan *softmax loss* dari *network prediction* untuk 1000 kelas. (Fan, 2014)

Berdasarkan Gambar 2.2 dan Gambar 2.3, ukuran *feature maps* yang terbentuk mengikuti lebar dari *input* atau unit layer sebelumnya dan jenis operasi yang dilakukan pada layer tersebut. Keseluruhan unit dalam *feature maps* akan menggunakan *weight* yang sama karena *convolutional filter* yang sama digunakan pada seluruh *receptive fields* dari *feature maps* sebelumnya.



Gambar 2.4. Konektivitas antara layer yang bertetangga (Fan, 2014).

Gambar 2.4 menunjukkan bahwa setiap unit di layer- k menerima input dari 3 unit pada layer $k-1$, oleh karena itu layer- k memiliki 3 *receptive fields*. Sama seperti pada layer- k , setiap unit pada layer



Gambar 2.5. Contoh *weight sharing* dari sebuah *feature map* (Fan, 2014)).

$k + 1$ juga memiliki 3 *receptive field* terhadap layer- k , namun memiliki 5 *receptive field* terhadap layer $k - 1$. Layer k dan layer $k - 1$ terhubung secara *sparsely connected* sedangkan layer $k + 1$ dan layer k terhubung secara *fully connected*.

Feature map pada layer m berisi tiga unit dan setiap unit memiliki tiga *receptive fields* dari layer $m - 1$ yang sudah dikonvolusi dengan filter berukuran 3. Filter digunakan pada seluruh unit dalam feature map, oleh karena itu garis pada Gambar 2.5 dengan warna yang sama memiliki nilai *weight* yang sama.

Interval pada *receptive field* dari *neighbouring unit* disebut dengan *stride*. *Receptive field* dari dua unit yang bertetangga dalam *feature map* yang sama memungkinkan terjadinya *overlap* jika ukuran *stride* lebih kecil dari lebar *receptive field*.

Shift invariance diperoleh melalui *weight sharing* dari *input image* yang di-*shift*, menyebabkan *feature maps* juga akan bergeser dengan jumlah yang sama. Dengan mempertimbangkan input berupa gambar, fitur elementer visual seperti *edge* dan *corner* akan diekstrak terlebih dahulu kemudian di-gabungkan pada layer lebih atas untuk membentuk *high-level feature*. Karena adanya kemungkinan bahwa fitur elementer tersebut dapat muncul pada berbagai bagian dari gambar, *shift invariance* penting untuk dipenuhi agar dapat mengambil keseluruhan fitur elementer visual pada gambar.

Weight sharing memiliki manfaat untuk mengurangi banyaknya parameter bebas/*free parameters* secara signifikan. Pada Gambar 2.2, *convolutional layer C3* memiliki 16 *feature maps*, setiap unit dalam *feature map* terhubung dengan 5×5 neighborhood di beberapa bagian dari *feature map* di layer sebelumnya yaitu *S2*. Banyaknya *free parameters* yang mungkin dapat dihitung melalui (ukuran *kernel+bias*) $\times |S2| \times |C3| = (5 \times 5 + 1) \times 6 \times 16$ termasuk dalam kompleksitas $\mathcal{O}(|S2||C3|)$. Dengan ukuran *kernel* merupakan bilangan bulat yang relatif kecil terhadap ukuran *input*, jumlah *free parameters* untuk proses *learning* tidak akan meningkat secara eksponensial terhadap ukuran *input*.

2.2.2 Sub-sampling pada CNN

Sub-sampling atau *pooling* merupakan salah satu konsep yang penting di CNN. Dengan melakukan *sub-sampling*, dimensi dari fitur dapat dikurangi sehingga beban komputasi (*computational cost*) dapat dikurangi melalui penghilangan *non-maximum local values*. Selain itu, *sub-sampling* dapat menyediakan *robustness* terhadap *noise* dan distorsi pada *feature maps*.

2.2.3 Loss Function

Selain *sub-sampling*, salah satu bagian yang juga penting dalam mendesain *neural network* adalah mengevaluasi performa dari keseluruhan sistem atau dalam kata lain menghitung *error* dalam sistem. Jenis *loss function* yang digunakan bergantung pada jenis *task* dari model yang dibuat. Untuk *neural network* dengan *task* regresi atau klasifikasi, *loss function* didefinisikan sebagai l untuk sebuah *training data*, dan L untuk dataset x .

$$L(x, W) = \frac{1}{N} \sum_{i=0}^N l(x_i, W) \quad (2.1)$$

- Minimasi *loss function* terhadap parameter model merupakan permasalahan optimisasi yang dinyatakan sebagai persamaan berikut.

$$W = \arg \min_w L(x, W) \quad (2.2)$$

Sum of Squares

Sum of squares merupakan *loss function* yang menghitung penjumlahan dari kuadrat dari selisih antara *predicted output* dari *neural network* dan *ground truth table* dari data pada skema *supervised learning*. *Sum of squares* didefinisikan sebagai berikut.

$$L = \frac{1}{2N} \sum_{I=1}^N ||O_I - y_I||^2 \quad (2.3)$$

Pada persamaan 2.3 diasumsikan bahwa *output* adalah $\mathbf{o} = o_1, o_2, \dots, o_n$ dan *ground truth table* adalah $\mathbf{y} = y_1, y_2, \dots, y_n$. *Sum of squares* menghitung *fitness* antara model dan data untuk *task regresi*.

Softmax

Layer *Softmax* menghitung *cross entropy* dari *softmax output neural network* dan biasanya digunakan untuk model dengan *task* klasifikasi. Fungsi *softmax* didefinisikan sebagai berikut.

$$L = - \sum_{j=1} y_j \log p_j Dengan P_j = \frac{e^{oj}}{\sum k e^{ok}} \quad (2.4)$$

Pada persamaan 2.4, *output vector* diasumsikan sebagai $\mathbf{o} = o_1, o_2, \dots, o_n$ dengan output softmax adalah $\mathbf{p} = p_1, p_2, \dots, p_n$ dengan bentuk sebagai *1-of-N encoding* dan dengan batasan $\sum_k o_k = 1$ dan $\sum_k p_k = 1$. Salah satu manfaat dalam menggunakan *softmax output* adalah total nilai dari seluruh *output unit* sama dengan 1 sehingga *output* dapat diinterpretasikan sebagai *probability distribution* dari seluruh kelas yang terdapat dalam problem *domain* klasifikasi.

2.3 Gradient-Based Parameter Optimization

Learning goal dari CNN adalah untuk meminimalkan *loss function* berdasarkan *network parameter* di setiap layer. *Gradient learning method* dikembangkan atas dasar bahwa *loss function* pada persamaan 2.1 dapat diminimalkan dengan mengestimasi efek dari variasi yang rendah dari nilai parameter *loss function*. Dua metode *gradient-based* yang sering digunakan adalah *Gradient Descent* dan *Stochastic Gradient Descent*.

2.3.1 Gradient Descent

1. *Gradient descent* melakukan optimasi berdasarkan *loss function* pada persamaan 2.1 sebagai fungsi dari parameter model W . Dasar dari *gradient-based updating method* adalah dengan melakukan *update* nilai W dengan *negative gradient* dari *loss function* pada suatu *mini-batch*.

$$W_{t+1} = W_t - \epsilon \frac{\partial L(W_t)}{\partial W} \quad (2.5)$$

Pada persamaan 2.5, ϵ adalah bilangan tetap dengan nilai yang relatif kecil, disebut sebagai *learning rate* atau *learning step*.

2.3.2 Stochastic Gradient Descent

Stochastic Gradient Descent merupakan perbaikan dari metode *gradient descent*. Pada *stochastic gradient descent*, nilai W di-*update* dengan menggunakan kombinasi linear dari *negative gradient* $\Delta L(W)$ dan *weight update* sebelumnya $\Delta(W_t)$ sehingga:

$$\Delta W_{(t+1)} = \mu \Delta W_t - \epsilon \nabla L(W_t) \quad W_{(t+1)} = W_t + \Delta W_{(t+1)} \quad (2.6)$$

Pada persamaan 2.6 *learning rate* ϵ adalah *weight* dari *negative gradient*. Momentum μ adalah *weight* dari hasil *update* sebelumnya.

2.4 Regularization

Regularization atau Regularisasi adalah metode yang umum digunakan pada bidang statistik untuk mengontrol kompleksitas model untuk menghindari *overfitting*. *Overfitting* adalah fenomena ketika *training error* menurun secara drastis sementara error pada *validation set* meningkat pada suatu titik tertentu. Semakin banyak parameter dari sebuah model, maka semakin tinggi resiko model *overfitting* data yang diberikan.

Salah satu metode yang dapat mengurangi *overfitting* yaitu dengan melakukan regularisasi berdasarkan pengukuran error untuk mengatur kompleksitas model. *Loss function* dengan menggunakan regularisasi dituliskan sebagai berikut.

$$L_{reg}(W) = L(W) + \eta r(W) \quad (2.7)$$

$L(W)$ adalah *loss function* yang telah dibahas pada subbab 2.2.3, dengan $r(W)$ adalah regularisasi parameter W dan η adalah level dari regularisasi.

2.5 Data Augmentation

Augmentasi data merupakan salah satu teknik yang dapat mengurangi *overfitting* dengan meningkatkan ukuran dataset dalam upaya yang seminimum mungkin. Data biasanya diaugmentasi dengan melakukan transformasi pada data atau dengan kata lain membuat salinan dari sumber data tanpa mengubah label yang tertera pada setiap bagian dari data tersebut. *Cropping* dan *flipping* merupakan dua transformasi yang paling sering digunakan dalam proses augmentasi data (A. Krizhevsky, 2012).

2.6 Dropout

Dropout merupakan salah satu teknik regularisasi yang melakukan skip beberapa *hidden neuron* pada saat iterasi *training* dengan probabilitas p . Neuron yang di-*skip* pada saat iterasi disebut dengan '*dropped out*', neuron ini tidak akan memiliki pengaruh baik pada *forward pass* atau *backward pass* pada iterasi tersebut.

2.7 Layer pada CNN

Dalam membuat model CNN, jenis layer yang tersusun akan mempengaruhi proses ekstraksi informasi dari gambar input. Sebagai dasar dari CNN, layer-layer pada CNN telah membuktikan keberagaman dan fleksibilitas baik dalam segi struktur maupun koneksi antar layer. Memodifikasi struktur layer dan koneksi dapat membawa model untuk dapat di-*training* lebih cepat dan membuat performa yang baik. Umumnya, layer pada CNN terdiri dari layer berikut.

2.7.1 Activation Layer

Activation layer pada umumnya merupakan sebuah fungsi aktivasi di bagian atas sebuah *output layer*. Tujuan utama dari penggunaan fungsi aktivasi ini adalah untuk membentuk *non-linearity* (ketidaklinieran) pada *neural network*. Tanpa adanya fungsi aktivasi, *neural network* hanya akan melakukan transformasi linier dari input ke output. Secara matematis, fungsi aktivasi ditulis sebagai berikut.

$$x^l = f(x^{l-1}) \quad (2.8)$$

Beberapa jenis fungsi aktivasi yang umum digunakan adalah RelU (*Rectified-Linear Unit*), Sigmoid dan TanH (*Hyperbolic Tangent*).

