PENGENALAN GESTUR TANGAN STATIS MENGGUNAKAN CNN DENGAN ARSITEKTUR EFFICIENT-NET B4

Bima Kusuma Wardana[1], Ema Rachmawati[2], Tjokorda Agung Budi Wirayuda [3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]bimakusumawardana@students.telkomuniversity.ac.id, [2]emarachmawati@telkomuniversity.ac.id,
[3]cokagung@telkomuniversity.ac.id

Abstrak

Komunikasi adalah suatu bentuk interaksi antara individu dengan individu lain, individu dengan kelompok, atau kelompok dengan kelompok. Komunikasi adalah suatu bentuk penyampaian isi pemikiran dari individu atau kelompok. Namun banyak sekali manusia berkebutuhan khusus yang kesulitan untuk berkomunikasi. Sering kali kita kesulitan untuk memahami orang-orang berkebutuhan khusus. Jika ada orang tunarungu, kita sulit untuk membuat mereka mengerti dengan apa yang kita bicarakan apalagi jika orang tunarungu tersebut tidak bisa menggunakan bahasa isyarat. Jika kita berkomunikasi dengan orang tuna wicara, orang tuna wicara tersebut kesulitan untuk membuat kita memahami apa yang mereka akan sampaikan, keadaan tersebut akan semakin sulit jika kita tidak bisa menggunakan bahasa isyarat. Oleh karena itu, pada tugas akhir ini dibangun sebuah sistem menggunakan metode *CNN* dengan arsitektur Efficient-Net B4 sebagai pengenalan bahasa isyarat dari gestur tangan. Sistem telah diuji pada *dataset ASL Alphabets* yang terdiri dari 87000 citra yang terbagi kedalam 29 kelas. Dataset dibagi sehingga didapat data *training* 70%, data validasi 15%, dan data *testing* 15%. Pengujian dengan skenario terbaik dilakukan dengan menggunakan *Cyclical Learning Rate (CLR)* yang membuat nilai *learning rate* pada saat proses *training* menjadi dinamis(berubah-ubah) pada rentang tertentu dan menggunakan *input size* yang lebih kecil yaitu 64x64x3 agar didapat kompleksitas yang lebih kecil. Hasil evaluasi dengan skenario terbaik meraih akurasi sebesar 99,81% untuk *testing* dan membutuhkan waktu 0,056 detik untuk mengenali gestur tangan.

Kata kunci : Komunikasi, gestur tangan, bahasa isyarat, CNN, Efficient-Net B4, Cyclical Learning Rate

Abstract

Communication is a shape of interaction between individuals and other individuals, between individuals and groups, or between groups and other groups. Communication is a shape of expression of their mind. But person with the disabilities find theirself difficult in communication. And we often find ourself difficult to understand what the person with a disability communicate to us. If we find a deaf person, it is difficult for us to make them understand what we are talking about, especially if the deaf person cannot use sign language. If we do communication with the mute person, it is difficult for the mute person to make us understand about what they are going to convey, the situation will be even more difficult if we don't know anything about sign language. Therefore, in this final task we build a sistem that can recognize the sign language using CNN with Efficient-Net B4 architecture. The sistem has been tested to ASL Alphabets Dataset wich consists of 87000 image divided into 29 classes. The dataset is splitted so that 70% training set, 15% validation set, 15% testing set. The best-skenario testing is done using Cyclical Learning Rate (CLR) which makes the learning rate value in training process used dynamically(fluctuative) in a certain range and using the smaller input size, 64x64x3 so it can decrease the complexity. The Evaluation results with the best skenario achieve 99,81% in accuracy and need 0,056 s to recognize a hand gesture.

Keywords: Communication, hand gesture, sign language, CNN, EfficientNetB4, Cyclical Learning Rate

1. Pendahuluan

Latar Belakang

Komunikasi adalah sesuatu yang lumrah dilakukan antar sesama manusia. Komunikasi adalah bentuk penyampaian informasi dari seseorang atau beberapa orang, kelompok, organisasi, atau masyarakat. Fungsi dari komunikasi itu sendiri adalah untuk menyampaikan informasi, menyampaikan pendapat, bentuk interaksi terhadap orang lain, menjaga jalinan hubungan antar orang agar tetap baik, dan untuk mengekspresikan diri. Komunikasi dikatakan berhasil jika kedua pihak atau lebih yang melakukan komunikasi bisa menerima dengan benar atas komunikasi yang disampaikan. Ada beberapa jenis komunikasi yang biasa digunakan menurut cara penyampaiannya, yaitu komunikasi secara verbal, non verbal, dan tertulis.

Orang-orang difabel yang tidak bisa berkomunikasi secara verbal biasanya berkomunikasi secara non-verbal. Komunikasi non verbal yang biasa digunakan orang-orang difabel adalah bahasa isyarat. Namun, banyak sekali orang yang tidak bisa memahami bahasa isyarat. Hal itu membuat terjadinya keterbatasan dalam berkomunikasi. Oleh karena itu, diperlukan sebuah sistem untuk mengatasi keterbatasan dalam berkomunikasi tersebut. Sistem yang perlu dibangun dari permasalahan itu adalah sistem yang bisa menerjemahkan bahasa isyarat. Hal itu adalah ide yang mendasari tercetusnya Tugas Akhir ini, penulis termotivasi untuk membuat komunikasi menjadi lebih mudah untuk siapapun. Sistem yang dibangun adalah sistem yang bisa mengenali bahasa isyarat huruf alfabet yang diperagakan menggunakan gestur tangan secara statis.

Dalam penelitian yang dilakukan sebelumnya tentang *static hand gesture* oleh Atoany Nazareth Fierro Radilla dan Karina Ruby Perez Daniel yang berjudul "*Siamese Convolutional Neural Network for ASL Alphabet Recognition*"[1]. Dalam penelitian tersebut menggunakan metode *CNN* dengan arsitektur *Siamese* untuk membedakan kemiripan dari gestur di setiap kelasnya dengan mengalokasikan 30 *epoch* untuk melakukan proses *training*. Dalam penelitian itu dihasilkan hasil evaluasi performa sistem yang cukup bagus yaitu dengan tingkat akurasi sebesar 96%. Kesalahan hasil prediksi sering terjadi pada kelas M, kelas N, kelas R dan kelas U. Banyak prediksi yang harusnya memprediksi kelas N, tetapi memprediksi kelas M, begitu juga sebaliknya. Banyak prediksi yang harusnya memprediksi kelas M , tetapi memprediksi kelas N. Hal yang sama terjadi pada kelas R dan U. Penelitian serupa juga dilakukan oleh Shweta Upadhyay, Dr. R.K. Sharma, Dr. Prashant Singh Rana pada tahun 2020 dari Thapar Institute of Engineering and Technology, Patiala yang berjudul "*Sign Language Recognition with Visual Attention*"[2]. Pada penelitian tersebut, penulis melakukan pengenalan bahasa isyarat menggunakan *dataset ASL Alphabet* yang terdiri dari 29 kelas, yaitu kelas alfabet A-Z ditambah dengan kelas 'hapus', 'spasi', dan '*no gesture*'. Untuk keperluan pengujian, penulis melakukan pembagian dataset ke dalam *training set* dan *testing set* dengan perbandingan 9:1. Pendekatan yang dilakukan adalah dengan menggunakan metode *Faster-RCNN*. Dari pendekatan tersebut hasil evaluasi mencapai tingkat akurasi sebesar 94%.

Lalu ada penelitian lain yang sudah dilakukan oleh Akm Ashiquzzaman, Hyunmin Lee, Kwangki Kim, Hye-Young Kim, Jaehyung Park, dan Jinsul Kim, dengan judul "*Compact Spatial Pyramid Pooling Deep Convolutional Neural Network Based Hand Gestures Decoder*"[3]. Dalam penelitian tersebut, penulis menggunakan metode *CNN* namun dengan menambahkan *Spatial Pyramid Pooling* dalam arsitektur *CNN*-nya untuk mengenali bahasa isyarat menggunakan *dataset ASL Aphabet.* Pada penelitian tersebut didapatkan hasil evaluasi secara keseluruhan dengan tingkat akurasi, *recall*, dan *f1-score* mencapai 94%, presisi 95%, dengan waktu proses prediksi untuk 1 data adalah 0.013 s. Misklasifikasi paling banyak terjadi pada kelas M dengan banyak memprediksi kelas N. misklasifikasi terjadi karena gestur M dan N terlihat sangat mirip. Penelitian serupa juga pernah dilakukan oleh P. V. S. M. S. Kartik, Konjeti B. V. N. S. Sumanth, V. N. V. Sri Ram, P. Prakash pada tahun 2020 dengan judul "Sign Language Recognition with Visual Attention"[4]. Pada penelitian tersebut, dataset yang digunakan adalah ASL Alphabet yang terdiri dari 29 kelas antara lain 26 kelas alfabet huruf A-Z ditambah dengan kelas 'hapus', 'spasi', dan '*no gesture*'. Untuk keperluan training dan testing penulis membagi dataset tersebut kedalam training set dan testing set dengan perbandingan 9:1. Dalam penelitian tersebut penulis menggunakan pendekatan *deep learning* dengan *convolutional neural networks* dengan arsitektur ResNet-50. Hasil evaluasi yang diperoleh dari penelitian tersebut adalah tingkat akurasi yang mencapai 98,67%.

