Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023118098
Vol. 11, No. 6, Desember 2024, hlm.1159-1168 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

Sistem Rekognisi Citra Digital Bahasa Isyarat Menggunakan *Convolutional Neural Network* Dan *Spatial Transformer*

Mohammad Alfiano Rizky Mahardika[*1],Novanto Yudistira[2] , Achmad Ridok[3]

[1,2,3] Universitas Brawijaya, Malang
Email: [1]alfianodamarjati15@gmail.com, [2]yudistira@ub.ac.id, [3]acridokb@ub.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 22 November 2023, diterima untuk diterbitkan: 19 November 2024)

Abstrak

Bahasa isyarat merupakan hal yang sangat penting bagi suatu kelompok masyarakat, yaitu masyarakat bisu atau tuli. Untuk dapat berkomunikasi dengan masyarakat bisu atau tuli, orang yang tidak bisu atau tuli memerlukan bahasa isyarat tersebut untuk dapat mengerti maksud atau pikiran mereka yang bisu atau tuli. Sebagian besar percakapan pada bahasa isyarat dilakukan dengan menggunakan tangan, dimana tangan beserta jari-jarinya digunakan untuk membentuk pose atau bentuk yang unik, sehingga dapat dikenali sebagai maksud tertentu. Penulis mengusulkan dikembangkan sistem rekognisi citra digital untuk dapat mengenali bahasa isyarat tersebut. Dengan menggunakan metode *Convolutional Neural Network* (CNN) yang merupakan bagian dari *Deep Learning* atau *Machine Learning*, sistem akan mengenali pose atau bentuk dari citra bahasa isyarat yang dimasukkan, dan memberikan luaran yang sesuai dengan maksud dari pose atau bentuk dari citra bahasa isyarat tersebut. Penelitian ini dimulai dengan pengumpulan data, baik data sekunder dari internet maupun data pribadi yang diambil secara manual. Data kemudian melalui pemrosesan awal dan diklasifikasikan dengan CNN, lalu didapatkan hasil untuk dianalisis. Apabila hasil memuaskan, model akan diekspor untuk dimasukkan ke dalam aplikasi berbasis web untuk digunakan secara *real-time*. Berdasarkan hasil pengujian, model yang terbaik untuk arsitektur adalah model EfficientNet B4 dengan menggunakan *Hyperparameter optimizer* Adam dan *learning rate* 0.001 beserta *scheduler*. Digunakan *pretrained weights* untuk meningkatkan akurasi tersebut, dan ditambahkan *Spatial transformer* untuk mencoba membuat model menjadi lebih kokoh. Ditambah dengan *pretrained weights*, model diekspor untuk digunakan secara *real-time*. Hasil pengujian *real-time* menunjukkan bahwa model mampu mendeteksi setidaknya 23 dari 26 alfabet pada latar belakang yang abstrak. Apabila diuji pada latar belakang polos seperti hitam atau putih, model mampu mendeteksi seluruh 26 alfabet dengan probabilitas yang hampir sempurna. Hal ini menunjukkan bahwa metode yang digunakan sudah mampu mengatasi masalah yang disampaikan.

Kata kunci: *convolutional neural network, bahasa isyarat, spatial transformer, klasifikasi real-time*

SIGN LANGUAGE DIGITAL IMAGE RECOGNITION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK AND SPATIAL TRANSFORMER

Abstract

Sign language is very important for a group of people, namely the deaf or dumb. To be able to communicate with people who are mute or deaf, people who are not mute or deaf require sign language to be able to understand the intentions or thoughts of those who are mute or deaf. Most conversations in sign language are carried out using the hands, where the hands and their fingers are used to form unique poses or shapes, so that they can be recognized as having certain meanings. The author proposes to develop a digital image recognition system to be able to recognize sign language. By using the Convolutional Neural Network (CNN) method which is part of Deep Learning or Machine Learning, the system will recognize the pose or shape of the entered sign language image, and provide output that matches the meaning of the pose or shape of the sign language image. This research began with data collection, both secondary data from the internet and personal data taken manually. The data then goes through initial processing and is classified with CNN, then results are obtained for analysis. If the results are satisfactory, the model will be exported to be included in a web-based application for use in real-time. Based on the test results, the best model for the architecture is the EfficientNet B4 model with the Hyperparameter consisting of optimizer Adam and learning rate 0.001 along with the scheduler. Pretrained weights were used to improve accuracy, and Spatial transformers were added to try to make the model more robust. Coupled with pretrained weights, the model is exported for use in real-time. Real-time test results show that the model is able to detect at least 23 of the 26 alphabets on an abstract background. When tested on a plain background such as black or white,

the model was able to detect all 26 alphabets with almost perfect probability. This shows that the method used is able to overcome the problem presented.

Keywords: *convolutional neural network, sign language, spatial transformer, real-time classification*

1. PENDAHULUAN

Bahasa isyarat merupakan hal yang sangat penting bagi suatu kelompok masyarakat, yaitu masyarakat bisu atau tuli. Untuk masyarakat yang bisu atau tuli, bahasa isyarat adalah metode terpenting untuk berkomunikasi. Tanpa adanya bahasa isyarat, akan sulit bagi mereka yang bisu atau tuli untuk dapat menyatakan maksud atau pikiran mereka. Untuk dapat berkomunikasi dengan masyarakat bisu atau tuli, orang yang tidak bisu atau tuli memerlukan bahasa isyarat tersebut untuk dapat mengerti maksud atau pikiran mereka yang bisu atau tuli. Setiap orang harus memiliki kemampuan menggunakan bahasa isyarat, agar dapat berkomunikasi dengan mereka yang bisu atau tuli. Bahasa isyarat diekspresikan menggunakan tangan, lengan, serta wajah dan dimengerti menggunakan mata (Mayberry & Squires, 2006).

Pembuatan sistem klasifikasi bahasa isyarat sebelumnya pernah dilakukan oleh (Someshwar, *et al*, 2020) yang membuat asisten virtual untuk bahasa isyarat menggunakan *deep learning* dan tensorflow. Kesimpulan dari penelitian ini adalah untuk dapat mendeteksi bahasa isyarat, diperlukan *dataset* tertentu untuk setiap bentuk bahasa isyarat. Hal ini dikarenakan terdapat banyak bentuk bahasa isyarat di dunia, sehingga untuk dapat mendeteksi bahasa isyarat yang ada pada regional tertentu, diperlukan *dataset* bahasa isyarat dari lokasi tersebut. Selain itu, diperlukan pula latar belakang yang mendukung ketika mengambil gambar, sehingga tidak dapat digunakan di sembarang tempat. Penelitian ini lebih memfokuskan terhadap pembuatan aplikasinya, sehingga tidak terlalu menunjukkan terhadap akurasi atau performa yang lebih mendalam.

