Analisis Pembacaan Gerakan Bibir Menggunakan Gabungan Arsitektur Convolutional Neural Network dan Recurrent Neural Network

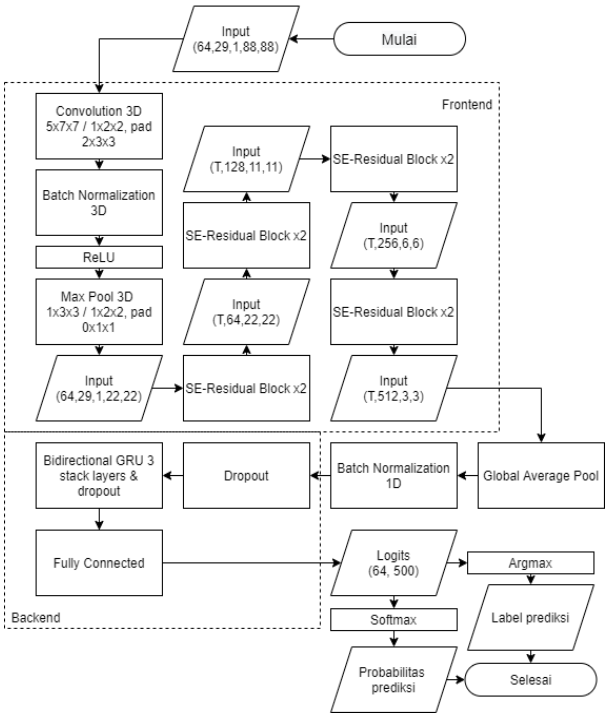
Akwila Feliciano Pradiptatmaka, Nanik Suciati, dan Dini Adni Navastara
Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: nanik@if.its.ac.id

***Abstrak*—Perkembangan bidang *deep learning* membawa pengaruh besar terhadap kemajuan teknologi. Pengucapan kata secara verbal dapat dibaca dengan pembelajaran mandiri melalui gerakan bibir. Dengan menggunakan arsitektur ekstraksi fitur dari suatu citra dan menggabungkannya dengan arsitektur klasifikasi secara *sequence*, permasalahan *visual speech recognition* ini dapat menjadi langkah awal untuk membantu para penyandang tuna rungu, yang dominan memahami komunikasi dari gerakan bibir lawan bicara ataupun dengan menggunakan bahasa isyarat. Selain itu, manfaat dari implementasi sistem ini dapat membantu lawan bicara dalam memahami pembicaraan pada kondisi audio yang terdistorsi. Arsitektur *deep learning* yang digunakan pada eksperimen ini mengacu pada implementasi dengan menggunakan arsitektur *convolution residual network* dan *recurrent neural network*, yang dapat mengklasifikasikan data berelasi secara *sequence* atau memiliki relasi secara strukturnya dari waktu ke waktu, untuk kasus ini berupa citra dari *frame* ke *frame*. Data yang digunakan untuk pembelajaran model berasal dari *dataset Lip Reading in the Wild* (LRW) yang merupakan potongan video dari pembicara stasiun berita BBC, dengan jumlah data sebanyak 500 kata dan mencapai 1000 data latih video yang berbeda-beda dalam bahasa Inggris.**

***Kata Kunci*—Convolutional Neural Network, Pembacaan Gerakan Bibir, Recurrent Neural Network, Visual-Speech Recognition.**

I. PENDAHULUAN

PERKEMBANGAN teknologi berbasis *deep learning* saat ini sudah canggih untuk melakukan deteksi objek hingga lakukan klasifikasi objek. Teknologi perekaman video dan pengolahan citra dengan memanfaatkan *deep learning* memberikan kemudahan untuk melakukan analisis pembacaan gerakan bibir yang akan diterjemahkan menjadi teks ataupun keluaran lainnya. Belakangan ini, beberapa penelitian mencari metode-metode efektif untuk melakukan *lip reading* menggunakan arsitektur *neural network*. Diantaranya, metode dengan menggabungkan arsitektur *convolutional neural network* (CNN) dengan *recurrent neural network* (RNN) dan beberapa metode *pre-processing* lainnya. Seperti pada penelitian [1], menggunakan *spatiotemporal convolutional neural network* (ST-CNN) dan *long-short term memory* (LSTM) mendapatkan skor akurasi sebesar 83%. Kemudian, pada [2] mengusulkan penggunaan RNN digantikan dengan *temporal convolutional networks* (TCN) [3], mendapatkan skor akurasi sebesar 85,3% disusul pada [4] yang menggunakan *AdamW* sebagai *optimizer* mendapat skor akurasi 87,9%. Walaupun skor akurasi tersebut tinggi, kompleksitas arsitekturnya sangat besar dan memakan waktu latih lebih lama. Salah satu penelitian [5] mengusulkan untuk melakukan optimasi pada pengolahan datanya dan menggunakan arsitektur *residual network* dan *gated recurrent unit*, mendapatkan hasil skor akurasi sebesar 88,4%.



Gambar 1. Diagram alir LipReadNet.

Pada eksplorasi ini akan melakukan implementasi sistem dan eksperimen analisis pembacaan gerakan bibir menggunakan arsitektur *residual network* yang akan dilatih lebih lanjut dengan *gated recurrent unit*. *Dataset* yang digunakan diambil dari *The Oxford-BBC Lip Reading in the Wild (LRW)*, milik program berita nasional Inggris, BBC. *Dataset* ini memiliki kata berbahasa Inggris dan terdiri dari video dengan panjang 25 *frame* per detik. Sebanyak 500 kata yang dijadikan label atau kelas untuk melatih model dan 1.000 pengucapan berbeda-beda. Informasi mengenai letak kata pada video ditulis pada *file metadata* yang juga ada pada *dataset*.

