Pengembangan Metode Neural Machine Translation Berdasarkan Hyperparameter Neural Network

(Studi Kasus: Bahasa Jerman – Inggris)

(*Neural Machine Translation Method Development Based on Neural Network Hyperparameter
(Case Study: German – English)*)

Muhammad Yusuf Aristyanto¹, Robert Kurniawan²

^{1,2}Politeknik Statistika STIS

Jl. Otto Iskandar Dinata No. 64C Jakarta 13330

E-mail: 221709870@stis.ac.id¹, robertk@stis.ac.id²

ABSTRAK

Manusia sebagai makhluk sosial yang selalu ingin berhubungan dengan manusia lainnya memaksa manusia untuk saling berkomunikasi. Di sinilah peran bahasa menjadi amat penting, karena dengan adanya bahasa, maka akan dengan mudah mengerti apa yang ingin disampaikan oleh orang lain. Untuk itu, perlu adanya media yang dapat membantu memahami berbagai bahasa di dunia, salah satunya adalah mesin penerjemah. Salah satu metode yang dapat digunakan untuk membuat mesin penerjemah adalah *Neural Machine Translation* (NMT). NMT yang sekarang sudah ada masih memiliki berbagai kekurangan dan perlu dilakukan pengembangan lebih jauh. Diantaranya pada masalah *overfitting* yang membuat modelnya tidak bisa melakukan generalisasi pada data lain yang diujikan. Banyak hal yang mempengaruhi performa untuk mengatasi *overfitting* dari NMT tersebut, salah satunya adalah ukuran *hyperparameter* yang digunakan dan arsitektur model yang digunakan. Penelitian ini bertujuan untuk mengembangkan arsitektur model NMT dan melakukan simulasi pada masing-masing *hyperparameter Neural Network* dan ukuran pada arsitektur modelnya, antara lain *batch size*, *epoch*, *optimizer*, *activation function*, dan *dropout rate*. Hasil yang didapatkan adalah model pengembangan dapat mengatasi masalah *overfitting* dari model sebelumnya dengan akurasi sebesar 71,95% dan skor BLEU sebesar 43,11% yang dilakukan pada data uji lainnya.

Kata kunci: Arsitektur, Neural Network, NMT, Hyperparameter

ABSTRACT

Humans as social beings who always want to be in touch with other human's force humans to communicate with each other. This is where the role of language becomes very important, because with language, it will be easy to understand what other people want to convey. For this reason, it is necessary to have a media that can help to understand the various languages in the world, one of which is machine translation. One method that can be used to make machine translators is Neural Machine Translation (NMT). The existing NMT still has various shortcomings and needs to be further developed. Among them is the problem of overfitting which makes the model unable to generalize to other data being tested. Many things affect the performance to overcome the overfitting of the NMT, one of which is the size of the hyperparameters used and the model architecture used. This study aims to develop the NMT model architecture and perform simulations on each of the Neural Network hyperparameters and the size of the model architecture, including batch size, epoch, optimizer, activation function, and dropout rate. The results obtained are the development model can overcome the overfitting problem of the previous model with an accuracy of 71.95% and a BLEU score of 43.11% which was carried out on other test data.

Keywords: Architecture, Neural Network, NMT, Hyperparameter

PENDAHULUAN

Bahasa adalah sesuatu yang penting bagi manusia untuk saling bersosialisasi. Bahasa memiliki sifat unik masing-masing, tiap bahasa memiliki ciri khasnya yang bahasa lain tidak miliki (Nugroho Aditya et al., 2015). Bahasa adalah media komunikasi bagi seseorang dalam menuangkan ide, konsep, perasaan, atau gagasannya kepada orang lain (Yahya, 2015). Hal ini berarti, tanpa adanya bahasa, manusia akan sulit untuk menjalankan kehidupannya sebagai makhluk sosial.

Keberagaman bahasa yang ada di dunia, menyebabkan komunikasi antar manusia relatif sulit. Sehingga, diperlukan media untuk menjembatani keberagaman tersebut, salah satunya adalah mesin penerjemah. Mesin penerjemah bertujuan untuk membantu memahami dan mempelajari bahasa lain yang tidak dimengerti sebelumnya. Mesin penerjemah menjadi salah satu solusi untuk menjembatani interaksi antar-suku dan

digunakan dalam pemeliharaan bahasa sebagai wujud dari kesadaran dalam melestarikan bahasa daerah (Pranata & Muljono, 2016).

