KETERANGAN GAMBAR OTOMATIS BERBAHASA INDONESIA MENGGUNAKAN CNN DAN LSTM DENGAN *ATTENTION*

Disusun Oleh:

Nama : Amiin Majiid Nugroho

NIM : 17523105

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

KETERANGAN GAMBAR OTOMATIS BERBAHASA INDONESIA MENGGUNAKAN CNN DAN LSTM DENGAN *ATTENTION*

TUGAS AKHIR

Disusun Oleh:

Nama : Amiin Majiid Nugroho

NIM : 17523105

Yogyakarta, 30 Juli 2021

Pembimbing,

(Ahmad Fathan Hidayatullah, S.T., M.Cs)

HALAMAN PENGESAHAN DOSEN PENGUJI

KETERANGAN GAMBAR OTOMATIS BERBAHASA INDONESIA MENGGUNAKAN CNN DAN LSTM DENGAN *ATTENTION*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 30 Juli 2021

Tim Penguji

Ahmad Fathan Hidayatullah, S.T., M.Cs.

Anggota 1

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.

Anggota 2

Septia Rani, S.T., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Amiin Majiid Nugroho
NIM	:	17523105

Tugas akhir dengan judul:

KETERANGAN GAMBAR OTOMATIS BERBAHASA INDONESIA MENGGUNAKAN CNN DAN LSTM DENGAN *ATTENTION*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apa pun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 30 Juli 2021

(Amiin Majiid Nugroho)

HALAMAN PERSEMBAHAN

Kepada kedua orang tua, saya ucapkan terima kasih yang sebesar-besarnya karena telah mendidik saya dengan sangat baik sampai saat ini dan terima kasih juga atas segala doa, dukungan, dan nasehat yang telah diberikan kepada saya selama pengerjaan laporan skripsi ini. Semoga selalu diberi rahmat berupa kesehatan oleh Allah *subhanahu wata 'ala.*

Terima kasih kepada Bapak Ahmad Fathan Hidayatullah, S.T., M.Cs. selaku dosen pembimbing, yang selama ini memberikan bimbingan dengan sangat sabar dan motivasi tinggi sehingga skripsi ini bisa terselesaikan.

HALAMAN MOTO

“Bersama kesulitan itu ada kemudahan”
(Q.S. Al Insyirah ayat 6)

“Pendidikan memiliki akar yang pahit, tapi buahnya manis”
(Aristoteles)

“Dengan ilmu kita menuju kemuliaan”
(Ki Hajar Dewantara)

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

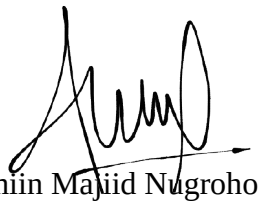
Allhamdulillahirobbil'alamin, segala puji bagi Allah *Subhanahu wata'ala* yang telah memberikan rahmat dan hidayah-Nya yang sangat banyak serta kemudahan kepada penulis sehingga penulis mampu menyelesaikan laporan skripsi dengan lancar. Shalawat serta salam tak lupa penulis panjatkan kepada junjungan Nabi Muhammad *Sallallahu 'alaihi wasallam* yang telah menuntun umat manusia menuju jalan yang diterangi yaitu jalan Islam.

Pada proses penulisan laporan tugas akhir ini, penulis mendapatkan banyak dukungan, bimbingan, dan motivasi dari banyak pihak. Oleh karena itu penulis hendak memberikan apresiasi dan terima kasih banyak kepada:

1. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia
2. Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Kaprodi Informatika UII
3. Ahmad Fathan Hidayatullah, S.T., M.Cs. selaku Dosen Pembimbing yang telah membimbing, memberikan banyak saran, masukan, serta dukungan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
4. Bapak dan Ibu Dosen Informatika UII, yang telah memberikan ilmu yang sangat bermanfaat bagi penulis.
5. Kedua orang tua penulis, yang telah senantiasa mendoakan, mendukung, dan memberikan motivasi kepada penulis.
6. Seluruh keluarga besar yang telah memberikan dukungan dan doa.
7. Teman-teman Halgib, yang selalu memberikan dukungan berupa motivasi yang sangat positif.
8. Semua orang yang telah mendoakan penulis.

Penulis sadar tugas akhir ini masih jauh dari kata sempurna, namun penulis berharap dengan diselesaikannya laporan tugas akhir ini bisa bermanfaat bagi banyak orang dan bisa dikembangkan menjadi hal yang lebih baik lagi.

Yogyakarta, 30 Juli 2021



(Amiin Majiid Nugroho)

SARI

Image captioning adalah proses untuk menghasilkan satu kalimat atau lebih untuk menjelaskan konten visual dari suatu gambar. *Image Captioning* bermanfaat untuk membantu kegiatan manusia memahami konten visual seperti keterangan pada citra medis, interaksi manusia dengan robot dan membantu mendeskripsikan gambar kepada tunanetra. Untuk menghasilkan deskripsi dari suatu gambar, diperlukan gabungan dua bidang ilmu, yaitu *computer vision* dan *natural language processing*. Tujuan dari penelitian ini adalah untuk menghasilkan deskripsi gambar (*caption*) berbahasa Indonesia. Untuk mencapai tujuan penelitian, peneliti menggunakan lima metodologi dalam penelitian yaitu pengumpulan data, *preprocessing*, pembuatan model, evaluasi, dan eksperimen. Penelitian dilakukan dengan menggunakan *dataset* gambar dari MS COCO. Metode yang digunakan dalam penelitian adalah CNN dan LSTM dengan *attention* untuk membangun model. Dalam penelitian ini, didapatkan dua model dengan ekstraksi fitur yang berbeda, yaitu Model 1 dengan ekstraksi fitur InceptionV3 dan Model 2 dengan ekstraksi fitur VGG16. *Caption* yang dihasilkan dievaluasi dengan skor BLEU dan pada Model 1 mampu memperoleh skor BLEU {1, 2, 3, 4} sebesar {34.93, 22.55, 14.75, 10.15} sedangkan untuk Model 2 memiliki skor {22.02, 14.29, 9.41, 6.28}. Model yang dibangun dengan LSTM cenderung lebih baik daripada model yang dibangun dengan GRU.

Kata kunci: *image captioning, CNN, LSTM*.

GLOSARIUM

Dataset	kumpulan data yang digunakan dalam pembuatan model
Epoch	banyaknya tahapan pelatihan.
Training set	data yang digunakan untuk melatih model
Validation set	data yang digunakan untuk proses validasi model

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI viii
GLOSARIUM ix
DAFTAR ISI x
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan Penelitian 2
1.4 Batasan Penelitian 2
1.5 Manfaat Penelitian 3
1.6 Sistematika Penulisan 3
BAB II LANDASAN TEORI 4
2.1 Dasar Teori 4
2.1.1 *Image Captioning* 4
2.1.2 *Convolutional Neural Network* (CNN) 4
2.1.3 InceptionV3 7
2.1.4 VGG16 7
2.1.5 *Long Short-Term Memory* (LSTM) 8
2.1.6 *Gated Recurrent Unit (GRU)* 9
2.1.7 *Attention Mechanism* 9
2.1.8 MS COCO 10
2.1.9 BLEU 11
2.1.10 Streamlit 12

2.2 Penelitian Sebelumnya 12
BAB III METODOLOGI PENELITIAN 15
3.1 Langkah – langkah Penelitian 15
3.2 Uraian Penelitian 16
3.2.1 Pengumpulan Data 16
3.2.2 *Preprocessing* 17
3.2.3 Pembuatan Model 18
3.2.4 Evaluasi 20
3.2.5 Eksperimen 20
3.2.6 Pembuatan *User Interface* 20
BAB IV HASIL DAN PEMBAHASAN 21
4.1 Pengumpulan Data 21
4.2 Preprocessing 22
4.2.1 *Preprocessing* Gambar 22
4.2.2 Inisialisasi Bobot ImageNet 22
4.2.3 Ekstraksi Fitur 23
4.2.4 *Preprocessing Caption* 23
4.3 Pembuatan Model 24
4.3.1 Konfigurasi Pembuatan Model 24
4.3.2 *Training Data* 28
4.4 Evaluasi 29
4.4.1 Hasil 29
4.4.2 Evaluasi dengan Skor BLEU 30
4.5 Eksperimen 36
4.6 Pembuatan User Interface 40
BAB V KESIMPULAN 42
5.1 Kesimpulan 42
5.2 Saran 42
DAFTAR PUSTAKA 43
LAMPIRAN 46

DAFTAR TABEL

Tabel 2.1 Daftar perbandingan penelitian yang dilakukan sebelumnya 13
Tabel 3.1 *Caption* setelah tanda baca dihilangkan..... 17
Tabel 3.2 Tokenisasi per kata 17
Tabel 3.3 Skenario pada Model 1 19
Tabel 3.4 Skenario pada Model 2 19
Tabel 4.1 Nilai *loss* dari masing-masing model..... 29
Tabel 4.2 Hasil *generate caption* pada Model 1 32
Tabel 4.3 Hasil *generate caption* pada Model 2 34
Tabel 4.4 Perbandingan skor BLEU pada masing-masing model 36
Tabel 4.5 Eksperimen dengan Model 1 37
Tabel 4.6 Eksperimen dengan Model 2 38

DAFTAR GAMBAR

Gambar 2.1 Contoh *image captioning* 4
Gambar 2.2 Arsitektur CNN 5
Gambar 2.3 Operasi konvolusi 5
Gambar 2.4 Ilustrasi *pooling* 6
Gambar 2.5 Ilustrasi *Fully Connected Layer* 6
Gambar 2.6 Arsitektur InceptionV3 7
Gambar 2.7 Arsitektur VGG16 8
Gambar 2.8 Arsitektur LSTM 8
Gambar 2.9 Arsitektur GRU 9
Gambar 2.10 *Attention* pada translasi bahasa 10
Gambar 2.11 Data gambar pada *dataset* MS COCO 10
Gambar 2.12 *Caption* gambar pada *dataset* MS COCO 11
Gambar 2.13 Contoh penerapan streamlit 12
Gambar 3.1 Alur Penelitian 15
Gambar 3.2 Data gambar pada MS COCO 16
Gambar 3.3 Alur model untuk *image captioning* 18
Gambar 3.4 Alur eksperimen 20
Gambar 4.1 Gambar yang digunakan dalam penelitian 21
Gambar 4.2 *Caption* yang telah dibuat 21
Gambar 4.3 *Preprocessing* gambar dengan InceptionV3 22
Gambar 4.4 *Preprocessing* gambar dengan VGG16 22
Gambar 4.5 Inisialisasi bobot yang ada di ImageNet 22
Gambar 4.6 Tahap ekstraksi fitur 23
Gambar 4.7 *Preprocessing* pada *caption* 23
Gambar 4.8 Kata setelah tokenisasi 24
Gambar 4.9 Jumlah setiap kata dari *caption* 24
Gambar 4.10 Melakukan *splitting dataset* 24
Gambar 4.11 Konfigurasi Model 1 25
Gambar 4.12 Konfigurasi Model 2 25
Gambar 4.13 Model dengan *attention mechanism* 26
Gambar 4.14 *Encoder* dengan CNN 26

Gambar 4.15 *Decoder* dengan LSTM.....27
Gambar 4.16 Konfigurasi saat melakukan *training* data 28
Gambar 4.17 Grafik *loss* dari Model 1 dan Model 2 dengan LSTM.....29
Gambar 4.18 Grafik *loss* dengan GRU 30
Gambar 4.19 Kode untuk menghitung *sentence* pada BLEU 31
Gambar 4.20 Perhitungan skor individu 1-gram skor BLEU 31
Gambar 4.21 Perhitungan skor kumulatif 4-gram skor BLEU 31
Gambar 4.22 Membuat fungsi *upload image* 40
Gambar 4.23 Membuat tampilan dan memanggil model.....40
.Gambar 4.24 Tampilan *User Interface* 41

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, pemrosesan gambar telah membuat kemajuan yang signifikan, seperti klasifikasi gambar dan deteksi objek. Manfaat dari kemajuan pada bidang klasifikasi gambar dan deteksi objek menjadi memungkinkan untuk secara otomatis menghasilkan satu kalimat atau lebih untuk menjelaskan konten visual dari suatu gambar, yang dikenal sebagai *Image Captioning*. Membuat deskripsi gambar yang lengkap dan alami secara otomatis memiliki manfaat yang besar, seperti pada pembuatan judul yang dilampirkan pada gambar berita, deskripsi yang terkait dengan gambar medis, pengambilan gambar berbasis teks, informasi yang diakses oleh pengguna tunanetra, sampai interaksi manusia dengan robot (Liu, Bai, Hu, & Wang, 2018).

Menghasilkan deskripsi dengan bahasa alami yang memiliki makna dari sebuah gambar membutuhkan tingkat pemahaman yang lebih tinggi dari klasifikasi dan deteksi gambar. Permasalahan tersebut sangat menarik karena menghubungkan antara *Computer Vision* dan *Natural Language Processing* yang merupakan dua bidang utama dalam *Artificial Intelligence* (You et al., 2016).

Penelitian tentang *image captioning* telah dilakukan sebelumnya dengan berbagai macam metode. Beberapa metode yang populer digunakan adalah *Convolutional Neural Network* (CNN) dengan *Recurrent Neural Network* (RNN). Oleh karena itu penelitian ini dilakukan dengan menggunakan metode (CNN) dan *Long Short-Term Memory* (LSTM) yang merupakan pengembangan dari RNN dengan penambahan *attention mechanism*. CNN digunakan untuk melakukan *encoding* pada gambar sedangkan LSTM yang merupakan sebuah jaringan syaraf berulang akan digunakan untuk *generate caption* (Pu et al., 2016). Tahapan penelitian ini dibagi menjadi lima tahapan, mulai dari pengumpulan data, lalu *preprocessing,* kemudian pembuatan model, setelah itu adalah evaluasi, dan yang terakhir adalah eksperimen.