Karena penulis berpandangan bahwa untuk terjadinya komunikasi yang efektif maka diperlukan suatu sistem yang dalam evaluasinya akurasinya yang hampir mendekati 100%, sehingga dalam sisi akurasi penulis berpandangan masih ada peluang untuk meningkatkannya agar menjadi seakurat mungkin. Penulis mengusulkan sebuah metode yang masih baru yaitu *CNN* dengan arsitektur *Efficient-Net B4* yang dirumuskan oleh Mingxing Tan dan Quoc V. Le pada tahun 2019[5]. Pendekatan tersebut dipilih karena metode tersebut dalam penelitian Mingxiang Tan dan Quoc V. Le[5], *Efficient-Net B4* mendapatkan hasil evaluasi yang lebih baik dari sisi akurasi dibandingkan dengan *ResNet-50* bahkan *ResNet-152*. Pada penelitian ini sistem dibangun untuk mengenali gestur tangan untuk mengenali dan menerjemahkan bahasa isyarat. Pada penelitian ini penulis membuat 4 skenario untuk mengenali gestur tangan sekaligus untuk mengetahui pengaruh perbedaan *input size* pada model dan penggunaan *learning rate* yang dinamis menggunakan *Cyclical Learning rate* atau *CLR* terhadap hasil evaluasi performa sistem dan efisiensi waktu. Cyclical Learning Rate pertama kali dipublikasikan oleh Leslie N. Smith pada tahun 2017[6]. Dimana pada penelitian tersebut, Leslie N. Smith melakukan perbandingan dengan melakukan training menggunakan model dengan macam-macam arsitektur. Setiap arsitekturnya dilakukan pengujian dengan menggunakan *CLR* dan tidak menggunakan *CLR*. Hasilnya pada semua model yang digunakan, pengujian yang menggunakan *CLR* mendapatkan hasil evaluasi yang lebih baik dan mengalami konvergen pada nilai *epoch* yang lebih kecil dibandingkan dengan pengujian tanpa menggunakan *CLR.*

Topik dan Batasannya

Berdasarkan latar belakang yang dijelaskan, permasalahan yang dapat diangkat dalam tugas akhir ini, yaitu membangun sebuah sistem yang dapat mengenali bahasa isyarat dengan gestur tangan statis menggunakan *CNN* dengan arsitektur *Efficient Net B4*. Performa sistem dalam mengenali bahasa isyarat dengan gestur tangan statis. Pengaruh variasi *input size* dan *CLR* terhadap performa sistem yang dibangun.

Batasan dari sistem yang dibangun dalam tugas akhir ini adalah sistem hanya mengenali huruf alphabet ditambah dengan isyarat untuk spasi, hapus, dan no gesture yang ada pada *dataset ASL Alphabet*(29 kelas). Penelitian ini tidak menerapkan *hand detection*, sehingga data inputan yang dimasukan sudah berupa citra tangan yang memperagakan bahasa isyarat dengan format citra .jpg. *Dataset* diperagakan dengan tangan kanan dari satu orang saja. Latar dalam dataset yang digunakan tidak terlalu beragam.

Tujuan

Tujuan yang ingin diraih dari tugas akhir ini adalah untuk membangun sebuah sistem yang bisa mengenali dan menerjemahkan bahasa isyarat dengan gestur tangan statis dengan menggunakan pendekatan *CNN* dengan arsitektur *Efficient-Net B4* dan menguji pengaruh variasi *input size* dan *CLR* terhadap performa dan efisiensi sistem.

Organisasi Tulisan

Pada bagian pertama menjelaskan pendahuluan, pada bagian kedua menjelaskan studi terkait, pada bagian ketiga menjelaskan sistem yang dibangun, pada bagian keempat menjelaskan evaluasi dari sistem yang dibangun dan pada bagian kelima menjelaskan tentang kesimpulan dan saran.

2. Studi Terkait

Penelitian mengenai *static hand gesture* sudah pernah dilakukan oleh Akm Ashiquzzaman, Hyunmin Lee, Kwangki Kim, Hye-Young Kim, Jaehyung Park, dan Jinsul Kim, dengan judul "*Compact Spatial Pyramid Pooling Deep Convolutional Neural Network Based Hand Gestures Decoder*"[3]. Penelitian tersebut menggunakan menggunakan *dataset* yang sama dengan yang digunakan oleh penulis yaitu *ASL Alphabets Dataset*. *Dataset* tersebut terdiri dari 29 kelas, 26 kelas alfabet dari huruf A hingga Z, ditambah dengan kelas 'del', 'nothing', dan 'space'. Penelitian tersebut membuktikan bahwa menggunakan pendekatan metode *DCNN* dengan menambahkan *Spatial Pyramid Pooling* dalam arsitektur *DCNN*-nya memperoleh hasil evaluasi secara keseluruhan dengan tingkat akurasi, recall, dan f1-score mencapai 94%, presisi 95%, dengan waktu proses prediksi untuk 1 data adalah 0.013 s. Misklasifikasi paling banyak terjadi pada kelas M dengan banyak memprediksi kelas N. misklasifikasi terjadi karena gestur M dan N terlihat sangat mirip.

Convolutional Neural Networks

Convolution Neural Network adalah perpaduan antara konvolusi citra untuk proses ekstraksi ciri, dan *neural network* untuk klasifikasi. Berdasarkan arsitektur LeNet [7], terdapat 4 macam *layer* utama pada sebuah *CNN* yaitu *convolution layer, relu layer, subsampling layer, dan fully connected layer*. Fungsi *layer* awal sebagai metode ekstraksi fitur, maka jumlah layer dalam sebuah CNN tidak memiliki aturan universal tergantung *dataset* yang digunakan. Karena hal tersebut, jumlah layer pada jaringan serta jumlah neuron pada masing-masing layer dianggap sebagai hyper parameter dan dioptimasi menggunakan pendekatan searching. Berikut ini adalah penjelasan mengenai masing-masing layer.

1. *Convolutional layer*, melakukan operasi konvolusi pada output dari layer sebelumnya. Layer tersebut adalah proses utama yang mendasari sebuah *CNN*. Konvolusi adalah suatu istilah matematis yang berarti mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang.
2. *ReLu layer* atau *Rectified Linear Unit Layer*, pada *layer* ini dapat diibaratkan seperti *thresholding* atau sama halnya seperti fungsi aktivasi pada jaringan syaraf tiruan.
3. *Subsampling layer*, Subsampling adalah proses mereduksi ukuran sebuah data citra. Dalam pengolahan citra, *subsampling* juga bertujuan untuk meningkatkan invariansi posisi dari fitur. .
4. *Fully connected layer*, *layer* yang biasanya digunakan dalam penerapan *MLP* dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linier.

Citra Digital

Citra digital merupakan citra yang telah disimpan dalam bentuk *file* sehingga dapat diolah dengan menggunakan komputer. Citra digital digunakan dalam berbagai bidang yang dapat membantu manusia dalam bekerja. Dalam penggunaan citra, tidak semua gambar digunakan, kadang-kadang hanya sebagian saja, membutuhkan beberapa perubahan seperti mengubah ukuran citra, mengubah tingkat kecerahan, serta menggabungkan dua citra atau lebih, proses tersebut biasanya disebut pengolahan citra[8].

Pengolahan citra memiliki berbagai macam jenis klasifikasi. Salah satunya adalah segmentasi citra. Segmentasi citra merupakan suatu proses memecah suatu citra digital menjadi banyak segmen/bagian daerah yang tidak saling bertabrakan(*non-overlapping*) dalam konteks citra digital daerah hasil segmentasi tersebut merupakan kelompok piksel yang bertetangga atau berhubungan[9].

Terdapat berbagai macam metode segmentasi citra misalnya *thresholding, clustering, compression based, color image-based, edge detection, partial differential equation-based, watershed transformation, model-based segmentation, active contour-model*, dan lain-lain[10].

Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = picture element), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu.

Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks 2.1 berikut :

$$f(x,y) = \begin{matrix} f(0,0) & \dots & f(0,M-1) \\ \dots & \dots & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{matrix} \quad (1)$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat ditulis sebagai fungsi intensitas f(x,y), dimana x adalah baris dan y adalah kolom merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantisasi) diperoleh besar baris M dan kolom N hingga citra membentuk matriks M x N dan jumlah tingkat keabuan piksel G[11].