Mesin penerjemah adalah sebuah media yang dapat melakukan pemindahan bahasa satu ke bahasa lain. Metode yang sering digunakan untuk membangun suatu mesin penerjemah antara lain adalah *Statistical Machine Translation* (SMT) dan *Neural Machine Translation* (NMT). SMT adalah suatu metode yang mengimplementasikan pendekatan model statistik. Masing-masing parameter dalam SMT bersumber dari *corpus parallel* jenis teks (Sujaini, 2020). SMT berguna untuk menanggulangi kekurangan dari pendekatan sebelumnya, seperti pendekatan berbasis contoh dan pendekatan berbasis aturan. Namun, kemampuan SMT masihlah belum mampu menangani masalah pada padanan kata dan panjang kalimat pada kalimat terjemahan (*corpus*) (Adji et al., 2011).

NMT adalah suatu metode baru dalam proses penerjemahan mesin yang di dalamnya terdapat arsitektur *Recurrent Neural Network* (RNN) pada bagian *encoder* dan *decoder*-nya (Hunston, 2002). Bahdanau (2015) menggunakan NMT berbasis *attention* (Bahdanau et al., 2015). Metode NMT berbasis *attention* adalah metode RNN yang menggunakan dua *layer Long Short Term Memory* (LSTM). Namun model tersebut belum mampu menangani masalah *overfitting* yang membuat modelnya tidak mampu melakukan generalisasi kepada data lainnya selain data latih. Kemudian, Liu (X. Liu et al., 2020) dan Sennrich (Sennrich et al., 2016) melakukan pengembangan model dari model dasar yang dibuat oleh Bahdanau untuk meningkatkan performa model dilihat dari akurasi. Namun, keduanya masih belum mampu mengatasi permasalahan *overfitting*.

NMT merupakan metode penerjemahan bahasa yang menggunakan *Neural Network*. *Neural Network* adalah suatu metode yang mengolah informasi atau data yang terinspirasi dari cara kerja jaringan otak (Asriningtias et al., 2015). *Neural Network* memiliki susunan berupa *node* dan relasi. Ada tiga jenis *node* diantaranya adalah *input layer*, *hidden layer*, dan *output layer*. Sedangkan fungsi dari relasi adalah untuk menggabungkan lebih dari satu *node* dengan suatu bobot yang memiliki arah aliran dalam suatu proses tertentu (Kusrini et al., 2009). *Neural Network* bermanfaat dalam memecahkan masalah terkait dengan pengenalan tren atau pola, prediksi, *data mining*, dan klasifikasi (Shukla et al., 2010). Salah satu keunggulannya adalah keandalannya dalam mengolah data (Larose, 2005).

Pada *Neural Network*, terdapat sebuah istilah yang bernama *hyperparameter* yang dibutuhkan dalam proses pelatihan model. Contoh *hyperparameter* antara lain *epoch*, *batch*, *split*, *learning rate*, *activation function*, *optimizer*, dan jumlah *layers*. Kinerja model menggunakan *Neural Network* dipengaruhi oleh *hyperparameter* yang menyusunnya, diantaranya *epoch*, *layer*, *split* data, dan jumlah data (Nurhikmat, 2018).

Selain itu, hal yang diperhatikan, yakni performa dari arsitektur model *Neural Network* yang dibangun. Arsitekturnya dilihat dari *layer* dan unit, serta algoritma pembelajarannya. Tidak ada ketetapan yang pasti dalam membentuk suatu arsitektur yang baik, tetapi ukuran arsitektur yang lebih kecil dapat mengakibatkan performa model kurang bagus. Sebaliknya, ukuran arsitektur yang lebih besar akan mempunyai sifat lemah ketika melakukan generalisasi data yang baru, serta membutuhkan waktu yang lama untuk melakukan pelatihan model (Che et al., 2011).

Berdasarkan permasalahan di atas, maka penelitian ini bertujuan melakukan modifikasi arsitektur model NMT dengan menambahkan *layer*, sehingga diharapkan dapat meningkatkan performa model, terutama untuk mengatasi *overfitting*. Selain itu, akan dilakukan simulasi pada *hyperparameter Neural Network* berdasarkan ukuran arsitektur modelnya, antara lain *batch size*, *epoch*, *optimizer*, *activation function*, dan *dropout rate*. Selanjutnya, model yang diperoleh diaplikasikan ke *dataset* yang berbeda.

METODE

Dataset Corpus Parallel

Korpus dapat diartikan sebagai sekumpulan teks berbentuk tulisan atau secara lisan yang berbentuk data dan dapat dibaca menggunakan suatu perangkat mesin (McEnery, 2006). Korpus dapat dikategorikan menjadi delapan macam, antara lain korpus, korpus umum, korpus komparatif, korpus paralel, korpus pemelajar, korpus pedagogis, korpus historis atau diakronis, dan korpus monitor (Hunston, 2002). Penelitian ini hanya akan menggunakan jenis korpus paralel. Korpus paralel (*Corpus Parallel*) merupakan kumpulan kalimat-kalimat yang saling berpasangan antara suatu bahasa dan terjemahan bahasa lainnya (Hadi, 2014).