II. LIPREADNET MODULE

Arsitektur yang digunakan merupakan modifikasi beberapa *module* dari arsitektur yang diusulkan oleh rujukan [5]. Beberapa metode seperti regularisasi *loss function*, *data augmentation*, dan konversi video menjadi *dataset* yang

Tabel 1.
Hasil uji coba data uji

No.	Akurasi	F1	*Recall*	*Precision*
#1	0,805	0,804	0,805	0,810
#2	0,008	6e-4	7,92e-3	3,3e-4
#3	0,796	0,795	0,796	0,804
#4	0,008	5,562e-4	7,92e-3	3,07e-4
#5	0,653	0,651	0,653	0,666
#6	0,819	0,818	0,819	0,821
#7	0,679	0,678	0,678	0,682
#8	0,676	0,675	0,676	0,678
#9	0,674	0,673	0,674	0,677
#10	0,669	0,668	0,669	0,672
#11	0,828	0,827	0,828	0,831
#12	0,698	0,697	0,698	0,701
#13	0,695	0,695	0,695	0,698
#14	0,661	0,659	0,661	0,664
Model *Pre-trained*	0,002	1,122e-4	0,002	8,286e-05

Tabel 2.
Variasi metode inisialisasi parameter

Variasi	Modul *Frontend*	Modul *Backend*
A	Kaiming Normal *fan_out* ReLU (*pretrained model*)	GRU: $Uniform(-k, k)$ dengan $k = \sqrt{1/hidden\_size}$ Linear: $Uniform(-k, k)$ dengan $k = \sqrt{1/features_{in}}$
B	Xavier Normal	GRU: $Normal(\mu = 0, \sigma = 1)$ Linear: $Uniform(-1,1)$
C	Kaiming Normal *fan_out* ReLU (*pretrained model*)	GRU: $Normal(\mu = 0, \sigma = 1)$ Linear: $Uniform(-1,1)$

dapat dilatih juga dimodifikasi sesuai *use case* dan lingkungan implementasi sistem yang dapat dilihat pada Gambar 1.

A. Frontend Module

Frontend Module digunakan untuk melakukan ekstraksi fitur citra yang menggunakan modifikasi dari ResNet-18 dengan menambahkan blok *Squeeze-Excitation* [6] sebelum tahapan perhitungan *identity mapping* dan menggantikan blok pertama menggunakan *3D Convolution* berukuran 5x7x7 (*frame sequence, height, width*). Blok *Squeeze-Excitation* memiliki tujuan untuk memaksimalkan kualitas dari fitur citra dengan merepresentasikan antara *channel* atau kanal tiap citranya. Tahapan *squeeze* menghasilkan distribusi global *channel* tiap fitur citra sehingga menyediakan informasi mengenai relasi tiap *channel* untuk bisa digunakan oleh layer selanjutnya. Tahapan *excitation* menghasilkan kumpulan informasi yang dihasilkan pada tahapan *squeeze* dengan menyediakan informasi parameter bobot tiap *channel* suatu fitur citra. Setelah tahapan *squeeze* dan *excitation*, dilanjutkan dengan *scaling* terhadap hasil kedua tahapan ini dengan fitur citra hasil konvolusi dari blok *residual* secara *channel-wise multiplication*.

B. Backend Module

Backend Module terdiri dari *Dropout layer*, *Bi-directional Gated Recurrent Unit layer*, dan *Fully Connected layer*. Sebelum memasuki *backend module*, dilakukan proses *average pooling* dari fitur citra keluaran *frontend module* bertujuan untuk mereduksi ukuran dimensi, dilanjut dengan tahapan normalisasi parameter. Hasil dari *backend module* merupakan *tensor logit* atau hasil probabilitas yang belum ternormalisasi, sehingga untuk mendapatkan probabilitas hasil prediksi digunakan fungsi aktivasi *softmax* dan menggunakan fungsi *argmax* untuk mendapatkan indeks dari *tensor logit* dengan nilai terbesar.

III. METODE

Implementasi yang digunakan pada eksperimen ini mengikuti rujukan [5] dengan beberapa penyesuaian parameter dan cara pengolahan datanya. Metode-metode yang digunakan pada eksperimen ini terdiri dari pengolahan data set, *data augmentation*, penggunaan metode inisialisasi parameter, penggunaan *learning rate scheduler*, dan *hyper-parameter tuning* model.

A. Pengolahan Data

Bentuk data yang didapatkan dari LRW *dataset* adalah video dengan ukuran 29 *frames* yang masing-masing memiliki metadata durasi suatu label target diucapkan oleh subjek. Metadata durasi diolah menjadi nilai 0 dan 1, merepresentasikan *frame* yang berisikan pengucapan label target oleh subjek dan disebut sebagai *word boundaries*. Data terdiri dari tiga jenis, yaitu data latih sebanyak kurang lebih 1000 video, data validasi sebanyak 50 video, dan data uji sebanyak 50 video untuk tiap label target pada masing-masing jenis data. Setiap data video dijadikan gambar berurutan (*sequence*) yang masing-masing gambar dilakukan *face alignment* untuk menyamaratakan posisi bibir subjek. Setelah proses *face alignment*, gambar dilakukan *cropping* secara manual berdasarkan indeks posisi *array* sebesar 96x96 piksel.