Penelitian dilaksanakan karena melihat pentingnya *image captioning* pada saat ini dan pada masa yang akan datang. Hasil yang diharapkan dari penelitian ini adalah pembuatan keterangan dari sebuah gambar secara otomatis dengan hasil berupa deskripsi gambar berbahasa Indonesia dengan tata bahasa yang benar dan senatural mungkin seperti deskripsi buatan manusia. Dengan dikembangkannya penelitian ini, diharapkan mampu untuk membantu pengembangan aplikasi yang lebih adaptif seperti pembuatan aplikasi bagi pihak kepolisian

untuk mendeskripsikan kondisi lalu lintas, membantu para dokter untuk mendeskripsikan citra medis dengan lebih rinci, dan membantu interaksi antara manusia dengan robot seperti untuk pembuatan *chatbot*.

1.2 Rumusan Masalah

Rumusan masalah berdasarkan latar belakang tersebut adalah:

a. Bagaimana cara membuat deskripsi gambar secara otomatis berbahasa Indonesia menggunakan CNN dan LSTM dengan *attention*?
b. Bagaimana performa CNN dan LSTM dengan *attention* dalam menghasilkan *caption* Berbahasa Indonesia?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

a. Membuat keterangan gambar (*caption*) berbahasa Indonesia secara otomatis.
b. Mengidentifikasi seberapa baik metode CNN dan LSTM dengan *attention* dalam melakukan *image captioning* berbahasa Indonesia.

1.4 Batasan Penelitian

Adapun lingkup dari penelitian ini adalah:

a. Deskripsi atau *caption* yang dihasilkan hanya berasal dari Bahasa Indonesia.
b. *Dataset* yang digunakan berasal dari MS COCO.
c. Setiap *dataset* memiliki satu *caption.*
d. Gambar yang digunakan dalam penelitian ini meliputi:
    1. Kegiatan olahraga: bisbol, tenis, skateboard, selancar, frisbee, papan ski salju, sepak bola.
    2. Kendaraan dan lalu lintas: sepeda, motor, mobil, bus, truk, kereta, pesawat, kapal, rambu lalu lintas.
    3. Ruangan dan bangunan: toilet, kamar tidur, ruang tamu, ruang TV, menara.
    4. Binatang: kucing, anjing, jerapah, gajah, domba, zebra, sapi, bison.
    5. Tempat: gunung salju, rerumputan, laut, pantai, pepohonan, semak-semak.
    6. Makanan: pizza, olahan makanan sayur, roti.
    7. Lain-lain: menelepon, membawa payung, berjalan-jalan.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah untuk mendapatkan keterangan (*caption)* dari sebuah gambar supaya mempermudah dan memperjelas informasi yang ada pada suatu gambar.

1.6 Sistematika Penulisan

Laporan tugas akhir ini disusun atas lima bagian, yang terdiri dari:

BAB I Pendahuluan, membahas latar belakang permasalahan, rumusan masalah, tujuan penelitian, batasan penelitian, metodologi penelitian, dan sistematika penulisan yang dilakukan dalam penelitian.

BAB II Landasan Teori, menjelaskan tentang dasar teori yang berkaitan dengan *image captioning* serta menampilkan penelitian terdahulu.

BAB III Metodologi Penelitian, membahas mengenai langkah-langkah yang dilakukan dalam penelitian tentang *image captioning* berbahasa Indonesia.

BAB IV Hasil dan Pembahasan, menampilkan dan menjelaskan hasil dari penelitian tentang *image captioning* berbahasa Indonesia dengan metode CNN dan LSTM dengan *attention*, mulai dari pengolahan data, pembuatan model, serta menampilkan hasil eksperimen yang dilakukan.

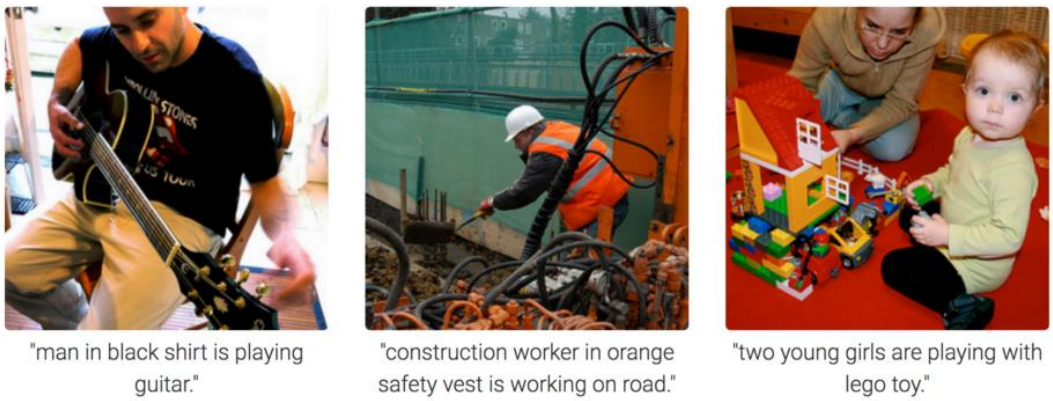
BAB V Kesimpulan Dan Saran, menjelaskan tentang kesimpulan dari penelitian ini serta memberikan saran untuk penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

2.1.1 *Image Captioning*

Image captioning merupakan teks yang diberikan pada suatu gambar dalam buku, surat kabar, maupun majalah yang isi tulisannya menjelaskan citra tersebut (Nursikuwagus, Munir, & Khodra, 2020). *Image captioning* bekerja dengan cara menggabungkan dua cabang utama dalam *Artificial Intelligence* yaitu *Computer Vision* dengan *Natural Language Processing* (You et al., 2016). Gambar 2.1 merupakan ilustrasi mengenai *image captioning.*

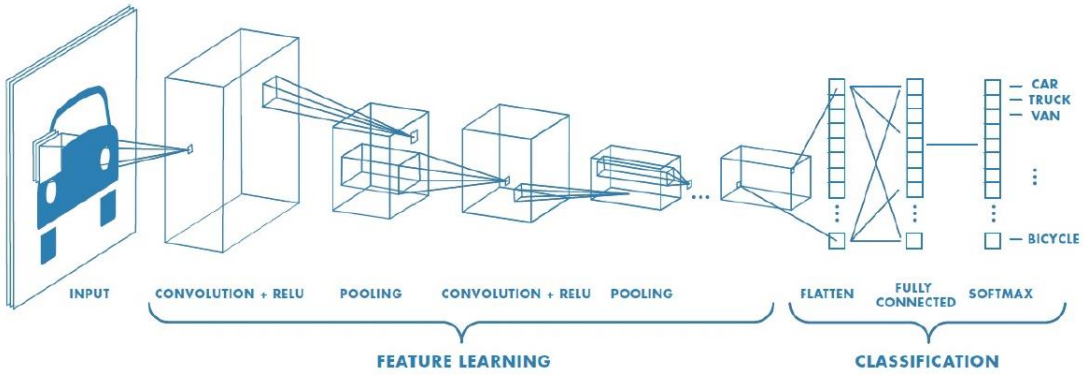


Gambar 2.1 Contoh *image captioning*

2.1.2 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan jaringan syaraf tiruan *feed-forward* di mana jaringan syaraf tersebut mempelajari struktur hierarki dengan mempelajari representasi fitur internal dan menggeneralisasi fitur dalam sebuah gambar secara umum seperti pengenalan objek dan permasalahan *computer vision* lainnya (Manaswi, 2018). Arsitektur dari CNN dapat dilihat pada Gambar 2.2.

Beberapa tahun terakhir telah melihat banyak arsitektur CNN yang dikembangkan dan membuat kemajuan luar biasa di bidang klasifikasi gambar. Jaringan *pre-trained* VGG16, VGG19, ResNet50, InceptionV3, dan Xception telah digunakan untuk berbagai tantangan klasifikasi gambar termasuk pencitraan medis (Manaswi, 2018).

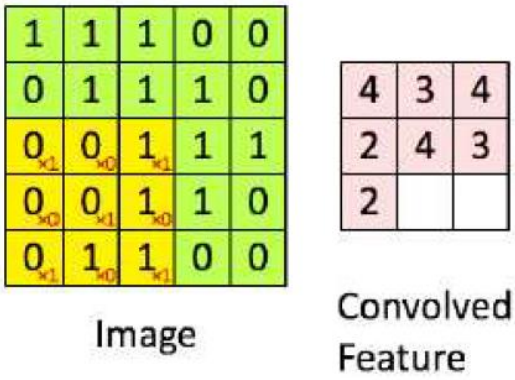


Gambar 2.2 Arsitektur CNN

Sumber: (Putra, 2019)

A. *Convolution Layer*

Lapisan Konvolusi (*Convolution layer)* pada CNN memiliki tugas untuk mengurangi kompleksitas perhitungan sesuai dengan prinsip *sliding window* dan *weight sharing* (Putra, 2019). Lapisan konvolusi terdiri dari filter dan peta gambar (Manaswi, 2018). Operasi konvolusi dapat diilustrasikan pada Gambar 2.3.

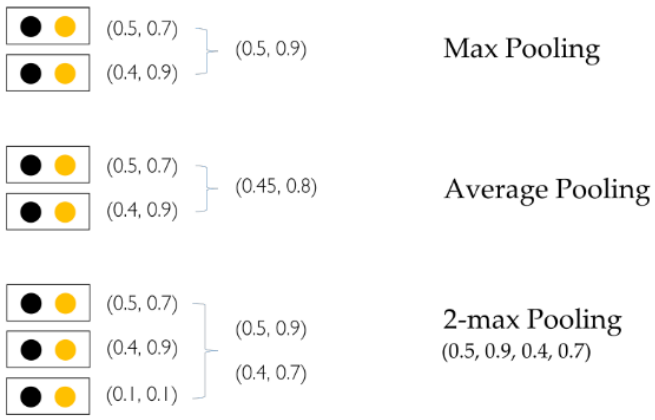


Gambar 2.3 Operasi konvolusi

Sumber: (Eka Putra, 2016)

B. *Pooling Layer*

Pooling layer adalah lapisan yang berfungsi untuk menampung suatu informasi yang informatif setelah melalui proses konvolusi sebelumnya (Putra, 2019), proses ini juga mereduksi ukuran data dari citra (Rohim, Sari, & Tibyani, 2019). Gambar 2.4 mengilustrasikan tentang *max pooling* yaitu mengambil nilai maksimal dari proses *pooling* dan *average pooling* yaitu mencari nilai rata-rata saat *pooling*.

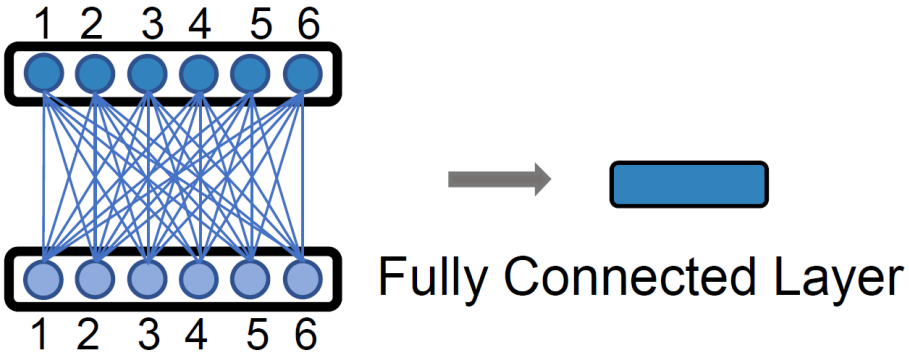


Gambar 2.4 Ilustrasi *pooling*

Sumber: (Putra, 2019)

C. *Fully Connected Layer*

Fully Connected Layer adalah lapisan jaringan syaraf tiruan *feed-forward*. Lapisan ini memiliki fungsi aktivasi nonlinier untuk menghasilkan probabilitas prediksi kelas. Lapisan ini digunakan menjelang akhir setelah semua fitur diidentifikasi dan diekstraksi oleh lapisan konvolusi dan telah dikonsolidasikan oleh lapisan penyatuan dalam jaringan. Di sini, *hidden layer* dan *output layer* adalah *fully connected layer* (Manaswi, 2018). Gambar 2.5 merupakan ilustrasi dari *fully connected layer*.



Gambar 2.5 Ilustrasi *Fully Connected Layer*

Sumber: (Zeng et al., 2020)

D. Fungsi Aktivasi ReLU

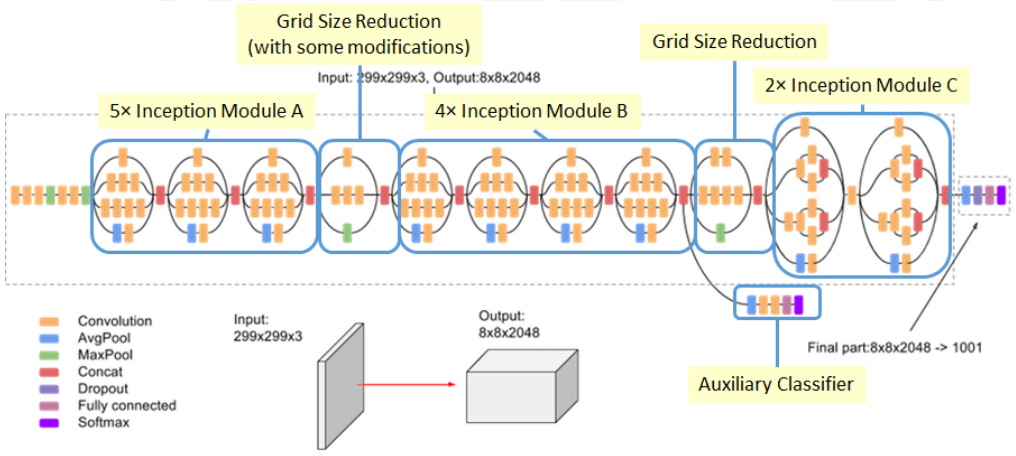
ReLU (*Rectification Linear Unit*) adalah fungsi untuk meningkatkan nilai representasi model dan mengenalkan non-linieritas (Zeng et al., 2020). ReLU secara konvensional digunakan sebagai fungsi aktivasi pada jaringan syaraf (Agarap, 2018).