Cyclical Learning rate

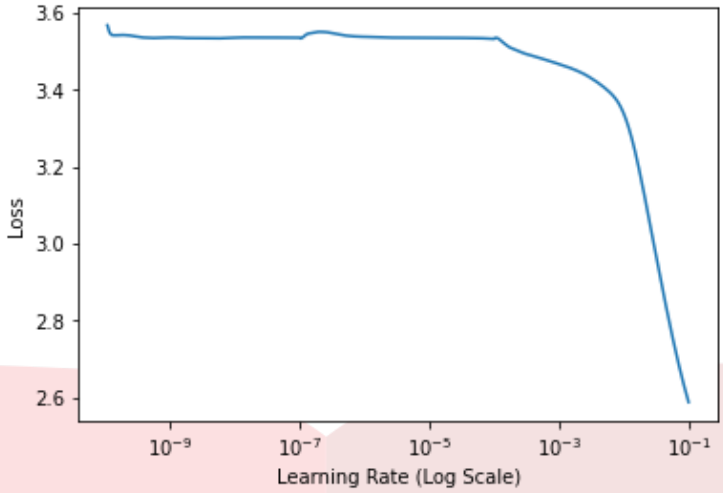
Learning rate adalah salah satu hiperparameter yang memiliki efek yang sangat besar pada saat melakukan proses *training* pada suatu model CNN. *Learning rate* adalah komponen paling penting untuk mendapatkan *training* yang efektif dan efisien dengan membuat proses *training* menjadi lebih cepat mendapatkan hasil yang maksimal dalam penggunaan nilai *learning rate* yang tepat. Penggunaan *learning rate* yang terlalu besar dapat membuat model menjadi lebih cepat dalam mempelajari suatu data, namun disisi lain penggunaan *learning rate* yang terlalu besar berdampak pada performa model yang tidak optimal. Sebaliknya penggunaan *learning rate* dengan nilai yang terlalu kecil dapat membuat model menjadi semakin lama dalam mempelajari suatu data. Untuk mengatasi masalah *learning rate* sehingga model menjadi lebih cepat dalam mempelajari suatu data dan tetap mendapatkan performa model yang optimal, Leslie N. Smith merumuskan suatu metode untuk membuat nilai *learning rate* selama proses *training* berubah-ubah dalam rentang nilai tertentu secara siklis, metode itu disebut dengan *Cyclical Learning rate* atau CLR.

CLR membuat suatu model bisa menurunkan dan menaikkan nilai *learning rate* secara siklis seiring dengan bertambahnya *step* dan *epoch*. Penggunaan *learning rate* yang dinamis dapat membantu model keluar dari *saddle point* pada saat proses *training*. Jika pada saat *training*, loss terjebak dalam saddle point, sangat sulit bagi model untuk keluar dari sad*dle poi*nt saat menggunakan *learning rate* secara statis, akibatnya loss dan akurasi *training* akan tersendat pada nilai tertentu yang masih rendah. Saddle point dapat diatasi dengan memperbesar nilai *learning rate*. Untuk bisa memperbesar *learning rate* maka penggunaan *learning rate* harus secara dinamis.

Learning Rate Finder

Dalam penggunaan *learning rate*, pendefinisian nilai *learning rate* harus dilakukan secara tepat, agar pada saat proses *training*, model bisa menemukan hasil terbaik atau nilai *loss* mencapai *global minima* atau nilai *loss* terkecil yang bisa dicapai. Untuk bisa menentukan nilai *learning rate* secara tepat, maka dibutuhkan sebuah modul yang bisa menemukan nilai *learning rate* yang optimal.

Learning Rate Finder(LRF) adalah suatu modul yang bisa menemukan *learning rate* optimum, dimana yang digunakan dasar untuk menentukan *learning rate* adalah nilai *loss* dari *dataset* yang digunakan terhadap model yang dibangun. Cara kerja dari *LRF* adalah mula mula dimasukkan rentang *learning rate* yang akan dicari, untuk rentang *learning rate* ini usahakan pada rentang mulai dari nilai yang sekecil-kecilnya sampai yang sebesar besarnya. Lalu masukkan model yang dibangun dan *dataset* yang akan digunakan untuk proses *training*. Selanjutnya LRF akan mengambil dataset secara *batch* per *batch* untuk dilakukan *training* dengan *learning rate* terkecil, pada data pada batch yang sudah dilakukan *training* tersebut akan dihasilkan nilai *loss*-nya, nilai *loss* akan dicatat. Training pada data pada *batch* selanjutnya akan dilakukan dengan nilai *learning rate* yang di-*increment*, hasil *loss* dari proses *training* ini akan dicatat juga. Proses tersebut terus berulang sampai nilai *learning rate* mencapai pada angka terbesar pada rentang nilai yang diberikan dan didapatkan catatan dari nilai *loss* dari training di semua nilai learning rate. Selanjutnya catatan nilai loss diplot untuk ditentukan nilai learning rate optimumnya.



Gambar 1 Contoh Grafik LRF

Pada gambar 1 ditunjukkan contoh grafik hasil plot *LRF*. Sumbu x adalah rentang nilai *learning rate*, sumbu y adalah nilai *loss*. Rentang nilai *learning rate* optimum terjadi pada saat nilai *loss* mengalami penurunan yang sangat drastis. Rentang nilai learning rate yang menyebabkan nilai loss menurun secara drastis disebut dengan *sweet spot*. Pada gambar 1, dapat disimpulkan bahwa '*sweet spot*' terjadi pada rentang nilai sekitar 10^{-2} sampai dengan 10^{-1}.

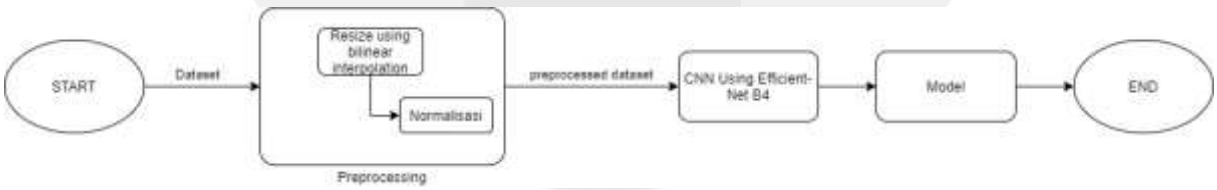
Efficient-Net

Meng-*upgrade* suatu *convolutional neural network* banyak dilakukan untuk mendapatkan akurasi yang lebih baik. Sebagai contoh *ResNet*[12] bisa di-*upgrade* dengan menambahkan lebih banyak *layer* sehingga menjadi *ResNet-18* sampai *ResNet-200*. Proses meng-*upgrade CNN* yang sering dilakukan adalah menambah kedalaman dengan menambahkan jumlah *layer*[12], atau menambah lebar dengan menambahkan jumlah *filter*[13], atau cara yang tidak sering dilakukan orang lain, namun saat ini mulai popular karena mulai digunakan orang adalah menambah resolusi citra masukan dengan memperbesar citra masukan[14]. Sebelum adanya Efficient-Net, untuk mendapatkan akurasi yang lebih baik, dilakukan upgrade pada salah satu komponen dari ketiga komponen. Penambahannya pun biasanya dilakukan secara semena-mena awalnya lalu dilakukan *tuning* secara manual.

Efficient-Net adalah suatu arsitektur *CNN* yang ditemukan dengan melakukan *scaling* secara teratur pada tiga komponen, yaitu kedalaman, lebar, dan resolusi[5]. Dalam penambahan kedalaman, lebar, ataupun resolusi, pasti akan membuat waktu yang dibutuhkan untuk memproses suatu data akan menjadi lebih lama. Dalam arsitektur ini, penambahan ketiga komponen tersebut dilakukan dengan sangat teratur sehingga didapatkan jumlah parameter yang lebih sedikit yang membuat waktu proses menjadi lebih cepat namun juga didapatkan juga akurasi yang baik dari model sebelumnya. Sehingga model ini menawarkan efisiensi, dan performa yang lebih baik dari model lain yang ada.

3. Sistem yang Dibangun

Pada sistem pengenalan bahasa isyarat yang dibangun pada Tugas Akhir ini memiliki dua alur yang harus dilewati, yaitu *training* dan *testing*. Pada saat *training*, secara garis besar sistem akan mempelajari *dataset*.

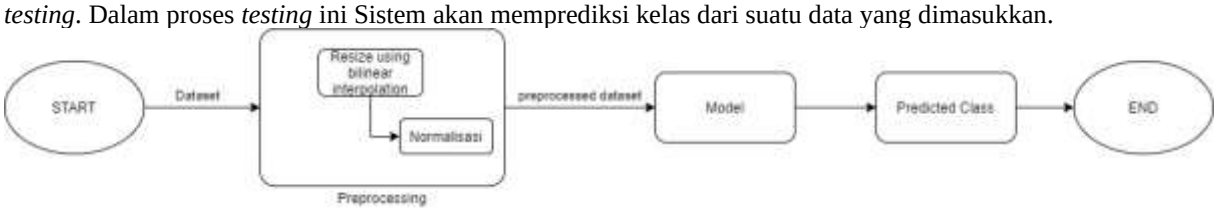


Gambar 2 Alur Proses *Training*

Seperti yang ditunjukkan pada gambar 2, mula mula *dataset* akan dilakukan *preprocessing*. Dalam *preprocessing* dilakukan *resize* menggunakan suatu metode *resize*, metode *resize* yang penulis gunakan adalah metode interpolasi *bilinear* yang dimiliki oleh modul *keras.preprocessing.image.ImageDataGenerator* dalam bahasa pemrograman Python. *Resize* ini dilakukan untuk mencocokkan *input shape* dari arsitektur Efficient-Net B4. Kemudian dalam *preprocessing* kita melakukan normalisasi citra. Normalisasi bertujuan untuk mengubah nilai-nilai dalam *array* citra menjadi nilai-nilai yang dapat diukur dengan skala umum[15]. Kemudian proses selanjutnya adalah mempelajari citra-citra hasil *preprocessing* menggunakan *classifier*, *classifier* yang penulis

gunakan adalah *CNN* dengan arsitektur *Efficient-Net B4*, untuk lebih detailnya mengenai arsitektur yang digunakan, penulis melampirkan *summary* dari model arsitektur *Efficient-Net B4*. hasil dari proses pembelajaran yang dilakukan oleh *EfficientNet B4* adalah berupa model.