Pada penelitian ini, peneliti menggunakan *dataset dummy* yang didapatkan dari situs web *manythings* (<https://www.manythings.org/anki/>) dengan judul *Tab-delimited Bilingual Sentence Pairs*. Situs tersebut berisi berbagai macam terjemahan dari berbagai bahasa di seluruh dunia. Di sini peneliti menyebutnya sebagai

pasangan kalimat (*Corpus Parallel*) dari bahasa satu ke bahasa lainnya. Pada Tabel 1 di bawah, ditunjukkan contoh sampel *corpus parallel* yang digunakan pada penelitian ini.

Tabel 1. Contoh *Corpus Parallel*

Inggris	Jerman
<i>Hello!</i>	<i>Sers!</i>
<i>Go away.</i>	<i>Geh weg!</i>
<i>That was ours.</i>	<i>Das war unsrer.</i>
<i>The river is wide.</i>	<i>Der Fluss ist breit.</i>
<i>He should thank you.</i>	<i>Er sollte dir danken.</i>
Inggris	Spanyol
<i>Jump.</i>	<i>Salte.</i>
<i>Go away.</i>	<i>Lárguese.</i>
<i>She is happy.</i>	<i>Ella es feliz.</i>
<i>It doesn't matter.</i>	<i>No es importante.</i>
<i>Tonight is the night.</i>	<i>Esta noche es la noche.</i>

Sumber: <https://www.manythings.org/anki>

Peneliti akan mengambil dua buah dataset, yang pertama digunakan untuk melakukan proses pelatihan model dan untuk mendapatkan kombinasi parameter *Neural Network* yang terbaik, yaitu Bahasa Inggris-Jerman. Sedangkan yang kedua akan digunakan untuk menguji model dan parameter terbaik tersebut, yaitu Bahasa Inggris-Spanyol. Selanjutnya, korpus paralel akan digunakan sebagai masukan dari metode *Neural Machine Translation*. Penggunaan Bahasa Inggris-Jerman dan Inggris-Spanyol dikarenakan kualitas dan kuantitas dataset yang bagus untuk dilakukan pemodelan.

Recurrent Neural Network (RNN)

RNN adalah salah satu bentuk arsitektur *Neural Network* yang pada saat pelatihan akan dipanggil secara berulang dalam memproses data sekuensial (P. Liu et al., 2016). Pemodelan dengan RNN dapat mengatasi berbagai permasalahan pengkategorisasian berbagai kalimat (Xia et al., 2018). Hal ini disebabkan oleh proses pelatihannya yang akan dipanggil secara berulang sehingga dapat menangani masukan dan keluaran data yang panjangnya dapat bervariasi. RNN adalah salah satu cabang dari metode *Neural Network* yang menggunakan perulangan (rekurensi) dengan memanfaatkan data *histories* (masa lalu). Oleh karena itu, studi terbaru menyebutkan kalau RNN dapat cukup kuat dalam mengatasi permasalahan klasifikasi (Firmansyah et al., 2020). RNN memiliki arsitektur seperti pada Gambar 1 berikut.

Gambar 1. Arsitektur RNN *folded* dan *unfolded* (Firmansyah et al., 2020).

Neural Machine Translation (NMT)

Neural Machine Translation atau biasa disebut sebagai NMT merupakan suatu metode baru dalam proses penerjemahan yang di dalamnya terdapat pengaplikasian dari arsitektur RNN pada bagian *encoder* dan *decoder*-nya (Hunston, 2002). Metode NMT yang digunakan oleh peneliti saat ini adalah metode berbasis *attention*, yang dilakukan oleh Bahdanau (Bahdanau et al., 2015). Untuk melakukan penerjemahan menggunakan jaringan saraf, yang pertama adalah menyambungkan bagian RNN *encoder* dengan *decoder*.

RNN *encoder* berfungsi memproses kata atau kalimat yang berbahasa sumber (manusia) menjadi suatu kalimat yang lebih bisa dipahami oleh mesin. Sedangkan RNN *decoder* akan mengambil hasil keluaran dari RNN *encoder* untuk digunakan sebagai masukan yang kemudian dapat menghasilkan terjemahan kata demi kata (Abidin, 2017). Gambar 2 berikut ini mengilustrasikan tentang bagaimana cara menyambungkan model *encoder* dan *decoder* bersama-sama. Arsitektur model dari suatu mesin terjemahan juga biasa dikenal dengan istilah model *encoder-decoder* (Papineni et al., 2002).