Sigmoid

Fungsi Sigmoid akan memetakan input ke interval $[0, 1]$. Fungsi sigmoid didefinisikan sebagai berikut.

$$\sigma(z) = \frac{1}{1 + \exp -z} \quad (2.9)$$

TanH

Fungsi TanH atau *hyperbolic tangent function* merupakan fungsi transformasi linier dari sigmoid ke interval [-1, 1]. Fungsi TanH dapat didefinisikan sebagai berikut.

$$\tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.10)$$

ReLU

Permasalahan yang ada pada fungsi sigmoid dan tanh yaitu output yang mudah jenuh ke angka 0 atau 1 pada sigmoid dan +1 dan -1 pada tanh. Output yang jenuh akan membuat hilangnya gradien dan menyebabkan turunnya kecepatan *training* dan memungkinkan untuk menjebak model pada area *local minimum*.

ReLU diperkenalkan pada tahun 2010 untuk meningkatkan kecepatan konvergensi dari *training neural networks* (V. Nair, 2010). ReLU memiliki kelebihan tidak akan jenuh pada suatu nilai dan memungkinkan untuk melakukan *training neural network* berukuran besar layaknya CNN.

ReLU didefinisikan sebagai berikut.

$$f(x) = \max(0, x) \quad (2.11)$$

Pada CNN, *layer output* pada umumnya merupakan *feature maps* dimensi dua. Pada persamaan 2.11, x adalah matrix dan max adalah elemen *twise* yang diterapkan pada setiap elemen matriks. Varian lain dari ReLU yaitu LeakyReLU memungkinkan adanya nilai yang kecil, bukan nol meski unit sedang tidak dalam keadaan aktif. Leaky ReLU didefinisikan sebagai berikut.

$$\begin{cases} x, & x > 0 \\ 0.01x & \text{selain } x > 0 \end{cases} \quad (2.12)$$

2.7.2 Pooling Layer

Pooling layer disebut juga sebagai *sub-sampling layer* atau *down-sampling layer* adalah layer yang digunakan untuk melakukan sub-sampel *feature maps* yang dihasilkan dari layer sebelumnya. *Pooling layer* memberikan manfaat bagi CNN dalam dua cara yaitu pertama mengurangi beban komputasi pada layer atas dengan mengurangi dimensi dari *feature maps*. Kedua, pooling layer juga memberikan *robustness* pada model terhadap *noise* dan *small distortions*.



Terdapat dua jenis *pooling layer* yang umum digunakan yaitu *average pooling* dan *max pooling*. *Average pooling* digunakan untuk menghitung nilai rata-rata dari setiap *pooling window* di seluruh *feature maps* dari layer l-1 sebagai unit yang berhubungan dengan layer l. Serupa dengan *average pooling*, *max pooling* digunakan untuk menghitung nilai maksimum dari setiap *pooling window* pada seluruh *feature maps* dari layer l-1.

2.7.3 Fully Connected Layer

Pada umumnya, *feature maps* dari *convolutional layer* terakhir divektorisasi (dibentuk menjadi vektor) dan dihubungkan secara penuh (*fully connected*) dengan output unit yang diikuti dengan *softmax loss* layer setelahnya. *Fully connected layer* juga disebut sebagai *special case* dari *densely connected convolutional layer* karena memiliki ukuran filter yang sama dengan *feature maps*.

2.7.4 Convolutional Layer

Pada *convolutional layer* l, *feature maps* dari layer sebelumnya dikonvolusi menggunakan *learnable kernel* untuk membentuk *output feature layer* tersebut. Operasi konvolusi dari fitur layer l-1 dengan layer l dapat dinyatakan sebagai berikut.

$$x_i^l = \sum_{j \in M_i} k_{ij}^l * x_j^{l-1} + b_i^l \quad (2.13)$$

Dengan $i = 1, 2, \dots, |M|$. $|M|$ adalah banyaknya *feature maps* pada layer l. ukuran kernel dinotasikan dengan k, menentukan ukuran dari *receptive fields* yang umumnya merupakan bilangan yang relatif kecil terhadap ukuran *feature maps*.

2.8 Transfer Learning

Teknologi *machine learning* telah mencapai kesuksesan di berbagai bidang disiplin ilmu khususnya dalam pengembangan model untuk klasifikasi, regresi dan *clustering* (X. Wu, 2008). Untuk membuat model tersebut, diperlukan data-data yang relevan dan seringkali dijumpai pengumpulan ulang data yang dibutuhkan untuk *training* dan pembuatan ulang model merupakan kegiatan yang memerlukan biaya yang banyak. Pada kasus seperti ini, *knowledge transfer* atau *transfer learning* antar task merupakan solusi yang dapat digunakan untuk mengatasi permasalahan tersebut.

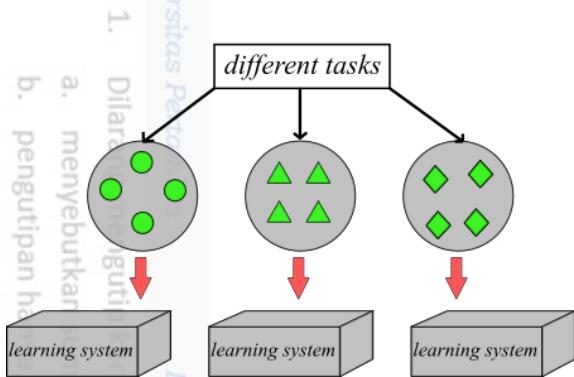
Kebutuhan akan *transfer learning* meningkat ketika data mudah kadaluwarsa/*outdated*. Kadaluwarsa dalam arti bahwa *labeled* data yang diperoleh pada suatu kurun waktu tertentu mungkin tidak memiliki distribusi yang sama dalam beberapa waktu kedepan (Sinno Jialin Pan, 2009).

2.8.1 Sejarah Transfer Learning

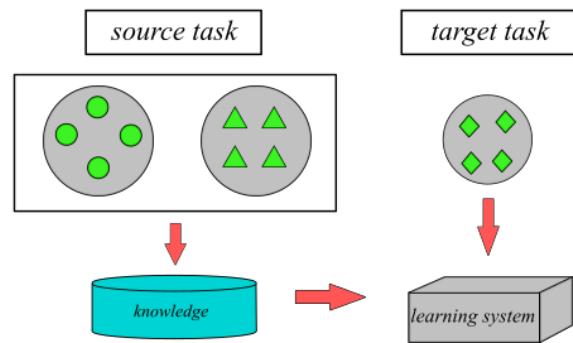
Algoritma *machine learning* membuat prediksi menggunakan model statistik yang di-*train* terlebih dahulu menggunakan *labeled* ataupun un *labeled* *training data*. Pada lingkungan *semi - super-*

vised classification, permasalahan yang mungkin terjadi adalah *labeled* data terlalu sedikit (jumlah *labeled* jauh lebih sedikit dari un *labeled* data) sehingga sulit untuk mebangun algoritma yang baik (Zhu, 2008). Sementara itu, *transfer learning* memungkinkan *domain*, *task* dan *distribusi* yang digunakan baik pada *training* dan *testing* data berbeda. Penelitian mengenai *transfer learning* termotivasi dari fakta bahwa manusia dapat secara cerdas mengaplikasikan pengetahuan yang dipelajari sebelumnya untuk mengatasi permasalahan yang baru dengan cara yang lebih cepat atau lebih baik dari sebelumnya.

Learning Process of Traditional Machine Learning



Learning Process of Transfer Learning



Gambar 2.6. Perbedaan *learning process* antara *traditional machine learning* dan *transfer learning* (Sinno Jialin Pan, 2009)

Pada gambar 2.6 dapat dilihat bahwa *traditional machine learning* melakukan pembelajaran untuk suatu *task* dari nol (*scratch*) sementara *transfer learning* mencoba untuk melakukan *transfer of knowledge* dari beberapa *task* sebelumnya yang ke *target task* ketika kualitas *training data* memiliki kualitas yang lebih rendah dari *training data* pada *task* sebelumnya (Sinno Jialin Pan, 2009).

2.8.2 Definisi Transfer Learning

Transfer *learning* didefinisikan sebagai berikut. Diberikan source *domain* D_s dan *learning task* T_s , target *domain* D_t dan *learning task* T_t , *transfer learning* ditujukan untuk meningkatkan kemampuan pembelajaran dari predictive function $f_t(\cdot)$ pada D_t menggunakan *knowledge*/pengetahuan dari D_s dan T_s , dengan kondisi $D_s \neq D_t$ atau $T_s \neq T_t$ (Sinno Jialin Pan, 2009). Contoh perbedaan yang mungkin terjadi pada *domain* adalah penggunaan bahasa yang berbeda atau *marginal probability distribution* yang berbeda antar source *domain* dan target *domain*.

Pada *transfer learning*, terdapat tiga isu untuk diteliti lebih lanjut: (1) apa yang akan di-*transfer*; (2) Bagaimana cara transfer; (3) Kapan harus melakukan transfer. Pada isu pertama, akan ditentukan bagian dari *knowledge* yang bisa di-*transfer* antar *domain* atau *task*. Beberapa *knowledge* spesifik pada individual *domain* atau *task*, beberapa *knowledge* mungkin mirip antar *domain* sehingga *knowledge* tersebut dapat digunakan untuk meningkatkan performa untuk target *domain* atau *task*. Algoritma *learning* perlu dikembangkan untuk menjawab permasalahan bagaimana transfer dapat dilakukan. Situasi dan *transferring skill* yang dibutuhkan untuk melakukan transfer merupakan dua aspek yang perlu diteliti dari isu penelitian kapan transfer harus dilakukan. Dalam kata lain, pe-

nelitian dilakukan untuk menjawab *knowledge* apa yang seharusnya tidak di- *transfer*. Kasus paling buruk adalah ketika transfer justru malah mengurangi performa dari *target domain learning*.

Beberapa pendekatan yang umum dilakukan dalam melakukan *transfer learning* yaitu melalui transfer pengetahuan (*transferring knowledge of instances*), transfer representasi fitur (*transferring knowledge of feature representation*), transfer pengetahuan dari parameter (*transferring knowledge of parameters*) dan transfer pengetahuan relasional (*transferring relational knowledge*).

Transferring Knowledge of Instances

2.

Pendekatan ini memanfaatkan bagian dari source *domain* data yang masih dapat digunakan kembali dengan *labeled* data yang ada pada *target domain* .

Transferring Knowledge of Feature Representations

Pendekatan *feature-representation- transfer* dalam *inductive transfer learning* menargetkan untuk mencari representasi fitur yang baik untuk meminimalkan divergensi dari *domain* dan error pada klasifikasi dan regresi. Untuk mencari representasi fitur yang baik, perlu mempertimbangkan kondisi dari data *source domain*. Jika *labeled* data tersedia dalam jumlah banyak dalam *source domain*, pembentukan representasi fitur dapat menggunakan metode *supervised learning*. Konsep dasar dari metode ini yaitu untuk mempelajari representasi dengan dimensi rendah yang digunakan dalam berbagai *task* yang serupa.

Sebaliknya, jika *labeled* data tidak tersedia pada *source domain* maka untuk membentuk representasi fitur dapat menggunakan *unsupervised learning*. Pada metode ini, *high-level feature* dipelajari melalui *sparse coding* (H. Lee, 2007). Ide yang mendasari *unsupervised learning feature construction* terdiri dari dua langkah, pertama, vektor basis level tinggi (*higher-level basis vector*) $b=b_1,b_2,\dots,b_s$ dipelajari pada *source domain* melalui perhitungan persamaan optimasi, kemudian *higher level features* pada basis vektor b dicari menggunakan persamaan optimasi. Pada tahap akhir, algoritma diskriminatif digunakan pada hasil tahap sebelumnya untuk mencari *higher level feature representation* untuk digunakan pada saat *training*, baik untuk label yang digunakan pada kasus klasifikasi atau model regresi yang digunakan pada *target domain* .