E. Fungsi Aktivasi Softmax

Fungsi aktivasi *softmax* merupakan sebuah fungsi aktivasi yang digunakan untuk mendapatkan hasil dari sebuah kalsifikasi (Zeng et al., 2020).

2.1.3 InceptionV3

InceptionV3 merupakan penerus dari generasi Inception sebelumnya. Arsitektur ini memiliki 24 juta parameter. Konvolusi di arsitektur ini menggunakan ReLu sebagai aktivasi. Selain itu arsitektur ini juga memiliki jumlah layer yang cukup banyak yaitu 48 layer. InceptionV3 memiliki *input image size* 299x299 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016). Ilustrasi mengenai arsitektur InceptionV3 dapat dilihat pada Gambar 2.6.

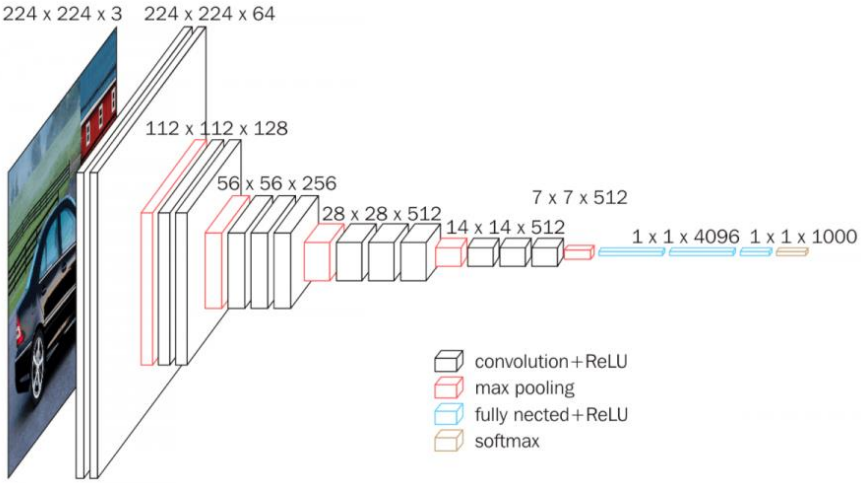


Gambar 2.6 Arsitektur InceptionV3

Sumber: (https://paperswithcode.com/method/inception-v3#)

2.1.4 VGG16

Visual Geometry Group (VGG) merupakan sebuah model yang diciptakan oleh Oxford. Model ini memiliki 2 jenis yaitu VGG16 dan VGG19, penggunaan nama dengan angka yang berbeda itu didasarkan pada kedalaman lapisan (*layer*) yaitu 16 dan 19 dan masing-masing memiliki 3 *fully connected layer* (Gollapudi, 2019). VGG16 memiliki ukuran konvolusi kecil yaitu 3x3 dengan *stride* 1, yang diikuti oleh beberapa *non-linear layer* (Mahdianpari, Salehi, Rezaee, Mohammadimanesh, & Zhang, 2018). VGG16 yang dikombinasikan dengan LSTM memiliki performa yang cukup baik dalam *corpus* yang kecil dan mampu menghasilkan deskripsi yang bisa merepresentasikan gambar (Pa Pa Aung, Pa Pa, & Nwe, 2020). Ilustrasi dari arsitektur VGG16 dapat dilihat pada Gambar 2.7.

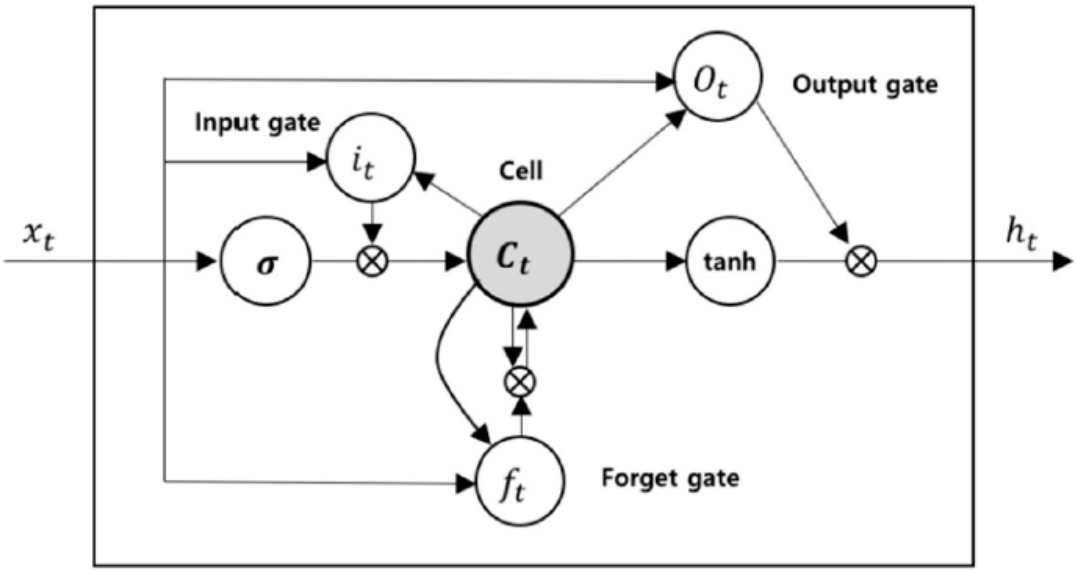


Gambar 2.7 Arsitektur VGG16

Sumber: (https://neurohive.io/en/popular-networks/vgg16/)

2.1.5 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) adalah arsitektur dari *Recurrent Neural Network* (RNN) yang telah dimodifikasi sedemikian rupa untuk mengatasi penyimpanan memori dalam jangka waktu yang lama karena telah menambahkan *memory cell*, LSTM dapat memecahkan masalah seperti terjadinya *vanishing gradient* (Manaswi, 2018).



Gambar 2.8 Arsitektur LSTM

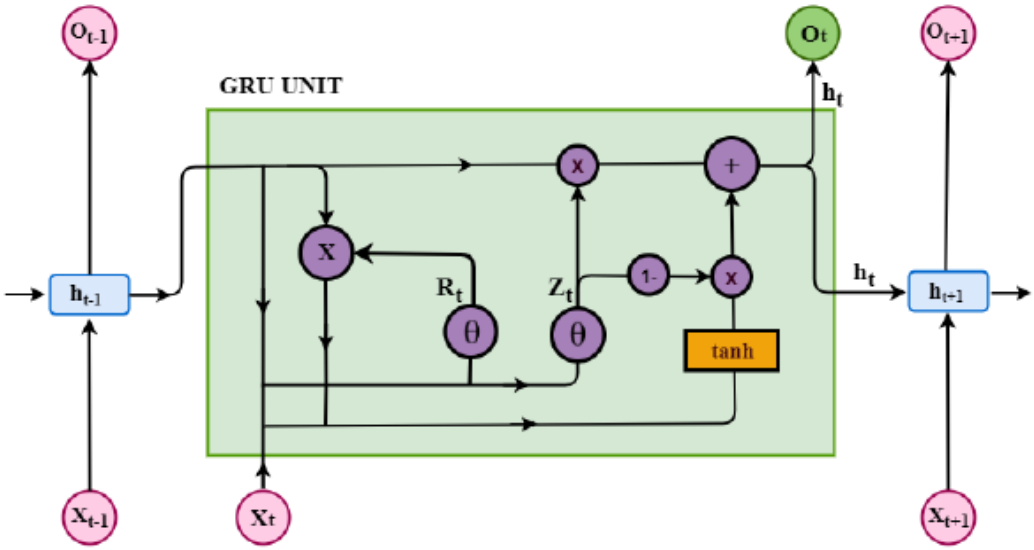
(H. Chung & Shin, 2018)

LSTM memiliki *memory cell dan* tiga unit *gate* seperti yang ditampilkan pada Gambar 2.8 yaitu *forget gate*, *input gate*, dan *output gate*. Ketiga *gate* tersebut memiliki fungsi masing-

masing. *Input gate* berfungsi untuk memutuskan suatu nilai *input* apakah akan diteruskan menuju *cell state* dan layak diperbaharui atau tidak. *Forget gate* berfungsi untuk memutuskan tentang memori mana yang perlu dilupakan atau dibuang dari *cell state*. *Output gate* berfungsi untuk memutuskan *output* yang dihasilkan.

2.1.6 *Gated Recurrent Unit (GRU)*

Gated Recurrent Unit (GRU) diperkenalkan oleh (J. Chung, Gulcehre, Cho, & Bengio, 2014). GRU sendiri merupakan salah satu arsitektur dari RNN. Tujuan utama dari GRU adalah membuat tiap *recurrent unit* mampu untuk menangkap sebuah *dependencies* dengan adaptif pada suatu skala waktu tertentu. Berbeda dengan LSTM, GRU memiliki 2 *gates* yaitu *reset gate* dan *update gates*. Setiap *gates* memiliku fungsi yang berbeda, *reset gates* berfungsi untuk memutuskan berapa banyak informasi pada masa lalu yang sudah tidak perlu dan kemudian dilupakan. *Update Gates* berfungsi untuk memutuskan informasi apa yang harus dibuang dan informasi mana yang harus ditambahkan. Arsitektur GRU dapat dilihat pada Gambar 2.9.



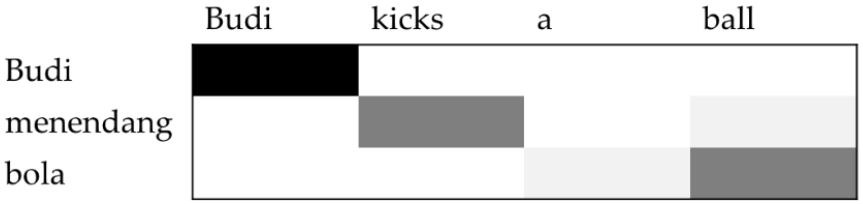
Gambar 2.9 Arsitektur GRU

Sumber: (Bibi et al., 2020)

2.1.7 *Attention Mechanism*

Attention mechanism adalah salah satu cara untuk memahami *neural network*. Hal ini dikarenakan *attention mechanism* mampu memprediksi *soft aligment* atau sebuah kata yang akan difokuskan saat akan memasukkan *input* informasi bahasa dengan menggunakan bobot

tertinggi. Mekanisme yang dimiliki oleh *attention mechanism* dapat mengatasi *input* yang panjang (Putra, 2019).



Gambar 2.10 *Attention* pada translasi bahasa

Sumber: (Putra, 2019)

Gambar 2.10 menunjukkan penerapan *attention mechanism* pada translasi bahasa. Gambar yang menunjukkan warna yang lebih gelap merepresentasikan sebuah bobot/titik fokus (*attention*) yang lebih tinggi.

2.1.8 MS COCO

Data yang digunakan berasal dari *dataset* MS COCO (*Microsoft COCO: Common Objects in Context*). MS COCO sendiri menyediakan ragam gambar yang cukup kompleks, objek di dalamnya dilabeli menggunakan segmentasi *per-instance*. *Dataset* berisi 91 objek dengan total 2,5 juta label berupa deskripsi gambar dengan Bahasa Inggris dalam 328.000 gambar (Lin et al., 2014). Contoh gambar yang ada di dalam *dataset* MS COCO dapat dilihat pada Gambar 2.12 dan contoh *caption* dapat dilihat pada Gambar 2.12.

Gambar 2.11 Data gambar pada *dataset* MS COCO

```
{"image_id": 410878,"id": 271860,"caption": "A motorcycle parked on a gravel road near a park."}
{"image_id": 477441,"id": 272042,"caption": "An airplane pulled up to a gate on the tarmac."},
{"image_id": 209613,"id": 272065,"caption": "A group of sheep surrounded by three dogs."},
{"image_id": 44279,"id": 272165,"caption": "A man in a kitchen pouring something from a pot."},
{"image_id": 103585,"id": 272212,"caption": "A picture of double sinks in a bathroom."},
{"image_id": 459467,"id": 272384,"caption": "A small wheeled airplane on an open runway."},
```


Gambar 2.12 *Caption* gambar pada *dataset* MS COCO

2.1.9 BLEU

BLEU *(Bilingual Evaluation Understudy)* adalah metode untuk evaluasi otomatis terjemahan mesin (Papineni, Roukos, Ward, Zhu, & Heights, 2001). Cara kerja skor BLEU adalah dengan membandingkan n-gram dari kandidat kalimat yang dihasilkan mesin dengan n-gram kalimat referensi, 1-gram atau *unigram* akan menjadi token dan *bigram* merupakan sebuah pasangan kata (Brownlee, 2017). Skor BLEU memiliki rentan nilai dari 0-1 yang mengindikasikan bahwa semakin mendekati angka 1 maka semakin akurat hasil *caption* tersebut, pada beberapa penelitian menggunakan skor BLEU yang dikalikan 100 (skala 0-100).