Gambar 2. Arsitektur Model *Encoder-Decoder* (Papineni et al., 2002)

Long-Short Term Memory (LSTM)

Menurut Burtsev (2018), LSTM adalah pengembangan khusus dari RNN. LSTM memiliki teknis yang mirip dengan RNN, namun LSTM akan mengganti *hidden unit* milik RNN menggunakan *memory block* yang terdiri dari 4 komponen yaitu *input gate*, *memory cells*, *forget gate*, dan *output gate* (Lample et al., 2016). Arsitektur LSTM ditunjukkan oleh Gambar 3 di bawah ini. Berikut adalah penjelasan dari tiap komponennya.

- *Input gate* berguna untuk melakukan *update* pada *memory cell*.
- *Output gate* berguna untuk menentukan *hidden state* berikutnya.
- *Forget gate* akan menentukan informasi yang harus dibuang atau disimpan.
- *Memory cells* berguna untuk mengingat *weight* seiring berubahnya interval waktu, tiga *gate* sebelumnya mengatur aliran *input* dan *output memory cell*.

Gambar 3. Arsitektur LSTM (Lample et al., 2016)

Hyperparameter

Metode pemodelan pada masa sekarang banyak melibatkan parameter dalam melakukan proses pelatihan data. Berbeda dengan parameter lain yang bisa saja berubah ketika atau seiring dengan berjalannya proses pelatihan data, parameter ini tidak dapat diubah-ubah, parameter ini biasa disebut *hyperparameter*. *Hyperparameter* penting digunakan dan didefinisikan karena dapat mempengaruhi keandalan dan performa dari suatu teknik pemodelan (Hutter & F., 2018). *Hyperparameter* adalah suatu parameter yang dapat digunakan untuk mengatur dan mengontrol sebuah metode pemodelan sehingga dapat menghasilkan performa prediksi yang lebih baik (Septiani & Rizal, 2021).

Tabel 2. *Hyperparameter Neural Network* yang akan Digunakan

Parameter	Nilai
<i>Batch Size</i>	16, 32, 64
<i>Epoch</i>	10, 30, 50
<i>Optimizer</i>	Adam, RMSprop, SGD
<i>Activation Function</i>	SoftMax, SoftPlus, SoftSign
<i>Dropout Rate</i>	0.2, 0.5, 0.8

Di sini peneliti akan menggunakan beberapa kombinasi dari *hyperparameter* dan ukuran arsitektur *Neural Network*, antara lain *batch size*, *epoch*, *optimizer*, *activation function*, dan *dropout rate* seperti pada Tabel 2 di atas. Berikut adalah penjelasan dari masing-masing *hyperparameter* yang digunakan.

- *Batch Size* adalah pembagian data menjadi beberapa bagian sehingga akan memudahkan beban pemrosesan memori. Nilai yang digunakan antara lain adalah 16, 32, 64 dengan mempertimbangkan angka kelipatan 2 yang dapat memudahkan proses pembagian dataset dalam pelatihan model
- *Epoch* adalah jumlah iterasi bagi seluruh *dataset* dalam proses pelatihan pada *Neural Network* sampai ke awal lagi. Nilai yang digunakan antara lain adalah 10, 30, 50 dengan mempertimbangkan waktu dari proses pelatihan yang lebih cepat
- *Optimizer* merupakan algoritma yang digunakan untuk mengubah atribut jaringan saraf Anda seperti bobot dan *learning rate* untuk mengurangi loss pada model. Nilai nilai yang digunakan antara lain adalah 10, 30, 50 dengan mempertimbangkan waktu dari proses pelatihan yang lebih cepat
- *Activation Function* mendefinisikan bagaimana jumlah tertimbang dari input diubah menjadi *output*. Menggunakan nilai berupa *SoftMax*, *SoftPlus*, dan *SoftSign* dengan mempertimbangkan kemiripan dari ketiganya
- *Dropout Rate* adalah regularisasi jaringan syaraf saat neuron dipilih secara acak dan tidak dipakai saat pelatihan. Menggunakan nilai masing-masing adalah 0.2, 0.5, dan 0.8 dengan mempertimbangkan pemerataan nilai dari nol sampai satu

Bilingual Evaluation Under Study (BLEU)