Transferring Knowledge of Parameters

Pada umumnya, pendekatan *transfer parameter* dalam *inductive transfer learning* dilakukan dengan menggunakan asumsi bahwa sebuah model dengan suatu *task* yang memiliki keterkaitan dengan suatu *task* lain, keduanya memiliki kemiripan pada beberapa parameter atau distribusi *hyper-parameter*. Pada pendekatan *multi-task learning*, *weight* dari *loss function* untuk *source* dan *target domain* bernilai sama, namun pada transfer learning, *weight* dari *source* dan *target* mungkin berbeda.

Pada pendekatan ini tidak menggunakan asumsi bahwa data yang diambil dari setiap *domain* bersifat independen dan terdistribusi secara identik. Pendekatan ini mencoba untuk melakukan transfer hubungan antara *source domain* data ke sebuah *target domain*.

2.8.3 Teknik-teknik *Transfer Learning*

Inductive Transfer Learning

Dalam *inductive transfer learning*, *target task* berbeda dengan *source task*, tidak memandang perbedaan dari *source domain* dan *target domain*. Kasus yang mungkin terjadi dalam *inductive transfer learning* yaitu:

1. *Labeled* data dengan jumlah banyak tersedia pada *source domain*. Pada kasus seperti ini, *inductive transfer learning setting* memiliki kemiripan dengan *multi-task learning setting*. Pada *inductive transfer learning* hanya bertujuan untuk mencapai performa yang tinggi pada *target task* melalui pemindahan *knowledge* dari *source task* sedangkan pada *multi-task learning* mencoba untuk mempelajari *target* dan *source task* secara bersamaan.
2. Tidak tersedianya *labeled* data pada *source domain*. Pada kasus ini, *inductive transfer learning setting* memiliki kemiripan dengan *self-taught learning*. Pada *setting* ini, *label space* diantara *source* dan *target domain* mungkin berbeda (dalam hal ini, data pada *source domain* tidak dapat langsung digunakan karena tidak adanya label pada data).

Transductive Transfer Learning

Pada *transductive transfer learning*, *source* dan *target task* sama, sementara pada *source* dan *target domain* berbeda. Pada situasi seperti ini, *labeled data* pada *target domain* tidak tersedia sementara pada *source domain* *labeled* data tersedia. Kasus yang mungkin terjadi pada *transductive transfer learning* yaitu:

1. *Feature space* antara *source* dan *target domain* berbeda, $\chi_s \neq \chi_t$.
2. *Feature space* antar *domain* sama, $\chi_s = \chi_t$, namun *marginal probability distributions* dari input data berbeda, $P(X_s) \neq P(X_T)$. Kasus ini memiliki kemiripan dengan *domain adaptation* untuk *transfer knowledge* seperti pada kasus klasifikasi teks (R. Raina, 2007).

Unsupervised Transfer Learning

Unsupervised transfer learning memiliki kemiripan dengan *inductive transfer learning setting*, yaitu *target task* yang berbeda namun memiliki keterkaitan dengan *source task*. Pada *unsupervised transfer learning* model berfokus pada penyelesaian *unsupervised learning task* pada *target domain*.

2.9 Batasan Transfer dan Negative Transfer

Negative transfer terjadi ketika data dan *task* dari *source domain* menurunkan performa dari pembelajaran pada *target domain* Sinno Jialin Pan (2009) . Selain mempertimbangkan data yang tersedia pada *source* dan *target domain* , jenis *task* juga perlu dipertimbangkan. Jika dua *task* sangat jauh dari kemiripan, maka *brute-force transfer* dapat menyebabkan penurunan performa pada target *task* (M. T. Rosenstein, 2005).

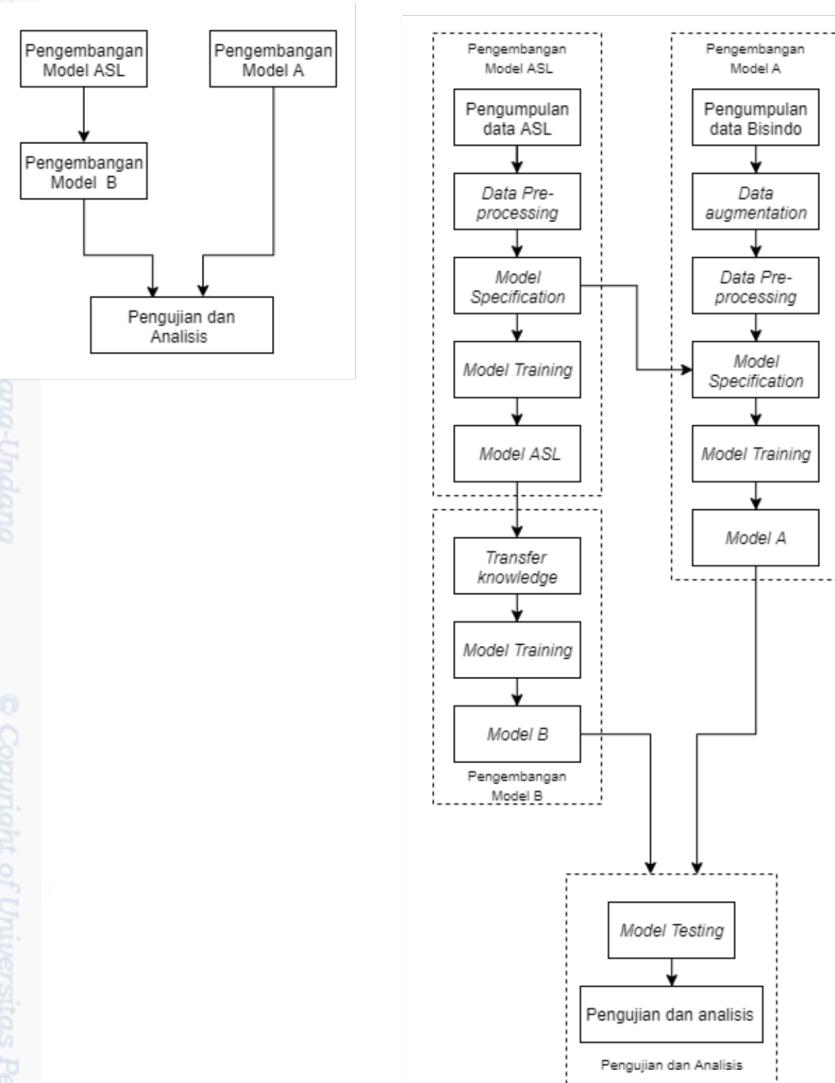
2.10 Penelitian sebelumnya mengenai Model Penerjemah Bisindo

(T. Handhika, 2018) mengusulkan model penerjemah gestur Bisindo melalui implementasi visi komputer (*computer vision*) dengan translasi mesin menggunakan *Hidden-Markov Model* (HMM). Jumlah *hidden state* dihitung menggunakan *Bayesian Inference Criterion* (BIC). Data yang digunakan merupakan gestur Bisindo dari 25 kata yang dikumpulkan menggunakan Microsoft Kinect Xbox. Model tersebut berhasil memperoleh akurasi sebesar 60%.

2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

3.1 Metodologi Penelitian

Penelitian ini dilakukan dengan melakukan eksperimen pembuatan Model A dan Model B. Model A dikembangkan dari algoritma *machine learning* CNN dengan menggunakan arsitektur yang sama dengan Model ASL yang telah teruji akurasinya. Model B dikembangkan melalui metode *transfer learning* dari Model ASL. Pada tahap akhir penelitian ini akan dibandingkan performa durasi *training*, akurasi saat *testing* dan *F1 score* setelah *training* untuk menentukan model Bisindo terbaik.



Gambar 3.1. Diagram tahapan penelitian (kanan: detail proses dari gambar kiri)

- ang-Undang © Copyright of Universitas Pertamina
- Dalam mengutip karya tulis ini, kecuali:
- c. pengutipan tidak merugikan Universitas Pertamina.
 2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

3.1.1 Pengembangan Model ASL

Pengumpulan Data ASL

Salah satu kriteria yang penting untuk dipertimbangkan dalam menentukan dataset yang akan digunakan pada Model ASL adalah jumlah dataset yang tersedia. Model ASL akan menjadi *source domain* untuk *transfer learning* Model B sehingga diperlukan data dengan jumlah yang banyak untuk setiap kelas dalam alfabet untuk dapat menghasilkan knowledge yang baik. Tidak ada angka tertentu yang harus dipenuhi dalam jumlah data, namun sebaiknya ada minimal 1000 gambar dalam satu kelas untuk suatu model yang akan digunakan sebagai *source domain* pada proses *transfer learning* (Ullah, 2017).

Data Pre-processing

Sebelum data diinput ke dalam model, terdapat beberapa proses yang dilakukan:

1. Memberi label pada data

Pemberian label dilakukan dengan membuat *image array* dan label. Untuk setiap gambar dalam folder/kelas yang sama akan diberi label yang sama. Contoh untuk folder A, setiap gambar akan diberi label 0.

2. Mengubah resolusi gambar menjadi persegi ($n \times n$).

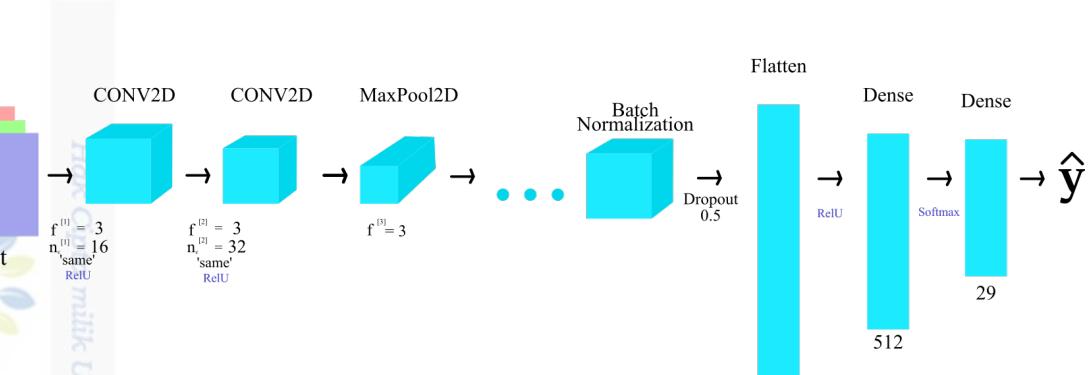
Penyesuaian resolusi gambar disesuaikan dengan kriteria layer pertama pada model.

3. Membagi data menjadi dua bagian: *data train* dan *data test*.

Setelah data berhasil diberi label, data kemudian dibagi dengan proporsi 0.88 untuk *training* dan 0.12 untuk *testing*.