Model adalah semacam kesimpulan yang didapat dari pemelajaran yang dilakukan oleh suatu metode *learning*[16]. Setelah kita mendapatkan model yang dihasilkan dari proses *learning*, kita akan melakukan proses *testing*. Dalam proses *testing* ini Sistem akan memprediksi kelas dari suatu data yang dimasukkan.



Gambar 3 Alur Proses *Testing*

Seperti yang ditunjukkan pada gambar 3, proses yang dilakukan pada proses *testing* mirip dengan proses yang dilakukan pada proses *training*. Mula-mula *dataset* akan masuk dalam *preprocessing*, dalam proses ini proses yang dilakukan sama dengan *preprocessing* pada tahap *training*, yaitu *scaling* dan normalisasi. Sebab dilakukan proses yang sama pada proses *preprocessing* adalah agar bentuk data yang diklasifikasikan pada saat *training* dan *testing* sama. Setelah melakukan tahap *preprocessing*, data hasil hasil *preprocessing* akan diprediksi kelasnya oleh model. Keluaran dari tahap ini adalah kelas hasil prediksi.

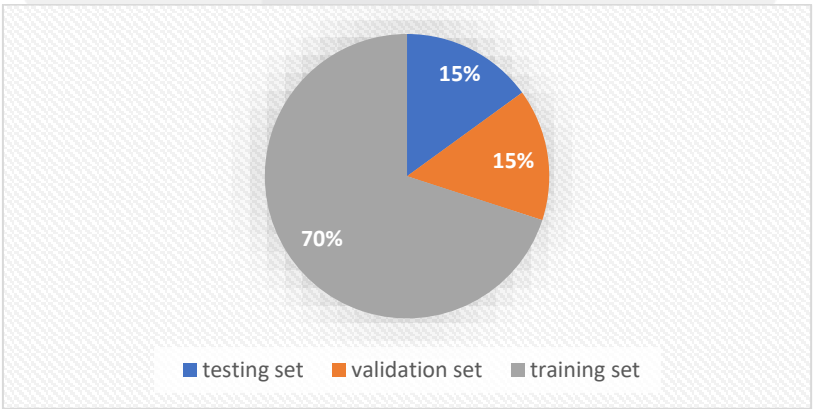
4. Eksperimen

Dataset

Dataset yang digunakan dalam penelitian ini adalah *ASL Alphabet Dataset* dari Kaggle [18]. *ASL Alphabet Dataset* adalah *dataset* gesture tangan dinamis, dimana *dataset* ini berisi citra-citra bahasa isyarat tangan yang memperagakan 26 huruf alfabet dan kelas '*del*'. '*nothing*', dan '*space*', sehingga *dataset* ini memiliki 29 kelas. Seperti yang ditunjukkan pada gambar 6, jumlah citra dari masing masing kelasnya memiliki 3000 citra. Sehingga secara total *dataset* ini memiliki 87000 citra. Dalam bahasa isyarat, huruf J dan Z adalah gestur dinamis. Namun dalam *dataset* ini penulis tetap menggunakan kelas J dan Z sebagai gestur statis, karena gestur J dan Z yang di-*capture* dalam *dataset* memiliki bentuk gestur yang berbeda dari kelas lain dan tidak memiliki kemiripan dengan kelas lain.

Gambar 4 Sample Kelas A (a), Kelas S (b), dan Kelas U (c)

Untuk bisa melakukan proses *training*, dibutuhkan data *training* dan data validasi, dan untuk melakukan proses *testing*, diperlukan data *testing*. Oleh karena itu penulis melakukan pembagian *dataset*.



Gambar 5 Presentase Pembagian Kelas

Komposisi pembagian yang kami lakukan seperti yang ditunjukkan pada gambar 5, data *training* mendapat 70% dari *dataset* keseluruhan, data validasi mendapat 15% dari *dataset* keseluruhan, dan data *testing* mendapat 15% dari *dataset* keseluruhan.

Preprocessing

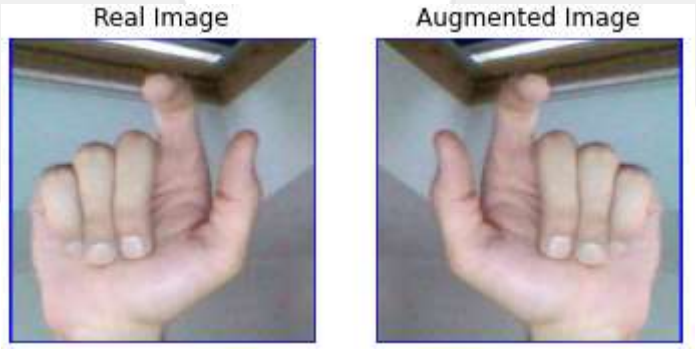
Dalam tahap ini penulis hanya melakukan proses *scaling* citra ke ukuran 112x112 px dan melakukan normalisasi citra. Alasan hanya melakukan proses tersebut adalah karena citra dalam *dataset* tersebut meskipun masih terdapat latar yang sekiranya tidak perlu namun *object of interest* sudah sangat ter-*highlight*. Keuntungan yang bisa didapat dari *preprocessing* yang tidak banyak adalah waktu pemrosesan keseluruhan akan lebih cepat dibanding jika melakukan banyak proses dalam tahap *preprocessing*.

Augmentasi *Dataset*

Augmentasi *dataset* bertujuan untuk menambahkan data dan variasi data agar sistem bisa mempelajari lebih banyak variasi data dalam tiap kelasnya. Penulis hanya melakukan augmentasi pada *training set* dan *validation set*, namun tidak dilakukan di *testing*. Augmentasi di *training set* bertujuan agar sistem mempelajari data yang lebih bervariasi sehingga model yang dihasilkan menjadi lebih '*robust'* terhadap suatu data. Augmentasi di *validation set* bertujuan untuk mengetahui bagaimana performa model dalam mempelajari data yang dilakukan augmentasi tadi. Tujuan tidak dilakukan augmentasi di *testing* set adalah penulis ingin melakukan *testing* terhadap data yang asli yang ada di dunia nyata. Jika dilakukan augmentasi pada *testing* set, *testing* yang dilakukan akan terkesan tidak dilakukan pada data yang ada pada dunia nyata. Dalam tahap ini penulis melakukan proses *horizontal flip*, namun tidak melakukan *rotating*, *zoom in*, dan *zoom out*.

Gambar 6 Kemiripan Kelas I (a) dan Kelas J (b)

Gambar 6 menunjukkan kelas I dan J, alasan tidak dilakukan proses *rotating* dalam augmentasi *dataset* adalah karena dikhawatirkan setelah dilakukan rotasi, sistem akan sulit membedakan kelas I dan J. Citra hasil augmentasi ditunjukkan pada gambar 7.



Gambar 7 Sampel Citra Teraugmentasi

Skenario Pengujian

Dalam skenario pengujian, penulis ingin mengetahui pengaruh perbedaan *input size* dan penggunaan *learning rate* yang dinamis menggunakan *cyclical learning rate* terhadap performa sistem dan efisiensi waktu terhadap *dataset ASL Alphabet*. Sehingga penulis merumuskan empat skenario pengujian seperti yang tercantum dalam tabel 1.

Tabel 1 Skenario Pengujian

Skenario	*Cyclical Learning Rate (CLR)*	*Input size*
1	No	112x112x3
2		64x64x3
3	Yes	112x112x3
4		64x64x3

Dengan dilakukannya empat skenario tersebut diharapkan bisa mengetahui pengaruh *CLR* dan perbedaan *input size* terhadap sistem. Pada proses *training* dilakukan sebanyak 30 *epoch* untuk masing-masing skenario, model diambil dengan membuat *callback* yang menyimpan *weight* model dari akurasi validasi tertinggi dari setiap *epoch*-nya. Selanjutnya *weight* terbaik dari proses *training* akan dimasukkan kedalam model. Selanjutnya dilakukan proses *testing* dan evaluasi sehingga didapat nilai akurasi, presisi, *recall* dan *f1-score* dari setiap skenario. Nilai akurasi dan waktu proses yang dijadikan pembanding untuk keempat skenario pengujian yang dilakukan.

Pengukuran Performa Sistem

Dalam penelitian ini, yang menjadi komponen pengukuran performa sistem dan evaluasi sistem adalah akurasi, presisi, *recall, f1-score*, dan waktu. Waktu yang dimaksud adalah waktu untuk melakukan *training* dan waktu untuk melakukan prediksi pada 1 data. Jika diketahui *TP* adalah *true positive*, yaitu jumlah kelas positif yang diklasifikasikan dengan benar. FP adalah *false positive*, yaitu jumlah kelas positif yang diklasifikasikan dengan salah. TN adalah *true negative*, yaitu jumlah kelas negatif yang diklasifikasikan dengan benar. FN adalah false negative, yaitu kelas negative yang diklasifikasikan dengan salah. Maka untuk mendapatkan nilai akurasi digunakan persamaan sebagai berikut:

$$Accuracy = \frac{TN + TP}{TN + TP + FP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

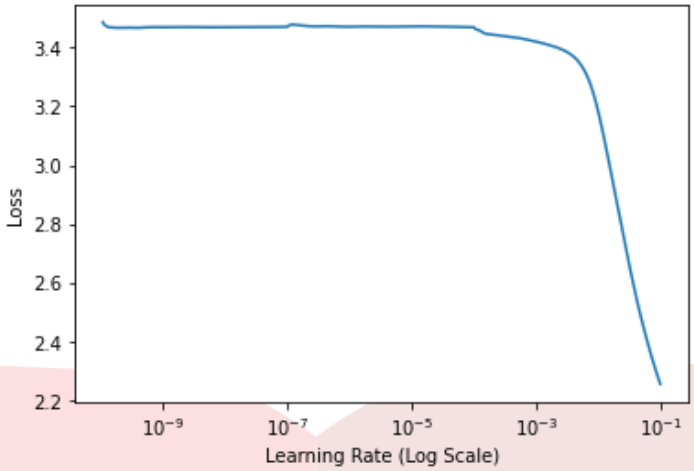
$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision} \quad (5)$$

Hasil Pengujian

A. Skenario 1 : *input size* 112x112x3 tanpa *CLR*

Untuk bisa melakukan perbandingan atas skenario-skenario pengujian yang dilakukan, perlu adanya tolak ukur dasar atau *benchmark* dasar dari pengujian-pengujian yang dilakukan. Dalam skenario 1, tujuan penulis melakukan skenario pengujian ini adalah agar skenario ini bertindak sebagai dasar atau *benchmark* dasar dari ketiga skenario pengujian lainnya.