Precision pada BLEU dihitung dengan menghitung jumlah kata yang ada pada *candidate* yang cocok dengan *reference* kemudian dibagi dengan jumlah total kata pada *unigram* yang ada pada hasil *candidate. Brevity Penalty* pada skor BLEU memiliki tugas untuk melakukan perhitungan agar kalimat yang pendek pada hasil *candidate* tidak memiliki skor yang tinggi dengan rumus pada persamaan (2.1). Kemudian skor BLEU diperoleh dari hasil perkalian antara *Brevity Penalty* dengan hasil rata-rata geometrik *modified precision score* dapat dilihat pada persamaan (2.2) (Papineni et al., 2001).

Rumus menghitung skor BLEU adalah sebagai berikut:

$$BP = \begin{cases} 1, & if\ c > r \\ exp^{\left(1-\frac{r}{c}\right)}, & if\ c \leq r \end{cases} \quad (2.1)$$

$$BLEU = BP.exp\left(\sum_{n-1}^{n} w_n \log\ p_n\right) \quad (2.2)$$

Keterangan:

BP = *Brevity Penalty*

c = jumlah kata dari hasil terjemahan otomatis (*candidate*)

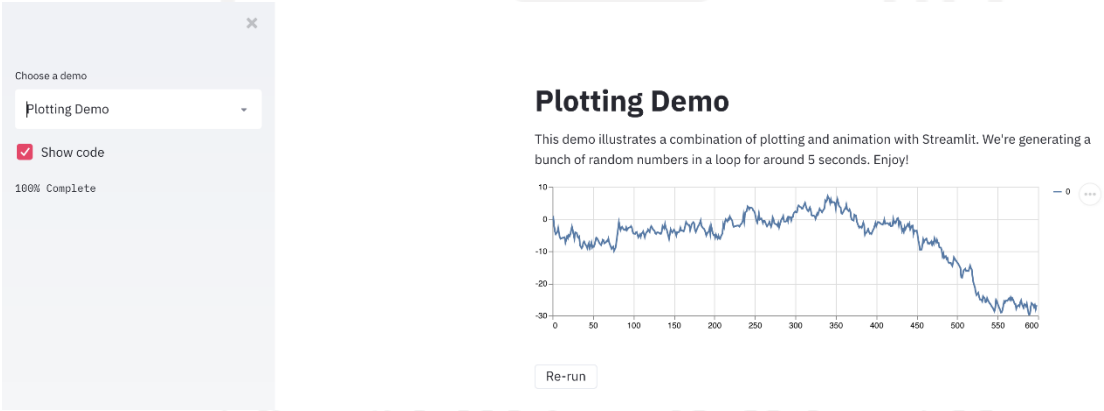
r = jumlah kata rujukan (*reference)*

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

p_n = jumlah n-gram hasil terjemahan yang sesuai dengan rujukan dibagi jumlah n-gram hasil terjemahan.

2.1.10 Streamlit

Streamlit merupakan *framework open-source* yang biasanya digunakan untuk membuat *interface* pada data sains dan *machine learning.* Penggunaan streamlit cukup mudah dikarenakan hanya perlu menggunakan bahasa pemrograman *Python.* Fitur seperti *slider, selectbox, checkkbox, text input* dan fitur lainnya membuat streamlit sangat baik dan cukup lengkap untuk melakukan kontrol terhadap web yang dibuat (Hidayatullah, 2021). Gambar 2.13 merupakan tampilan streamlit saat digunakan untuk membangun *intterface* sebuah program.



Gambar 2.13 Contoh penerapan streamlit

Sumber: (Hidayatullah, 2021)

2.2 Penelitian Sebelumnya

Penelitian mengenai pembuatan keterangan gambar (*image captioning*) sudah dilakukan oleh beberapa peneliti sebelumnya. Penelitian yang dilakukan (Al-muzaini, Al-yahya, & Benhidour, 2018) menggunakan MS COCO dan Flickr *dataset* yang diberi *caption* berbahasa Arab mampu menghasilkan keterangan gambar sesuai dengan gambar dan menghasilkan skor

BLEU yang lebih baik dari penelitian sebelumnya. Penelitian tersebut memperoleh skor 46.1 pada BLEU-1. Metode yang digunakan adalah gabungan antara RNN-CNN.

Penelitian tentang *image captioning* dilakukan oleh (Gu, Wang, Cai, & Chen, 2017) menggunakan metode CNN-GRU dan CNN-LSTM. *Dataset* yang digunakan pada penelitian tersebut adalah MS COCO dan flickr *dataset* dan mampu menghasilkan keluaran yang cukup baik. Skor BLEU yang didapat dengan kombinasi CNN-LSTM adalah 64.5 dan CNN-GRU menghasilkan skor BLEU-1 sebesar 71.4

Penelitian yang menggunakan gabungan metode CNN-LSTM oleh (Tan, Chan, & Chuah, 2019). Penelitian tersebut mampu mencapai tingkat *sparsity* tanpa kehilangan kinerja terhadap *baseline.* BLEU yang didapatkan pada penelitian yang menggunakan *dataset* dari MS COCO ini mencapai 73.8 pada BLEU-1. Metode lain yang digunakan adalah GRU dan memperoleh skor BLEU-1 sebesar 73.4.

Penggunaan *attention* dilakukan pada penelitian (You et al., 2016). Penelitian tersebut menggunakan *dataset* dari MS COCO. Hasil yang diperoleh mampu melampaui penelitian sebelumnya. Penelitian ini juga mampu menampilkan aspek semantik yang halus. Skor BLEU yang didapatkan adalah 91 pada BLEU-1.

Penelitian tentang *image captioning* berbahasa Inonesia telah dilakukan oleh (Nugraha, Arifianto, & Suyanto, 2019) dan (Fudholi et al., 2021). (Nugraha et al., 2019) meneliti tentang *image captioning* dengan gambar bersifat umum menghasilkan prediksi *caption* yang cukup baik dengan skor BLEU-1,2,3,4 {36.7, 17.8, 6.7, 2.0}. (Fudholi et al., 2021) melakukan *image captioning* dengan data gambar objek wisata di Yogyakarta menghasilkan skor BLEU {24.51}. Kedua penelitian tersebut menggunakan GRU sebagai *encoder* untuk memprediksi *caption*. Penelitian yang akan dilakukan, penulis memilih LSTM sebagai *encoder* untuk melakukan prediksi *caption.* Pemilihan LSTM dilakukan berdasarkan performa LSTM yang lebih baik daripada GRU (Tan et al., 2019). Perbandingan penelitian terdahulu terdapat pada Tabel 2.1.

Tabel 2.1 Daftar perbandingan penelitian yang dilakukan sebelumnya

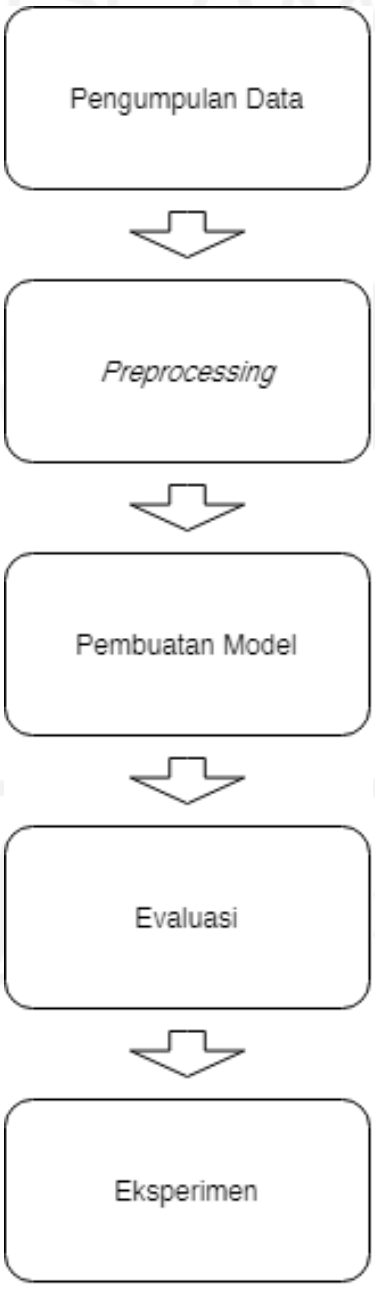
Penulis	**Judul**	**Metode**	**Hasil**
(Al-muzaini et al., 2018)	*Automatic Arabic image captioning using RNN-LSTM-based language model and CNN*	RNN-CNN	Mampu menghasilkan deskripsi gambar dengan berbahasa Arab. Model yang dibuat memperoleh skor 46.2 pada BLEU-1

(Gu et al., 2017)	*An Empirical Study of Language CNN for Image Captioning*	CNN	Memiliki performa yang baik ketika digunakan di Flickr30k dan MS COCO *dataset*, serta memperoleh skor BLEU-1 sebesar 72.6 pada MS COCO *dataset.*
(Tan et al., 2019)	*Image Captioning with Sparse Recurrent Neural Network*	RNN	Mencapai tingkat *sparsity* hingga 97,5% tanpa kehilangan kinerja yang signifikan terhadap *baseline*. Memperoleh skor BLEU yang cukup baik dengan LSTM maupun GRU yaitu 73.8 dan 73.4 pada BLEU-1.
(You et al., 2016)	*Image Captioning with Semantic Attention*	RNN-*semantic attention*	Hasilnya mampu memberikan hasil yang melampaui standar sebelumnya, selain itu mampu menampilkan aspek semantik dengan halus. Memperoleh skor 91 untuk BLEU-1.
(Nugraha et al., 2019)	*Generating Image Description on Indonesian Language using Convolutional Neural Network and Gated Recurrent Unit*	*(CNN dan GRU)*	BLEU-1 36.7 BLEU-2 17.8 BLEU-3 6.7 BLEU-4 2.0
(Fudholi et al., 2021)	*Image Captioning with Attention for Smart Local Tourism using EfficientNet*	*CNN – GRU attention*	BLEU 24.51

BAB III
METODOLOGI PENELITIAN

3.1 Langkah – langkah Penelitian

Gambar 3.1 mengilustrasikan langkah-langkah yang dilakukan pada penelitian ini. Tahapan penelitian terdiri dari pengumpulan data, *preprocessing*, pembuatan model, evaluasi, dan eksperimen.



Gambar 3.1 Alur Penelitian

3.2 Uraian Penelitian

Berikut ini adalah uraian yang menjelaskan bagan alur tentang langkah penelitian ini mulai dari pengumpulan data sampai eksperimen.

3.2.1 Pengumpulan Data

Data yang digunakan pada penelitian ini diambil dari MS COCO *dataset*. *Dataset* tersebut dipilih karena memiliki data yang kompleks dan berisi 91 objek dengan total 2,5 juta label dalam 328.000 gambar (Lin et al., 2014). Pada MS COCO *dataset* sudah dilengkapi dengan *caption* berbahasa inggris berjumlah 5 (lima) *caption* untuk setiap gambar.

Penelitian ini memilih 10.000 gambar secara acak dari dataset. Adapun sampel data gambar dapat dilihat pada Gambar 3.2. Gambar yang telah dipilih kemudian diberi *caption* secara manual berjumlah satu *caption* untuk satu gambar, jadi total *caption* yang dibuat adalah 10.000, kemudian *file* berisi *caption* tersebut disimpan dalam format json. Tahap pelabelan/pemberian *caption* dilakukan dengan cara:

a. Pelabelan data dilakukan secara manual oleh penulis, dengan memberi satu label/*caption* pada tiap-tiap gambar.
b. *Caption* yang dibuat memuat aspek utama dalam gambar seperti gambar manusia disertai kegiatan yang dilakukan, gambar kendaraan, gambar hewan, dan gambar Makanan.
c. *Caption* yang dibuat menggunakan susunan dengan minimal mengandung kalimat yang menggambarkan situasi dalam gambar, bukan hanya menggambarkan nama objek yang ada pada gambar.

Gambar 3.2 Data gambar pada MS COCO

3.2.2 *Preprocessing*

Tahap *preprocessing* adalah tahapan sebelum melakukan pembuatan model. Pada tahapan ini dilakukan ekstraksi fitur menggunakan 2 (dua) arsitektur dari CNN yaitu InceptionV3 dan VGG16. *Preprocessing* gambar dengan InceptionV3 dilakukan dengan cara mengubah ukuran citra menjadi ukuran menjadi 299x299 sesuai dengan format dari InceptionV3, sedangkan saat menggunakan VGG16 maka format ukuran yang digunakan adalah 224x224. Setelah proses mengubah ukuran gambar, tahap berikutnya adalah melakukan inisialisasi dan memuat bobot dari *ImageNet* dengan data yang telah dilatih sebelumnya.

Tahap *preprocessing* selanjutnya adalah melakukan *preprocessing* terhadap teks. Teks yang dimaksud adalah *caption* yang telah dibuat. Langkah awalnya adalah memilih 5.000 kata yang paling banyak digunakan dalam *caption.* Teks tersebut digunakan untuk membentuk kosakata yang nantinya akan digunakan untuk menghasilkan prediksi *caption*. Pada tahapan ini, juga dilakukan penghilangan tanda baca karena tidak diperlukan seperti pada Tabel 3.1.

Tabel 3.1 *Caption* setelah tanda baca dihilangkan

Sebelum	**Sesudah**
keranjang buah berisi buah-buahan seperti jeruk, pisang, dan nanas	keranjang buah berisi buah buahan seperti jeruk pisang dan nanas
sebuah gunting, sebuah tang, sebuah pisau diletakkan di atas meja.	sebuah gunting sebuah tang sebuah pisau diletakkan di atas meja

Setelah *caption* tersebut dihilangkan tanda bacanya, langkah selanjutnya adalah melakukan tokenisasi terhadap setiap kata yang ada di dalam *caption*. Tokenisasi merupakan proses pemisahan teks ke dalam unit-unit kecil seperti pada Tabel 3.2.