BLEU adalah suatu metode yang dapat melakukan evaluasi mengenai kualitas hasil terjemahan suatu mesin penerjemah dari bahasa satu ke bahasa lainnya. BLEU bekerja dengan cara mengukur skor presisi dari modified n-gram dari hasil terjemahan prediksi dengan terjemahan aslinya serta menggunakan konstanta *brevity penalty*. Rumus BLEU ditunjukkan pada persamaan (1), (2), dan (3) berikut (Papineni et al., 2002):

$$BP_{BLEU} = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } c \leq r \end{cases} \dots\dots\dots (1)$$

$$P_n = \frac{\sum_{C \in \text{corpus}} n\text{-gram} \in C \sum \text{count}_{clip}(n\text{-gram})}{\sum_{C \in \text{corpus}} n\text{-gram} \in C \sum \text{count}(n\text{-gram})} \dots\dots\dots (2)$$

$$BLEU = BP_{BLEU} \cdot e^{\sum_{n=1}^N w_n \log p_n} \dots\dots\dots (3)$$

Keterangan:

BP = *brevity penalty*

c = jumlah kata dari hasil terjemahan otomatis

r = jumlah kata rujukan

P_n = *modified precision score*

w_n = $1/N$ (standar nilai N untuk BLEU adalah 4)

Confussion Matrix

Confusion matrix adalah metode yang berfungsi untuk mengevaluasi performa atau keandalan dari suatu model pembelajaran mesin dalam mengklasifikasikan suatu data (Rohim et al., 2019). *Confusion matrix* secara umum dapat digunakan untuk menghitung beberapa matriks antara lain *precision*, *accuracy*, dan *recall* (Powers, 2007). *Confusion matrix* digambarkan seperti pada Tabel 3 berikut.

Tabel 3. *Confussion Matrix*

Keterangan		Nilai Asli	
		Benar	Salah
Nilai Prediksi	Benar	<i>True Positive</i> (TP)	<i>False Positive</i> (FP)
	Salah	<i>False Negative</i> (FN)	<i>True Negative</i> (TN)

Sumber: (Rohim et al., 2019)

Akurasi adalah suatu nilai yang mewakili ketepatan dari suatu model dalam mengklasifikasikan data. Akurasi biasa digunakan untuk pengukur seberapa tepat kah suatu metode klasifikasi, akurasi didefinisikan seperti Persamaan (4).

$$Akurasi = \frac{TP+TN}{TP+FP+FN+TN} \dots\dots\dots (4)$$

Akurasi juga dapat dituliskan seperti pada Persamaan (5), yakni ketika keseluruhan data latih merupakan sesuatu yang selalu bernilai benar (Kulkarni et al., 2020). Nilai N mewakili banyaknya percobaan klasifikasi yang dilakukan.

$$Akurasi = \frac{N_{Benar}}{N} \dots\dots\dots (5)$$

HASIL DAN PEMBAHASAN

Pengembangan Metode NMT

Pada proses pengembangan model NMT ini, terdapat beberapa *default value* untuk *hyperparameter* yang digunakan antara lain:

- Digunakan hanya sebanyak 20.000 sampel data, karena mempertimbangkan performa dan kapasitas mesin komputasi yang digunakan
- *Split dataset* yang digunakan yakni 6:4, dengan mempertimbangkan jumlah data, sehingga diharapkan tidak terjadi ketimpangan antara data *train* dan *test*.
- *Epoch* yang digunakan adalah sebanyak 50, dengan mempertimbangkan lamanya waktu pemodelan.
- *Batch size* yang dipilih penelitian ini adalah 16, dengan mempertimbangkan kapasitas RAM yang tersedia.
- Penggunaan '*SoftMax*' sebagai *Activation Function* karena keandalannya dalam melakukan klasifikasi.
- *Layer Dropout*, menggunakan 0.5 sebagai *input*.
- *Optimizer* menggunakan '*Adam*' karena keandalannya dalam mengoptimalkan bobot.

Hasil pengembangan model NMT yang akan terbagi menjadi 4 model, antara lain model dasar dari Bahdanau, kemudian model yang dioptimalkan dengan metode *Callbaks*, lalu model dengan penambahan *layer Dropout*, dan terakhir adalah model yang dioptimalkan dengan metode *Cross Validation*. Berikut adalah penjelasan selengkapnya dan hasil yang didapatkan dari tiap model dan metode pengembangan yang digunakan.