Untuk mendapatkan nilai *learning rate* yang optimal, penulis harus menemukan *sweet spot*, yaitu pada saat rentang *learning rate* tertentu loss mengalami penurunan yang tajam, saat itulah dimana sistem dapat melakukan learning dengan cepat. Untuk mendapatkan *sweet spot*, penulis menggunakan *learning rate finder*.



Gambar 8 Hasil plot *learning rate finder* dari model dengan *input size* 112x112x3

Gambar 7 menunjukkan hasil plot *learning rate finder*. Dari hasil plot tersebut ditunjukkan bahwa *loss* mengalami penurunan yang tajam pada saat nilai *learning rate* sekitar 10^{-2}, asumsi 10^{-2}, sampai dengan nilai *learning rate* mencapai nilai 10^{-1}. Sehingga dapat disimpulkan bahwa *sweet spot* berada pada rentang nilai *learning rate* dari 0,01 hingga 0,1. Karena dalam skenario pengujian ini tidak menggunakan *Cyclical Learning rate* atau *CLR*, maka nilai *learning rate* diambil dari rentang nilai *sweet spot*, penulis mendefinisikan nilai *learning rate* dengan nilai 0,1. Tidak ada alasan khusus mengenai pengambilan *learning rate* dengan nilai terbesar dalam rentang nilai *learning rate* pada *sweet spot*.

Tabel 2 Hasil Evaluasi Setiap Kelas Skenario 1

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
A	0	0.99	0.96	0.97	450
B	1	0.89	1.00	0.94	450
C	2	1.00	1.00	1.00	450
D	3	1.00	1.00	1.00	450
E	4	1.00	0.96	0.98	450
F	5	1.00	1.00	1.00	450
G	6	0.98	0.98	0.98	450
H	7	0.95	0.99	0.97	450
I	8	0.88	0.97	0.92	450
J	9	0.93	0.99	0.93	450
K	10	0.88	1.00	0.94	450
L	11	0.99	0.99	0.99	450
M	12	0.99	0.93	0.96	450
N	13	0.96	0.96	0.96	450
O	14	1.00	1.00	1.00	450
P	15	0.99	1.00	0.99	450
Q	16	0.99	0.98	0.98	450
R	17	0.99	0.97	0.98	450
S	18	0.99	0.88	0.93	450
T	19	1.00	0.92	0.96	450
U	20	0.99	0.87	0.93	450
V	21	1.00	0.9	0.94	450
W	22	0.99	0.99	0.99	450
X	23	0.99	0.92	0.95	450
Y	24	1.00	0.92	0.96	450
Z	25	1.00	0.97	0.93	450
del	26	1.00	0.96	0.98	450
nothing	27	0.71	1.00	0.83	450
space	28	1.00	0.99	1.00	450

Tabel 3 Hasil Evaluasi Keseluruhan Skenario Pengujian 1

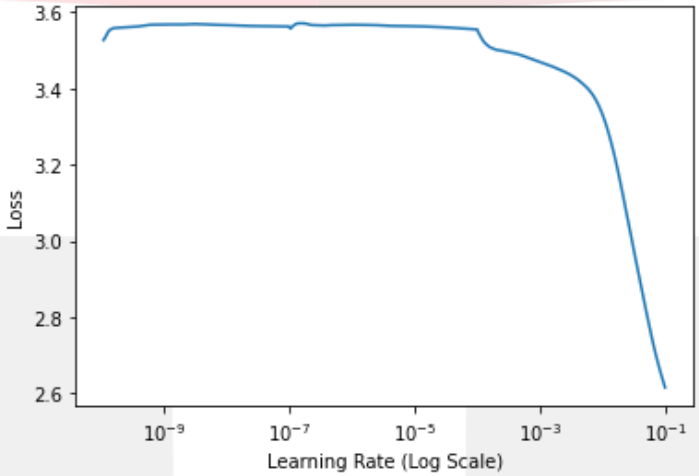
Akurasi	Presisi	*Recall*	*F1-Score*	Waktu Prediksi	Waktu *Training*
96,25%	96,82%	96,25%	96,34%	0,112 s	12524,5 s

Seperti yang ditunjukkan pada tabel 2 dan 3, hasil evaluasi skenario 1 didapatkan akurasi 96,25%, dimana itu berarti sistem berhasil melakukan prediksi dengan benar terhadap 96,25% bagian dari *testing* set. Presisi didapatkan dengan nilai 96,82%, recall 96,25%, *f1-score* 96,34%, dan waktu yang dibutuhkan untuk melakukan prediksi pada 1 data adalah 0,112 s. Seluruh hasil evaluasi dari skenario 1 akan dijadikan tolak ukur untuk skenario lainnya.

B. Skenario 2: *input size* 64x64x3 tanpa *CLR*

Dalam skenario 2 ini, berdasarkan pada skenario 1, penulis ingin mengetahui apa yang terjadi pada hasil kinerja sistem dan efisiensi waktu, jika menginputkan data yang sama dengan ukuran citra yang lebih kecil, yaitu dengan *input size* 64x64x3. Diinputkan ukuran citra yang lebih kecil diharapkan bisa mengurangi penggunaan *resource* namun tidak mengurangi kinerja sistem.

Untuk mendapatkan nilai *learning rate* yang optimal, penulis harus menemukan *sweet spot*, yaitu pada saat rentang *learning rate* tertentu *loss* mengalami penurunan yang tajam, saat itulah dimana sistem dapat melakukan learning dengan cepat. Untuk mendapatkan *sweet spot*, penulis menggunakan *learning rate finder*.



Gambar 9 Hasil plot *learning rate finder* dari model dengan *input size* 64x64x3

Gambar 8 menunjukkan hasil plot *learning rate finder*. Dari hasil plot tersebut ditunjukkan bahwa *loss* mengalami penurunan yang tajam pada saat nilai *learning rate* sekitar 10^{-2}, asumsi 10^{-2}, sampai dengan nilai *learning rate* mencapai nilai 10^{-1}. Sehingga dapat disimpulkan bahwa *sweet spot* berada pada rentang nilai *learning rate* dari 0,01 hingga 0,1. Karena dalam skenario pengujian ini tidak menggunakan *Cyclical Learning rate* atau *CLR*, maka nilai *learning rate* diambil dari rentang nilai *sweet spot*, penulis mendefinisikan nilai *learning rate* dengan nilai 0,1. Tidak ada alasan khusus mengenai pengambilan *learning rate* dengan nilai terbesar dalam rentang nilai *learning rate* pada *sweet spot*

Tabel 4 Hasil Evaluasi Setiap Kelas Skenario 2

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
A	0	0.99	0.18	0.30	450
B	1	0.99	0.41	0.58	450
C	2	0.95	0.77	0.85	450
D	3	0.99	0.58	0.73	450
E	4	0.71	0.32	0.44	450
F	5	0.99	0.56	0.71	450
G	6	0.13	0.87	0.22	450
H	7	0.22	0.66	0.33	450
I	8	0.61	0.55	0.58	450
J	9	0.2	0.3	0.24	450
K	10	0.95	0.19	0.32	450
L	11	0.91	0.67	0.77	450
M	12	0.21	0.4	0.27	450
N	13	0.18	0.59	0.28	450

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
O	14	0.82	0.52	0.63	450
P	15	0.45	0.52	0.48	450
Q	16	0.87	0.39	0.53	450
R	17	0.66	0.45	0.53	450
S	18	0.88	0.03	0.06	450
T	19	1.00	0.04	0.08	450
U	20	0.5	0.08	0.13	450
V	21	0.36	0.51	0.42	450
W	22	0.99	0.17	0.29	450
X	23	0.3	0.4	0.34	450
Y	24	0.94	0.2	0.33	450
Z	25	0.96	0.15	0.26	450
del	26	0.83	0.09	0.16	450
nothing	27	0.91	0.5	0.64	450
space	28	0.83	0.47	0.6	450

Tabel 5 Hasil Evaluasi Skenario Pengujian 2

Akurasi	Presisi	Recall	F1-Score	Waktu Prediksi	Waktu *Training*
39,78%	70,08%	39,78%	41,83%	0,053 s	4625 s

Seperti yang ditunjukkan pada tabel 4 dan 5, hasil evaluasi skenario 1 didapatkan akurasi 39,78%, dimana itu berarti sistem berhasil melakukan prediksi dengan benar terhadap 39,78% bagian dari *testing* set. Presisi didapatkan dengan nilai 70,08%, *recall* 39,78%, *f1-score* 41,83%, dan waktu yang dibutuhkan untuk melakukan prediksi pada 1 data adalah 0,064 s. Hasil evaluasi yang relatif kecil pada skenario pengujian ini kemungkinan besar terjadi karena model belum mencapai konvergen dengan 30 *epoch* yang diberikan. Hal tersebut dibuktikan pada saat penulis menambahkan 20 epoch lagi pada training, didapatkan akurasi validasi sebesar 99%.