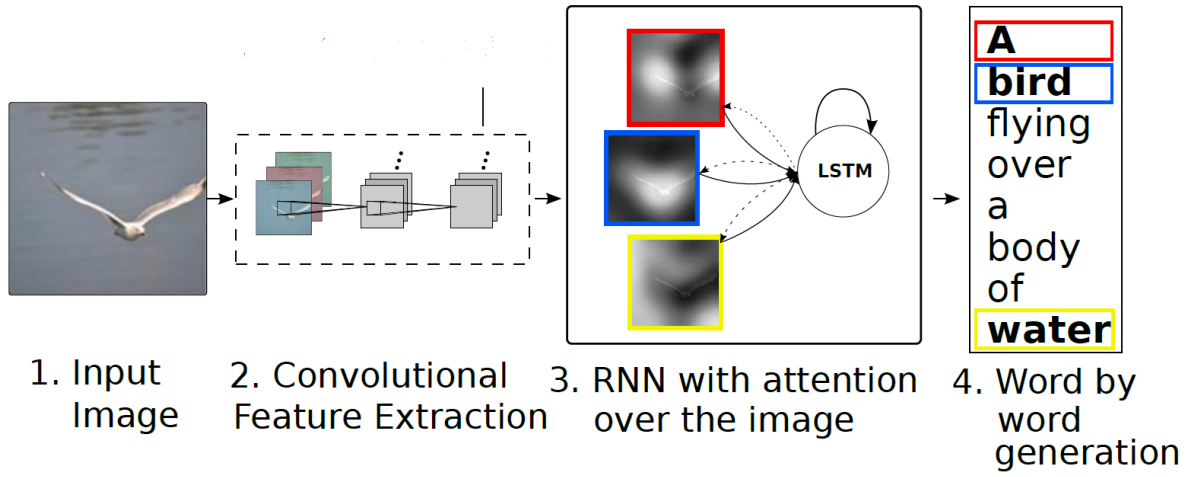
Tabel 3.2 Tokenisasi per kata

Sebelum	**Sesudah**
keranjang buah berisi buah-buahan seperti jeruk pisang dan nanas	“keranjang” “buah” “berisi” “buah” “buahan” “seperti” “jeruk” “pisang” “dan” “nanas”

sebuah gunting sebuah tang sebuah pisau diletakkan di atas meja.	“sebuah” “gunting” “sebuah” “tang” “sebuah” “pisau” “diletakkan” “di” “atas” “meja”

3.2.3 Pembuatan Model

Model akan dibuat dengan alur sesuai Gambar 3.3. Pertama gambar akan masuk ke ekstraksi fitur dari CNN yang di dalamnya adalah InceptionV3 atau VGG16. Proses ekstraksi tersebut menghasilkan vektor, yang masing-masing merupakan representasi dimensi yang sesuai dengan bagian dari gambar. Untuk mendapatkan korespondensi antara vektor fitur dan bagian dari gambar 2-D, ekstraksi fitur dari lapisan dilewatkan menuju *Fully Connected Layer*. Hal tersebut memungkinkan dekoder untuk secara selektif fokus pada bagian tertentu dari suatu gambar dengan memilih *subset* dari semua vektor fitur. Kemudian dekoder (LSTM) melakukan prediksi berdasarkan vektor konteks yang didapatkan.



Gambar 3.3 Alur model untuk *image captioning*

Sumber: (Xu et al., 2015).

Sebelum model dibuat, data yang ada dibagi menjadi dua yaitu *training set* dan *validation set* dengan rasio 80:20 (Fudholi et al., 2021) sehingga jumlah data yang digunakan yaitu sejumlah 8.000 untuk *training set* dan 2.000 untuk *validation set*, data tersebut dibagi secara acak. *Training set* adalah data yang digunakan sebagai data pelatihan sedangkan *validation set* adalah data yang digunakan untuk memvalidasi hasil keluaran.

Tabel 3.3 Skenario pada Model 1

Batch Size	***Embedding Dimension***	***Units***	***Features Shape***	***Attention Features Shape***	***Epoch***
64	256	512	2048	64	10

Tabel 3.4 Skenario pada Model 2

Batch Size	***Embedding Dimension***	***Units***	***Features Shape***	***Attention Features Shape***	***Epoch***
64	256	512	512	49	10

Model dibangun dengan LSTM dan akan dibandingkan dengan model yang dibangun menggunakan GRU. Semua model sama-sama menggunakan *batch size* berukuran 64. Pembagian dengan *batch size* dilakukan karena tidak bisa melewatkan keseluruhan *dataset* ke dalam jaringan syaraf sekaligus. Jadi, *dataset* harus dibagi ke dalam sejumlah bagian yang lebih kecil. Ukuran pada *batch size* akan mempengaruhi durasi untuk pelatihan. Diilustrasikan apabila menggunakan 1 *batch* maka semua (N) data diproses *forward* kemudian di hitung rata-rata eror lalu melakukan *backprop* dan terakhir melakukan *update* bobot. Berbeda dengan *Minibatch*, yaitu beberapa *batch* data diproses *forward* kemudian menghitung rata-rata eror lalu melakukan *backprop* dan terakhir melakukan *update* bobot.

Terdapat perbedaan antara Model 1 dan Model 2 pada *Features Shape* dan *Attention Features Shape*, perbedaan tersebut didapatkan karena ukuran vektor yang telah diekstrak pada Model 1 yang menggunakan InceptionV3 adalah (64, 2048) sedangkan pada Model 2 yang menggunakan VGG16 adalah (49, 512). Kemudian *vector* tersebut masuk ke CNN *encoder* yang terdiri dari *Fully Connected Layer*. Skenario pada Model 1 dapat dilihat pada Tabel 3.3 dan skenario pada Model 2 dapat dilihat pada Tabel 3.4.

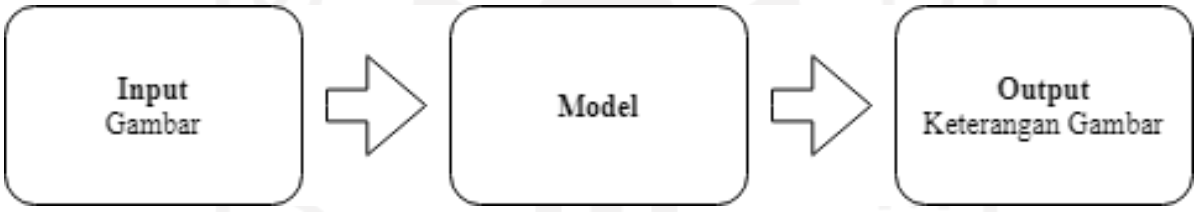
Pada proses *training*, fitur yang telah diekstrak dijalankan menuju *encoder*. Output *encoder*, *hidden state*, dan *input decoder* akan diteruskan menuju *decoder*. Kemudian akan dihitung *loss* dengan menggunakan kata yang digunakan untuk memprediksi. Langkah terakhir adalah melakukan kalkulasi terhadap gradien kemudian menerapkannya ke *optimizer* dan *backprogate*.

3.2.4 Evaluasi

Evaluasi pada penelitian bertujuan untuk mengukur kinerja dari model yang telah dibuat. Hasil dari *generate caption* akan dinilai dengan skor BLEU. Skor BLEU yang semakin tinggi mengindikasikan bahwa *caption* yang telah dibuat semakin menyamai referensi. Pada tahap evaluasi ini akan dibandingkan hasil *generate caption* yang dihasilkan sistem dengan *caption* asli.

3.2.5 Eksperimen

Gambar 3.4 mengilustrasikan alur eksperimen. Eksperimen dilakukan dengan cara *generate caption* dengan gambar yang tidak ada di *dataset*, tujuannya adalah untuk melihat apakah hasil *caption* dari model yang telah dibuat dapat menghasilkan deskripsi sesuai dengan gambar dan membentuk kalimat secara baik atau tidak. Selanjutnya gambar akan ditampilkan di dalam tabel berdampingan dengan hasil *generate caption* dari Model 1 dan Model 2.



Gambar 3.4 Alur eksperimen

3.2.6 Pembuatan *User Interface*

User interface akan dibangun dengan menggunakan streamlit. Streamlit memungkinkan untuk membangun sebuah *website* sederhana untuk menampilkan dan mempermudah orang lain untuk melihat hasil dari model yang dibangun. Berikut ini adalah proses pembuatan *interface* yang dilakukan pada penelitian ini:

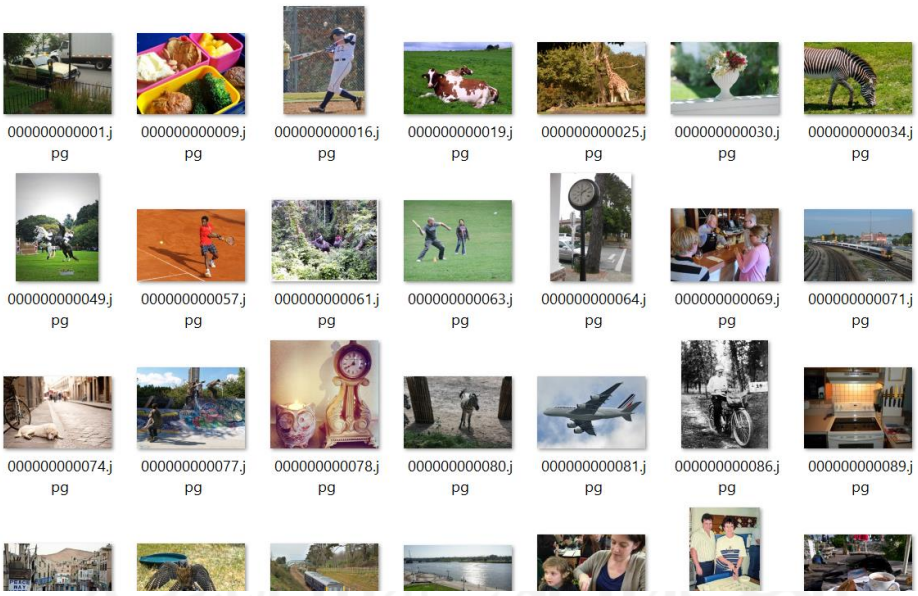
a. Mengunggah gambar.
b. Model melakukan prediksi *caption.*
c. Menampilkan hasil prediksi *caption.*

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Data diperoleh dengan cara mengunduh langsung ke *website* MS COCO. Data yang diambil adalah data berupa gambar, tanpa perlu mengambil data *annotations* yang berisi *caption* gambar, dikarenakan *caption* yang disediakan pada *dataset* tersebut berbahasa Inggris, sedangkan pada penelitian ini keluaran yang diinginkan adalah *caption* berbahasa Indonesia.

Data yang digunakan berjumlah 10.000 data gambar. Oleh karena itu, *caption* dibuat sendiri oleh penulis disesuaikan data jumlah gambar yang dipilih yaitu berjumlah 10.000 *caption*, kemudian *caption* tersebut disimpan di dalam format json. Gambar 4.1 adalah kumpulan data gambar yang akan digunakan dan Gambar 4.2 merupakan hasil pembuatan *caption.*



Gambar 4.1 Gambar yang digunakan dalam penelitian

```
{"image_id": 72,
 "captionindonesia": "beberapa jerapah sedang memakan daun yang ada di
                      pohon"
},
{"image_id": 19,
 "captionindonesia": "dua sapi sedang berbaring di padang rumput"
},
{"image_id": 25,
 "captionindonesia": "seekor jerapah sedang memakan dedaunan yang ada di
                      pohon"
 },
```

Gambar 4.2 *Caption* yang telah dibuat

4.2 Preprocessing

4.2.1 *Preprocessing* Gambar

Tahap *preprocessing* gambar pada penelitian ini menggunakan dua model arsitektur CNN yaitu InceptionV3 dan VGG16 yang diterapkan di masing-masing model. Model 1 menggunakan InceptionV3 dan Model 2 menggunakan VGG16. Langkah awal melakukan *preprocessing* ini adalah mengubah format ukuran gambar disesuaikan dengan ukuran *input* dari masing-masing ekstraksi fitur yang digunakan, untuk Model 1 dijelaskan pada Gambar 4.3 menggunakan ukuran (299x299) sedangkan pada Model 2 pada Gambar 4.4 menggunakan ukuran (224x224). Nilai 3 pada *channels* menandakan ukuran konvolusi pada tiap arsitektur.

```
1     #Preprocess the images using InceptionV3
2     def load_image(image_path):
3         img = tf.io.read_file(image_path)
4         img = tf.image.decode_jpeg(img, channels=3)
5         img = tf.image.resize(img, (299, 299))
6         img = tf.keras.applications.inception_v3.preprocess_input(img)
7         return img, image_path
```

Gambar 4.3 *Preprocessing* gambar dengan InceptionV3

```
1     #Preprocess the images using VGG16
2     def load_image(image_path):
3         img = tf.io.read_file(image_path)
4         img = tf.image.decode_jpeg(img, channels=3)
5         img = tf.image.resize(img, (224, 224))
6         img = tf.keras.applications.vgg16.preprocess_input(img)
7         return img, image_path
```

Gambar 4.4 *Preprocessing* gambar dengan VGG16

4.2.2 Inisialisasi Bobot ImageNet

Gambar 4.5 adalah tahapan melakukan inisialisasi bobot ImageNet. Tahap tersebut dilakukan dengan cara membuat model *tf.keras* dengan konvolusi terakhir dari masing-masing arsitektur. Untuk Model 1 adalah (8x8x2048) sedangkan Model 2 menggunakan (7x7x512).

```
1  image_model = tf.keras.applications.InceptionV3(include_top=False,
2                                                   weights='imagenet')
3  new_input = image_model.input
4  hidden_layer = image_model.layers[-1].output
5
6  image_features_extract_model = tf.keras.Model(new_input, hidden_layer)
```