Model Specification

CNN merupakan arsitektur yang memiliki reputasi yang baik dalam implementasi mengenali gestur tubuh manusia (Bheda and Radpour, 2017). Model dikembangkan menggunakan arsitektur CNN dengan 6 *convolution layer*, tiga *max pooling layer*, satu *batch normalization layer*, satu *flatten* dan *dropout layer*, dua layer *dense* dengan fungsi aktivasi ReLU dan Softmax (Modi, 2019). Model di-*compile* menggunakan *Adam optimizer* dan *loss function Categorical Crossentropy*. Berikut adalah skema dari arsitektur CNN yang digunakan untuk pengembangan Model ASL.



Gambar 3.2. Arsitektur CNN yang digunakan (Modi, 2019)

- 1.
 2. **Model training**
- c. pengutipan tidak merugikan Universitas Pertamina.
- Dilarang mengubahnya tanpa izin, memberikan perekembangan pada karya tulisnya.

3.1.2 Pengembangan Model A

Pengumpulan Data Bisindo

Dataset Bisindo diambil melalui proses observasi. Dalam proses observasi, penggunaan alfabet Bisindo khususnya pada gerakan tangan akan didokumentasikan dalam bentuk gambar dengan alat bantu kamera. Dataset yang diharapkan dari proses observasi ini adalah data tanpa *background clutter* dan data yang dikumpulkan adalah data positif yaitu gambar yang terkumpul hanya data yang memuat alfabet Bisindo.

Data Augmentation

Data yang telah dikumpulkan pada tahap sebelumnya kemudian diaugmentasi untuk menghindari *overfitting* (lihat sub bab 2.4) akibat jumlah data untuk setiap kelas yang diperoleh dalam kurun waktu yang telah ditentukan relatif sedikit dari jumlah dataset pada umumnya. Pada penelitian ini, data diaugmentasi menggunakan tiga teknik transformasi gambar: *flip*, *rotate* dan memberikan *Gaussian Noise* pada gambar. Jumlah dataset ditingkatkan dengan membuat salinan baru dari gambar asal menggunakan ketiga teknik tersebut.

Data Pre-processing

Sama seperti pada model ASL, data yang telah diaugmentasi kemudian diberi label kemudian resolusi gambar diubah menjadi persegi nxn. Selanjutnya data dibagi menjadi *data train* dan *data test* dengan perbandingan yang sama yaitu 0.88 *data train* dan 0.12 *data test*.

Model Specification

Model A dikembangkan menggunakan arsitektur yang sama pada Model ASL. Hal ini bertujuan untuk melakukan perbandingan yang adil antara Model A dan Model B yang keduanya dibangun menggunakan arsitektur yang sama namun dengan pendekatan yang berbeda (Model A tanpa *parameter-transfer* dari Model ASL, Model B menggunakan *parameter-transfer* dari Model ASL).

Model training

Sama seperti pada pengembangan model ASL, Model A di-*training* menggunakan *kernel* dan GPU dari Kaggle untuk mendapatkan *training* dan *validation* accuracy dari model.

3.1.3 Pengembangan Model B

Pada tahap ini, strategi dan teknik *transfer learning* ditentukan dengan pertimbangan *source domain* (ASL) dan *target domain* (Bisindo). Bagian dari domain yang mempengaruhi teknik *transfer learning* yang digunakan adalah tersedianya label di *target domain* dan *source domain* serta *task* dari kedua domain (Sinno Jialin Pan, 2009). Pada penelitian ini, baik *source domain* dan *target domain* memiliki *task* yang relatif sama yaitu klasifikasi dan *labeled data* tersedia pada keduanya. Penelitian ini menggunakan teknik *transductive transfer learning* dengan pemindahan pengetahuan dilakukan melalui *transferring knowledge of parameters*. *Learning parameter* yang sudah disimpan pada Model ASL digunakan kembali pada Model B melalui konsep *layer freezing*. Model B dibuat dari Model ASL dengan melakukan *freezing* pada beberapa layer bawah dari Model ASL dan mengganti layer terakhir dengan Dense Layer 26 (26 untuk alfabet). Dengan melakukan *layer freezing*, *training* dilakukan dengan menggunakan data Bisindo menggunakan learning parameter yang sudah tersimpan (*weight* dan *bias* dari Model ASL).

3.1.4 Pengujian dan Analisis

Hipotesis diuji menggunakan akurasi model baik pada saat *training* dan saat melakukan *testing*. Pada saat *training*, nilai akurasi model akan dihitung berdasarkan error yang dihasilkan oleh model ketika proses pembelajaran dilakukan. Nilai error diperoleh melalui persamaan *loss function*, semakin kecil nilai *loss* dari suatu model, maka semakin tinggi akurasi yang dihasilkan saat *training*. Oleh karena model dibuat untuk melakukan klasifikasi, maka *loss function* yang digunakan adalah

Categorical Crossentropy. Selain akurasi pada saat *training*, lamanya waktu yang diperlukan untuk *training* juga akan diuji untuk mengetahui dampak *parameter-transfer* pada durasi *training*.

Pada saat *testing*, model akan diberikan sekumpulan gambar untuk diprediksi kelas dari masing-masing gambar. Hasil prediksi kemudian dipisahkan berdasarkan kelompok *true positive*, *true negative*, *false positive* dan *false negative*. *true positive* adalah model benar/berhasil dalam memprediksi kelas positif. Dalam penelitian ini, contoh *true positive* adalah gambar yang memuat huruf A diklasifikasikan sebagai huruf A oleh model. Sebaliknya, *true negative* adalah model benar/berhasil dalam memprediksikan kelas negatif. Ambil contoh sampel gambar yang tidak memuat huruf A (misal huruf C), berhasil diklasifikasikan sebagai huruf lain selain A. *false positive* terjadi ketika model tidak berhasil mengklasifikasikan gambar yang memuat huruf A ke dalam kelas A. Begitu pula *false negative* adalah ketika model tidak berhasil melakukan klasifikasi sampel negatif (tidak memuat huruf A) ke dalam kelas selain A.

Nilai angka presisi dapat dihitung menggunakan persamaan berikut.

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \quad (3.1)$$

Nilai angka recall dapat dihitung menggunakan persamaan berikut.

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \quad (3.2)$$

Kedua nilai tersebut digunakan untuk menghitung *F1 score* dari model yang didefinisikan sebagai berikut:

$$F1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.3)$$

Angka *precision* merupakan besaran yang sesuai untuk mengukur keakuratan dan angka presisi dari sebuah model dalam memprediksi sampel positif dengan hasil prediksi positif. Besaran ini digunakan untuk mengevaluasi model ketika *cost* dari *false positive* tinggi. Contoh model yang relevan untuk dievaluasi dengan besaran ini yaitu pendekripsi *spam e-mail*. *false positive* dalam model ini yaitu ketika *e-mail* yang bukan *spam* (*actual negative*) diprediksi sebagai *spam*. Pengguna *e-mail* mungkin dapat melewatkannya informasi penting akibat *e-mail* penting dikategorikan sebagai *spam*.

Recall menghitung jumlah dari *true positive* yang dihasilkan melalui prediksi oleh model. Besaran ini umumnya digunakan untuk mengevaluasi model ketika *cost* dari *false negative* tinggi. Contoh model yang sesuai dievaluasi dengan besaran ini yaitu model pendekripsi penyakit tertentu. Hal yang terjadi ketika *false positive* dihasilkan dari model ini yaitu seseorang yang sakit (*actual positive*) diprediksi tidak sakit (*predicted negative*) tentu akan membahayakan karena penyakit tersebut tidak terdeteksi.

F1 merupakan besaran yang digunakan untuk mengevaluasi keseimbangan antara *precision* dan *recall*. Karena *cost* dari kesalahan prediksi model pada penelitian ini tidak terlalu tinggi, oleh karena itu *F1 score* dipilih sebagai besaran yang digunakan untuk mengevaluasi model saat *testing*.



BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengembangan Model ASL

4.1.1 Library dan Packages yang digunakan

Untuk melakukan pengembangan Model ASL terdapat beberapa *library* dan *packages* yang diperlukan untuk mempermudah proses pengembangan. Berikut adalah *library* dan *packages* yang digunakan untuk mengembangkan Model ASL.

Tabel 4.1. *Library* yang digunakan untuk membuat model ASL dan penggunaannya dalam model

Nama <i>Library/Packages</i>	Penggunaan dalam Model
keras.models : Sequential	Penulisan urutan layer saat pendefinisian model agar dibaca secara sekuelisial.
keras.layers : Conv2D, MaxPool2D, Flatten, Dense, Dropout, BatchNormalization	Menyusun layer yang digunakan dalam model.
keras.preprocessing.image : ImageDataGenerator	Digunakan untuk <i>preprocessing</i> data
cv2	Pengolahan gambar, <i>load</i> gambar dan tampilan gambar.
Numpy	Menyimpan data ke <i>dataframe</i> khususnya <i>array</i> .
Glob	Mengembalikan daftar file dalam bentuk <i>full path</i> .
matplotlib : pyplot	Menampilkan grafik hasil <i>training</i> .
random	Generate angka acak yang digunakan untuk mengakses dataset secara acak.
sklearn.model_selection:train_test_Split	Memisahkan data ke dalam <i>train data</i> dan <i>validation data</i> .
keras.utils.np_utils: to_categorical	Mengubah data label ke bentuk <i>hot vector</i> .
keras.callbacks: TensorBoard, ReduceLROnPlateau, EarlyStopping	<i>Callback</i> yang akan mempengaruhi jalannya <i>training</i> model.
skimage.transform: resize	Mengubah dimensi dari gambar.
sklearn.model_selection: train_test_split	Membagi data menjadi <i>train data</i> , <i>validation data</i> dan <i>test data</i> .
sklearn.metrics : f1_score	Menghitung F1 score dari hasil prediksi model.

4.1.2 Dataset yang Digunakan

Data ASL yang digunakan dalam penelitian ini yaitu ASL Alphabet (Akash, 2017). Data tersebut merupakan kumpulan gambar isyarat alfabet ASL yang diambil dari tampak depan. Data terdiri dari 87000 gambar yang disimpan dalam 29 folder masing-masing kelas yang terdiri dari 26 folder alfabet, folder spasi, folder *delete* dan folder *nothing*. Dari jumlah tersebut, ada lebih dari 1000 gambar pada setiap kelas sehingga dataset dapat dikatakan layak untuk digunakan sebagai *source domain* untuk *transfer learning*.

4.1.3 Strategi Pembagian Data

Terdapat 87000 gambar untuk *training* dengan resolusi 200x200 piksel dari 29 kelas yang terdiri dari 26 alfabet dan 3 tambahan kelas yaitu spasi, *delete* dan *nothing* (tidak memuat isyarat apapun pada gambar). Selain itu, terdapat 29 gambar dengan satu gambar masing-masing kelas untuk melakukan *testing*. Dari 87000 gambar tersebut, 57420 gambar digunakan untuk *training data*, 19140 untuk *validation data* dan 10440 gambar digunakan sebagai *test data*. Dengan demikian, rasio dari data yang digunakan untuk *training*, *validation* dan *test data* berturut-turut 0.66 : 0.22 : 0.12.