Pada model arsitektur dasar (Gambar 4), terdapat beberapa lapis *layers*, antara lain:

- *Embedding*, metode yang digunakan untuk merepresentasikan variabel diskrit sebagai vektor kontinu
- LSTM, untuk mengatasi ketergantungan jangka panjang (*long term dependencies*) pada masukannya
- *Repeat Vector*, jembatan antara *encoder* dan *decoder*
- *Time Distributed*, membuat vektor dengan panjang yang sama dengan jumlah fitur yang dihasilkan dari lapisan sebelumnya

1. Model Dasar

Gambar 4. Arsitektur Model Dasar (Bahdanau et al., 2015)

Gambar 5. Tren Akurasi dan Loss pada Model Dasar

Terlihat pada Gambar 5, loss pada model mencapai titik terendah pada epoch ke-11. Namun, secara berangsur-angsur loss model semakin bertambah. Hal ini mengindikasikan bahwa pada model dasar terjadi *overfitting*. Selain itu, skor BLEU yang diperoleh pada data *train* adalah sebesar 0,658279 dan pada data test sebesar 0,198291. Perbedaan skor yang jauh pada data *train* menunjukkan bahwa model dasar ini mengalami *overfitting*. *Overfitting* adalah suatu kondisi saat performa model sangat bagus hanya pada data latih, sebaliknya memiliki performa yang jauh lebih buruk pada data uji (Belkin et al., 2019).

2. Model Dasar dengan Penambahan Callbacks

Penambahan *Callbacks* pada model dasar dilakukan dengan menggunakan fungsi *EarlyStopping*. Beberapa parameter yang relevan dalam fungsi *EarlyStopping* tersebut adalah:

- *monitor*: nilai yang dipantau, yaitu: *val_loss*

- *min_delta*: perubahan minimum dalam nilai yang dipantau. Pada penelitian ini menggunakan *min_delta*=0 yang berarti bahwa proses pelatihan akan dihentikan jika perubahan absolut dari nilai yang dipantau kurang dari 0
- *patience*: jumlah *epoch* tanpa peningkatan setelah itu pelatihan akan dihentikan. Pada penelitian ini menggunakan *patience* 3, sehingga jika dalam 3 *epoch* masih lebih besar, maka pelatihan akan dihentikan
- *restore_best_weights*: setel metrik ke *True* jika ingin mempertahankan bobot terbaik setelah dihentikan

Gambar 6. Tren Akurasi dan *Loss* pada Model dengan *Callbacks*

Pada Gambar 6, *loss* pada model mencapai titik terendah pada *epoch* ke-14, dengan *train accuracy* sebesar 0,888117 dan *test accuracy* sebesar 0,677500. Skor BLEU yang dihasilkan pada data *train* adalah sebesar 0,519729 dan pada data *test* sebesar 0,162981. Perbedaan skor yang jauh pada data *train* menunjukkan bahwa model dengan penambahan *Callbacks* ini masih mengalami *overfitting*.

3. Model Dasar dengan Penambahan *Layer Dropout*

Pada Gambar 7, *loss* pada model mencapai titik terendah pada *epoch* ke-25. Model dengan penambahan *Callbacks* juga memiliki akurasi data *train* dan *test* yang cenderung lebih kecil, yakni sebesar 0,821433 dan 0,694925, dengan skor BLEU yang dihasilkan adalah 0,517304 dan 0,187630. Selisih skor yang jauh pada data *train* menunjukkan bahwa model dengan penambahan *Dropout* ini masih mengalami *overfitting*.

Gambar 7. Tren Akurasi dan *Loss* pada Model dengan Penambahan *Layer Dropout*

Layer Dropout adalah suatu metode yang meregularisasi Neural Network dengan cara mengacak beberapa *neuron* dan tidak memakainya dalam proses pelatihan. Kegunaan utama dari *Dropout* adalah untuk mempercepat pelatihan model pembelajaran mesin serta dapat mengurangi kejadian *overfitting* pada model (Santoso & Ariyanto, 2018). Model arsitektur dengan penambahan *layer Dropout* ditunjukkan seperti pada Gambar 8 di bawah ini.

Gambar 8. Arsitektur Model Dasar dengan Penambahan *Layer Dropout*

4. Pengimplementasian *Cross Validation*

Gambar 9. Tren Akurasi dan *Loss* pada Model dengan *Cross Validation*

Pada Gambar 9 di atas, *loss* pada model mencapai titik terendah pada *epoch* ke-22. Kemudian kalau juga selisih akurasi dari data *train* dan data *test* yang cenderung lebih kecil, yakni sebesar 0,8292 dan 0,7195. Ini adalah indikasi dari turunnya *overfitting*. Selain itu, skor BLEU yang dihasilkan pada data *train* adalah sebesar 0,429079 dan pada data *test* sebesar 0,431115. Perbedaan skor yang kecil menunjukkan bahwa model dengan *Cross Validation* ini sudah tidak mengalami *overfitting*.

Simulasi Hyperparameter Neural Network

Pada penelitian ini, dilakukan simulasi dengan data yang sama seperti sebelumnya dengan menggunakan model terbaik dan terakhir sebagai hasil dari subbab sebelumnya. Simulasi ini dilakukan dengan menggunakan bantuan fungsi *Grid Search CV* yang secara otomatis akan melakukan pelatihan model dari berbagai kombinasi nilai *hyperparameter* yang ada. Hasilnya bisa dilihat pada Tabel 4 dengan parameter terbaik diberi warna hijau.