Gambar 4.5 Inisialisasi bobot yang ada di ImageNet

4.2.3 Ekstraksi Fitur

Gambar 4.6 adalah tahapan untuk melakukan ekstraksi fitur sesuai ukuran konvolusi terakhir masing-masing arsitektur. Tahap ini menggunakan InceptionV3 dan VGG16 yang dijalankan dengan Google Colab menggunakan *runtime* GPU. *Batch* yang digunakan pada tahap tersebut adalah 16, sehingga setiap satu iterasi langsung mengeksekusi 16 gambar, jadi total ada 625 iterasi untuk 10.000 gambar. Proses eksekusi pada model 1 memerlukan waktu 70 menit, sedangkan Model 2 hanya 45 menit. Keluaran dari tahapan ini adalah sebuah *file* dengan ekstensi *.npy* untuk masing-masing gambar.

```
1     # Get unique images
2     encode_train = sorted(set(img_name_vector))

3     image_dataset = tf.data.Dataset.from_tensor_slices(encode_train)
      image_dataset = image_dataset.map(
4       load_image, num_parallel_calls=tf.data.AUTOTUNE).batch(16)
5
6     for img, path in image_dataset:
        batch_features = image_features_extract_model(img)
7       batch_features = tf.reshape(batch_features,
8                                   (batch_features.shape[0],-1,
9                                    batch_features.shape[3]))
10
        for bf, p in zip(batch_features, path):
          path_of_feature = p.numpy().decode("utf-8")
11        np.save(path_of_feature, bf.numpy())
```

Gambar 4.6 Tahap ekstraksi fitur

4.2.4 *Preprocessing Caption*

Penelitian ini hanya membatasi 5.000 kosakata terbanyak pada *dataset*. Setelah banyaknya kosakata dipilih, tanda baca dan simbol yang ada pada *dataset* dihapus dengan fungsi *filters* karena tidak diperlukan. Proses tersebut dapat dilihat pada Gambar 4.7 baris kode 2-4. Proses selanjutnya adalah tokenisasi, yaitu dengan cara memisahkan setiap kata dengan memanfaatkan spasi yang ada pada teks seperti pada Gambar 4.7 baris kode 6-8, tahap ini digunakan untuk memisahkan kata dan memperoleh kosakata yang nantinya akan digunakan sebagai *caption* prediksi. Hasil dari tokenisasi kata dapat dilihat pada Gambar 4.8 sedangkan banyaknya kata yang sering muncul dapat dilihat pada Gambar 4.9.

```
1  top_k = 5000
2  tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=top_k,
3                      oov_token="<unk>",
4                      filters='!"#$%&()*+.,-/:;=?@[\]^_`{|}~')
5
6  tokenizer.fit_on_texts(train_captions)
7  tokenizer.word_index['<pad>'] = 0
8  tokenizer.index_word[0] = '<pad>'
```

Gambar 4.7 *Preprocessing* pada *caption*

```
{1: '<unk>',
 2: '<start>',
 3: '<end>',
 4: 'di',
 5: 'sedang',
 6: 'sebuah',
 7: 'seorang',
 8: 'yang',
 9: 'dengan',
 10: 'dan',
```

Gambar 4.8 Kata setelah tokenisasi

```
OrderedDict([('<start>', 10000),
             ('sebuah', 4091),
             ('mobil', 279),
             ('diparkirkan', 27),
             ('di', 6950),
             ('meteran', 12),
             ('parkir', 59),
             ('yang', 3056),
             ('ada', 308),
             ('pinggir', 497),
             ('jalan', 799),
```

Gambar 4.9 Jumlah setiap kata dari *caption*

4.3 Pembuatan Model

4.3.1 Konfigurasi Pembuatan Model

Model 1 akan dibuat sesuai dengan skenario pada Tabel 3.3 dan Model 2 disesuaikan dengan skenario pada Tabel 3.4. Sebelum model dibuat, langkah awal yang dilakukan adalah membagi data menjadi *training set* dan *validation set*. Pembagian data dilakukan dengan rasio 80:20, dengan pembagian sebanyak 8.000 data digunakan untuk *training set* dan 2.000 data untuk *validation set.* Selanjutnya, data tersebut dibagi secara acak dengan *random_state = 0*. Proses melakukan *spliting data* dapat dilihat pada Gambar 4.10.

```
1  # Create training and validation sets using an 80-20 split
2  img_name_train, img_name_val, cap_train, cap_val =
3                                     train_test_split(img_name_vector,
4                                     cap_vector,
5                                     test_size=0.2,
6                                     random_state=0)
7
8  print("Number of training captions: %d" % len(cap_train))
9  print("Number of validation captions: %d" % len(cap_val))
```

Gambar 4.10 Melakukan *splitting dataset*

Setelah *dataset* berhasil dibagi menjadi *training* dan *validation*, proses selanjutnya adalah mengatur konfigurasi masing-masing model seperti yang ada di Gambar 4.11 untuk Model 1 dan Gambar 4.12 untuk Model 2. Konfigurasi pada Model 1 dan Model 2 disesuaikan dengan *feature shape* masing-masing arsitektur. Model 1 memiliki ukuran (64, 2048) sehingga variabel *feature shape* diisi dengan 2048 dan variabel *attention_feature_shape* diisi dengan 64. Adapun ukuran model 2 adalah (49, 512) maka *feature_shape* diisi 512 dan *attention_feature_shape* adalah 49. Kemudian membuat model dengan pendekatan *attention mechanism*.

```
1       BATCH_SIZE = 64
2       BUFFER_SIZE = 100
3       embedding_dim = 256
4       units = 512
5       vocab_size = top_k + 1
6       num_steps = len(img_name_train) // BATCH_SIZE
7       # Shape of the vector extracted from InceptionV3 is (64, 2048)
8       # These two variables represent that vector shape
9       features_shape = 2048
10      attention_features_shape = 64
```

Gambar 4.11 Konfigurasi Model 1

```
1       BATCH_SIZE = 64
2       BUFFER_SIZE = 100
3       embedding_dim = 256
4       units = 512
5       vocab_size = top_k + 1
6       num_steps = len(img_name_train) // BATCH_SIZE
7       # Shape of the vector extracted from InceptionV3 is (49, 512)
8       # These two variables represent that vector shape
9       features_shape = 512
10      attention_features_shape = 49
```

Gambar 4.12 Konfigurasi Model 2

Pada Gambar 4.13, *attention layer* digunakan untuk melakukan perhitungan dengan tujuan menghasilkan *context vector*. Skor *attention* didapatkan dari perkalian antara *hidden state* dengan *decoder layer* sebelumnya. *Softmax* digunakan untuk mengubah skor tersebut agar menjadi sebuah nilai probabilitas. *Attention* skor akan dikalikan dengan *image feature vector* untuk menjadi *context vector* pada a*ttention layer*, *context vector* akan ditambahkan dengan *embedding vector* sebelum masuk menuju *decoder.*

```
1   class BahdanauAttention(tf.keras.Model):
2     def __init__(self, units):
3       super(BahdanauAttention, self).__init__()
4       self.W1 = tf.keras.layers.Dense(units)
5       self.W2 = tf.keras.layers.Dense(units)
6       self.V = tf.keras.layers.Dense(1)
7
8     def call(self, features, hidden):
9       # features(CNN_encoder output) shape == (batch_size, 64,
```

```
10      embedding_dim)
11
12      # hidden shape == (batch_size, hidden_size)
13      # hidden_with_time_axis shape == (batch_size, 1, hidden_size)
14      hidden_with_time_axis = tf.expand_dims(hidden, 1)
15
16      # attention_hidden_layer shape == (batch_size, 64, units)
17      attention_hidden_layer = (tf.nn.tanh(self.W1(features) +
18                                self.W2(hidden_with_time_axis)))
19
20      # score shape == (batch_size, 64, 1)
21      # This gives you an unnormalized score for each image feature.
22      score = self.V(attention_hidden_layer)
23
24      # attention_weights shape == (batch_size, 64, 1)
25      attention_weights = tf.nn.softmax(score, axis=1)
26
27      # context_vector shape after sum == (batch_size, hidden_size)
28      context_vector = attention_weights * features
29      context_vector = tf.reduce_sum(context_vector, axis=1)
30
31      return context_vector, attention_weights
```

Gambar 4.13 Model dengan *attention mechanism*

CNN *encoder* pada Gambar 4.14 baris 1-8 digunakan untuk melewatkan fitur yang telah diekstrak dan disimpan sebelumnya. Fitur yang telah diekstrak tersebut dilewatkan melalui *fully connected layer* setelah semua fitur telah melalui proses konvolusi. CNN *encoder* menggunakan fungsi aktivasi ReLu pada baris 10-13.

```
1   class CNN_Encoder(tf.keras.Model):
2       # Since you have already extracted the features and dumped it
3       # This encoder passes those features through a Fully connected
4   layer
5       def __init__(self, embedding_dim):
6           super(CNN_Encoder, self).__init__()
7           # shape after fc == (batch_size, 64, embedding_dim)
8           self.fc = tf.keras.layers.Dense(embedding_dim)
9
10      def call(self, x):
11          x = self.fc(x)
12          x = tf.nn.relu(x)
13          return x
```

Gambar 4.14 *Encoder* dengan CNN

Context vector pada Gambar 4.15 diperoleh dari langkah Gambar 4.13 kemudian digabungkan dengan *decoder output.* Selanjutnya, *context vector* dimasukkan ke dalam *Decoder RNN cell* untuk menghasilkan *hidden state* baru. Proses ini dilakukan berulang kali sampai *decoder* mencapai titik ‘<end>’ atau batas panjang maksimum yang telah ditentukan sebelumnya. *Output* terakhir diperoleh dengan melewatkan *hidden state* baru menuju *Linear*

layer yang berfungsi sebagai *classifier* untuk memberikan probabilitas terhadap kata yang diprediksi sebelumnya.

```
class RNN_Decoder(tf.keras.Model):
  def __init__(self, embedding_dim, units, vocab_size):
    super(RNN_Decoder, self).__init__()
    self.units = units

    self.embedding = tf.keras.layers.Embedding(vocab_size,
embedding_dim)
    self.lstm = tf.keras.layers.LSTM(self.units,
                                     return_sequences=True,
                                     return_state=True,

recurrent_initializer='glorot_uniform')
    self.fc1 = tf.keras.layers.Dense(self.units)
    self.fc2 = tf.keras.layers.Dense(vocab_size)

    self.attention = BahdanauAttention(self.units)

  def call(self, x, features, hidden):
    # defining attention as a separate model
    context_vector, attention_weights = self.attention(features,
hidden)

    # x shape after passing through embedding == (batch_size, 1,
embedding_dim)
    x = self.embedding(x)

    # x shape after concatenation == (batch_size, 1, embedding_dim +
hidden_size)
    x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

    # passing the concatenated vector to the LSTM
    output, state, cell = self.lstm(x)

    # shape == (batch_size, max_length, hidden_size)
    x = self.fc1(output)

    # x shape == (batch_size * max_length, hidden_size)
    x = tf.reshape(x, (-1, x.shape[2]))

    # output shape == (batch_size * max_length, vocab)
    x = self.fc2(x)

    return x, state, attention_weights

  def reset_state(self, batch_size):
    return tf.zeros((batch_size, self.units))
```

Gambar 4.15 *Decoder* dengan LSTM

4.3.2 *Training Data*

Tahap *training* dilakukan untuk melatih model yang telah dibuat, keseluruhan proses *training* dapat dilihat pada Gambar 4.16. *Training* pada penelitian ini menggunakan 10 *epoch* pada masing-masing model. Tahap nomor 7-13 bertujuan untuk melihat data *loss* pada data *training* dan nomor 15 -21 untuk melihat *loss* pada *validation*, semakin kecil angka *loss* yang didapat maka model semakin baik, setelah itu dapat di lihat di grafik apakah model mengalami *overfitting* atau tidak. Pada proses *training* ini juga dilakukan penyimpanan model ke dalam bentuk *checkpoint* dengan tujuan untuk mengambil model terbaik pada setiap *checkpoint* yang dilalui, setiap satu *epoch,* proses tersebut disimpan ke dalam *checkpoint*.