Beberapa cara dilakukan untuk menentukan *cropping* bibir yang efektif untuk mendapatkan area sekitar mulut yang penting dalam pendeteksian gerakan. Cara pertama menggunakan *face landmark* dari *library DLib* untuk mendapatkan posisi area mulut dan melakukan *cropping* terhadap area tersebut dan dilakukan *rescale* menjadi 96x96

Tabel 3.
Hasil uji coba data validasi

#	*Hyper-parameter*				Akurasi	*Loss*
#2	200 Data Latih	*B32*	*End-to-End* (Metode Inisialisasi B)	*Epoch* (12,80)/92, *Mixup Alpha* 0,2, *LSR Factor* 0,1, *Dropout* 0,1, *Cosine Annealing Scheduler*	7,438e-3 (*overfitting*)	0,187
#4		*B32*	*End-to-End* (Metode Inisialisasi B)	*Epoch* 20/80, *Mixup Alpha* 0,2, *LSR Factor* 0,1, *Dropout* 0,1, *Cosine Annealing Scheduler*	5,84e-3 (7,84e-3)	5,98
#6	320 Data Latih	*B64*	*Transfer Learning* (Metode Inisialisasi A)	*Epoch* (6,14)/20, *MixUp Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, SGDR *Scheduler*	0,808	2,441
#7				*Epoch* 20, *MixUp Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, *Cosine Annealing Scheduler*	0,688 (*overfitting*)	2,753
#8				*Epoch* 20, *MixUp Alpha* 0,1, *LSR Factor* 0,4, *Dropout* 0,1, *Cosine Annealing Scheduler*	0,687	3,909
#9				*Epoch* 20, *LSR Factor* 0,2, *Dropout* 0,1, *SGDR Scheduler*	0,643 (0,682)	2,891 (2,748)
#10				*Epoch* 40, *MixUp Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, *SGDR Scheduler*	0,641 (0,679)	2,896 (2,760)
#14				*Epoch* 20, *Mixup Alpha* 0,4, *LSR Factor* 0,1, *Dropout* 0,1, *Cosine Annealing Scheduler*	0,674 (*underfitting*)	2,175
#1	500 Data Latih	*B32*	*Warm Restart* (Metode Inisialisasi A)	*Epoch* (9,41)/50, *MixUp Alpha* 0,4, *LSR Factor 0,1*, *Dropout* 0,2, *Cosine Annealing Scheduler*	0,808 (*underfitting*)	0,057
#13			*Transfer Learning* (Metode Inisialisasi C)	*Epoch* 20, *Mixup Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, SGDR *Scheduler*	0,666 (0,708)	2,828 (2,689)
#3		*B64*	*Transfer Learning* (Metode Inisialisasi A)	*Epoch* 48/80, *Mixup Alpha* 0,2, *LSR Factor* 0,1, *Dropout* 0,2, *Cosine Annealing Scheduler*	0,781 (*underfitting*)	0,106
#11			*Transfer Learning* (Metode Inisialisasi C)	*Epoch* (6,13)/20, *Mixup Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, SGDR *Scheduler*	0,780 (0,835)	2,373
#5	1000 Data Latih	*B64*	*Transfer Learning* (Metode Inisialisasi C)	*Epoch* 27/80, *Mixup Alpha* 0,4, *LSR Factor* 0,1, *Dropout* 0,2, *Cosine Annealing Scheduler*	0,659 (*underfitting*)	2,225
#12			*Transfer Learning* (Metode Inisialisasi C)	*Epoch* 19/40, *Mixup Alpha* 0,1, *LSR Factor* 0,2, *Dropout* 0,1, SGDR *Scheduler*	0,710	2,660

***Catatan**:
Transfer Learning: *Pretrained* Modul *Frontend* + *Freezed* Parameter
End-to-End Learning: Latih model dari awal
Warm Restart Learning (*Fine Tuning*): *Pretrained Model* + Latih Ulang

untuk semua gambar. Cara kedua, menggunakan *face landmark Dlib* untuk mendapatkan area rahang subjek dan dilakukan *cropping* pada area tersebut. Cara ketiga, menggunakan indeks posisi dari *array* yang digunakan untuk mendapatkan area sekitar mulut sebesar 96x96. Berdasarkan uji coba yang dilakukan, hasil cara pertama memiliki kekurangan dari kejelasan dan kejernihan gambar yang dihasilkan karena setiap subjek memiliki ukuran pengambilan gambar yang berbeda-beda. Cara kedua memiliki kekurangan karena area yang didapatkan terlalu luas dan tidak spesifik terhadap area mulut sehingga terdapat beberapa area yang tidak penting untuk dipelajari oleh model. Cara ketiga digunakan untuk semua pemrosesan data pada ketiga jenis data dan gambar-gambar tersebut disimpan menjadi *Python pickle* untuk mengurangi ruang penyimpanan hasil pengolahan data video menjadi gambar berurut.

B. Augmentasi Data

Proses augmentasi data yang dilakukan sesuai jenis datanya. Untuk data latih, dilakukan *random cropping* atau melakukan pemotongan gambar secara acak dengan keluaran gambar berdimensi 88 x 88. *Cropping* atau pemotongan *frame* diambil dari bagian *top* (atas) suatu gambar dan *heigtht* (tinggi) dengan nilai 88 piksel. Selain bagian atas, pemotongan gambar dimulai dari nilai *left* (kiri) suatu gambar dan *width* (lebar) dengan nilai 88 piksel. Pemilihan nilai *top* (atas) dan *left* (kiri) ditentukan secara acak dengan nilai [0, h_0-88) untuk *top* dan [0, w_0-88) untuk *left*. Nilai acak tersebut merupakan hasil dari distribusi uniform, dengan h_0 adalah tinggi gambar aslinya dan w_0 adalah lebar gambar aslinya.