Berikut adalah sampel gambar dari dataset yang digunakan:



Gambar 4.1. Sampel gambar data ASL

1. Dilarang mengungkapkan karya tulis
a. berdasarkan sumber berbeda;
b. peruntukan keperluan pendidikan, penulisan karya ilmiah atau penelitian;
c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

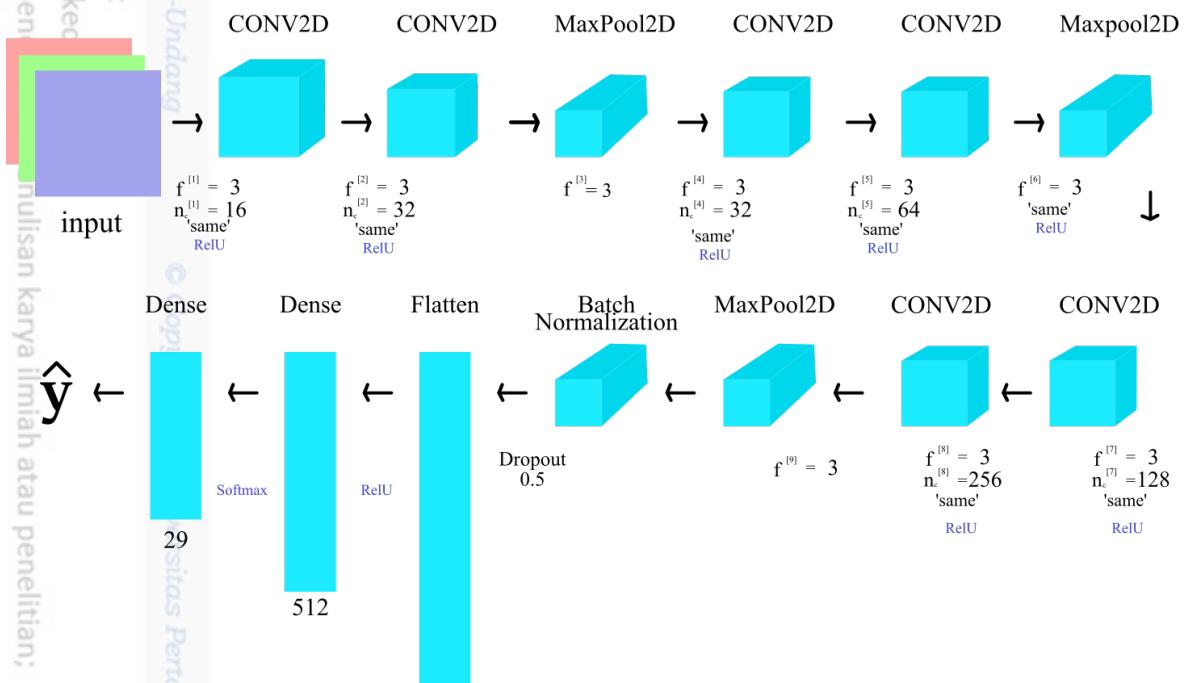
4.1.4 Data Pre-processing

Sebelum data dimasukkan ke dalam model, terdapat beberapa operasi yang dilakukan:

1. Pemberian label pada gambar Dua array dibuat untuk menyimpan informasi mengenai gambar dan label dari gambar tersebut. Array X digunakan untuk menyimpan konten dari sebuah gambar. Array y dibuat untuk memuat label dari gambar tersebut. Untuk setiap gambar dalam folder yang sama maka akan diberi label yang sama. Contoh array X[0] merupakan array berisi gambar isyarat huruf C maka y[0] berisi angka 2 sebagai label dari gambar tersebut yang ditulis dalam format *hot vector* (contoh untuk angka 3 : [0, 0, 1, 0, ..., 0, 0]) dengan menggunakan fungsi *to_categorical*.
2. Mengubah resolusi gambar menjadi persegi ($n \times n$). Resolusi gambar yang disimpan adalah gambar yang resolusinya telah disesuaikan menggunakan fungsi transformasi *reshape* dari Skimage dengan n sebesar 64.
3. Membagi data menjadi tiga bagian: *train*, *validation* dan *test data*. Setelah data berhasil diberi label, data kemudian dibagi dengan proporsi 0.66 untuk *training*, 0.22 untuk *validation* dan 0.12 untuk *test*. Pembagian dilakukan menggunakan fungsi *train_test_split*.

4.1.5 Arsitektur Model

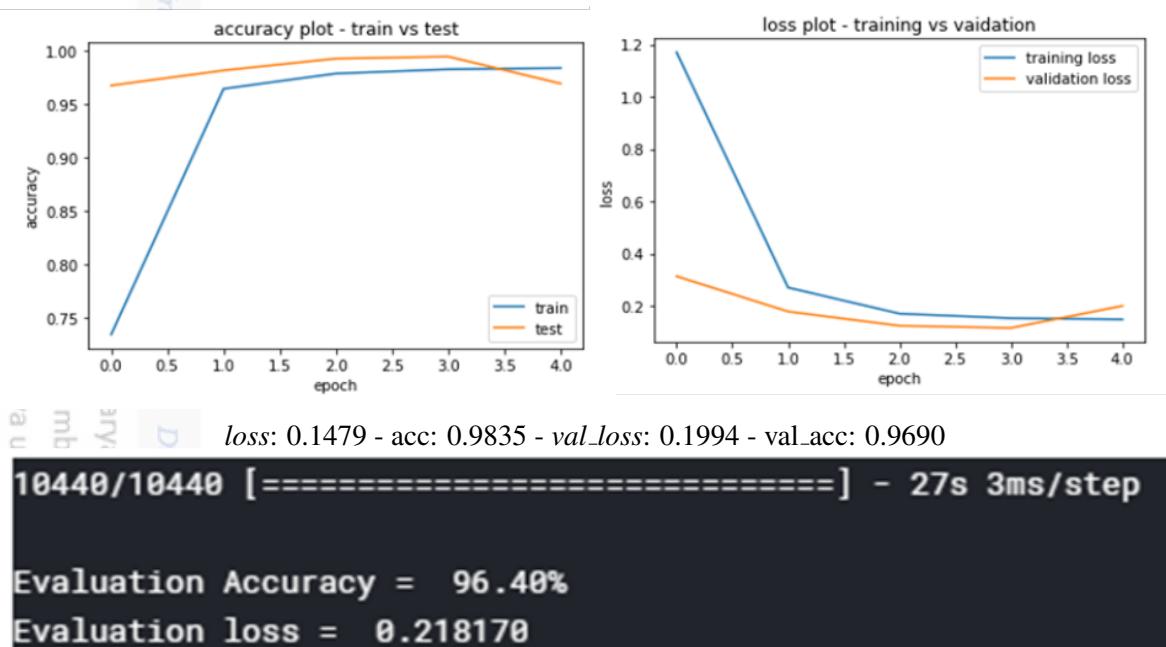
Konfigurasi layer pada model CNN yang digunakan adalah sebagai berikut.



Gambar 4.2. layer pada model CNN diambil dari Model ASL referensi (Modi, 2019)

4.1.6 Hasil *Training*

Model di-*training* dengan menggunakan teknik *early stopping* untuk menghindari terjadinya *overfit*. *Early stopping* adalah teknik yang dapat mencegah *overfitting* dengan menghentikan *training* ketika *validation accuracy* mengalami penurunan. Angka kesabaran/*patience* dari *early stopping* diatur pada angka 2, dengan demikian, jika dalam dua *epoch* terjadi penurunan *validation accuracy*, maka proses *training* akan dihentikan.



Gambar 4.3. Grafik akurasi dan loss dari Model ASL dengan 5 epoch saat *training* dan *testing*

Dari gambar 4.3 dapat dilihat bahwa dalam proses *learning*, model mengalami peningkatan baik pada *train accuracy* dan pada *validation accuracy* mengindikasikan bahwa model melakukan pembelajaran dengan baik. Hasil akhir dari Model ASL yaitu *training accuracy* = 0.9835, *validation accuracy* = 0.9690 dan *test accuracy* = 96.40

4.2 Pengembangan Model A

4.2.1 Library yang digunakan

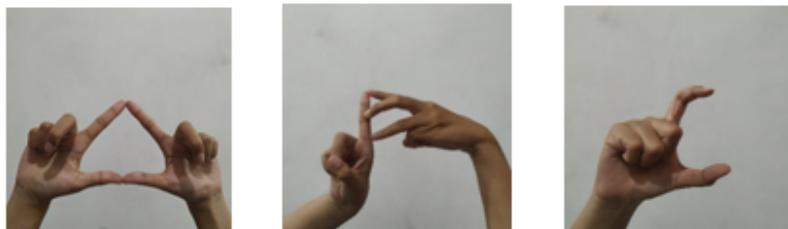
Library yang digunakan sama seperti model ASL, hanya ada beberapa tambahan library untuk melakukan *data augmentation* dan *cross validation*.

Tabel 4.2. Library yang digunakan untuk *data augmentation* dataset Bisindo

Nama Library/Packages	Penggunaan dalam Model
scipy : ndarray	Mengembalikan gambar menjadi nilai array.
Skimage : transform, util	Membaca gambar dan membuat salinan dari gambar melalui transformasi seperti <i>rotate</i> , <i>flip</i> dan <i>random noise</i> .
sklearn.model_selection : KFold	Membagi data menjadi k bagian sama rata (<i>fold</i>).

4.2.2 Dataset yang digunakan

Dataset yang digunakan adalah gambar alfabet Bisindo dengan jumlah 2659 gambar (hasil augmentasi) yang disimpan dalam 26 folder, merepresentasikan kelas yang akan diklasifikasi yaitu alfabet (a-z). Pada penelitian ini dataset yang digunakan dibuat berdasarkan hasil observasi penggunaan Bisindo dari responden yang mempelajari Bisindo.

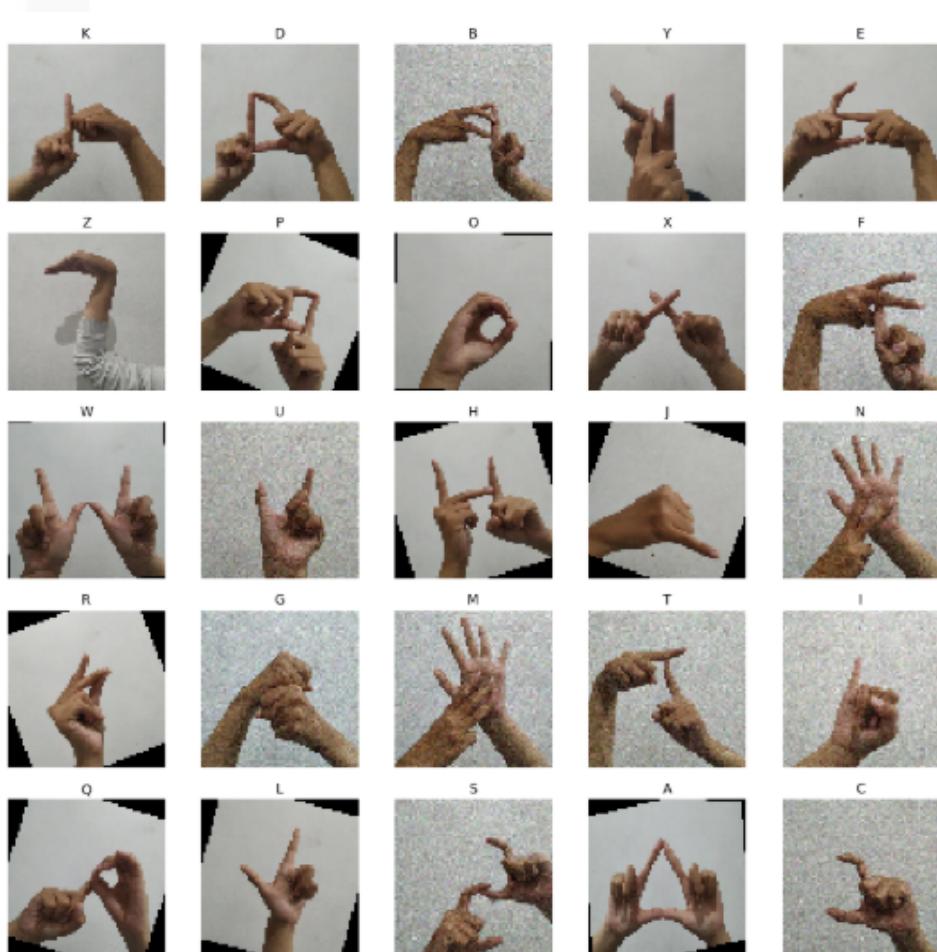


Gambar 4.4. Sampel Dataset Bisindo

4.2.3 Data Augmentation

Proses augmentasi data dilakukan dengan memperbanyak gambar yang semula berjumlah 26 gambar (masing masing alfabet satu gambar) menjadi 100 gambar pada tiap huruf menggunakan transformasi gambar: *flip*, *rotate* dan memberikan *Gaussian Noise* pada gambar.