Epoch berjumlah 10 dirasa cukup karena telah menggambarkan kondisi data. Semakin banyaknya *epoch* yang bertambah, semakin sering *weight* yang ada pada *neural network* diubah (di-*update*) dan kurva akan berubah dari *underfitting* ke kurva optimal lanjut ke kurva *overfitting*, hal tersebut dapat digambarkan dengan grafik.

```
1   EPOCHS = 10
2   checkpoint_every_n_epochs = 1
3
4   for epoch in range(start_epoch, EPOCHS):
5       start = time.time()
6
7       #For Train
8       total_loss = 0
9       for (batch, (img_tensor, target)) in enumerate(dataset):
10          batch_loss, t_loss = train_step(img_tensor, target)
11          total_loss += t_loss
12      # storing the epoch end loss value to plot later
13      loss_plot.append(total_loss / num_steps)
14
15      #For Test
16      total_loss_test = 0
17      for (batch, (img_tensor, target)) in enumerate(test_dataset):
18          batch_loss, t_loss = test_step(img_tensor, target)
19          total_loss_test += t_loss
20      # storing the epoch end loss value to plot later
21      test_loss_plot.append(total_loss_test / num_steps)
22
23      if (epoch+1) % checkpoint_every_n_epochs == 0:
24        print("Checkpointing model after %d epochs of training." %
25              (epoch+1))
26        ckpt_manager.save(epoch+1)
27
28      print ('Epoch {} TrainLoss {:.6f} ValLoss {:.6f}'.format(epoch +
29             1,(total_loss/num_steps),(total_loss_test/num_steps)))
30      print ('Time taken for 1 epoch {} sec\n'.format(time.time() -
32             start))
```

Gambar 4.16 Konfigurasi saat melakukan *training* data

4.4 Evaluasi

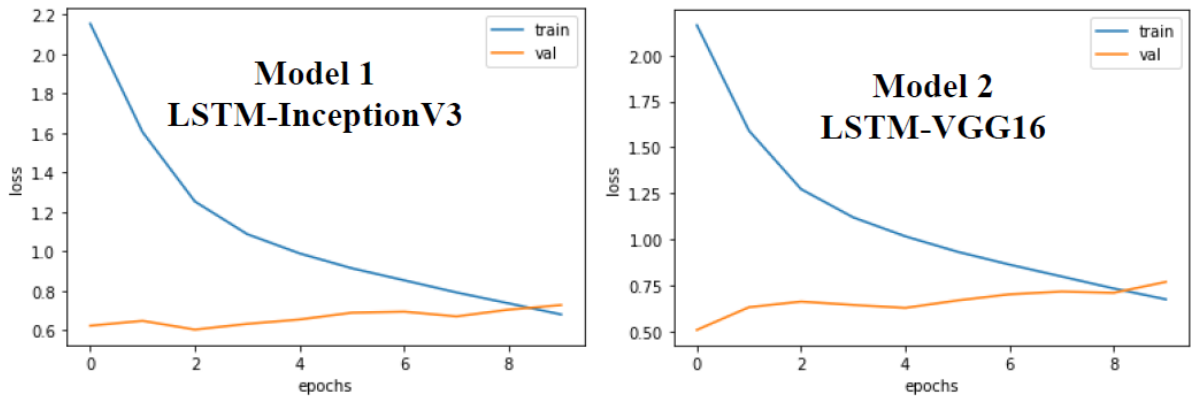
4.4.1 Hasil

Pada saat proses *training* selesai, didapatkan nilai *loss* dari masing-masing model yang telah dibuat. Nilai tersebut ditampilkan pada Tabel 4.1. Berdasarkan tabel tersebut Model 1 memiliki nilai *training loss* sebesar 0.69, lebih kecil daripada Model 2 yang memiliki nilai *training loss* 0.70. *Validation loss* Model 1 dan Model 2 memiliki nilai yang cenderung lebih kecil daripada *loss* yang dihasilkan oleh model yang dibangun dengan GRU. Grafik nilai *loss* dapat dilihat pada Gambar 4.17 dan Gambar 4.18.

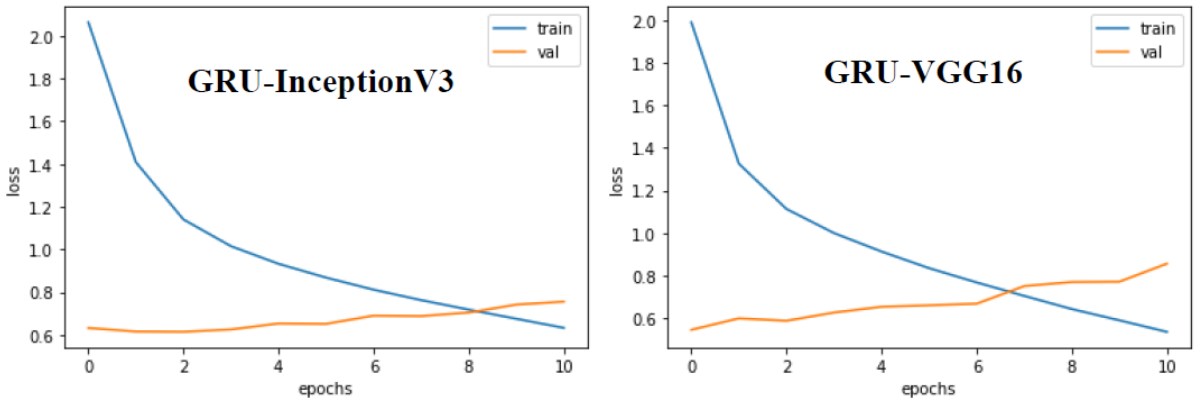
Tabel 4.1 Nilai *loss* dari masing-masing model

	Model 1 InceptionV3	**Model 2 VGG16**	**GRU InceptionV3**	**GRU VGG16**
Training Loss	0.69	0.70	0.63	0.53
Validation Loss	0.71	0.75	0.75	0.85

Gambar 4.17 dan Gambar 4.18 menampilkan grafik dari Model 1 dan Model 2 pada proses *training* dengan e*poch* 10. Kedua model menampilkan grafik yang menunjukkan kenaikan nilai *validation loss*. Pada model 1, grafik menunjukkan kenaikan dan perpotongan antara nilai *validation loss* dan *training loss* pada *epoch* ke-9, sedangkan pada model 2 grafik nilai *validation loss* mulai naik dan mengalami perpotongan dengan *training loss* pada *epoch* ke-8. Hal tersebut menunjukkan semua model yang telah dibuat sama-sama mengalami *overfitting*. *Overfitting* merupakan kondisi pada saat model terlalu fokus pada *training* dalam *dataset* tertentu, sehingga apabila model diberikan data yang baru, prediksi yang dilakukan bisa tidak tepat.



Gambar 4.17 Grafik *loss* dari Model 1 dan Model 2 dengan LSTM



Gambar 4.18 Grafik *loss* dengan GRU

Gambar 4.18 adalah grafik pada saat *training* dari model yang dibuat dengan GRU. Model tersebut adalah InceptionV3-GRU dan VGG16-GRU. Dilihat dari gambar tersebut, kedua model GRU mengalami *overfitting* hampir sama dengan Model 1 dan Model 2 yang menggunakan LSTM. Model GRU-VGG16 mengalami kenaikan nilai *loss* pada *validation set* lebih besar daripada model GRU-InceptionV3.

4.4.2 Evaluasi dengan Skor BLEU

Tahap evaluasi digunakan untuk menilai seberapa bagus *caption* yang telah berhasil dibuat oleh masing-masing model dengan menggunakan skor BLEU. Hasil *caption* akan dibandingkan dengan *caption* asli dari gambar yang telah dibuat sebelumnya. Penghitungan skor BLEU dilakukan dengan skor BLEU-1, BLEU-2, BLEU-3 dan, BLEU-4. Hasil evaluasi menunjukkan gambar dengan *caption* asli dibandingkan dengan hasil prediksi *caption* dan disertai skor BLEU untuk masing-masing gambar.

Rentang penilaian skor BLEU adalah dari 0-1 (sebagian menggunakan format penilaian dikali 100 jadi apabila skor BLEU sempurna maka memiliki skor 100). Skor 1 hanya akan di dapatkan apabila kalimat yang dihasilkan mesin benar-benar sama dengan referensi kalimat yang sudah ada. Untuk Mencari Skor BLEU dapat menggunakan *Python* dengan memakai *Natural Language Toolkit Library* (NLTK), hal ini mempermudah untuk membandingkan kalimat yang dihasilkan mesin dengan referensi.

A. *Sentence* BLEU

Sentence BLEU digunakan untuk menilai sebuah kalimat dibandingkan dengan satu kalimat atau lebih kalimat referensi. Kalimat-kalimat tersebut harus dijadikan *list* untuk

membentuk sebuah token terlebih dahulu, Gambar 4.19 merupakan potongan kode program untuk menghitung skor *sentence* BLEU.

```
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'a', 'test'], ['this', 'is' 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate)
print(score)
```

Gambar 4.19 Kode untuk menghitung *sentence* pada BLEU

B. Individu dan Kumulatif BLEU

Skor BLEU memungkinkan untuk mengatur bobot secara spesifik untuk setiap n-gram, hal tersebut memudahkan untuk menghitung tipe BLEU yang diinginkan. N-gram individu hanya mencocokkan gram dari urutan tertentu, misalnya kata tunggal (1-gram) atau pasangan kata (2-gram). Bobot spesifik untuk tiap indeks mengacu pada urutan gram. Untuk menghitung skor BLEU pada 1-gram yang cocok, hanya perlu memberi bobot 1 pada posisi 1-gram dan 0 di posisi 2-3-4 (1, 0, 0, 0). Cara menghitung skor individu 1-gram pada BLEU ditunjukkan dengan Gambar 4.20.

```
# 1-gram individual BLEU
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'small', 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate, weights=(1, 0, 0, 0))
print(score)
```

Gambar 4.20 Perhitungan skor individu 1-gram skor BLEU

Sedangkan kumulatif n-gram adalah penghitungan individu n-gram dengan semua perhitungan rata-rata secara geometrik. Secara *default sentence_bleu()* merupakan perhitungan dengan 4-gram skor BLEU atau biasa disebut dengan BLEU-4. Bobot dari BLEU-4 adalah ¼ (25%) atau 0.25 untuk tiap 1-gram, 2-gram, 3-gram, dan 4-gram. Gambar 4.21 adalah cara untuk menghitung kumulatif 4-gram skor BLEU.

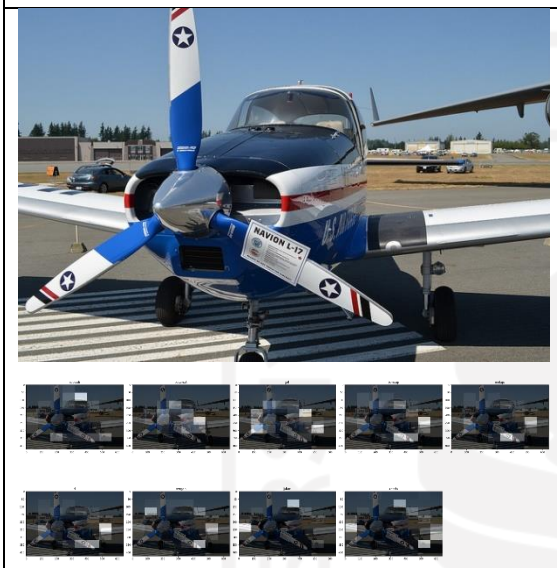
```
# 4-gram cumulative BLEU
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'small', 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate, weights=(0.25, 0.25, 0.25,
                                                     0.25))
print(score)
```

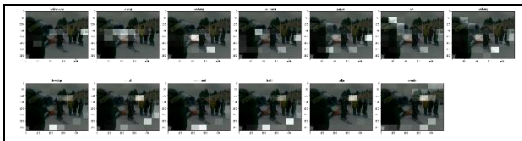
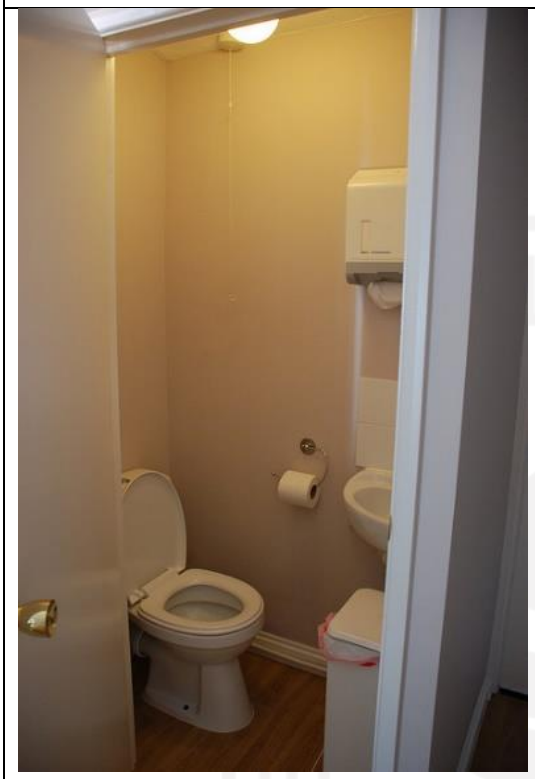
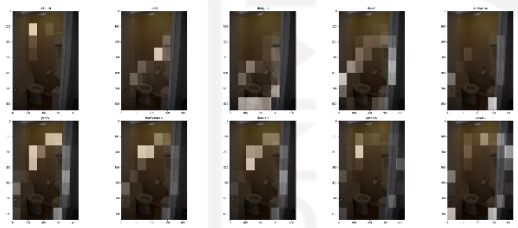
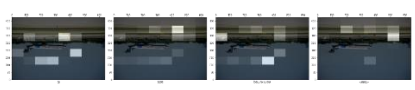
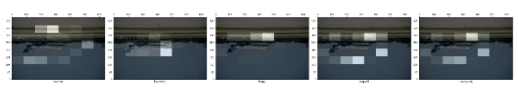
Gambar 4.21 Perhitungan skor kumulatif 4-gram skor BLEU

C. Hasil skor BLEU

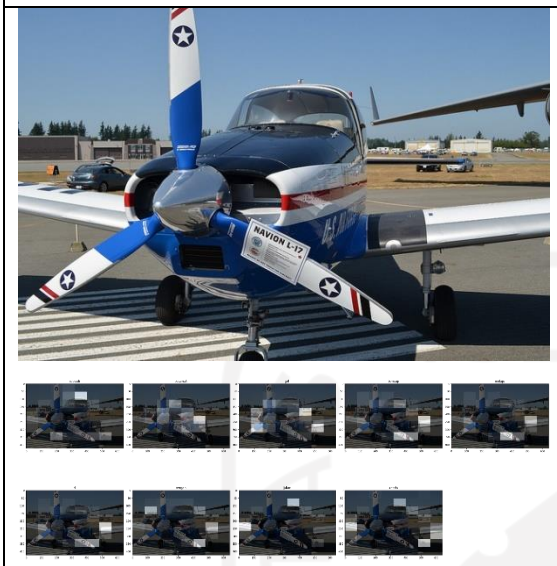
Hasil skor BLEU dari Model 1 ditampilkan pada Tabel 4.2 sedangkan Model 2 ditampilkan pada Tabel 4.3. Hasil *generate caption* dari kedua model tersebut dibandingkan dengan *caption* asli dari gambar, kemudian dihitung skor BLEU masing-masing gambar.