Tabel 4. Hasil Simulasi pada *Hyperparameter Neural Network*

<i>Hyperparameter</i>	Nilai	Rata-Rata Akurasi	Standar Deviasi Akurasi
<i>Batch Size</i>	16	0,612167	0,034584
	32	0,631333	0,020233
	64	0,633083	0,009790
<i>Epoch</i>	10	0,641833	0,000717
	30	0,549167	0,052383
	50	0,459250	0,034202
<i>Optimizer</i>	<i>Adam</i>	0,464500	0,137227
	<i>RMSProp</i>	0,478667	0,112564
	<i>SGD</i>	0,701500	0,005672
<i>Activation Function</i>	<i>SoftMax</i>	0,594250	0,072630
	<i>SoftPlus</i>	0,533250	0,085778
	<i>SoftSign</i>	0,119083	0,134133
<i>Dropout Rate</i>	0,2	0,643917	0,004789
	0,5	0,629000	0,027116
	0,8	0,642583	0,005606

Implementasi Menggunakan Data Lain

Untuk menguji arsitektur model dan *hyperparameter* terbaik yang sudah didapatkan sebelumnya, digunakanlah data lain, yakni *dataset corpus parallel* Bahasa Spanyol-Inggris untuk mengujinya. Hasilnya ditunjukkan pada Tabel 5 berikut ini.

Tabel 5. Perbandingan Hasil Pemodelan dengan Data Lain

Data	Skor BLEU (<i>Train</i>)	Skor BLEU (<i>Test</i>)
Jerman-Inggris	0,454968	0,458266
Spanyol-Inggris	0,453146	0,457280

Jika dibandingkan, maka skor BLEU pada masing-masing bahasa cenderung mirip, yakni di sekitar 45%. Hal ini berarti, dengan perbedaan bahasa, arsitektur model dan *hyperparameter* yang digunakan terbukti mampu untuk diimplementasikan pada *dataset* yang berbeda sekalipun.

KESIMPULAN

Model yang telah dikembangkan dengan penambahan *callbacks*, *layer dropout*, dan metode *cross validation* dapat mengatasi *overfitting* pada model dasar. Sehingga mendapatkan akurasi sebesar 71,95% dan skor BLEU sebesar 43,11% pada data *test*. Kemudian hasil dari simulasi *hyperparameter* antara lain mendapatkan ukuran dari masing-masing *hyperparameter* sebagai berikut, *batch size* (64), *epoch* (10), *optimizer* (*SGD*), *activation function* (*SoftMax*), dan *dropout rate* (0,2). Yang terakhir, implementasi model dan *hyperparameter* terbaik pada *dataset corpus parallel* Bahasa Spanyol-Inggris menghasilkan skor BLEU yang sangat mirip dengan *dataset corpus parallel* Bahasa Jerman-Inggris, yakni 0,453146 pada data *train* dan 0,457280 data *test*. Dengan hasil tersebut, masih terdapat beberapa hal yang dapat dilakukan untuk meningkatkan performa model, antara lain dengan menambahkan jumlah sampel *dataset* yang digunakan untuk pelatihan model, karena penelitian ini hanya menggunakan sebanyak 10% dari total baris *dataset corpus parallel* yang digunakan. Kemudian jika melakukan penambahan sampel, perlu diperhatikan pula kapasitas RAM yang sangat besar. Karena pada penelitian ini menggunakan *Google Colaboratory* yang mempunyai

kapasitas RAM terbatas. Disarankan juga untuk menggunakan lebih banyak kombinasi nilai pada *hyperparameter* dan juga melakukan simulasi secara simultan, karena penelitian ini hanya menggunakan masing-masing tiga nilai *hyperparameter* yang dilakukan secara parsial. Terakhir bisa dilakukan penambahan *layer* lain untuk meningkatkan performa model, misalnya *Flatten* dan *Dense*, karena penelitian ini hanya menggunakan dua *layer Dropout* untuk mengurangi *overfitting* pada model.