Berikut adalah sampel hasil gambar yang sudah diaugmentasi.

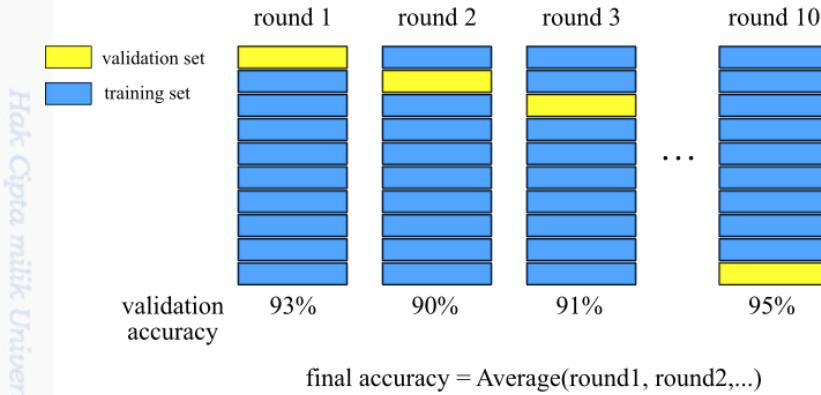


Gambar 4.5. Dataset Bisindo hasil augmentasi: *crop, rotate dan Gaussian Noise*

4.2.4 Hasil Training

Model ditraining menggunakan *kernel* Kaggle dengan *batch size* 64 dan 5 *epoch* menggunakan teknik *k-fold cross-validation*. Teknik ini memungkinkan *training* dilakukan pada data dengan jumlah yang terbatas. Setiap sampel data dibagi menjadi k grup dengan jumlah data dalam satu *fold* sama besar. Grup k akan di-train sebagai *validation data* dan grup lainnya (1,.. k-1) sebagai *training*. Berikut adalah ilustrasi *training* dalam teknik *k-fold cross-validation*.

1. **menyebutkan sumber sesuai kaidah kecendekian;**
 - b. pengutipan hanya untuk keperluan pendidikan penulisan kegiatan ilmiah atau penelitian;
 - c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.



Gambar 4.6. Ilustrasi *training* menggunakan *k-fold cross validation*

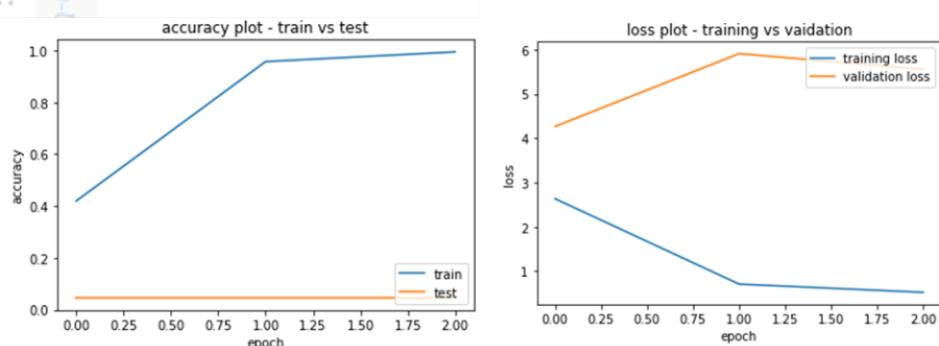
1. Dilarang mengungkapkan hasil penelitian dan/atau karya ilmiah kecuali:

 - a. menyertakan sumber asal dalam penulisan;
 - b. pengutipan tidak merugikan Universitas Pertamina.

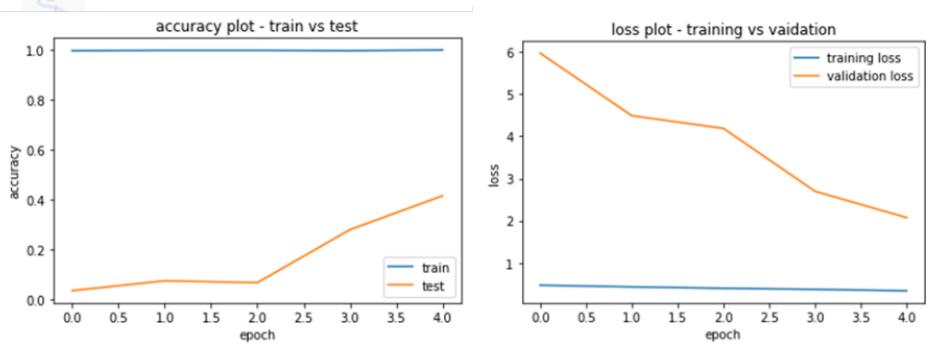
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

Gambar 4.6 menunjukkan pembagian *training data* dan *validation data* pada setiap *round training*. Contoh menggunakan $k=10$ sehingga data dibagi menjadi sepuluh kelompok dengan jumlah sama besar. *Round* pertama menggunakan grup pertama sebagai *validation data* dan seterusnya hingga grup ke-10. Akurasi akhir diperoleh melalui nilai rata - rata akurasi *training* dan *validation* dari seluruh *round*.

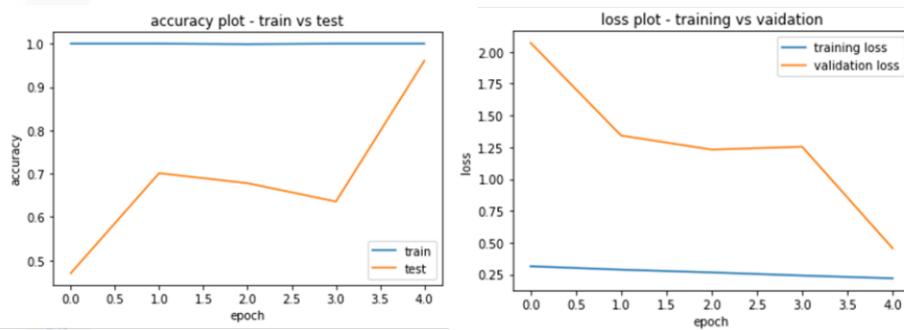
Berikut adalah hasil *training* Model A menggunakan *k-fold cross-validation* dengan $k=3$. Tidak ada aturan tertentu mengenai nilai k , umumnya pada angka 5-10. Karena jumlah dataset Bisindo yang digunakan sangat sedikit (2659 gambar), k yang tinggi akan menyebabkan jumlah gambar yang sedikit dalam satu grup dan tidak dapat merepresentasikan data dengan baik, oleh karena itu angka 3 dipilih sebagai k untuk melakukan *k-fold cross-validation*.



Gambar 4.7. Grafik akurasi dan loss dari Model A saat *round* pertama



Gambar 4.8. Grafik akurasi dan loss dari Model A saat *round* kedua



Gambar 4.9. Grafik akurasi dan loss dari Model A saat *round* ketiga

1.

Melalui gambar 4.7 dapat dilihat bahwa model tidak menghasilkan akurasi yang baik pada validasi dan proses *training* dihentikan pada *epoch* ke-2 akibat *callback* dari *early stopping*. Pada *round* selanjutnya dapat dilihat melalui gambar 4.8 dan 4.9 validation loss berkurang seiring kenaikan *epoch*. Hal ini merupakan indikasi yang baik bagi model. Pada *round* 2 dan 3 tidak terjadi *early stopping* sehingga dapat dikatakan bahwa *validation accuracy* mengalami peningkatan pada kedua *round* ini.

Setelah melalui tiga *round training*, rata-rata train dan *validation accuracy* dari ketiga *round* dihitung untuk mengetahui kinerja rata – rata dari model.

Tabel 4.3. Rata-rata loss dan accuracy Model A untuk seluruh *round*

<i>train_accuracy</i>	0.46846
<i>train_loss</i>	2.71451
<i>val_accuracy</i>	0.47391
<i>val_loss</i>	2.69424

Berdasarkan tabel 4.3 diperoleh rata – rata hasil akurasi yang tidak terlalu tinggi dikarenakan performa model di *round* pertama masih menghasilkan akurasi yang sangat rendah. Hasil akhir dari Model A yaitu *train accuracy* = 0.46846 dan *validation accuracy* = 0.47391. Lamanya waktu yang diperlukan untuk melakukan *training* Model A yaitu 2 menit 39 detik, tepatnya 7 ms/ step.

4.3 Pengembangan Model B

4.3.1 Teknik Transfer Learning

Pada penelitian ini, *transfer learning* untuk membuat Model B terbatas hanya pada metode *transfer learning* parameter dari Model ASL dalam setting *inductive transfer learning* (lihat sub bab 2.8.3 dan 2.8.4). Memindahkan *learning parameter* dari Model ASL ke Model B dilakukan melalui langkah berikut.