Penelitian lain yang mendalami performa dari model *deep learning*-nya yaitu oleh (Yuan, *et al*, 2017) tentang klasifikasi wajah dan penelitian oleh (Kochgaven, et al, 2021) tentang deteksi citra Covid-19. Kedua penelitian ini tidak melakukan klasifikasi terhadap bahasa isyarat, tetapi metode untuk mendapatkan performa yang terbaik dijadikan penulis sebagai referensi. Kesimpulan dari penelitian kedua adalah untuk dapat mendeteksi wajah, ada sangat banyak elemen yang perlu diperhatikan seperti cahaya, pose, angle atau posisi kamera, dan lainnya. Tidak akan mudah untuk dapat mendapatkan hasil yang sempurna, tetapi CNN mampu mendapatkan hasil yang baik dikarenakan pembelajaran fiturnya yang kuat, sehingga mampu bertahan pada lingkungan yang kompleks. Hal yang sama juga dapat diaplikasikan terhadap bahasa isyarat, sebagaimana disebutkan pada kesimpulan penelitian pertama. Kesimpulan dari penelitian ketiga adalah X-Ray serta CT-Scan yang termasuk kepada radiografi dada sangatlah membantu untuk mendeteksi penyakit-penyakit, dan penelitian menggunakan *Transfer Learning* dari PyTorch tadi mendapatkan hasil yang memuaskan, dengan akurasi sebesar 97,78%. Hasil ini sangat memuaskan dan menjadikan penulis tertarik pada metode yang digunakan untuk dicoba diberikan pada aplikasi rekognisi bahasa isyarat.

Penelitian terakhir adalah tentang penggunaan metode Spatial Transformer dan optimisasi stokastik pada rekognisi rambu lalu lintas oleh (Arcos-García, *et al*, 2018). Penelitian ini menggunakan metode *Spatial Transformer* dan mampu mendapatkan akurasi sebesar 99,71% pada dataset dengan jumlah data uji sebanyak 12.630. Hasil ini menunjukkan kalau metode *Spatial Transformer* pantas untuk digunakan untuk mencoba meningkatkan akurasi dari penelitian.

Oleh karena itu, penulis mengusulkan dikembangkan sistem rekognisi citra digital untuk dapat mengenali bahasa isyarat tersebut. Dengan menggunakan metode *Convolutional Neural Network* (CNN) yang merupakan bagian dari *Deep Learning* atau *Machine Learning* dan dengan metode *Spatial Transformer*, sistem akan mengenali pose atau bentuk dari citra bahasa isyarat yang dimasukkan, dan memberikan luaran yang sesuai dengan maksud dari pose atau bentuk dari citra bahasa isyarat tersebut.

2. METODE PENELITIAN

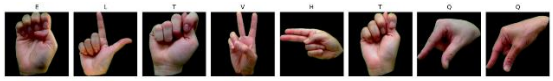
2.1 Data Penelitian

Data yang digunakan untuk *training* model adalah data sekunder yang didapat secara *open-source* dari berbagai halaman web tertentu yang menyediakan *dataset-dataset* secara gratis dan legal. Untuk penelitian ini, digunakan *dataset-dataset* dengan tema bahasa isyarat. Adapun situs yang menyediakan set data tersebut bernama Kaggle.

Data sekunder yang akan digunakan ada dua macam, yaitu data bahasa isyarat dengan *background* hitam dan data bahasa isyarat dengan *background* putih. Data dengan *background* hitam merupakan data publik dari situs Kaggle (Thakur, 2019) sebagaimana ditunjukkan pada Gambar 1. Sedangkan data dengan *background* putih juga merupakan data publik dari situs Kaggle (Lanang, 2021) sebagaimana ditunjukkan pada Gambar 2. *Dataset background* hitam memiliki jumlah sebanyak 70 gambar, sedangkan *dataset background* putih memiliki jumlah sebanyak 21 gambar.

Selain data sekunder dari internet, digunakan pula data yang diambil oleh penulis secara pribadi. Pengambilan data pribadi ini dilakukan untuk

menguji adanya perbedaan hasil pada pengujian apabila menggunakan data dengan *background* yang berbeda dan mengetahui seberapa besar pengaruh yang dimiliki oleh *background* tersebut terhadap keseluruhan percobaan, sebagaimana permasalahan ini disebutkan sebelumnya oleh (Someshwar, *et al*, 2020). Data gambar yang diambil akan memiliki pose dan bentuk bahasa isyarat yang sama dengan data sekunder dari internet tersebut, tetapi akan memiliki *background* yang bervariasi. Data yang diambil memiliki dua macam *background*, yaitu data dengan *background* karpet bermotif ditunjukkan pada Gambar 3 dan data dengan *background* pemandangan pagi ditunjukkan pada Gambar 4. *Dataset background* karpet memiliki jumlah sebanyak 54 gambar, sedangkan *dataset background* pemandangan memiliki jumlah sebanyak 40 gambar.



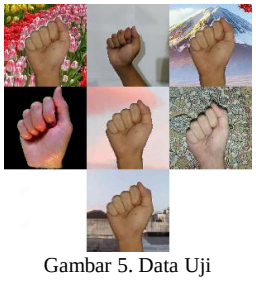
Gambar 1. Dataset Background Hitam

Gambar 2. Dataset Background Putih

Gambar 3. Dataset Background Pemandangan

Gambar 4. Dataset Background Karpet

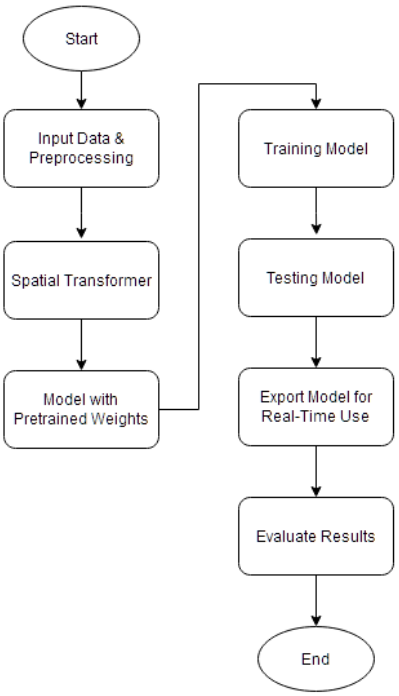
Data uji yang akan digunakan adalah *dataset* yang menggabungkan keempat *dataset* yang digunakan untuk *training* dan *validation*. Selain itu, ditambahkan pula tiga set gambar yang tidak berada dalam data *training* maupun *validation*, yaitu *dataset* dengan *background* bunga, *dataset* dengan *background* langit merah, dan *dataset* dengan *background* gunung. Penyusunan *dataset* uji per alfabet adalah satu gambar dari setiap *dataset background* hitam, putih, karpet, dan pemandangan ditambah tiga gambar dari setiap *dataset background* bunga, langit merah, dan gunung sebagaimana ditunjukkan pada Gambar 5. Total sebanyak 13 gambar untuk setiap alfabet, menjadikan 338 total gambar data uji.