DAFTAR PUSTAKA

- Abidin, Z. (2017). Penerapan Neural Machine Translation untuk Eksperimen Penerjemahan secara Otomatis pada Bahasa Lampung – Indonesia. *Prosiding Seminar Nasional Metode Kuantitatif 2017*, 978, 53–68.
- Adji, T. B., Astuti (2011). Statistical-based machine translation for prepositional phrase using Link Grammar. *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011, July*. <https://doi.org/10.1109/ICEEI.2011.6021644>
- Asriningtias, S. R. (2015). Optimasi Training Neural Network Menggunakan Hybrid Adaptive Mutation. *Eecis*, 9(1), 79–84.
- Bahdanau, D. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.
- Belkin, M. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences of the United States of America*, 116(32), 15849–15854. <https://doi.org/10.1073/pnas.1903070116>
- Che, Z. (2011). *Feed-Forward Neural Networks Training : A Comparison Between Genetic Algorithm And Back-Propagation Learning Algorithm*. 7(10), 5839.
- Firmansyah, M. R. (2020). Klasifikasi Kalimat Ilmiah Menggunakan Recurrent Neural Network. *Prosiding The 11th Industrial Research Workshop and National Seminar*, 11(1), 488–495.
- Hadi, I. (2014). Uji Akurasi Mesin Penerjemah Statistik (MPS) Bahasa Indonesia Ke Bahasa Melayu Sambas Dan Mesin Penerjemah Statistik (MPS) Bahasa Melayu Sambas Ke Bahasa Indonesia. *Jurnal Sistem Dan Teknologi Informasi*, 2, 1–6.
- Hunston, S. (2002). *Corpora in Applied Linguistics*. Cambridge University Press. <https://doi.org/10.1017/CBO9781139524773>
- Hutter, J. N. (2018). Hyperparameter Importance Across Datasets. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2367–2376.
- Kulkarni, A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Data Democracy* (pp. 83–106). Elsevier. <https://doi.org/10.1016/B978-0-12-818366-3.00005-8>
- Kusrini, T. E. (2009). *Algoritma Data Mining* (T. A. Prabawati (ed.)). Penerbit Andi.
- Lample, G. (2016). Neural architectures for named entity recognition. *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, 260–270. <https://doi.org/10.18653/v1/n16-1030>
- Larose, D. (2005). *Discovering Knowledge in Data*. Wiley Interscience.
- Le, T. A. (2018). Application of a hybrid Bi-LSTM-CRF Model to the task of Russian named entity recognition. *Communications in Computer and Information Science*, 789, 91–103. https://doi.org/10.1007/978-3-319-71746-3_8
- Liu, P. (2016). Recurrent neural network for text classification with multi-task learning. *IJCAI International Joint Conference on Artificial Intelligence, 2016-Janua*, 2873–2879.
- Liu, X. (2020). *Very Deep Transformers for Neural Machine Translation*. <http://arxiv.org/abs/2008.07772>
- McEnery, T. (2006). *Corpus Based Language Studies: An Advanced Resource Book*. Oxon: Routledge.
- Nugroho Aditya, R. (2015). Penerjemahan Bahasa Indonesia dan Bahasa Jawa Menggunakan Metode Statistik Berbasis Frasa. *Seminar Nasional Teknologi Informasi Dan Komunikasi, 2015(Sentika)*.
- Nurhikmat, T. (2018). Implementasi Deep Learning untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) pada Citra Wayang Golek. *Universitas Islam Indonesia*.
- Papineni, K. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. <https://doi.org/10.1002/andp.19223712302>
- Powers, D. M. W. (2007). *Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation*. December.
- Pranata, J., & Muljono. (2016). *Mesin Penerjemah Bahasa Indonesia-Jawa Menggunakan Metode Statistik Berbasis Frasa*. 5, 1–5. <http://eprints.dinus.ac.id/18206/>
- Rohim, A. (2019). Convolution neural network (cnn) untuk pengklasifikasian citra makanan tradisional.

- Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 3(7), 7038–7042.
- Sennrich, R. (2016). Improving neural machine translation models with monolingual data. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 1, 86–96. <https://doi.org/10.18653/v1/p16-1009>
- Septiani, A. (2021). *Klasifikasi Suara Paru Normal dan Abnormal dengan Menggunakan Discrete Wavelet Transform dan Support Vector Machine*. 8(1), 731–742.
- Shukla, A. (2010). *Real Life Applications of Soft Computing*. CRC Press.
- Sujaini, H. (2020). Peningkatan Akurasi Mesin Penerjemah Bahasa Inggris - Indonesia dengan Memaksimalkan Kualitas dan Kuantitas Korpus Paralel. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 7(3), 471. <https://doi.org/10.25126/jtiik.2020732076>
- Xia, W. (2018). Novel architecture for long short-term memory used in question classification. *Neurocomputing*, 299, 20–31. <https://doi.org/10.1016/j.neucom.2018.03.020>
- Yahya, M. B. (2015). *Penerapan Metode Interpolation Search Untuk Melakukan Pencarian Pada Aplikasi Translator Huruf Pegon Jawa – Bahasa Indonesia*. Universitas Islam Negeri Maulana Malik Ibrahim.