Setelah *random cropping*, dilakukan *random horizontal flip*, yaitu melakukan transformasi pembalikan *frame* berdasarkan y-*axis* (sumbu y), secara acak dengan probabilitas p. Nilai *p* pada implementasi eksperimen ini bernilai $p = 0.5$. Untuk data validasi dan data uji, transformasi yang dilakukan adalah *center cropping*. Transformasi *center cropping*, melakukan pemotongan *frame* yang pada titik tengah suatu *frame*, diambil dari nilai *top* dan *left*, dengan keluaran berdimensi $crop_{height}$ x $crop_{width}$. Nilai titik awal pemotongan didapatkan dari perhitungan $round((frame\_height - crop\_height)/2.0)$ untuk nilai dari atas gambar dan $round((frame\_width - crop\_width)/2.0)$ untuk nilai dari kiri gambar. Semua jenis data sebelum melakukan transformasi *random cropping*, *random horizontal flip*, ataupun *center cropping*, telah dilakukan normalisasi (1/255.), konversi warna menjadi hitam-putih, dan standardisasi distribusi normal pada kanal warna hitam-putih. Secara keseluruhan, data latih, data validasi, dan data uji memiliki keluaran gambar berdimensi 88 x 88 dengan kanal warna hitam-putih dan bernilai antara 0 dan 1.

C. MixUp Data Augmentation

Tahapan *MixUp* digunakan sebagai *data augmentation* untuk data latih sebelum dilakukan pembelajaran model. *MixUp* [7] bertujuan untuk mengurangi *overfitting* dengan menggabungkan *input* dari suatu label target dengan label target lainnya secara interpolasi antara *input* dan λ yang merupakan distribusi beta (1). Label target yang dihasilkan dari *MixUp* terdiri dari dua bagian yang mewakili tiap label yang diproses, $target_A$ dan $target_B$. Informasi *word boundaries* tiap *frame* juga dilakukan *augmentation* sehingga memiliki korelasi yang sesuai untuk model *recurrent*. Selain *word boundaries*, perhitungan nilai *loss*

Gambar 2. Hasil prediksi percobaan ke-6.

Gambar 3. Hasil prediksi percobaan ke-7.

Gambar 4. Hasil prediksi percobaan ke-8.

dan jumlah prediksi benar dari hasil *argmax* dilakukan interpolasi sesuai nilai λ yang digunakan untuk masing-masing label target.

$$\lambda \sim f(x|\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{(\alpha-1)}(1-x)^{\beta-1} \quad (1)$$

$$\hat{x} = \lambda x_i + (1-\lambda)x_j \quad (2)$$

$$\hat{w} = \lambda w_i + (1-\lambda)w_j \quad (3)$$

$$target_A = y_i;\ target_B = y_j \quad (4)$$

$$loss = \frac{1}{n}(\lambda l(f(\hat{x}), target_a) + (1-\lambda) l((f(\hat{x}), target_b)) \quad (5)$$

D. Label Smoothing Regularization

Umumnya pada klasifikasi, untuk mengukur kesalahan prediksi suatu model digunakan fungsi *cross entropy* (8) yang merupakan gabungan fungsi aktivasi *softmax* (6) dengan *negative log likelihood*. Peran dari fungsi aktivasi *softmax* adalah normalisasi probabilitas dari hasil keluaran model klasifikasi linear, yang disebut sebagai *logit* (z_k). Persamaan (7) merupakan distribusi *ground truth* $q(k)$ atau label target aslinya, yang didapatkan dari penjumlahan terhadap matriks *one hot encoding*. Tujuan dari pembelajaran klasifikasi, yaitu memaksimalkan keakuratan suatu model dalam melakukan prediksi label dan meminimalkan nilai *negative log likelihood loss*. Untuk meminimalkan nilai *loss* tersebut, perlu terlebih dahulu memaksimalkan log *likelihood* dengan menghasilkan hasil prediksi benar (z_y) lebih dominan dibandingkan hasil prediksi yang salah ($z_{k!=y}$).

$$p(k|z) = softmax(z) = \frac{\exp(z_k)}{\sum_{k=1}^{K} \exp(z_k)} \quad (6)$$

$$\sum q(k|z) = 1 \quad (7)$$

$$l(q,p) \sim H(q,p) = -\sum_{k=1}^{K} q(k) log\ p(k) \quad (8)$$

$$\frac{\partial l}{\partial z_k} = p(k) - q(k) \quad (9)$$

Berdasarkan analisis pada salah satu penelitian, permasalahan akan muncul ketika *logits k=y* (z_y) atau hasil prediksi model benar terhadap label aslinya lebih banyak secara kuantitas dibandingkan prediksi label kelas lainnya

Gambar 5. Hasil prediksi percobaan ke-9.

Gambar 6. Hasil prediksi percobaan ke-10.

Gambar 7. Hasil prediksi percobaan ke-11.

($z_{k!=y}$). Model akan cenderung percaya diri terhadap hasil prediksinya sehingga menjadikannya *overfit* terhadap data di luar distribusinya. Dari (9), yang merupakan gradien hasil *backpropagation* dengan nilai antara -1 dan 1, perbedaan *logits* (z_k) yang besar dapat mengurangi kemampuan model untuk memperbaharui parameter pembelajarannya. Untuk itu, mereka mengusulkan regularisasi terhadap fungsi *loss* dengan memodifikasi pembobotan antara $q(z_k)$ dengan $p(z_k)$. $\delta_{k,y}$ dan $\frac{1}{K}$ pada (10) merupakan *kronocker delta* atau *dirac delta* dari distribusi diskrit dan $\frac{1}{K}$ merupakan distribusi uniform $u(k)$ dengan K adalah total jumlah kelas. Secara intuisi, *kronocker delta* merupakan *one hot encoding* yang mana bernilai 1 jika indeks dari matriks tersebut merupakan label target dan 0 untuk lainnya [8]. Nilai $q(z_k)$ akan maksimal jika *kronocker delta* adalah matriks identitas.