C. Skenario 3: *input size* 112x112x3 dengan CLR

Dalam skenario 3 ini, penulis ingin mengetahui bagaimana efek penggunaan *learning rate* yang dinamis menggunakan *Cyclical Learning rate* atau CLR terhadap hasil kinerja sistem. Untuk memakai *learning rate* yang dinamis, penulis harus mengetahui rentang nilai *sweet spot* pada *learning rate*. Dikarenakan model, data, dan ukuran input yang sama dengan skenario 1, penulis menggunakan nilai rentang *learning rate* berdasarkan plot *learning rate finder* pada skenario 1 yang ditunjukkan pada gambar 9. Dari gambar tersebut disimpulkan bahwa rentang *sweet spot*-nya adalah 0,01 sampai dengan 0,1, sehingga penulis menetapkan penggunaan *learning rate* dinamis pada rentang nilai tersebut.

Tabel 6 Hasil Evaluasi Setiap Kelas Skenario 3

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
A	0	1,00	1,00	1,00	450
B	1	1,00	1,00	1,00	450
C	2	1,00	1,00	1,00	450
D	3	1,00	1,00	1,00	450
E	4	1,00	1,00	1,00	450
F	5	1,00	1,00	1,00	450
G	6	1,00	1,00	1,00	450
H	7	1,00	1,00	1,00	450
I	8	1,00	1,00	1,00	450
J	9	1,00	1,00	1,00	450
K	10	1,00	1,00	1,00	450
L	11	1,00	1,00	1,00	450
M	12	0,97	1,00	0,98	450
N	13	1,00	0,98	0,99	450
O	14	1,00	1,00	1,00	450
P	15	1,00	1,00	1,00	450
Q	16	1,00	1,00	1,00	450
R	17	1,00	1,00	1,00	450
S	18	0,99	1,00	1,00	450
T	19	1,00	1,00	1,00	450

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
U	20	1,00	1,00	1,00	450
V	21	1,00	1,00	1,00	450
W	22	1,00	1,00	1,00	450
X	23	1,00	0,99	1,00	450
Y	24	1,00	1,00	1,00	450
Z	25	1,00	1,00	1,00	450
del	26	1,00	1,00	1,00	450
nothing	27	1,00	1,00	1,00	450
space	28	1,00	1,00	1,00	450

Tabel 7 Hasil Evaluasi Skenario Pengujian 3

Akurasi	Presisi	Recall	F1-Score	Waktu Prediksi	Waktu *Training*
99,84%	99,84%	99,84%	99,84%	0,114 s	12351,1 s

Seperti yang ditunjukkan pada tabel 6 dan 7, hasil evaluasi skenario 1 didapatkan akurasi 99,84%, dimana itu berarti sistem berhasil melakukan prediksi dengan benar terhadap 99,84% bagian dari *testing* set. Presisi didapatkan dengan nilai 99,84%, *recall* 99,84%, *f1-score* 99,84%, dan waktu yang dibutuhkan untuk melakukan prediksi pada 1 data adalah 0,114 s.

D. Skenario 4: *input size* 64x64x3 dengan CLR

Dalam skenario 4 ini, penulis ingin mengetahui bagaimana efek penggunaan *learning rate* yang dinamis menggunakan *Cyclical Learning rate* atau CLR terhadap hasil kinerja sistem dan juga ingin mengetahui bagaimana efek penggunaan *input size* yang lebih kecil terhadap hasil kinerja sistem dan efisiensi waktunya. Untuk memakai *learning rate* yang dinamis, penulis harus mengetahui rentang nilai *sweet spot* pada *learning rate*. Dikarenakan model, data, dan ukuran *input* yang sama dengan skenario 2, penulis menggunakan nilai rentang *learning rate* berdasarkan plot *learning rate finder* pada skenario 2 yang ditunjukkan pada gambar 10. Dari gambar tersebut disimpulkan bahwa rentang *sweet spot*-nya adalah 0,01 sampai dengan 0,1, sehingga penulis menetapkan penggunaan *learning rate* dinamis pada rentang nilai tersebut.

Tabel 8 Hasil Evaluasi Setiap Kelas Skenario 4

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
A	0	0,99	1,00	1,00	450
B	1	0,99	1,00	1,00	450
C	2	1,00	1,00	1,00	450
D	3	1,00	1,00	1,00	450
E	4	0,99	1,00	1,00	450
F	5	1,00	1,00	0,99	450
G	6	1,00	1,00	1,00	450
H	7	1,00	1,00	1,00	450
I	8	1,00	1,00	1,00	450
J	9	1,00	1,00	1,00	450
K	10	0,99	1,00	0,99	450
L	11	1,00	1,00	1,00	450
M	12	1,00	1,00	1,00	450
N	13	1,00	1,00	1,00	450
O	14	1,00	1,00	1,00	450
P	15	1,00	1,00	1,00	450
Q	16	1,00	1,00	1,00	450
R	17	1,00	1,00	1,00	450
S	18	1,00	0,98	0,99	450
T	19	1,00	1,00	1,00	450
U	20	1,00	0,99	0,99	450
V	21	1,00	0,99	0,99	450
W	22	1,00	1,00	1,00	450
X	23	1,00	1,00	1,00	450
Y	24	1,00	1,00	1,00	450
Z	25	1,00	1,00	1,00	450
del	26	1,00	1,00	1,00	450

Kelas	Index Kelas	Presisi	Recall	F1-Score	Support
nothing	27	1,00	1,00	1,00	450
space	28	1,00	1,00	1,00	450

Tabel 9 Hasil Evaluasi Skenario Pengujian 4

Akurasi	Presisi	Recall	F1-Score	Waktu Prediksi	Waktu *Training*
99,81%	99,81%	99,81%	99,81%	0,056 s	6665,2 s

Seperti yang ditunjukkan pada tabel 8 dan 9, hasil evaluasi skenario 1 didapatkan akurasi 99,81%, dimana itu berarti sistem berhasil melakukan prediksi dengan benar terhadap 99,81% bagian dari *testing* set. Presisi didapatkan dengan nilai 99,81%, *recall* 99,81%, *f1-score* 99,81%, dan waktu yang dibutuhkan untuk melakukan prediksi pada 1 data adalah 0,068 s.

Analisis Hasil Pengujian

Penggunaan *learning rate* harus didasari pada hasil pencarian *learning rate* menggunakan *Learning rate finder* atau *LRF*. Pemilihan *learning rate* dalam *LRF* harus dipastikan bahwa rentang nilai atau nilai yang diambil adalah rentang nilai dimana *sweet spot* berada. *Sweet spot* adalah rentang nilai *learning rate* yang menyebabkan nilai loss mengalami penurunan yang sangat signifikan.

Tabel 10 Summary Skenario Pengujian

Skenario	*Cyclical Learning Rate (CLR)*	*Input size*	*Sweet Spot*	*Learning Rate*
1	No	112x112x3	$10^{-2} – 10^{-1}$	10^{-1}
2		64x64x3	$10^{-2} – 10^{-1}$	10^{-1}
3	Yes	112x112x3	$10^{-2} – 10^{-1}$	$10^{-2} – 10^{-1}$
4		64x64x3	$10^{-2} – 10^{-1}$	$10^{-2} – 10^{-1}$

Secara kebetulan model dengan *input size* 64x64x3 dan model dengan *input size* 112x112x3 memiliki rentang nilai *sweet spot* yang sama yaitu dengan rentang *learning rate* 0,01 sampai 0,1. Untuk penggunaan *learning rate* sendiri, bagi skenario pengujian yang tidak menggunakan *learning rate* secara dinamis, nilai *learning rate* didefinisikan dengan nilai 0,1, dan bagi skenario pengujian yang menggunakan *learning rate* secara dinamis, nilai *learning rate* didefinisikan dengan rentang nilai yang sama dengan nilai *sweet spot*-nya, yaitu 0,01 sampai 0,1.

Tabel 11 Summary Hasil Evaluasi dari Setiap Skenario Pengujian

Skenario	Akurasi	Presisi	Recall	F1-Score	Waktu Prediksi	Waktu *Training*
1	96,25%	96,82%	96,25%	96,34%	0,112 s	12524,5 s
2	39,78%	70,08%	39,78%	41,83%	0,053 s	4625 s
3	99,84%	99,84%	99,84%	99,84%	0,114 s	12351,1 s
4	99,81%	99,81%	99,81%	99,81%	0,056 s	6665,2 s

Dalam keempat skenario pengujian yang dilakukan, jika ditinjau dari sisi efektifitas dan efisiensi waktu, skenario 4 adalah skenario pengujian terbaik. Namun karena karakteristik dari dataset yang tidak terlalu banyak memiliki variasi khususnya dalam keberagaman latar, dibutuhkan satu skenario pengujian tambahan dengan menggunakan dataset sejenis yang memiliki variasi latar didalamnya. Pengujian tambahan ini dilakukan menggunakan *test model* pada skenario 3. Sampel dataset ditunjukkan pada gambar 9.