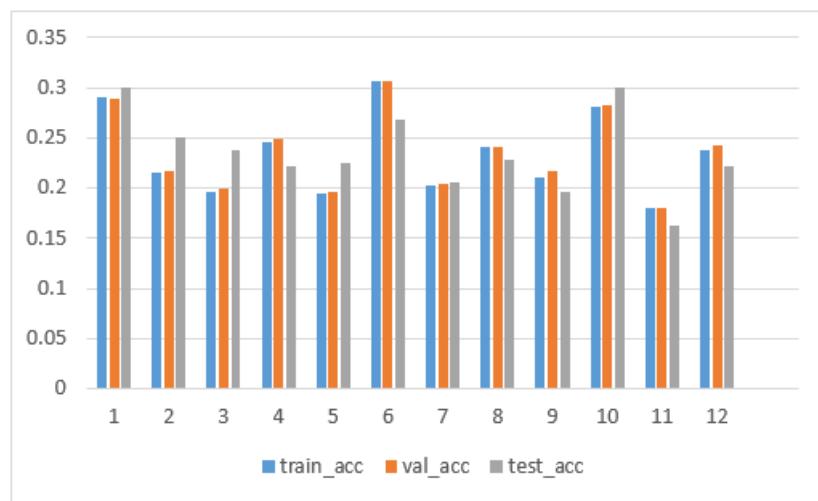
1. Model ASL yang telah selesai di-*training* disimpan dalam format .h5.
2. Dataset Bisindo disiapkan dengan melakukan *preprocessing* yang sama seperti pada tahap pengembangan Model A.
3. Model ASL di-*load* dan layer terakhir dari model tersebut dihapus dari model dengan menggunakan fungsi `layer.pop()`.
4. *Dense layer* dengan jumlah kelas 26 (kelas dari dataset Bisindo) ditambahkan pada layer terakhir dari model.
5. *Learning parameter* dari Model ASL digunakan kembali dengan menggunakan teknik *layer freezing* yaitu membekukan (membuat konstan, tidak berubah saat *training*) *learning parameter* seperti *weight* dan *bias* yang sebelumnya telah diperoleh melalui *training* Model ASL. Layer pertama hingga terakhir dibekukan dengan menggunakan fungsi `layers.trainable = False`. Model kemudian di-*training* dengan menggunakan *learning parameter* dari ASL menggunakan konfigurasi yang sama dengan *training* Model A dan Model ASL

4.3.2 Hasil Training

Untuk mencari model dengan akurasi terbaik, mula-mula *layer freezing* dilakukan dari layer pertama hingga satu layer di bawah layer teratas (output layer). Setelah *layer freezing*, model di-*compile* dan *training* menggunakan dataset Bisindo teknik *k-fold cross-validation* dengan konfigurasi yang sama dengan Model A yaitu dengan $k=3$ dan 5 *epoch*. Percobaan dilakukan berulang kali dengan mengubah banyaknya layer yang di-freeze dihitung dari layer input model. Untuk setiap konfigurasi layer yang di-freeze, rata – rata *loss* dan *accuracy* hasil *training* akan diidentifikasi kemudian model di-*test* menggunakan *data test* dan dicatat akurasinya.

Gambar 4.10. Grafik rata-rata akurasi dari konfigurasi *layer freezing*.

(sumbu-y adalah akurasi, sumbu x adalah banyaknya *frozen layer* terhitung dari layer *input* model.)



Model terdiri dari 13 layer dan 12 percobaan *layer freezing* dilakukan dengan melakukan *freezing* layer-1 sampai ke layer-12, layer-1 sampai layer-11, dan seterusnya hingga hanya layer pertama yang di-freeze. Dari percobaan tersebut, *transfer learning parameter* hanya berhasil mencapai akurasi tertinggi yaitu 30.63% pada *train accuracy*, 30.66% pada *validation accuracy* dan 30% pada *testing accuracy*. Model yang dipilih adalah model dengan hasil *test accuracy* tertinggi, yaitu model dengan jumlah layer yang di-freeze adalah 1 layer dan 10 layer (*consecutive freeze* dari layer pertama). Lama waktu yang diperlukan untuk melakukan *training* model bergantung pada jumlah layer yang di-freeze. Berikut adalah tabel durasi *training* untuk setiap konfigurasi *layer freezing*.

Tabel 4.4. Durasi *training* terhadap banyaknya *frozen layer*

Banyaknya layer yang di-freeze	Waktu <i>training</i>
1	7 ms/step
2	7 ms/step
3	7 ms/step
4	7 ms/step
5	5 ms/step
6	5 ms/step
7	5 ms/step
8	3 ms/step
9	3 ms/step
10	3 ms/step
11	3 ms/step
12	3 ms/step

Dari Tabel 4.4 dapat dilihat bahwa semakin banyak *frozen layer* maka waktu yang dibutuhkan untuk *training* juga semakin cepat. Hal ini disebabkan oleh *non-trainable parameter* akan meningkat seiring dengan meningkatnya jumlah *frozen layer*. Semakin banyak *non-trainable parameter* maka semakin cepat proses *training* karena tidak perlu melakukan perhitungan ulang (*update*) untuk mencari nilai *weight* dan *bias* yang optimal untuk *training* (lihat sub bab 2.3).

1. Dilarang mengutip karya ilmiah tanpa izin.
 - a. menyebutkan sumber dalam tesis, skripsi, laporan, bentuk pendidikan, penulisan karya ilmiah atau penelitian;
 - b. pengutipan hanya boleh dilakukan dalam bentuk ringkas;
 - c. pengutipan tidak merugikan Universitas Pertamina.
2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

4.4 Pengujian dan Analisis

4.4.1 Testing Model

Hasil evaluasi model

Setelah proses *training*, model dievaluasi menggunakan fungsi *model.evaluate()*. Pada fungsi ini, model yang telah di-*training* dievaluasi dengan membandingkan hasil prediksi data test (*X_test*) dari model dengan label sebenarnya (*Y_test*).

1.

a. Model A

```
320/320 [=====] - 1s 2ms/step
Evaluation Accuracy = 94.38%
Evaluation loss = 0.491370
```

Gambar 4.11. Hasil evaluasi Model A

2.

b. Model B

```
320/320 [=====] - 1s 2ms/step
Evaluation Accuracy = 30.00%
Evaluation loss = 13.130595
```

Gambar 4.12. Hasil evaluasi Model B

Gambar 4.11 dan 4.12 merupakan hasil prediksi tehadap gambar *test data* kemudian dievaluasi dengan membandingkan kelas hasil prediksi dengan kelas sebenarnya dari *test data*.

F1 Score Model

1. Model A

```
array([1.        , 1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 0.90322581, 0.93333333,
       1.        , 1.        , 0.95      , 0.9       , 0.53846154,
       1.        , 1.        , 0.64864865, 1.        , 1.        ,
       0.91666667, 1.        , 1.        , 1.        , 0.96      ,
       1.        ])
```

Gambar 4.13. *F1 Score* Model A untuk setiap kelas dalam alfabet (kiri atas: *F1 Score* huruf A, tersusun alfabetis dari kiri ke kanan)



2. Model B

```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.53333333,
       0.28779221, 0.22680412, 0.72727273, 0.        , 0.        ,
       0.        , 0.        , 0.15384615, 0.        , 0.        ,
       0.        , 0.31034483, 0.325      , 0.        , 0.        ,
       1.        ])
```

Gambar 4.14. *F1 Score* Model B untuk setiap kelas dalam alfabet (kiri atas: *F1 Score* huruf A, tersusun alfabetis dari kiri ke kanan)

1. *Dilakukan pengujian yang dilakukan pada test data*
 2. *Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.*
- a. *Setelah melakukan prediksi pada test data, *F1 Score* dihitung untuk setiap kelas berdasarkan hasil prediksi dari model. *F1 Score* yang bernilai nol merupakan skor yang diperoleh ketika model tidak menghasilkan kelas tersebut sebagai hasil prediksi.*

4.4.2 Analisis *Transfer Learning* pada Model A dan B

Jika ditinjau dari definisi *transfer learning* yang telah dibahas di sub bab 2.8.2, kondisi transfer yang terjadi ketika memindahkan *knowledge* dari Model ASL ke Model B yaitu adanya perbedaan *domain* $D_s \neq D_t$ dengan *task* yang serupa $\mathbb{T}_s = \mathbb{T}_t$.

Domain D terdiri dari dua komponen: feature space X dan distribusi marjinal (*marginal probability distribution*) $P(X)$, dengan $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. Sangat mungkin terjadi adanya *generalization error* akibat perbedaan *domain* dan karakteristik di dalam *domain* tersebut. *Generalization error* merupakan besaran yang mengukur seberapa baik model yang di-train menggunakan sampel dengan sebuah distribusi tertentu dapat melakukan generalisasi dengan baik pada sampel baru yang memiliki distribusi serupa (Kouw).

Selain *generalization error*, pada penelitian ini tidak dilakukan penelitian lebih lanjut antara *source domain* (ASL) dan *target domain* (Bisindo). Hal yang dapat dilakukan dalam menguji *domain* salah satunya adalah dengan mengukur seberapa berbeda satu *domain* dengan yang lainnya. Besarnya perbedaan dari kedua *domain* dapat diidentifikasi dengan menggunakan *symmetric difference hypothesis divergence* ($\mathcal{H}\Delta\mathcal{H}$ -divergence) (Kouw).

Sebuah *task* terdiri dari dua komponen: *label space* Y dan *objective predictive function* $f(\cdot)$. *Task* dinotasikan dengan $\mathbb{T} = \{Y, f(\cdot)\}$. Pada penelitian ini *source task* tersusun dari *label space* dan *predictive function* alfabet ASL. *Label space* dari Model ASL dan Model Bisindo tidaklah sama, pada Model ASL terdapat tambahan label seperti space, delete dan nothing. Hal ini juga dapat mempengaruhi *predictive function* pada Model Bisindo sebagai konsekuensi dari *transfer learning parameter*.

Perbedaan gestur yang signifikan antara ASL dan Bisindo juga dapat mempengaruhi *feature map* yang terbentuk dalam proses *learning*. Untuk membentuk isyarat alfabet ASL digunakan dengan menggunakan satu tangan sedangkan pada Bisindo mayoritas alfabet menggunakan kedua tangan

4.4.3 Perbandingan Performa Model A dan Model B

Durasi training

Lamanya durasi *training* dari model ditentukan oleh banyaknya *trainable parameter* pada masing-masing model.

Tabel 4.5. Jumlah Parameter dalam setiap Konfigurasi Model

Model	Banyaknya frozen layer	Trainable parameter	Non-trainable parameter	Waktu training
A	0	940.506	512	7 ms/step
B	1	940.506	512	7 ms/step
	2	940.058	960	7 ms/step
	3	935.418	5.600	7 ms/step
	4	935.418	5.600	7 ms/step
	5	926.170	14.848	5 ms/step
	6	907.674	33.344	5 ms/step
	7	907.674	33.344	5 ms/step
	8	833.818	107.200	3 ms/step
	9	538.650	402.368	3 ms/step
	10	538.650	402.368	3 ms/step
	11	538.138	402.800	3 ms/step
	12	538.138	402.800	3 ms/step

Berdasarkan tabel 4.5, dapat dilihat hubungan antara banyaknya *non-trainable parameter* dengan waktu *training*. Semakin banyak *non-trainable parameter* maka semakin cepat waktu yang dibutuhkan untuk *training*, hal ini karena *non-trainable parameter* pada saat *training* tidak akan *update* dengan nilai dari *training* data sehingga proses *training* menjadi lebih cepat. Oleh karena itu, hipotesis dari penelitian ini terjawab bahwa penggunaan *transfer learning* khususnya *transfer learning parameter* dapat mengurangi waktu *training* secara signifikan. Selanjutnya akan dibahas mengenai performa dari kedua model yang telah dibuat.