$$q'(k) = (1-\epsilon)\delta_{k,y} + \frac{\epsilon}{K} \quad (10)$$

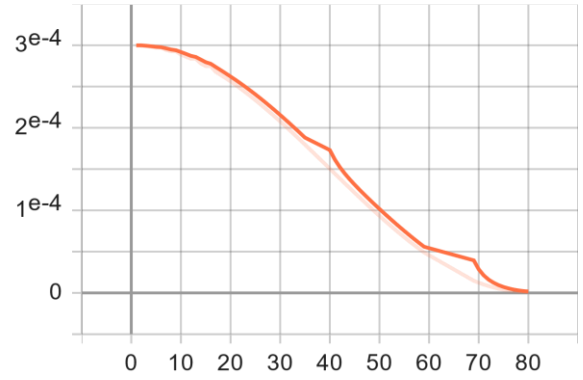
$$q'(k) = (1-\epsilon)\, p(k|z) + \frac{\epsilon}{K} \quad (11)$$

$$H(q',p) = (1-\epsilon)H(q,p) + \epsilon H(u,p) \quad (12)$$

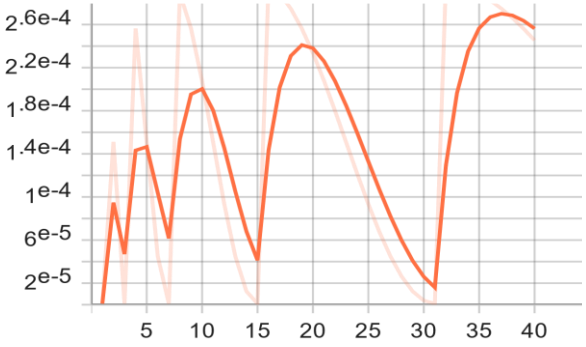
Untuk meringankan komputasi mendapatkan *one hot encoding*, (10) digantikan dengan (12) pada eksperimen ini, yaitu *cross-entropy loss* dengan probabilitas dari fungsi aktivasi s*oftmax* dan nilai ϵ disebut sebagai *smoothing factor* atau LSR *factor*.

E. Inisialisasi Parameter

Sebelum melakukan proses pelatihan model, parameter dari setiap modul akan dilakukan inisialisasi secara acak. Motivasi dilakukannya inisialisasi parameter model adalah tahapan ini dapat membantu model untuk konvergen jika digunakan metode inisialisasi yang tepat. Tanpa melakukan inisialisasi parameter (nilai parameter bernilai 0), dapat membuat model mempelajari fitur yang sama selama proses pelatihan dan mencegah model mempelajari data yang baru [9]. Penggunaan nilai parameter yang terlalu kecil dapat menyebabkan model menjadi lama untuk konvergen dan jika menggunakan nilai parameter yang besar menyebabkan model mudah menjadi divergen.



Gambar 4. Grafik cosine annealing scheduler percobaan ke-4.



Gambar 5. Grafik SGDR scheduler percobaan ke-10.

Inisialisasi parameter yang digunakan pada implementasi eksperimen ini dibagi berdasarkan modulnya. Pada modul frontend digunakan beberapa metode, yaitu metode Inisialisasi Parameter *Xavier Normal* (14) dan Inisialisasi Parameter *Kaiming Normal* (15) dengan nilai I pada (3,13) adalah ukuran dimensi *kernel* dari *convolution layer*. Untuk percobaan yang menggunakan *pretrained* model yang diambil dari penelitian [5], digunakan metode Inisialisasi Parameter Kaiming Normal dengan *gain non-linearity ReLU* bernilai √2 dan mode fan_{out}.

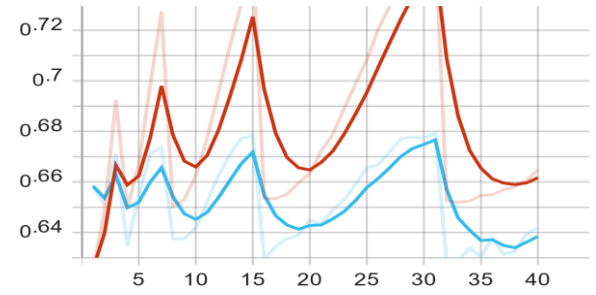
$$fan_{out} = \prod_{i=1}^{I} kernelsize_i * nfeatures_{out}$$

$$fan_{in} = \prod_{i=1}^{I} kernelsize_i * nfeatures_{in}$$

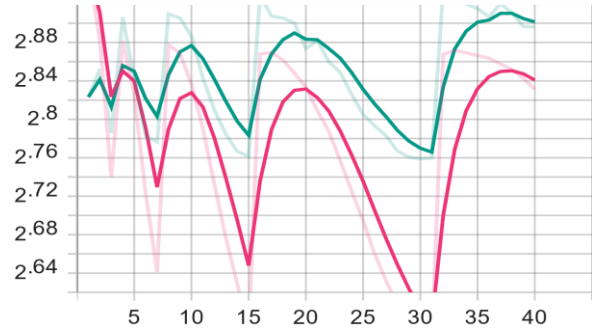
$$std = \sqrt{2/(fan_{in} + fan_{out})} \quad (14)$$

$$std = gain / \sqrt{fan_{\{in,out\}}} \quad (15)$$

Inisialisasi parameter pada modul *backend* mengikuti rujukan [5] dengan *layer GRU* menggunakan nilai acak dari distribusi normal untuk setiap parameter bobot, dan *layer Linear* menggunakan nilai acak dari distribusi uniform[-1,1] untuk parameter bobot. Pada eksperimen ini, juga membandingkan penggunaan inisialisasi distribusi uniform bawaan dari *framework PyTorch* dengan $\mathcal{U}(-k, k)$ dengan $k = \sqrt{(\frac{1}{hiddensize})}$ untuk GRU dan $k = \sqrt{(\frac{1}{nfeatures_{in}})}$ untuk Linear. Sedangkan untuk *trainable parameter layer* lainnya seperti *Batch Normalization layer* menggunakan inisialisasi bobot dengan nilai konstan 1, bias dengan nilai 0, dan standar nilai parameter lainnya yang ditentukan pada *framework PyTorch*, dengan nilai $\gamma = 1$, $\beta = 0$ untuk proses afinitas $y = \gamma x + \beta$.