Gambar 5. Data Uji

2.2 Perancangan Model

Pada perancangan model *Convolutional Neural Network* dari sistem rekognisi citra digital bahasa isyarat diawali dengan adanya pemrosesan awal dengan mengubah dimensi dari data dan juga mengkonversi data menjadi berbentuk Tensor agar dapat diberikan sebagai input kepada model CNN. Setelah dilakukan pemrosesan awal, dibuatlah arsitektur dari model CNN yang akan melakukan pembelajaran mendalam atau *Deep Learning*.



Gambar 6. Diagram Alir Perancangan Model

Penentuan Arsitektur dari model CNN ini dilakukan dengan memasukan lapisan-lapisan *neural network* yang terbaik untuk data yang digunakan. Dalam arsitektur model akan ditambahkan modul *Spatial Transformer*, yang ditambahkan sebelum bagian utama dari *neural network*. Data akan melewati lapisan-lapisan *Spatial Transformer* terlebih dahulu sebelum memasuki bagian *neural network* utama. Setelah arsitektur dari model CNN telah ditentukan, dilakukan proses pelatihan atau *Training* pada *dataset*. *Training* atau pelatihan dilakukan dengan tujuan memberikan model pengetahuan yang dibutuhkannya untuk dapat mengklasifikasikan dan merekognisi data digital. Terakhir, dilakukan pengujian dengan data uji untuk menentukan seberapa baik kinerja model dalam pekerjaan mengklasifikasi dan merekognisi *dataset* digital tersebut. Apabila kinerja model masih kurang baik, maka perlu dilakukan perbaikan atau *Tweaking* pada model dengan tujuan mendapatkan hasil yang lebih baik. Pada penelitian ini, target yang dicari adalah nilai akurasi yang tinggi. Alur perancangan model ini dilakukan sebagaimana pada Gambar 6.

Arsitektur yang akan digunakan dalam penelitian sistem rekognisi citra digital bahasa isyarat menggunakan *Convolutional Neural Network* akan menggunakan berbagai *pretrained weights* yang telah disediakan oleh *library* PyTorch. Contoh model dan *weights*-nya yang disediakan oleh *library* PyTorch yaitu ResNet, AlexNet, atau EfficientNet. Model dan *weights*-nya sudah dilatih sebelumnya menggunakan *dataset* ImageNet-1k dengan tujuan membantu meningkatkan akurasi dari model. Selain menggunakan *pretrained weights*, arsitektur model yang akan digunakan juga akan memiliki penambahan *Spatial Transformer* berupa lapisan *Localization*, *fully connected*, dan *sampler*. Penambahan *Spatial Transformer* diposisikan sebelum bagian utama arsitektur, tepatnya di awal arsitektur. Data akan melewati lapisan *Spatial Transformer* terlebih dahulu untuk ditransformasi sebelum memasuki lapisan utama dari *Convolutional Neural Network*.

3. DASAR TEORI

3.1 Bahasa Isyarat

Bahasa isyarat diekspresikan menggunakan tangan, lengan, serta wajah dan dimengerti menggunakan mata (Mayberry & Squires, 2006). Bahasa isyarat juga memiliki struktur dan peraturan untuk berbagai kata, kalimat, atau percakapan, sebagaimana bahasa lain pada umumnya. Sebagian besar percakapan pada bahasa isyarat dilakukan dengan menggunakan tangan, dimana tangan beserta jari-jarinya digunakan untuk membentuk pose atau bentuk yang unik, sehingga dapat dikenali sebagai maksud tertentu.

Pada negara tertentu, bahasa isyarat yang dikembangkan dapat berbeda dengan bahasa isyarat dari negara lain. Contoh dari bahasa isyarat yang ada pada suatu negara tertentu yaitu American Sign Language (ASL) yang merupakan bahasa isyarat populer yang berasal dari Amerika Serikat. Selain itu, terdapat pula bahasa isyarat dari Inggris yaitu British Sign Language (BSL). Selain kedua negara tersebut, terdapat Australian Sign Language (Auslan), Italian Sign Language (LIS), Japanese Sign Language (JSL), dan lainnya (Mayberry & Squires, 2006).

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah jaringan saraf/*neuron* lapisan banyak yang tidak sepenuhnya tersambung. CNN berisi lapisan *convolutional*, lapisan *sampling* atau *sub-sampling*, dan lapisan tersembunyi yang masih berupa lapisan *convolutional* atau *sampling*. Setiap lapisan *convolution* pada CNN diikuti dengan lapisan penghitung untuk dilakukan pemerataan dan ekstraksi. Adapun perhitungan yang dilakukan pada lapisan *convolutional* adalah sebagai berikut:

$$H(x,y) = b + [F(x,y) \times G(x,y)] \\ = b + \sum_{j=(-\infty)}^{\infty} \sum_{k=(-\infty)}^{\infty} F(j,k) \times G(x-j, y-k) \quad (1)$$

Keterangan:
F = *Filter* Lapisan
G = *Input feature map*
H = *Output feature map*
b = Bias
x = Sumbu X pada data dan *filter*
y = Sumbu Y pada data dan *filter*
j = *Increment* untuk sumbu X
k = *Increment* untuk sumbu Y

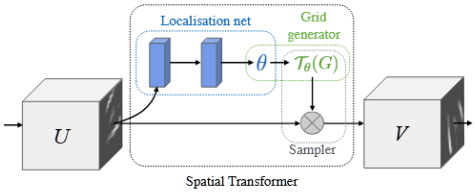
Suatu *input feature map* G dengan nilai $G(x,y)$ akan dimasukkan ke dalam lapisan konvolusi. Kernel F yang berada di dalam lapisan konvolusi memiliki nilai $F(x,y)$ untuk menjadi pengali dari *input* tersebut. Untuk setiap j pada sumbu x dan setiap k pada sumbu y, nilai *input* $G(x-j, y-k)$ akan dikalikan dengan nilai *kernel* $F(j,k)$ dan hasilnya akan dijumlahkan sesuai dengan ukuran data *input* pada nilai j dan k. Hasil dari total penjumlahan tersebut akan dijumlahkan dengan nilai *bias* b untuk menjadi nilai *output* $H(x,y)$. Perhitungan akan dilanjutkan ke *pixel* selanjutnya pada *input* G hingga seluruh *pixel* pada *output feature map* H mendapatkan hasil.