(a) (b) (c)

Gambar 10 Sampel *Dataset* dengan Beragam Latar. (a) kelas A, (b) kelas S, (c) kelas L

Dataset yang digunakan pada skenario ini adalah *ASL Alphabet Test* yang memiliki 29 kelas yang diperagakan dengan gestur yang sama dengan dataset yang digunakan pada 4 skenario sebelumnya, namun lebih memiliki variasi dari sisi latar dan variasi peraga. Pada skenario ini, penulis berharap bisa mengetahui performa pengujian dengan menggunakan dataset yang lebih bervariasi.

Dalam skenario pengujian tambahan ini didapatkan hasil evaluasi seperti pada tabel 10. Pengujian tambahan dengan model skenario 4 ini mendapatkan tingkat akurasi sebesar 42%, presisi sebesar 54%, *recall* sebesar 42%, dan *f1-score* sebesar 43%.

Tabel 12 Hasil Evaluasi Skenario Pengujian Tambahan

Akurasi	**Presisi**	***Recall***	***F1-Score***
42%	54%	42%	43%

Dari hasil yang didapatkan pada saat menggunakan *dataset* yang lebih bervariasi, performa sistem belum bisa mencapai hasil yang optimal. Hal ini disebabkan karena tidak dilakukan *preprocessing* yang lebih baik. Meskipun *object of interest* sudah sangat terhighlight namun tetap saja ada latar-latar yang diproses sehingga latar yang seharusnya tidak dipelajari oleh sistem menjadi ikut terpelajari.

A. Pengaruh Perbedaan *Input Size*

Hasil yang didapatkan dari keempat skenario pengujian yang sudah dilakukan, *input size* sangat berpengaruh pada hasil performa sistem yang dibangun. Ada beberapa aspek yang terpengaruh oleh perbedaan *input size* model, yaitu pada aspek akurasi dan aspek efisiensi waktu. Pada skenario pengujian 1 dan 2 didapatkan hasil yang berbeda pada segi akurasi dan efisiensi waktu.

Dari aspek akurasi, skenario yang menggunakan *input size* berukuran 64x64x3 tanpa CLR memperoleh hasil yang kurang maksimal jika dibandingkan dengan skenario dengan *input size* 112x112x3 tanpa CLR, hal ini disebabkan karena *input size* yang lebih kecil akan membuat '*field'* suatu model untuk mempelajari suatu data menjadi lebih sempit. Dampak dari '*field'* yang terlalu sempit adalah sistem akan memerlukan lebih banyak *epoch* untuk mendapatkan performa yang lebih baik, atau bahkan dampak yang lebih buruknya adalah performa model akan menjadi lebih kecil seiring dengan mengecilnya *input size*. Menurut percobaan yang dilakukan oleh Suresh Prasad Kannojia dan Gaurav Jaiswal pada tahun 2018[17] yang meneliti tentang efek variasi resolusi citra inputan terhadap suatu model *CNN*. Dalam penelitian itu mengemukakan bahwa performa model akan menjadi semakin kecil seiring dengan mengecilnya resolusi citra inputan.

Dari aspek efisiensi waktu, skenario yang menggunakan *input size* 64x64x3 memerlukan waktu *training* yang lebih cepat dibandingkan skenario yang menggunakan *input size* 112x112x3. Hal ini disebabkan karena semakin kecil *input size* dari model, maka akan membuat kompleksitasnya menjadi semakin kecil juga. Dengan model yang memiliki kompleksitas yang berbeda, dimana model satu memiliki kompleksitas yang lebih kecil dari model lainnya, jika diproses menggunakan perangkat yang sama, maka waktu proses model yang memiliki kompleksitas lebih kecil akan lebih cepat dari waktu proses model yang memiliki kompleksitas yang lebih besar.

B. Pengaruh *Cyclical Learning rate*

Gambar 11. *Model loss* untuk setiap skenario pengujian. Kolom pertama adalah grafik *training loss* dan *validation loss* dari setiap skenario pengujian. Kolom kedua adalah tren dari grafik *validation loss*. (a) dan (b) adalah hasil model loss dari skenario pengujian dengan *input size* 112x112x3 dengan tidak menggunakan *CLR*. (c) dan (d) adalah hasil *model loss* dari skenario pengujian dengan *input size* 64x64x3 tanpa menggunakan *CLR*. (e) dan (f) adalah hasil *model loss* dari skenario pengujian dengan *input size* 112x112x3 dengan menggunakan *CLR*. (g) dan (h) adalah hasil *model loss* dari skenario pengujian dengan *input size* 64x64x3 dengan menggunakan *CLR*.

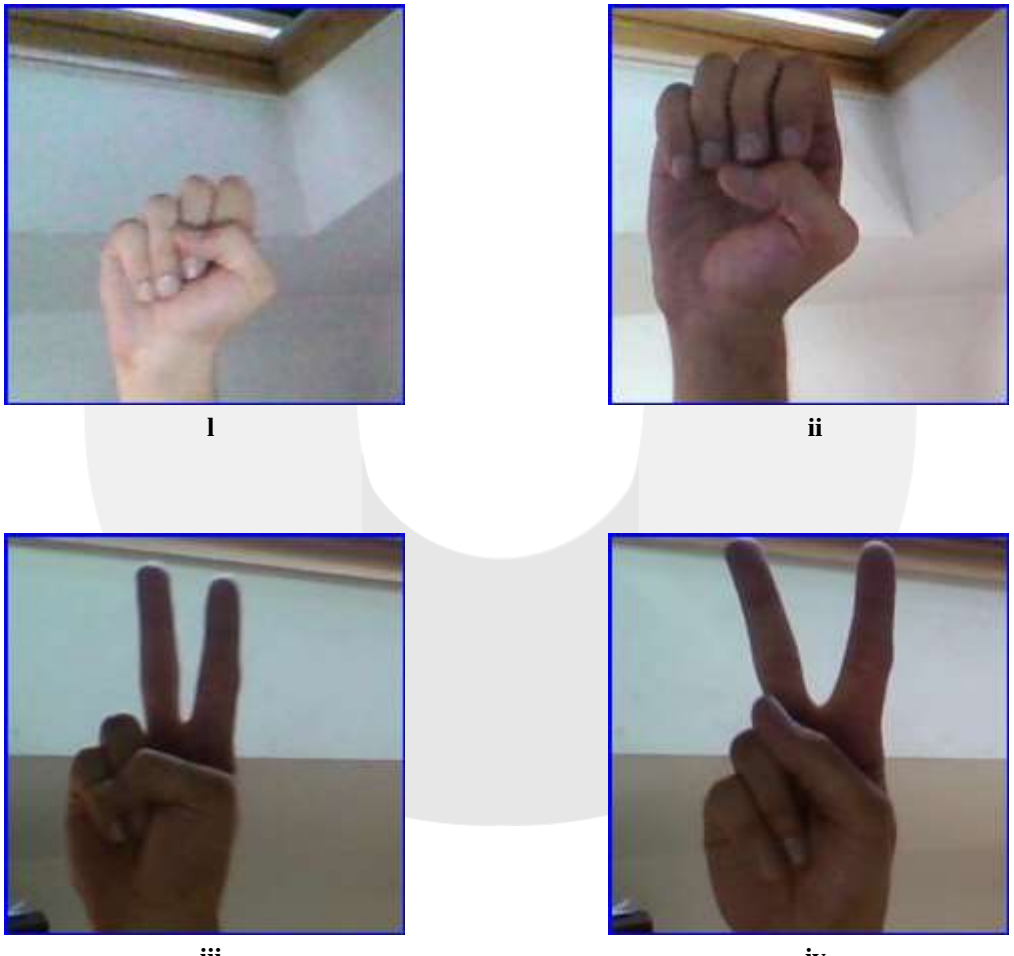
Skenario pengujian yang tidak menggunakan *CLR* berlaku sebagai *benchmark* atau tolak ukur dasar untuk pengujian yang menggunakan *CLR*. Penulis ingin mengetahui bagaimana pengaruh *CLR* terhadap performa model khususnya dalam hal tingkat akurasi dan bagaimana pengaruhnya dalam pencarian performa terbaik. Pada skenario pengujian dengan *input size* 112x112x3 mengalami peningkatan performa pada saat menggunakan *CLR*. Hal yang sama terjadi pada skenario pengujian dengan *input size* 64x64x3. Hal itu disebabkan oleh pemberlakuan *learning rate* secara dinamis yang membuat proses *training* menjadi lebih cepat untuk menemukan hasil terbaik. Pada hasil *model loss* yang ditunjukkan pada gambar 10 a, c, e, dan g, *validation loss* terlihat tidak konsisten naik atau tidak konsisten turun, sehingga penulis sulit untuk mengetahui perbandingan dari setiap *model loss*-nya. Oleh karena itu

penulis membuat gambar 10 b, d, f, dan h yang menampilkan grafik *validation loss* dan tren yang dikalkulasi secara eksponensial melalui Microsoft Excel.