Gambar 6. Nilai akurasi epoch percobaan ke-10 dengan SGDR scheduler.



Gambar 7. Nilai loss epoch percobaan ke-10 dengan SGDR scheduler.

IV. HASIL DAN ANALISIS EKSPERIMEN

Pada eksperimen ini, dilakukan dua skenario uji coba dengan menggunakan data validasi dan data uji yang didapatkan dari LRW. Metrik atau penilaian yang digunakan sebagai acuan model belajar dengan baik atau tidak menggunakan metrik akurasi dan nilai *loss* atau galat dari prediksi label oleh model dengan label target. Penilaian ini diakumulasikan dalam satu iterasi pelatihan atau satu *epoch*.

Pada uji coba menggunakan data validasi, *hyperparameter* yang digunakan terdiri dari variasi banyaknya *epoch*, banyaknya data latih, nilai probabilitas *dropout layer*, ukuran *batch*, metode inisialisasi parameter, dan metode *learning rate scheduler*. Metode inisialisasi parameter yang digunakan terdiri dari empat variasi yang ditunjukkan pada Tabel 2. dengan jumlah data validasi masing-masing percobaan sebanyak 50 data. Nilai *weight decay* untuk parameter dari optimasi Adam disesuaikan untuk semua eksperimen sebesar 1e-4 dengan *learning rate* awal sebesar 3e-4. Waktu pelatihan model yang lama, sekitar 3 hari untuk satu percobaan dengan *epoch 80* dan 1000 data latih, menjadikan beberapa uji coba tahapan ini tidak selesai sepenuhnya atau menggunakan data latih yang minimal karena keterbatasan sumber daya, yaitu dengan alasan penggunaan sumber daya yang dipakai secara bersama. Tabel 3 merupakan hasil uji coba pada saat pelatihan model dan prediksi menggunakan data validasi.

A. Hasil Data Uji

Tahapan uji coba pada skenario ini memiliki tahapan yang serupa dengan uji coba menggunakan data validasi namun pada tahapan ini digunakan data uji dari LRW *dataset*. Metrik yang digunakan untuk tahapan dengan data uji adalah nilai akurasi, nilai F1, nilai *precision*, dan nilai *recall*. Hasil uji coba data uji tertera pada Tabel 1.

Tabel 4.
Durasi waktu pelatihan

Nomor Uji Coba	Durasi Pelatihan	Interupsi
#1	1 Hari 3 Jam 53 Menit	Ya; (9,41)/50
#2	1 Hari 18 Jam 16 Menit	Ya; (12,80)/92
#3	1 Hari 2 Jam 23 Menit	Ya; 48/80
#4	18 Jam 38 Menit	Ya; 20/80
#5	2 Hari 7 Jam 30 Menit	Ya; 27/80
#6	15 Jam 29 Menit	Ya; (6,14)/20
#7	9 Jam 23 Menit	Tidak; 20
#8	9 Jam 20 Menit	Tidak; 20
#9	9 Jam 22 Menit	Tidak; 20
#10	20 Jam 20 Menit	Tidak; 40
#11	15 Jam 45 Menit	Ya; (6,13)/20
#12	1 Hari 3 Jam 7 Menit	Ya; 19/40
#13	18 Jam 53 Menit	Tidak
#14	12 Jam 23 Menit	Tidak

B. Hasil Prediksi

Pada bagian ini diperoleh hasil prediksi. Hasil prediksi diperoleh dengan melakukan percobaan sebanyak 11. Hasil dari percobaan dari percobaan ke-6 sampai ke-11 ditampilkan pada Gambar 2, Gambar 3, Gambar 4, Gambar 5, Gambar 6 dan Gambar 7.

Dari hasil prediksi, beberapa kata memiliki bentuk pengucapan konsonan yang mirip. Dalam fonologi, istilah ini disebut *minimal pair* yang merupakan pasangan kata yang memiliki pengucapan dengan artikulasi yang serupa namun konstruksi organ yang berbeda. Terdapat tiga jenis artikulasi yang menggunakan area mulut, yaitu menggunakan kedua bibir (*bilabial*), satu bibir dan gigi (*labiodental*), serta lidah dan mulut (*linguolabial*) [10]. Batasan arsitektur model ini hanya dapat mendeteksi perbedaan gerakkan area mulut/kedua bibir (*bilabial*) sehingga ambiguitas kata yang diprediksi sangat memungkinkan.

Error! Reference source not found. merupakan hasil prediksi dari model terbaik dalam uji coba ini. Dalam prediksinya, kata "***message***" dan kata "***manchester***" memiliki cara artikulasi yang mirip, hanya perbedaan gerakan lidah sehingga confidence level prediksi sangat tinggi (92,0%). Terlebih, ambiguitas kata jamak dan kata tunggal yang berakhiran "***-es***" atau "***-s***" sering kali menjadi masalah dalam arsitektur model yang diimplementasikan.