3.3 Transfer Learning

Transfer Learning merupakan salah satu teknik dari penggunaan *Deep Learning*, dimana model yang sebelumnya telah melewati tahap *training* digunakan kembali untuk mengekstrak dan *tuning* lebih lanjut pada model lain untuk memperbaiki akurasi. Keuntungan dari penggunaan teknik *Transfer Learning* terletak pada penghematan waktu penjalanan proses *training* serta penghematan *resource* dikarenakan menggunakan data yang lebih sedikit. Teknik ini dapat digunakan untuk mendeteksi berbagai macam objek (Galvez et al., 2018).

3.4 Spatial Transformer

Spatial Transformer merupakan modul yang bisa ditambahkan ke dalam *Convolutional Neural Network* yang memungkinkan manipulasi spasial data di dalam jaringan (Jaderberg, *et al*., 2015). Modul yang dapat dibedakan ini dapat dimasukkan ke dalam arsitektur *Convolutional Neural Network* yang sudah ada, memberikan *neural network* kemampuan untuk secara aktif mentransformasikan *feature map* secara spasial, tergantung pada *feature map* itu sendiri tanpa pengawasan pelatihan tambahan atau modifikasi pada proses pengoptimalan. Tahapan dari Spatial Transformer digambarkan pada Gambar 7.



Gambar 7. Diagram Alir Perancangan Model

Pada Gambar 7, mekanisme *Spatial Transformer* dibagi menjadi tiga bagian, yaitu jaringan *Localization*, *Sampling Grid*, dan *Sampler*. Jaringan *Localization* mengambil *feature map input* dan mengeluarkan parameter transformasi Teta (θ) yang harus diterapkan pada *feature map* melalui sejumlah *hidden layer*. Hasil output jaringan *Localization* berupa parameter transformasi Teta (θ) akan digunakan untuk membuat *Sampling Grid*, yang merupakan sekumpulan titik di mana *map input* harus diambil sampelnya untuk menghasilkan *output* yang telah diubah. Hal ini dilakukan oleh *generator grid*, yang merupakan komponen *Spatial Transformer* yang memiliki fungsi eksklusif untuk melakukan transformasi invers dari *output*. Terakhir, *feature map* dan *Sampling Grid* yang dihasilkan oleh *Grid Generator* diambil sebagai *input* untuk *Sampler*, komponen lain dari *Spatial Transformer* selain *Grid generator*. Tujuan dari *Sampler* adalah untuk menghasilkan *output map* yang di-*sampling* dari *input* pada titik-titik *grid*. *Sampler* mengulangi entri *grid* pengambilan sampel dan mengekstrak nilai *pixel* yang sesuai dari *input map* menggunakan interpolasi *bilinear*. *Output* dari ketiga tahapan *Spatial Transformer* tersebut kemudian akan diteruskan ke jaringan konvolusi setelahnya.

$$\begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} = \tau_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,1} & \theta_{2,3} & \theta_{2,3} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \quad (2)$$

Keterangan:

τ_θ = Transformasi Afin 2D A_θ
G_i = *Grid* dari *Output Feature Map*
A_θ = Matriks Transformasi Afin θ
x_i^s = Sumber Kordinat Sumbu X
y_i^s = Sumber Kordinat Sumbu Y
x_i^t = Target Kordinat Sumbu X
y_i^t = Target Kordinat Sumbu Y
θ = Luaran *Localization*

Kordinat (x_i^t, y_i^t) adalah target kordinat pada titik-titik *grid* (G_i) dan berasal dari *output feature map*. Kordinat (x_i^s, y_i^s) adalah sumber kordinat dari *input feature map* yang menentukan titik sampel. *Output* dari lapisan *Localization* yaitu θ menjadi penentu transformasi target, dan mungkin saja mengambil berbagai transformasi. A_θ adalah matriks transformasi afin θ. Transformasi yang didefinisikan adalah seperti *cropping*, *translation*, *rotation*, *scale*, dan *skew* untuk diterapkan pada *input feature map*, dan hanya membutuhkan 6 parameter (6 elemen A_θ) yang diproduksi oleh lapisan *Localization*.

3.5. Confusion Matrix

Confusion Matrix adalah salah satu metode pengukuran kinerja untuk masalah klasifikasi dalam machine learning, di mana *output*-nya dapat berupa dua kelas atau lebih. *Confusion Matrix* dapat berupa tabel dengan empat macam kombinasi berbeda dari nilai prediksi dan nilai sebenarnya. Keempat macam kombinasi tersebut adalah *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FP).

Tabel 1. Confusion Matrix

	Positif Sebenarnya	**Negatif Sebenarnya**
Positif Prediksi	True Positive	False Positive
Negatif Prediksi	False Negative	True Negative

Keempat kombinasi yang ditunjukkan pada Tabel 1 dapat digunakan untuk melakukan perhitungan metrik-metrik untuk mengevaluasi hasil prediksi sistem. Metrik yang digunakan untuk mengevaluasi yaitu *Accuracy* atau akurasi, *Precision*, *Recall*, dan *F1-Score*. Adapun rumus dari setiap metrik tersebut ditunjukkan pada persamaan-persamaan berikut:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (6)$$

Keterangan:

TP = *True Positive*
TN = *True Negative*
FP = *False Positive*
FN = *False Negative*

Perhitungan akurasi dilakukan dengan membagi seluruh prediksi yang benar $(TP + TN)$ oleh keseluruhan data $(TP + FP + TN + FN)$ dan digunakan untuk mengetahui keefektifan secara menyeluruh dari sistem. Perhitungan *Precision* digunakan untuk mengetahui kesepakatan kelas label data dengan label positif yang diberikan oleh sistem, dengan cara membagi nilai *True Positive* (TP) dan penjumlahan *True Positive* dan *False Positive* $(TP + FP)$. Perhitungan *Recall* digunakan untuk

mengetahui keefektifan dari sistem untuk mengidentifikasi label yang positif, dilakukan dengan membagi nilai *True Positive* (*TP*) dengan penjumlahan *True Positive* dan *False Negative* ($TP + FN$). Perhitungan *F1-Score* digunakan untuk mengetahui hubungan antara data dengan label positif dan data yang dilabelkan positif oleh sistem (Sokolova & Lapalme, 2009).