Tren dari *validation loss* dari setiap skenario pengujian yang telah dilakukan yang ditunjukkan pada gambar 10, penggunaan *CLR* terbukti membuat tren dari *validation loss* pada saat training menjadi lebih cepat mengalami penurunan terhadap jumlah *epoch*-nya. Dalam penelitian ini, meskipun tren dari *validation loss* dari skenario yang menggunakan *CLR* lebih mendekati *training loss*, penulis tidak bisa mengatakan bahwa model yang menggunakan *CLR* yang penulis bangun tidak mengalami *overfit* karena itu adalah tren dari grafik *validation loss* bukan grafik *validation loss*-nya. Namun penulis juga tidak bisa mengatakan bahwa model yang penulis bangun mengalami *overfit* karena beberapa alasan, pertama, *validation loss* dari model yang menggunakan *CLR* tidak stabil atau naik turun dengan masif, kedua, *validation loss* dan *training loss* dari model yang diambil selisihnya sangat kecil dan hasil *testing* mendapatkan hasil dengan tingkat akurasi *testing* yang hampir sama dengan tingkat akurasi pada *validation*. Kesimpulan yang dapat diambil dari penggunaan *CLR* adalah dengan menggunakan *CLR* pencarian model terbaik menjadi lebih cepat, model terbaik yang didapatkan akan lebih baik jika dibandingkan dengan saat tidak menggunakan *CLR*, dan penggunaan *CLR* bisa membuat tren validation loss pada saat training menjadi lebih mendekati training loss.

C. Analisis Misklasifikasi

Meninjau dari tabel 11 terkait *summary* hasil semua skenario pengujian, bisa disimpulkan bahwa jika meninjau akurasi dan efisiensi waktu, skenario 4 adalah skenario dengan hasil terbaik. Namun dalam skenario terbaik yang ditemukan masih terjadi misklasifikasi atau kesalahan klasifikasi pada saat pengujian. Dalam *confusion matrix* yang dicantumkan dalam lampiran, misklasifikasi terjadi pada kelas S, dimana model yang harusnya memprediksi kelas S namun model memprediksi kelas E. Misklasifikasi selanjutnya terjadi pada kelas V, dimana model yang harusnya memprediksi kelas V namun model memprediksi kelas K.



l **ii**

iii **iv**

Gambar 12 Kemiripan Gestur. (i) Kelas S, (ii) Kelas E, (iii) Kelas V, (iv) Kelas K

Gambar 11 menunjukkan kemiripan gestur yang mengalami misklasifikasi. Misklasifikasi yang terjadi adalah kelas S yang salah memprediksi kelas E dan kelas V yang salah memprediksi kelas K. Jika ditinjau dari gestur

yang diperagakan dalam dataset, gestur huruf S memang terlihat sangat mirip dengan gestur huruf E, begitu juga gestur V yang terlihat sangat mirip dengan gestur K. Kemiripan dari gestur-gestur inilah yang membuat model melakukan misklasifikasi.

5. Kesimpulan

Dalam penelitian ini, gestur tangan diklasifikasikan dari *dataset ASL Alphabet* menggunakan *CNN* dengan arsitektur Efficient-Net B4. Penelitian ini mencoba menerapkan variasi *input size* dan penggunaan *CLR* guna mengetahui efeknya pada tingkat akurasi dan efisiensi waktu. Berdasarkan pada penelitian yang telah dilakukan, variasi *input size* berdampak besar pada efisiensi waktu, dimana *input size* yang semakin kecil akan menyebabkan waktu proses menjadi lebih cepat, hal ini disebabkan karena *input size* yang lebih kecil akan menurunkan kompleksitas, namun disisi lain model akan lebih lama dalam menemukan hasil terbaiknya karena *input size* yang kecil akan membuat "*field*" untuk mempelajari suatu data akan lebih kecil sehingga memerlukan lebih banyak *epoch* untuk menemukan hasil terbaiknya. Penggunaan *learning rate* secara dinamis menggunakan *CLR* berefek baik pada penelitian ini, dimana saat model menggunakan *CLR*, model menjadi lebih cepat dalam menemukan hasil terbaiknya. Hasil terbaik yang didapatkan saat menggunakan *CLR* pun lebih besar dibandingkan hasil terbaik model saat tidak menggunakan *CLR*.

Dalam pemilihan skenario pengujian terbaik, penulis memilih skenario pengujian yang hasilnya memenuhi dua aspek, yaitu efisiensi waktu dan tingkat akurasi. Dalam penelitian ini hasil terbaik diperoleh pada saat menggunakan *input size* 64x64x3 dengan menggunakan *learning rate* secara dinamis menggunakan *CLR* dengan akurasi yang didapat bernilai 99,81%, waktu *training* 6665,2 s, dan waktu untuk memprediksi 1 data adalah 0.056s. Jika dibandingkan, akurasi skenario *input size* 112x112x3 dengan *CLR* lebih besar dari skenario *input size* 64x64x3 (tabel 4 dan tabel 5), namun selisih akurasi yang hanya 0.03% tidak membuat akurasi menjadi aspek utama dalam pemilihan skenario terbaik dari kedua skenario tersebut, waktu *training* dan waktu memprediksi 1 data yang lebih menjadi penentu dari pemilihan skenario terbaik dari 2 skenario tersebut.

Model yang dibangun memang masih mengalami misklasifikasi, meskipun secara keseluruhan performa model khususnya dalam tingkat akurasi sudah sangat baik, bahkan sudah lebih baik dibandingkan penelitian yang dilakukan oleh Akm Ashiquzzaman dan kawan kawan [3]. Namun pada saat dilakukan pengujian dengan dataset dengan latar yang lebih beragam, hasil evaluasinya menjadi tidak baik, jika dilakukan lebih banyak *preprocessing* , sangat memungkinkan model akan menjadi lebih *robust* untuk di-*deploy*. *Preprocessing* yang sangat penting agar model menjadi lebih 'robust' adalah segmentasi, karena dengan dilakukan segmentasi model hanya akan mempelajari *object of interest* saja tanpa memproses latar-latar yang tidak perlu, sehingga karna latar tidak diproses, saat gestur diperagakan dilatar yang berbeda-beda, model tetap berfokus pada gestur tangannya saja.

References

[1] A. N. Fierro Radilla and K. R. Perez Daniel, "Siamese Convolutional Neural Network for ASL Alphabet Recognition," *Comput. y Sist.*, vol. 24, no. 3, pp. 1211–1218, 2020, doi: 10.13053/cys-24-3-3481.

[2] S. Upadhyay, R. K. Sharma, P. Singh Rana, and R. Sharma, "Sign Language Recognition with Visual Attention," 2020.

[3] A. Ashiquzzaman, H. Lee, K. Kim, H. Y. Kim, J. Park, and J. Kim, "Compact spatial pyramid pooling deep convolutional neural network based hand gestures decoder," *Appl. Sci.*, vol. 10, no. 21, pp. 1–22, 2020, doi: 10.3390/app10217898.

[4] P. V. S. M. S. Kartik, K. B. V. N. S. Sumanth, V. N. V. S. Ram, and P. Prakash, "Sign Language to Text Conversion Using Deep Learning," in *Inventive Communication and Computational Technologies*, 2021, pp. 219–227.

[5] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[6] L. N. Smith, "Cyclical learning rates for training neural networks," *Proc. - 2017 IEEE Winter Conf. Appl. Comput. Vision, WACV 2017*, no. April, pp. 464–472, 2017, doi: 10.1109/WACV.2017.58.

[7] M. Deshpande, "Machine Learning for Human Beings," 2018.

[8] A. Budiarti, "Bab 2 landasan teori," *Apl. dan Anal. Lit. Fasilkom UI*, pp. 4–25, 2006.

[9] H. C. Andrews, "Digital Image Processing," *Computer (Long. Beach. Calif).*, vol. 7, no. 5, pp. 17–19, 1974, doi: 10.1109/MC.1974.6323522.

[10] H. Cheng, L. Yang, and Z. Liu, "Survey on 3D Hand Gesture Recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1659–1673, 2016, doi: 10.1109/TCSVT.2015.2469551.

[11] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2015-Octob, pp. 1–7, 2015, doi: 10.1109/CVPRW.2015.7301342.

[12] S. Xie, R. Girshick, and P. Doll, "Aggregated Residual Transformations for Deep Neural Networks http://arxiv.org/abs/1611.05431v2," *Cvpr*, 2017, [Online]. Available: https://github.com/facebookresearch/ResNeXt%0Ahttp://arxiv.org/abs/1611.05431v2.

[13] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," *Br. Mach. Vis. Conf. 2016, BMVC 2016*, vol. 2016-Septe, pp. 87.1-87.12, 2016, doi: 10.5244/C.30.87.

[14] Y. Huang *et al.*, "GPipe: Efficient training of giant neural networks using pipeline parallelism," *Adv. Neural Inf. Process. Syst.*, vol. 32, no. NeurIPS, 2019.

[15] Z. Xu, X. Yang, X. Li, and X. Sun, "The effectiveness of instance normalization: A strong baseline for single image dehazing," *arXiv*, pp. 1–13, 2018.

[16] J. Qiu *et al.*, "Modeling and predicting learning behavior in MOOCs," *WSDM 2016 - Proc. 9th ACM Int. Conf. Web Search Data Min.*, pp. 93–102, 2016, doi: 10.1145/2835776.2835842.

[17] S. P. Kannojia and G. Jaiswal, "Effects of Varying Resolution on Performance of CNN based Image Classification An Experimental Study," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 9, pp. 451–456, 2018, doi: 10.26438/ijcse/v6i9.451456.