C. Penggunaan Learning Rate Scheduler

Penggunaan *Cosine Annealing Scheduler* memiliki nilai akurasi yang kurang maksimal dibandingkan rata-rata penggunaan SGDR *Scheduler* dan dapat menghasilkan model yang *overfit* dengan hyper-parameter yang sama (#6 dan #7) berdasarkan 0. Dengan SGDR *Scheduler*, setiap iterasi memiliki *learning rate* yang bervariasi dan berubah secara periodik yang memaksa model untuk keluar dari *local minima* atau *plateau* sehingga tidak membutuhkan lama untuk mencapai konvergen. Sedangkan *Cosine Annealing Scheduler* memiliki perubahan setiap *epoch* yang semakin mengecil nilai *learning rate*-nya sehingga dapat menjadikan model lama untuk konvergen ketika terjebak di daerah *plateau*. Sedangkan kekurangan dari SGDR *Scheduler* adalah nilai akurasi dan *loss* yang tidak stabil akibat perubahan *learning rate* setiap iterasi data. Hal ini dapat terjadi jika model atau penggunaan *optimizer* yang sensitif terhadap nilai *learning rate*.

Error! Reference source not found. menampilkan grafik perubahan *learning rate* dari *Cosine Annealing Scheduler* dan Gambar 9 menampilkan grafik perubahan *learning rate* dari SGDR *Scheduler*. Efek dari penggunaan SGDR *Scheduler* diperlihatkan pada **Error! Reference source not found.** yang menampilkan nilai akurasi tiap *epoch* dari data latih (berwarna merah) dan data validasi (berwarna biru) pada percobaan ke-10 (#10). **Error! Reference source not found.** memperlihatkan efek penggunaan SGDR *Scheduler* terhadap perubahan nilai *loss* pada percobaan ke-10, dengan data latih (berwarna hijau) dan data validasi (berwarna merah muda).

D. Penggunaan Label Smoothing Regularizaiton

Berdasarkan data dari Tabel 3, percobaan ke-7 dan ke-8 memiliki nilai akurasi yang tidak beda signifikan, namun untuk nilai *loss* antara kedua percobaan berbeda jauh, dengan nilai 2,753 untuk percobaan ke-7 dan 3,909 untuk percobaan ke-8. Data dari percobaan ke-1, ke-3, dan ke-5 dengan nilai LSR *factor* 0,1, memberikan nilai *loss* yang lebih kecil dibanding penggunaan LSR *factor* 0,2 atau 0,4 untuk pelatihan *transfer learning* dan *fine tuning*. Dikarenakan percobaan yang dilakukan ini menggunakan nilai *epoch* yang kecil, LSR *factor* yang digunakan pada percobaan ke-5 ke atas menggunakan LSR *factor* 0,2 atau 0,4, yang berbeda dengan rekomendasi dari implementasi D. Feng, S. Yang, S. Shan, dan X. Chen.

Berdasarkan salah satu penelitian, cara kerja *inductive bias* yang diberikan oleh *label smoothing regularization* tidak dapat dijelaskan secara teori. Namun, dalam penerapan untuk permasalahan yang memiliki label target atau kelas dengan relasi antar datanya yang sangat mirip, regularisasi ini dapat membantu model agar tidak *over-confidence* dan dapat memberikan efek negatif generalisasi pada label target yang minoritas. Tujuan dari regularisasi ini tidak lain untuk mempertahankan relasi *mutual exclusive* antar data. Walaupun dapat membantu meningkatkan bias terhadap data yang mayoritas, cara regularisasi ini tidak membantu untuk permasalahan yang membutuhkan sensitivitas terhadap nilai probabilitas prediksi karena akan menurunkan nilai *confidence level*.

E. Penggunaan MixUp

Berdasarkan Tabel 3, penggunaan *MixUp* dengan nilai *alpha* 0,1 ke atas menjadikan model *underfit* pada percobaan ke-1, ke-3, dan ke-5. Untuk mengetahui apakah hasil *underfit* diakibatkan oleh nilai probabilitas *dropout*, pada percobaan ke-14 digunakan nilai probabilitas *dropout* senilai

0,1 dan tetap menghasilkan model yang *underfit* untuk nilai *alpha* yang lebih besar dari 0,1. Selain itu, untuk mengetahui apakah penggunaan *MixUp Data Augmentation* berpengaruh terhadap nilai akurasi model, pada percobaan ke-9 tidak dipergunakan metode *MixUp* dengan *hyper-parameter* dan jumlah data latih yang sama pada percobaan ke-6. Hasil percobaan ke-6 memberikan nilai akurasi yang lebih tinggi dibandingkan tanpa penggunaan *MixUp* walaupun nilai *MixUp alpha* yang digunakan relatif kecil. Namun, untuk perbandingan percobaan ke-9 dengan percobaan ke-10 yang memiliki nilai *epoch* lebih besar (*epoch* 20 dan *epoch* 40), hasil akurasi penggunaan tanpa *MixUp* lebih tinggi serta nilai *loss* yang lebih rendah walaupun perbedaannya tidak signifikan. Dari data ini, faktor perbandingan nilai *epoch* dan penggunaan *MixUp* perlu diperhatikan. Penggunaan jumlah data latih juga menjadi faktor penentu nilai akurasi. Pada percobaan ke-11, *hyper-parameter* yang digunakan sama dengan percobaan ke-9 namun berbeda pada jumlah data latihnya, menghasilkan nilai akurasi yang lebih tinggi untuk percobaan ke-11 yang menggunakan *MixUp*.

Komposisi perbandingan antara nilai *epoch*, *MixUp alpha*, dan jumlah data latih tidak begitu jelas. Namun, beberapa kesimpulan yang dapat diambil dari data uji coba tersebut adalah penggunaan *MixUp* dapat membantu menaikkan nilai akurasi pada jumlah data yang cukup banyak. Efeknya akan lebih terlihat jelas (dan lebih cepat) pada penggunaan *epoch* yang kecil, seperti pada perbandingan percobaan ke-11 dan ke-12, dengan catatan pada percobaan ke-12, pelatihan model mengalami interupsi dan tidak dilanjutkan sampai akhir *epoch*. Dari hasil percobaan tersebut, penggunaan nilai *MixUp alpha* yang baik berada pada nilai 0,1. Kesimpulan ini membedakan dengan implementasi rujukan [5] yang menggunakan nilai *MixUp alpha* 0,2 untuk model terbaiknya.