3.5 Flask

Flask adalah sebuah kerangka kerja aplikasi web yang memberikan alat, pustaka, dan teknologi untuk membuat aplikasi web. Flask merupakan modul python untuk membuat aplikasi berbasis web yang cukup mudah untuk digunakan. Dengan menggunakan flask, suatu model *deep learning* dapat diintegrasikan ke dalam aplikasi web untuk dapat digunakan (Indu, *et al.*, 2023).

4. HASIL DAN PEMBAHASAN

Pengujian pertama yang akan diuji adalah pengaturan arsitektur CNN dengan model-model yang memiliki *pretrained weights*. Adapun model-model yang akan diuji yaitu ResNet18 dan ResNet50 (He, *et al.*, 2016), AlexNet (Krizhevsky, *et al.*, 2012), dan EfficientNet B4 (Tan & Le, 2019). Selain arsitektur CNN, hal lain yang akan diuji adalah *Hyperparameter* dari proses pelatihan. Parameter yang akan diuji adalah *optimizer* dan *learning rate*. Berbagai *optimizer* yang akan diuji seperti SGD, RMSProp, dan Adam. Sedangkan untuk *learning rate*, pengujian hanya akan menguji penggunaan *learning rate scheduler* untuk menilai keefektifan *scheduler* dalam membantu proses *training*. Setelah didapat model dan pengaturan *hyperparameter* yang terbaik, dilakukan pengujian penggunaan *pretrained weights* dari arsitektur model untuk mencoba menaikkan hasil akurasi. *Spatial Transformer* juga akan diuji coba untuk melihat apakah penggunaannya mampu membuat akurasi meningkat. Terakhir, setelah didapatkan hasil yang maksimum dari kombinasi pengaturan yang terbaik, dilakukan uji coba secara *real-time*.

Pengujian secara *real-time* dilakukan dengan melakukan *export* terhadap model dengan hasil yang terbaik, lalu diuji dengan gambar nyata yang diambil menggunakan kamera atau *webcam*. Pengujian secara *real-time* juga akan memiliki variasi terhadap pengujiannya, dimana masing-masing dari keempat dataset akan diuji coba melakukan klasifikasi secara *real-time* terhadap tiga macam latar belakang, yaitu latar belakang putih polos, latar belakang hitam polos, dan latar belakang abstrak atau bermacam-macam warna. Uji coba latar belakang abstrak dilakukan untuk melihat pengaruh dari variasi warna pada latar belakang. Setelah melihat performa dari masing-masing *dataset*, dilakukan uji coba menggunakan *dataset* gabungan dari keempat *dataset* untuk mencoba mendapatkan hasil yang terbaik. Pengujian model, *hyperparameter*, *pretrained weights*, dan *Spatial Transformer* akan menggunakan akurasi sebagai metrik utama dengan mencatat juga waktu jalannya proses training. Khusus untuk *learning rate*, ditambahkan metrik berupa *epoch* hingga konvergen untuk menilai seberapa cepat model mampu mencapai titik konvergen pada proses pelatihan.

4.1 Pengujian Arsitektur Model

Dalam pengujian ini digunakan empat tipe model. ResNet18 cukup populer di kalangan *dataset* kecil, tetapi juga layak digunakan pada *dataset* besar. Selain ResNet18, model populer lain yang juga digunakan adalah AlexNet. ResNet50 juga digunakan untuk membandingkan dengan tipe sebelumnya. Terakhir, EfficientNet B4 digunakan karena jumlah *trainable parameter*-nya yang tinggi, tetapi tidak terlalu besar untuk mencegah *overfitting*. Pengujian dilakukan dengan menggunakan *Hyperparameter* yang sama, yaitu *optimizer* Adam, *learning rate* 0.001, dan *loss Cross Entropy*. Pengujian ini tidak menggunakan *Spatial Transformer*. Hasil dari pengujian ditunjukkan pada Tabel 2.

Tabel 2. Hasil Pengujian Arsitektur Model

Model	Akurasi			Waktu *Training*
	Train	*Validation*	*Test*	
ResNet18	99,89%	100%	54,73%	3585s
AlexNet	99,93%	100%	57,39%	**3350s**
ResNet50	99,83%	100%	57,10%	4167s
Efficient Net B4	**99,96%**	100%	**78,10%**	4924s

Hasil pengujian pada Tabel 2 menunjukkan bahwa model dengan arsitektur EfficientNet B4 memiliki akurasi *testing* yang tertinggi sebesar 78,10%. Hasil ini menunjukkan bahwa model EfficientNet B4 mampu mendeteksi alfabet dalam data uji hingga didapatkan 78,10% kebenarannya. Selain memiliki akurasi *testing* yang tertinggi, akurasi *training* dan *validation* dari model EfficientNet B4 juga yang terbesar, yaitu sebesar 99,96% untuk *training* dan 100% untuk *validation*. Seluruh model mendapatkan akurasi *training* yang tidak begitu jauh dari satu sama lain, dengan perbedaan antara hasil terendah dan tertinggi hanya sebesar 0,13%. Akurasi validation tampak merata untuk keempat arsitektur yang diuji. Waktu pelatihan tercepat jatuh kepada model AlexNet dengan waktu 3350 detik atau hanya 55 menit dan terlama pada model EfficientNet B4 dengan waktu 4924 detik atau 1 jam 22 menit. Hasil ini menunjukkan bahwa model EfficientNet B4 memiliki kemampuan terbaik dalam prediksi data uji dibanding model lainnya, walau waktu pelatihannya memakan waktu paling lama.