Akurasi *training* dan *testing*

Performa dari model dinilai berdasarkan akurasi saat *training* dan *testing*. Berdasarkan hasil yang diperoleh pada pengembangan Model B, *transfer learning parameter* justru menurunkan performa dari model ketika *training*. Hal ini dapat dilihat dengan melakukan perbandingan dengan Model A yang disusun dari arsitektur yang sama namun tidak menggunakan *learning parameter* dari Model ASL. Dalam melakukan pengembangan Model B, percobaan dilakukan dengan melakukan *layer freezing* dari jumlah layer paling maksimum untuk yaitu 13 hingga jumlah paling minimum yaitu 1. Berdasarkan percobaan tersebut, terlihat pada gambar 4.10 dapat dilihat bahwa semakin banyak

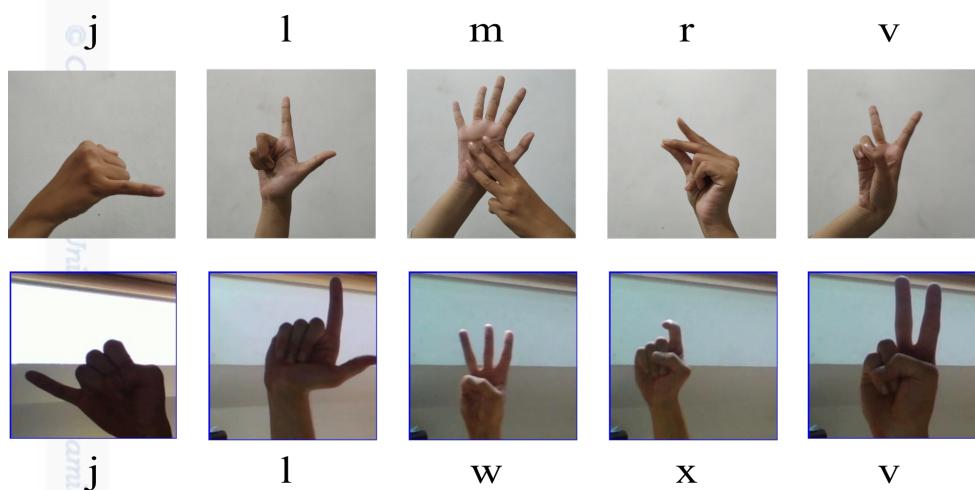
jumlah layer yang di-freeze, akurasi saat *training* model semakin menurun. Hal ini mengindikasikan bahwa *learning parameter* dari Model ASL tidak dapat menggeneralisasi *feature maps* dari dataset Bisindo dengan baik, sehingga ketika semakin sedikit *frozen layer*, akurasi justru semakin meningkat. Dalam kata lain, pemindahan pengetahuan dari parameter tidak perlu dilakukan.

Selain akurasi saat *training*, akurasi saat *testing* juga digunakan untuk membandingkan performa Model B terhadap Model A. Model A menghasilkan akurasi 94.3% pada data test sedangkan Model B hanya menghasilkan akurasi 30%. Tentu hal ini tidaklah aneh karena pada saat *training*, Model B tidak menghasilkan performa yang baik sehingga akan mempengaruhi performa pada saat *testing*.

Jika dilihat dari *F1 Score* yang dihasilkan, Model A masih jauh lebih unggul atas keberhasilannya pada *test data* yang hampir dapat memprediksi huruf dengan akurat. Adapun huruf yang masih belum dapat diprediksi dengan akurat yaitu huruf O dan R. *F1 Score* dari kedua huruf tersebut berturut-turut sebesar 0.5384 dan 0.6486. Kedua huruf tersebut memiliki *F1 Score* yang kurang akurat dikarenakan minimnya kualitas data hasil sampel dalam pengumpulan data.

Sampel huruf O dikembangkan dengan melakukan augmentasi dari dua gambar asal yang keduanya hampir mirip satu sama lain. Gestur huruf O yang membentuk lingkaran juga mempengaruhi fitur yang terbentuk dari data hasil augmentasi, khususnya rotasi dari gambar asal. Karena huruf O membentuk lingkaran sehingga besar kemungkinan rotasi tidak dapat meningkatkan keberagaman fitur dalam sampel. Sampel huruf R dibuat dari hasil augmentasi gambar-gambar yang menangkap gerakan jentikan jari sebagai isyarat huruf R. Tentunya ketika gerakan jentikkan jari disimpan dalam gambar mengakibatkan distribusi dari fitur yang semakin beragam sehingga dengan jumlah hasil augmentasi yang sama dengan kelas yang lain, model tidak dapat mempelajari dengan baik fitur dari huruf R.

Di sisi lain, Model B hanya berhasil memprediksi huruf J, K, L, M, R, V dan W. Jika kita melihat karakter dari gestur masing - masing alfabet pada ASL dan Bisindo, terdapat beberapa huruf yang memiliki kemiripan yaitu C, I, J, L, O, V, Y, dan *delete* (pada ASL menyerupai huruf Z pada Bisindo).



Gambar 4.15. Contoh huruf yang diprediksi oleh Model B serta kemiripan gestur antara Bisindo (atas) dan ASL (bawah)

Berdasarkan kelas yang berhasil diprediksi dari Model B, dapat disimpulkan bahwa pemindahan *learning parameter* dari Model ASL akan membuat Model B mempelajari huruf Bisindo yang memiliki kemiripan dengan huruf ASL. Sehingga untuk dapat mengenali keseluruhan alfabet Bisindo, pemindahan *knowledge* khususnya *learning parameter* dari Model ASL tidak dapat meningkatkan performa Model B dalam memprediksi seluruh huruf pada alfabet Bisindo. Oleh karena itu, Model A memiliki performa yang lebih baik dibandingkan Model B dalam mengenali keseluruhan alfabet Bisindo.

Setelah dilakukan pengembangan dan pengujian Model A dan Model B, diperoleh akurasi Model A yang lebih baik dari Model B baik pada saat *training* maupun *testing*, oleh karena itu dapat disimpulkan bahwa model yang dibuat dengan dataset yang terbatas belum tentu memiliki akurasi yang lebih baik dari model hasil *transfer parameter*. Jumlah dataset yang sangat terbatas dapat ditingkatkan kualitasnya melalui proses *data augmentation* dan *transfer learning parameter* dapat menurunkan performa jika karakteristik kedua *domain* sangat berbeda sehingga tidak menutup kemungkinan model dengan dataset yang terbatas seperti Model A dapat melakukan pembelajaran lebih baik dibandingkan Model B yang menyimpan parameter hasil *training* dari Model ASL.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dengan dilaksanakannya penelitian ini telah dihasilkan model penerjemah alfabet Bisindo menggunakan implementasi algoritma CNN dengan akurasi 94.38% sebagai upaya untuk meningkatkan aksesibilitas Tuli pengguna Bisindo.

Model penerjemah tersebut merupakan model terpilih dari dua model yang telah dikembangkan dengan menggunakan pendekatan yang berbeda. Model A dikembangkan menggunakan pemindahan arsitektur dari *source model* (Model ASL) dan Model B dikembangkan menggunakan pemindahan arsitektur beserta *learning parameter* yang tersimpan dalam *source model* tersebut. Dengan percobaan yang dilakukan dalam penelitian ini, pemindahan *learning parameter* dari Model ASL kepada Model B mengurangi durasi *learning* yang signifikan namun menyebabkan Model B hanya dapat mempelajari kemiripan fitur antara *source* dan *target domain* (huruf yang memiliki kemiripan antara ASL dan Bisindo). Oleh karena itu, pemindahan arsitektur tanpa menggunakan *learning parameter* merupakan metode pengembangan yang sesuai untuk kasus seperti ini, sehingga *parameter-transfer* masih belum dapat mengatasi keberagaman gestur ASL dan Bisindo.

5.2 Saran

Luaran dari penelitian ini adalah model penerjemah yang disimpan dalam bentuk .h5. Bentuk ini tidak dapat digunakan secara praktis dalam berkomunikasi sehingga masih perlu dikembangkan dan dikemas dalam bentuk aplikasi agar dapat digunakan secara langsung oleh masyarakat.

Penelitian ini juga terbatas pada alfabet Bisindo, pengembangan yang mungkin dilakukan dari penelitian ini yaitu memperluas cakupan penerjemahan hingga pada penerjemahan kata dan kalimat. Penerjemahan kata dapat dikembangkan dengan menambah dataset kata. Setelah penerjemahan kata, penerjemahan kalimat dapat dibuat dengan menambahkan algoritma LSTM dari model hasil penelitian ini.

Ada banyak pendekatan *transfer learning* yang mungkin digunakan dalam pembuatan model penerjemah ini, saat ini Model B baru dikembangkan menggunakan pendekatan *parameter-transfer* dari Model ASL dan masih belum menghasilkan akurasi yang memuaskan sehingga kedepannya dapat dikembangkan dengan pendekatan lain (lihat sub bab 2.8.3) ataupun menggunakan *pre-trained* model lain.

Sebelum melakukan *transfer learning*, akan lebih baik jika dilakukan penelitian mengenai *domain* dan *task* yang digunakan untuk mengetahui teknik dan pendekatan yang sesuai dalam melakukan *transfer learning*. Tanpa mengkaji *domain* dan *task* terlebih dahulu, *transfer learning* dikhawatirkan menjadi *brute-force transfer* yang justru memperburuk performa dari model yang dihasilkan.



<https://www.kaggle.com/riestiyazain/model-a-dan-b-bisindo>

LAMPIRAN A

Tautan menuju Kernel

1. Dilarang mengutip karya tulis ini, kecuali:

- a. menyebutkan sumber sesuai kaidah kecendekiaan;
- b. pengutipan hanya untuk keperluan pendidikan, penulisan karya ilmiah atau penelitian;
- c. pengutipan tidak merugikan Universitas Pertamina.

2. Dilarang mempublikasikan atau memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa adanya izin dari Universitas Pertamina.

DAFTAR PUSTAKA

- A. Krizhevsky, I. Sutskever, G. E. H. (2012). Imagenet classification with deep convolutional neural network. *Advances in Neural Information Processing Systems*, pages 1097–1105.
- Almuhamarram (2013). Berkas:isyaratbisindo.jpg.
- Bheda, V. and Radpour, D. (2017). Using deep convolutional networks for gesture recognition in american sign language. *CoRR*, abs/1710.06836.
- Ehsan Othman, A. A.-H. (2018). Automatic arabic document classification based on the hrwtd algorithm. *Journal of Software Engineering and Applications*.
- Fan, C. (2014). Survey of convolutional neural network.
- Gilang Gumelar, Hanny Hafiar, P. S. (2018). Bahasa isyarat indonesia sebagai budaya tuli melalui pemaknaan anggota gerakan untuk kesejahteraan tuna rungu. *INFORMASI:Kajian Ilmu Komunikasi*, pages 66–67.
- H. Lee, A. Battle, R. R. A. Y. N. (2007). Efficient sparse coding algorithms. In *Proceedings of The 19th Annual Conference on Neural Information Processing Systems*. MIT Press.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lisa Torrey, J. S. (2009). *Handbook of Research on Machine Learning Applications*. IGI Global.
- M. T. Rosenstein, Z. Marx, L. P. K. (2005). To transfer and not to transfer. *NIPS-05 Workshop on Inductive Transfer*.
- Modi, R. (2019). Asl classifier using keras.
- R. Raina, A. Battle, H. L. B. P. A. Y. N. (2007). Self taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*.
- Sinno Jialin Pan, Q. Y. (2009). A survey on transfer learning. In *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*.
- T. Handhika, R. I. M. Zen, M. D. P. L. I. S. (2018). Gesture recognition for indonesian sign language (bisindo). In *2nd International Conference on Statistics, Mathematics, Teaching and Research*. IOP Publishing.
- Ullah, A. (2017). Trainig a model good enough for transfer learning.
- Utama, A. (2015). Penyandang tunarungu desak pemerintah aplikasikan bisindo.
- V. Nair, G. E. H. (2010). Rectified linear unit improve restricted boltzmann machines. In *27th International Conference on Machine Learning (ICML-10)*.
- X. Wu, V. Kumar, J. R. Q. J. G. Q. Y. H. M. (2008). Top 10 algorithms in data mining. knowledge information system. *Knowledge Information System*, pages 1–37.
- Zhu, X. (2008). Semi-supervised learning literature survey. *Comput Sci, University of Wisconsin-Madison*, 2.