F. Penggunaan Metode Inisialisasi Parameter

Tabel 2 menampilkan penggunaan beberapa metode inisialisasi parameter yang digunakan pada eksperimen ini. Secara bawaan (*default*), *framework PyTorch* sudah melakukan inisialisasi parameter sehingga setiap pembuatan objek layer, parameter bobot (*weight*) dan bias memiliki nilai parameter non-nul. Metode inisialisasi parameter A untuk modul *backend* merupakan metode yang digunakan oleh *framework PyTorch*.

Berdasarkan hasil uji coba data validasi, hasil akurasi yang didapatkan tidak membawa perbedaan antara penggunaan metode inisialisasi parameter A dan C untuk proses latih *transfer learning*. Namun, pada hasil uji coba dengan metode inisialisasi parameter B untuk proses latih *end-to-end*, kurangnya *epoch* menyebabkan nilai akurasi yang sangat kecil dibanding dengan hasil A dan C. Rujukan [5] dalam implementasinya menggunakan inisialisasi parameter *Kaiming Normal* untuk modul *frontend* dan menggunakan standar *framework PyTorch* untuk modul *backend* (pada Tabel 1 merupakan metode A).

G. Dampak dari Interupsi Pelatihan Model

Tabel 3 memperlihatkan hasil validasi dengan beberapa *hyper-parameter* yang digunakan. Pada kesempatan eksperimen ini, tidak semua uji coba selesai dilakukan atau tidak semua uji coba dilatih tanpa mengalami interupsi. Implementasi kode yang digunakan pada pelatihan model sudah diberikan mekanisme untuk melanjutkan proses pelatihan terakhir sebelum mengalami interupsi (*checkpoint* model). Untuk alasan tersebut, setiap uji coba yang dilakukan mengalami interupsi, diberikan tanda kurung atau/dan garis miring pada nilai *epoch*. Sebagai contoh pada percobaan ke-12 memiliki nilai *epoch* 19/40 yang berarti model dilatih mengalami interupsi dan tanpa pelanjutan pelatihan dengan target nilai *epoch* 40 dan realisasi nilai *epoch* 19. Tabel 4 menunjukkan variasi waktu dan nilai *epoch* yang diberikan.

V. KESIMPULAN

Pengubahan *learning rate* secara periodik menggunakan SGDR *Scheduler* memaksa model untuk keluar dari *local minima* dan dapat mengurangi *overfitting* dan *underfitting* dengan nilai *epoch* yang kecil, walaupun memiliki risiko tidak stabilnya hasil akurasi dan nilai *loss* pada akhir *epoch*. Memanfaatkan LSR dan *MixUp* dengan nilai yang tepat dapat membantu model untuk mencapai konvergen dengan nilai akurasi tertinggi 0,828 untuk data uji dan 0,910 untuk data latih dengan nilai *loss* terendah masing-masing 2,373 dan 2,441 pada penggunaan *dataset* LRW.

Penggunaan metode regularisasi *Label Smoothing* dengan nilai 0,2 atau 0,4 dapat membantu meningkatkan nilai akurasi pada pelatihan model dengan nilai *epoch* yang kecil dan penggunaan data latih yang minimal, walaupun terdapat efek negatif pada nilai *loss* dan *confidence level* dari prediksi label target. 5. Penggunaan *MixUp data augmentation* dengan nilai 0,1 menghasilkan nilai akurasi yang lebih tinggi dibandingkan tanpa menggunakan metode tersebut pada pelatihan model dengan nilai *epoch* yang kecil untuk pelatihan model *transfer learning* ataupun *fine tuning*.

Beberapa uji coba yang dilakukan mengalami interupsi saat pelatihan sehingga dapat menyebabkan hasil pelatihan yang tidak akurat untuk tiap percobaan skenario yang ada. Berdasarkan uji coba yang dilakukan, penggunaan metode inisialisasi parameter *Xavier Normal* untuk modul *frontend*, $\mathcal{N}(\mu = 0, \sigma = 1)$ untuk GRU, dan $\mathcal{U}(-1,1)$ untuk *fully-connected layer*, yang dilatih secara *end-to-end*, tidak memberikan hasil yang baik karena nilai *epoch* yang digunakan sangat minim.

DAFTAR PUSTAKA

[1] T. Stafylakis and G. Tzimiropoulos, "Combining Residual Networks With LSTMS For Lipreading," in *Interspeech Conference*, 2017, doi: 10.21437/Interspeech.2017-85.

[2] B. Martinez, P. Ma, S. Petridis, and M. Pantic, "Lipreading Using Temporal Convolutional Networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6319--6323, doi: 10.1109/ICASSP40776.2020.9053841.

[3] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9915 LNCS, pp. 47–54, doi: 10.1007/978-3-319-49409-8_7.

[4] P. Ma, B. Martinez, S. Petridis, and M. Pantic, "Towards Practical Lipreading with Distilled and Efficient Models," *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, New Jersey, pp. 7608--7612, 2021.

[5] D. Feng, S. Yang, S. Shan, and X. Chen, "Learn an Effective Lip Reading Model without Pains," in *ArXiv*, 2020.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132--7141, doi: 10.1109/TPAMI.2019.2913372.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[8] E. W. Weisstein, "Kronecker Delta.," Wolfram Math World, 2020. https://mathworld.wolfram.com/KroneckerDelta.html.

[9] K. Katanforoosh, "Initializing Neural Networks," *DeepLearning.AI*, 2019. https://www.deeplearning.ai/ai-notes/initialization/.

[10] P. Ladefoged and I. Maddieson, *The Sounds of the World's Languages*. New Jersey: Blackwell Publishers, 1996.