4.2 Pengujian *Hyperparameter*

Pengujian *Hyperparameter* dilakukan dengan menentukan pengaturan *Hyperparameter* yang dapat

memberikan hasil yang terbaik, dan pengaturan *Hyperparameter* yang akan diuji yaitu *optimizer* dan penggunaan *learning rate scheduler*. Untuk model pengujian digunakan model EfficientNet B4 untuk mencari kombinasi *Hyperparameter* yang terbaik. Jumlah *epoch* yang digunakan adalah 20, dan *Learning Rate* yang digunakan yaitu 0.001. Pengujian *optimizer* dilakukan dengan mencoba satu per satu *optimizer* yang dapat digunakan. Opsi *optimizer* yang akan dicoba yaitu Adam, SGD, dan RMSProp. Pengujian ini juga tidak menggunakan *Spatial Transformer*. Hasil dari pengujian ditunjukkan pada Tabel 3.

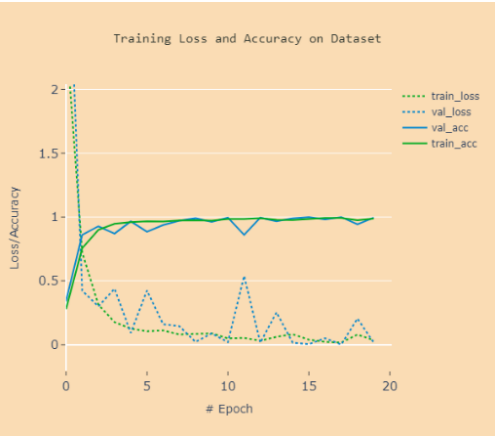
Tabel 3. Hasil Pengujian *Optimizer*

Optimizer	**Akurasi**			**Waktu**
	Train	***Validation***	***Test***	***Training***
Adam	99,96%	100%	**78,10%**	4924s
SGD	4,79%	3,85%	3,84%	**4707s**
RMSProp	**99,98%**	100%	57,39%	4767s

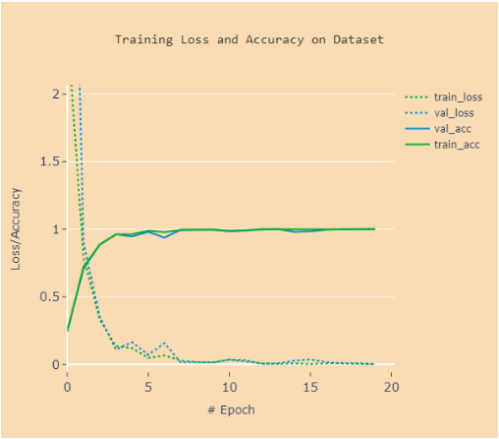
Berdasarkan hasil pengujian Tabel 3, dapat dilihat bahwa model yang menggunakan *optimizer* SGD terlihat tidak berkembang. Akurasi yang didapat terlihat tidak mampu mencapai 5%, baik akurasi *training*, *validation*, maupun *testing*. Hal ini diduga karena *optimizer* SGD memiliki sifat yang stokastik atau *random* dalam memilih data latih untuk setiap *step*, sehingga membutuhkan lebih banyak *epoch* dibandingkan dengan *optimizer* lain. Model dengan *optimizer* RMSProp terlihat cukup baik, akan tetapi akurasi *testing*-nya tidak sebaik dengan model yang menggunakan *optimizer* Adam. Hasil ini diduga dikarenakan *optimizer* Adam merupakan hasil perkembangan gabungan dari kedua *optimizer*. Setelah *optimizer*, pengujian *Hyperparameter* selanjutnya adalah penggunaan *learning rate scheduler*.

Tabel 4. Hasil Pengujian Arsitektur Model

Learning Rate	**Akurasi**		***Epoch* mencapai Konvergen**
	Train	***Test***	
LR = 0.001	98,83%	76,33%	Belum Konvergen
LR = 0.001 + *Scheduler*	**99,96%**	**78,10%**	***Epoch* ke-8**



Gambar 8. Grafik Proses Training Tanpa Scheduler



Gambar 9. Grafik Proses Training dengan Scheduler

Pengujian *learning rate scheduler* dilakukan dengan menguji performa dari pelatihan apabila penggunaan *scheduler* diberikan. Metrik utama yang akan diperhatikan adalah jumlah *epoch* yang diperlukan dari model hingga mencapai titik konvergen, atau titik dimana akurasi pelatihan tidak lagi fluktuatif. Untuk nilai *learning rate* yang akan digunakan yaitu 0.001, dan *learning rate scheduler* yang digunakan adalah *Exponential Learning Rate Scheduler* dari *library* 'torch optim'.

Sebagaimana ditunjukkan pada Tabel 4, Gambar 8, dan Gambar 9, pengujian penggunaan *Learning Rate Scheduler* pada model CNN tampak memiliki pengaruh terhadap proses pelatihan maupun hasil dari *output* model tersebut. Dengan adanya *Learning Rate Scheduler*, proses pelatihan model terlihat menjadi lebih stabil. Perubahan terhadap nilai *learning rate* menjadikan proses pelatihan menjadi lebih teratur dan tidak fluktuatif, sebagaimana dapat dilihat pada Gambar 9. Hasil yang fluktuatif berarti model yang dilatih masih belum mampu memberikan hasil yang konsisten. Hal ini memiliki pengaruh terhadap hasil *output*, dibuktikan dengan akurasi yang didapat terlihat lebih kecil dibandingkan dengan model yang menggunakan *scheduler*. Gambar 8 menunjukkan bahwa pengujian yang tidak menggunakan *scheduler* tampak fluktuatif dan tidak teratur. Selain itu, hal ini juga memiliki pengaruh terhadap hasil *output*, dibuktikan dengan akurasi yang didapat terlihat lebih kecil dibandingkan dengan model yang menggunakan *scheduler* sebagaimana ditunjukkan pada Tabel 4. Baik akurasi *training* maupun *testing*, penggunaan *scheduler* terlihat mampu menaikkan akurasi pada model yang dilatih

4.3. Pengujian Penggunaan *Pretrained Weights*

Pengujian penggunaan *pretrained weights* dilakukan dengan menggunakan parameter 'pretrained=True' ketika memanggil model menggunakan *library torch* untuk menggunakan *weights* yang sebelumnya telah dilatih dan disediakan pada pemanggilan model. Model yang akan digunakan sama seperti pengujian sebelumnya, yaitu ResNet18, AlexNet, ResNet50 dan EfficientNet B4.

Tujuan pengujian adalah untuk mengetahui seberapa besar pengaruh *pretrained weights* pada proses pelatihan dan hasilnya. Pengujian menggunakan *optimizer* Adam dan *learning rate* 0.001 serta *scheduler*-nya. Pengujian ini tidak menggunakan *Spatial Transformer*.

Tabel 5. Hasil Pengujian *Pretrained Weights*

Model	Akurasi			Waktu *Training*
	Train	*Validation*	*Test*	
ResNet18	99,89%	100%	54,73%	3585s
ResNet18 *Pretrained*	**100%**	100%	84,91%	3566s
AlexNet	99,93%	100%	57,39%	3350s
AlexNet *Pretrained*	99,96%	99,04%	75,44%	**3345s**
ResNet50	99,83%	100%	57,10%	4167s
ResNet50 *Pretrained*	99,89%	99,52%	65,09%	4119s
EfficientNet B4	99,96%	100%	78,10%	4924s
EfficientNet B4 *Pretrained*	99,96%	100%	**100%**	4870s

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 5, dapat dilihat bahwa terdapat peningkatan pada akurasi *testing* untuk seluruh model. Peningkatan tersebut berbeda-beda pada setiap model, berkisar antara angka 8% hingga 30%. Model ResNet18 mendapatkan peningkatan akurasi *testing* yang paling besar, yaitu 30,18%. Model AlexNet juga mendapatkan peningkatan, yaitu sebesar 18,05%. Model ResNet50 mendapatkan peningkatan paling kecil, yaitu hanya sebesar 7,99%. Terakhir, model EfficientNet B4 mendapatkan peningkatan sebesar 21,9% dan mampu mencapai akurasi maksimum, yaitu 100%. Hasil ini menunjukkan bahwa model EfficientNet B4 sudah mampu mendeteksi seluruh data uji secara sempurna diduga karena tingkat kesulitan untuk mendeteksi data uji tidak begitu tinggi. Dari keempat model, dapat dilihat bahwa model EfficientNet B4 masih memiliki akurasi terbesar, bahkan mampu mencapai angka maksimum. Hal ini menunjukkan bahwa model EfficientNet B4 merupakan model yang terbaik dari keempat model yang diuji dalam mengklasifikasikan dataset yang diberikan.

4.4. Pengujian Penggunaan *Spatial Transformer*

Pengujian ini dilakukan dengan menambahkan lapisan *localization* dan lapisan *Fully Connected* yang merupakan bagian dari *Spatial Transformer Network*. Lapisan-lapisan ini ditambahkan tepat sebelum dimasukkan ke model EfficientNet B4 yang menjadi arsitektur utama. Pengujian akan menggunakan *optimizer* Adam dan *learning rate* 0.001 serta *scheduler*-nya.

Tabel 6. Hasil Pengujian *Spatial Transformer*

Model	Akurasi *Train*	Akurasi *Test*	Waktu *Training*
EfficientNet B4	99,96%	78,10%	**4924s**
EfficientNet B4 dengan *Spatial Transformer*	100%	86,09%	5950s
EfficientNet B4 *Pretrained* dengan *Spatial Transformer*	100%	**100%**	6537s

Dari hasil pengujian yang ditunjukkan pada Tabel 6, terlihat terdapat peningkatan akurasi *Test* pada model EfficientNet B4 dengan *Spatial Transformer*, yaitu sebesar 8%. Akurasi yang didapat apabila EfficientNet B4 *Pretrained* menggunakan *Spatial Transformer* mampu mencapai akurasi maksimum pada data *Test*, yaitu sebesar 100%. Adanya peningkatan akurasi menunjukkan bahwa *Spatial Transformer* memiliki kemampuan untuk meningkatkan kemampuan dari model dalam klasifikasi, sehingga pantas untuk digunakan untuk model akhir. Model akhir ini mampu mendapatkan nilai maksimum dari data *testing*, sehingga siap untuk diuji coba secara *real-time*.

4.5. Pengujian Secara *Real-Time*

Pengujian secara *real-time* dilakukan dengan mencoba *dataset* satu per satu terhadap suatu kondisi atau tampilan latar belakang tertentu dan menilai keefektifan masing-masing data terhadap masing-masing tampilan latar belakang, yaitu latar belakang putih, hitam, dan abstrak. Setelah itu, keempat *dataset* akan digabungkan menjadi satu untuk dicoba melihat keefektifannya terhadap masing-masing tampilan latar belakang. Model yang digunakan adalah model dengan pengaturan serta arsitektur yang terbaik, yaitu EfficientNet B4 *Pretrained* dengan *Spatial Transformer*. Data gambar akan di-*convert* menjadi *single channel* agar mempermudah prediksi melalui aplikasi.

Tabel 7. Hasil Pengujian Secara *Real-Time*

Dataset	Total Benar & Akurasi	Latar Belakang Pengujian		
		Putih	Hitam	Abstrak
Latar Belakang Putih	Total Benar	22	23	13
	Akurasi	85%	88%	50%
Latar Belakang Hitam	Total Benar	13	16	6
	Akurasi	50%	62%	23%
Latar Belakang Karpet	Total Benar	9	12	3
	Akurasi	35%	46%	12%
Latar Belakang Pemandangan	Total Benar	9	4	3
	Akurasi	35%	15%	12%
Gabungan	Total Benar	**26**	**26**	**23**
	Akurasi	**100%**	**100%**	**88%**

Tabel 7 menunjukkan bahwa akurasi yang didapat menggunakan *dataset* latar belakang hitam terlihat masih kurang baik. *Dataset* masih mampu melakukan klasifikasi pada latar belakang pengujian

yang polos, seperti latar belakang putih dan hitam. Total prediksi yang benar masih mampu mencapai 50% dari seluruh alfabet pada latar belakang putih dan sebesar 62% pada latar belakang hitam. Hasil pengujian pada latar belakang abstrak terlihat hanya sebesar 23%.

Dapat juga dilihat pada Tabel 7 bahwa *dataset* dengan latar belakang putih mampu melakukan klasifikasi dengan baik pada latar belakang pengujian putih dan hitam polos, tetapi tidak terlalu buruk pada klasifikasi dengan *background* abstrak. Total benar yang didapat pada latar belakang putih sebesar 85% dan latar belakang hitam 88% dari seluruh alfabet. Hasil pengujian pada latar belakang abstrak hanya sebesar 50%, tetapi hasil ini masih lebih baik jika dibandingkan dengan *dataset* berlatar belakang hitam.

Hasil pengujian *dataset* berlatar belakang pemandangan terlihat kurang baik. Pada latar belakang putih, total benar yang didapat hanya sebesar 35% dari seluruh alfabet, hasil yang lebih buruk jika dibandingkan dengan kedua dataset di pengujian sebelumnya. Tetapi, pengujian pada latar belakang hitam terlihat lebih buruk, hanya sebesar 15%. Pengujian pada latar belakang abstrak terlihat tidak jauh, yaitu hanya sebesar 12%.

Dataset berlatar belakang karpet yang ditunjukkan pada Tabel 7 terlihat lebih baik daripada pengujian pada *dataset* berlatar belakang pemandangan, tetapi masih kurang jika dibandingkan dengan *dataset* berlatar belakang hitam dan putih. Pengujian pada latar belakang putih terlihat mendapatkan total benar sebesar 35%, dan pengujian pada latar belakang hitam terlihat mendapatkan total benar sebesar 46% dari seluruh alfabet. Tetapi, pengujian pada latar belakang abstrak masih terlihat buruk, yaitu hanya 12%.

Setelah keempat dataset diatas digabungkan, terlihat hasil pengujian secara *real-time* yang didapat cukup memuaskan. Sebagaimana ditunjukkan pada Tabel 7, hasil pengujian *dataset* gabungan pada latar belakang putih terlihat mencapai 100% total benar. Pengujian pada latar belakang hitam juga mencapai 100% total benar. Terakhir, pengujian pada latar belakang abstrak juga terlihat baik, yaitu sebesar 88%.

5. KESIMPULAN

Hasil penelitian sudah mampu dengan baik melakukan prediksi pada setiap alfabet, akan tetapi masih memiliki kekurangan jika diuji pada tampilan dengan latar belakang yang abstrak atau bermacam-macam warna. Adapun saran untuk penelitian selanjutnya, penulis menyarankan menggunakan model lain yang lebih baik agar mendapatkan performa yang lebih bagus. Selain itu, dapat digunakan suatu metode khusus yang dapat membedakan beberapa alfabet yang cukup mirip antara satu sama lain. Pengambilan *dataset* yang lebih profesional juga disarankan dengan harapan mampu mendapatkan akurasi yang lebih baik dan lebih dapat diandalkan. Penambahan *dataset* yang lebih bervariasi diharapkan mampu meningkatkan hasil probabilitas pada setiap alfabet. Terakhir, aplikasi yang digunakan untuk pengujian *real-time* mungkin dapat digunakan aplikasi berbasis *mobile* seperti *android*.

DAFTAR PUSTAKA

ARCOS-GARCÍA, Á., ALVAREZ-GARCIA, J.A. AND SORIA-MORILLO, L.M., 2018. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural Networks*, *99*, pp.158-165.

BAGHEL, R., PAHADIYA, P. AND SINGH, U., 2022, June. Human Face Mask Identification using Deep Learning with OpenCV Techniques. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)* (pp. 1051-1057). IEEE.

CHIRODEA, M.C., NOVAC, O.C., NOVAC, C.M., BIZON, N., OPROESCU, M. AND GORDAN, C.E., 2021, July. Comparison of tensorflow and pytorch in convolutional neural network-based applications. In *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)* (pp. 1-6). IEEE.

DAS, P., AHMED, T. AND ALI, M.F., 2020, June. Static hand gesture recognition for american sign language using deep convolutional neural network. In *2020 IEEE region 10 symposium (TENSYMP)* (pp. 1762-1765). IEEE.

GALVEZ, R.L., BANDALA, A.A., DADIOS, E.P., VICERRA, R.R.P. AND MANINGO, J.M.Z., 2018, October. Object detection using convolutional neural networks. In *TENCON 2018-2018 IEEE Region 10 Conference* (pp. 2023-2027). IEEE.

HE, K., ZHANG, X., REN, S. AND SUN, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

INDU, M., SWETHA, N. AND SARITHA, C., 2023, March. Smart Chatbot for College Information Enquiry Using Deep Neural Network. In *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 991-994). IEEE.

JADERBERG, M., SIMONYAN, K. AND ZISSERMAN, A., 2015. Spatial transformer networks. *Advances in neural information processing systems*, *28*.

JALAL, M.A., CHEN, R., MOORE, R.K. AND MIHAYLOVA, L., 2018, July. American sign language posture understanding with deep neural networks. In *2018 21st International*

Conference on Information Fusion (FUSION) (pp. 573-579). IEEE.

JU, Y., WANG, X. AND CHEN, X., 2019, April. Research on OMR recognition based on convolutional neural network tensorflow platform. In *2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)* (pp. 688-691). IEEE.

KINGMA, D.P. AND BA, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

KRIZHEVSKY, A., SUTSKEVER, I. AND HINTON, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

LANANG, A.A.M., 2021. *Datasets SIBI Sign Language Alphabets*. Kaggle. Tersedia pada: < https://www.kaggle.com/datasets/mlanangafkaar/datasets-lemlitbang-sibi-alphabets> [Diakses 10 Juli 2023].

MAYBERRY, R.I. AND SQUIRES, B., 2006. Sign language acquisition. *Encyclopedia of language and linguistics*, *11*, pp.739-43.

RASCHKA, S. AND MIRJALILI, V., 2019. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd.

SARKAR, D., BALI, R. AND GHOSH, T., 2018. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing Ltd.

SOKOLOVA, M. AND LAPALME, G., 2009. A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), pp.427-437.

SOMESHWAR, D., BHANUSHALI, D., CHAUDHARI, V. AND NADKARNI, S., 2020, July. Implementation of Virtual Assistant with Sign Language using Deep Learning and TensorFlow. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 595-600). IEEE.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. AND SALAKHUTDINOV, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

TAN, M. AND LE, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

TAQI, A.M., AWAD, A., AL-AZZO, F. AND MILANOVA, M., 2018, April. The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 140-145). IEEE.

TASKIRAN, M., KILLIOGLU, M. AND KAHRAMAN, N., 2018, July. A real-time system for recognition of American sign language by using deep learning. In *2018 41st international conference on telecommunications and signal processing (TSP)* (pp. 1-5). IEEE.

THAKUR, A., 2019. *American Sign Language Dataset*. Kaggle. Tersedia pada: < https://www.kaggle.com/datasets/ayuraj/asl-dataset> [Diakses 10 Juli 2023].

YUAN, L., QU, Z., ZHAO, Y., ZHANG, H. AND NIAN, Q., 2017, March. A convolutional neural network based on TensorFlow for face recognition. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 525-529). IEEE.