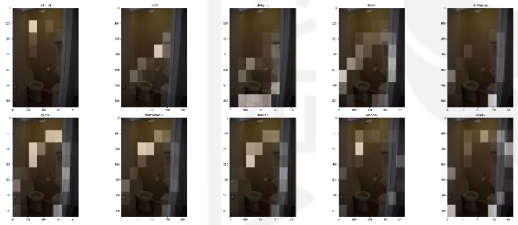
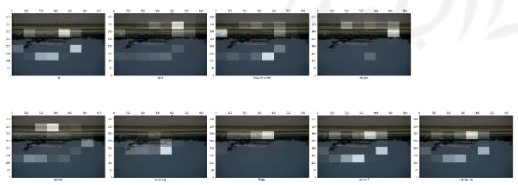
Tabel 4.2 Hasil *generate caption* pada Model 1

Gambar	*Caption* Asli	Prediksi	Skor
	sebuah pesawat dengan baling baling berhenti di pinggir jalan	sebuah pesawat jet bersiap melaju di tengah jalan	bleu-1: 44.1 bleu-2: 23.5 bleu-3: 22.6 bleu-4: 20.6
	beberapa orang sedang bermain bisbol di lapangan bisbol sambil dilihat oleh banyak penonton	beberapa orang berbaju putih sedang berusaha menangkap bola bisbol di lapangan bisbol	bleu-1: 53.6 bleu-2: 42.3 bleu-3: 32.4 bleu-4: 24.1
	sekumpulan orang sedang bersiap untuk bermain papan ski	beberapa orang sedang bermain papan ski sedang bersiap untuk menuruni bukit salju	bleu-1: 58.3 bleu-2: 51.4 bleu-3: 37.9 bleu-4: 32.0

			
 	sebuah toilet putih berada di samping wastafel	sebuah toilet dengan kloset berwarna putih berdekatan dengan jendela	bleu-1: 33.3 bleu-2: 20.4 bleu-3: 21.6 bleu-4: 20.7
  	sebuah pesawat bersiap untuk mendarat di bandara	sebuah pesawat sedang mendarat di bandara	bleu-1: 70.5 bleu-2: 59.8 bleu-3: 42.6 bleu-4: 32.4

Tabel 4.3 Hasil *generate caption* pada Model 2

Gambar	*Caption* Asli	Prediksi	Skor
	sebuah pesawat dengan baling baling berhenti di pinggir jalan	sebuah pesawat dengan baling berhenti di pinggir jalan	bleu-1: 88.2 bleu-2: 88.2 bleu-3: 83.0 bleu-4: 74.2
	beberapa orang sedang bermain bisbol di lapangan bisbol sambil dilihat oleh banyak penonton	beberapa orang sedang bermain bisbol nya	bleu-1: 25.9 bleu-2: 25.4 bleu-3: 24.7 bleu-4: 23.6
	sekumpulan orang sedang bersiap untuk bermain papan ski	sekumpulan orang sedang bersiap untuk bermain papan ski di sebuah membawa layang layang di lahan salju	bleu-1: 50.0 bleu-2: 48.3 bleu-3: 46.7 bleu-4: 44.2

 	sebuah toilet putih berada di samping wastafel	sebuah toilet dengan cermin putih berada di samping cermin	bleu-1: 66.6 bleu-2: 57.7 bleu-3: 46.0 bleu-4: 35.4
 	sebuah pesawat bersiap untuk mendarat di bandara	sebuah pesawat putih bersiap untuk mendarat di bandara	bleu-1: 67.0 bleu-2: 58.0 bleu-3: 42.4 bleu-4: 30.1

Tahap berikutnya adalah menghitung rata-rata skor BLEU dari BLEU-1, BLEU-2, BLEU-3 dan BLEU-4 untuk melihat model mana yang memiliki performa lebih baik dalam menghasilkan deskripsi. Skor BLEU didapatkan dengan memberi perhitungan bobot pada masing-masing n-gram sebagai berikut:

BLEU-1 = (1, 0, 0, 0)

BLEU-2 = (0.5, 0.5, 0, 0)

BLEU-3 = (0.33, 0.33, 0.33, 0.33)

BLEU-4 = (0.25, 0.25, 0.25, 0.25)

Tabel 4.4 Perbandingan skor BLEU pada masing-masing model

BLEU-N	**Model-1**	**Model-2**	**GRU - InceptionV3**	**GRU - VGG16**
BLEU-1	34.93	22.02	27.03	20.93
BLEU-2	22.55	14.29	16.33	13.93
BLEU-3	14.75	9.41	9.39	9.68
BLEU-4	10.15	6.28	5.78	6.73

Tabel 4.4 menampilkan hasil rata-rata skor BLEU semua model. Dilihat dari Tabel 4.4 terlihat bahwa Model 1 memiliki skor yang lebih tinggi dibandingkan dengan model lainnya, baik Model 2 yang dibangun dengan LSTM maupun Model lain yang dibangun dengan GRU. Akan tetapi skor yang didapatkan masih tergolong kecil, skor tertinggi pada BLEU-1 sebesar 34.93. Percobaan lain pada saat proses *generate caption*, penulis menemukan beberapa kali skor 0 pada semua BLEU dari semua model yang dibuat, yang artinya *caption* yang dihasilkan sama sekali berbeda dari referensi.

4.5 Eksperimen

Eksperimen bertujuan untuk menilai secara kualitatif model yang telah dibuat apakah mampu membuat *caption* secara benar sesuai konteks atau tidak. Hasil eksperimen ditampilkan dalam Tabel 4.5 dan Tabel 4.6. Eksperimen dilakukan dengan cara mengambil beberapa gambar dari internet. Model 1 dan Model 2 digunakan untuk melakukan *generate caption* terhadap gambar yang dipilih, setelah itu hasil dari kedua Model 1 dan Model 2 dibandingkan untuk mengetahui model mana yang menghasilkan deskripsi gambar yang lebih tepat.

Tabel 4.5 menampilkan hasil dari *generate caption* dari Model 1 dan menghasilkan *caption* "dua jerapah berada di pohon" yang menandakan bahwa *caption* yang dihasilkan masih salah, walaupun memiliki objek yang benar yaitu jerapah.

Tabel 4.5 Eksperimen dengan Model 1

Gambar	Deskripsi	Keterangan
	seorang pemain sepak bola sedang bermain sepak bola	Benar
	seorang pria melompat dengan skateboard sambil bermain skateboard	Benar
	beberapa orang sedang bermain layang layang layang dan sekelompok orang perempuan sedang bermain layang layang di taman	Benar

	beberapa orang sedang berada di dekat sepeda	salah
	dua jerapah berada di pohon	Salah

Tabel 4.6 menampilkan hasil pengujian dari Model 2 dan menghasilkan *caption* “seorang pria mengendarai sepeda motor dan memakai dasi lain” yang berarti *caption* yang dihasilkan salah dan tidak menggambarkan situasi yang terjadi di dalam gambar.

Tabel 4.6 Eksperimen dengan Model 2

Gambar	**Deskripsi**	**Keterangan**
	sekelompok anak sedang bermain sepak bola sepak bola	Benar

	seorang pria melompat dengan papan skateboard di arena skateboard di arena skateboard	Benar
	sekelompok orang sedang menerbangkan layang layan	Benar
	seorang pria mengendarai sepeda motor dan memakai dasi lain	Salah
	tiga jerapah	Benar

4.6 Pembuatan User Interface

User interface digunakan untuk mengunggah gambar lalu memprediksi *caption* dari gambar tersebut. Pembuatan *user interface* ini dimulai dari membuat fungsi untuk mengunggah gambar dengan fungsi *def upload_image()* dan mendefinisikan format gambar yang dapat diproses seperti pada Gambar 4.22.

```
def upload_image():
    args = { 'baseball' : 'baseball.jpeg' }

    img_upload  = st.file_uploader(label= 'Upload Image', type =
                                   ['png', 'jpg', 'jpeg','webp'])
    img_open = args['baseball'] if img_upload is None else img_upload

    image = Image.open(img_open)
    rgb_im = image.convert('RGB')
    rgb_im.save("saved_image.jpeg")

    st.image(image,use_column_width=True,caption="Your image")

```

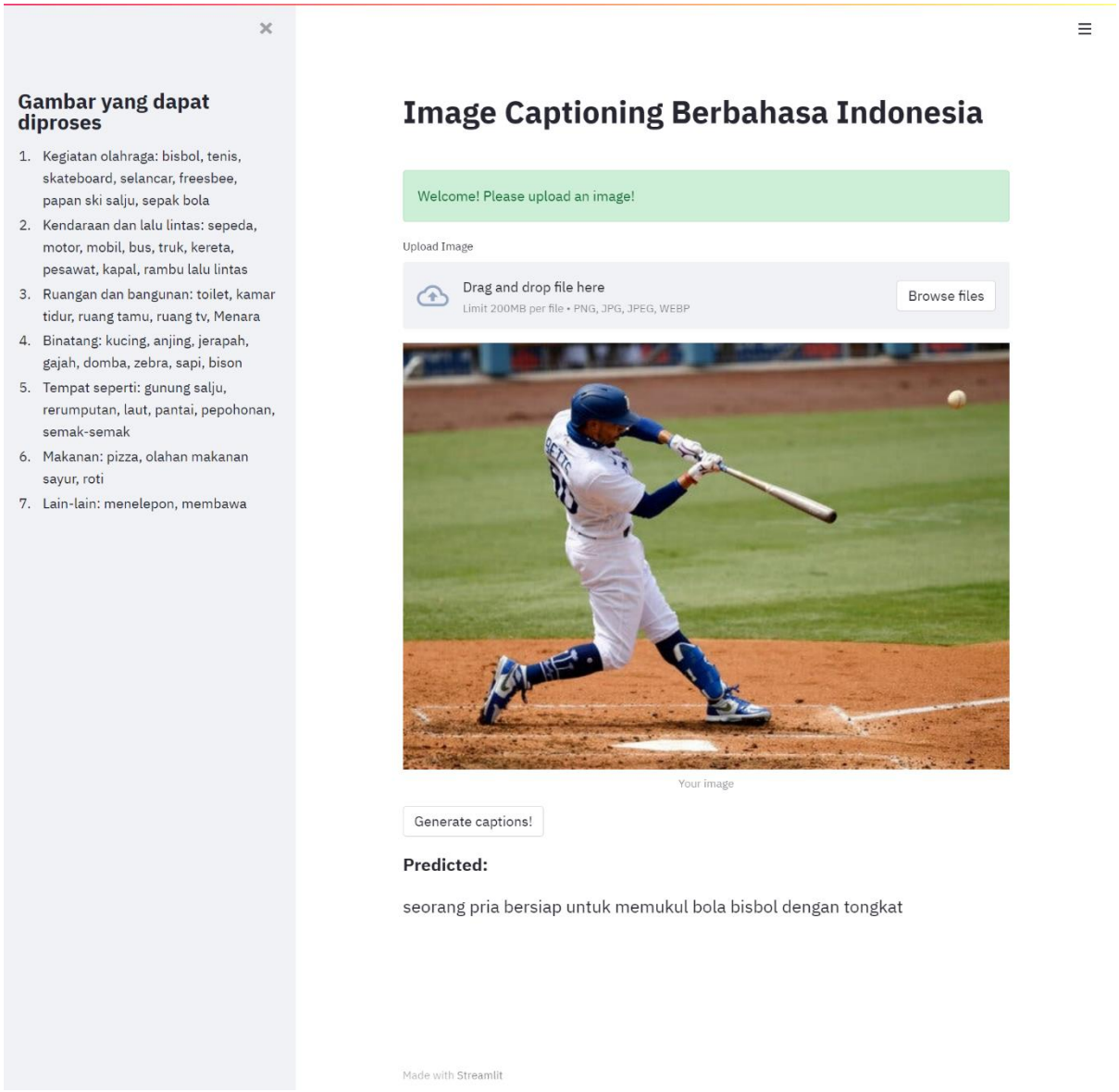
Gambar 4.22 Membuat fungsi *upload image*

Selanjutnya adalah mengubah ukuran *input* dari gambar sesuai pada Gambar 4.3. Setelah proses tersebut, gambar dilewatkan melalui *encoder* dan *decoder* model seperti pada Gambar 4.13, Gambar 4.14, dan Gambar 4.15. Langkah terakhir adalah membuat tampilan untuk pengguna seperti pada Gambar 4.23. Fungsi *load_model()* digunakan untuk memanggil model yang telah disimpan sebelumnya, kemudian *result, attention_plot = evaluate("saved_image.jpeg")* digunakan untuk membentuk keluaran hasil prediksi *caption.*

```
if __name__ == '__main__':
    encoder, decoder = load_model()

    st.title("Image Captioning Berbahasa Indonesia")
    st.text("")
    st.text("")
    st.success("Welcome! Please upload an image!")

    upload_image()

    if st.button('Generate captions!'):
        result, attention_plot = evaluate("saved_image.jpeg")

        styled=f'''<p style="font-size:0px;"><b>Predicted:</b></p>
        <p style="font-size: 20px;">{' '.join(result)}</p>'''
        st.markdown(styled, unsafe_allow_html=True)
```

Gambar 4.23 Membuat tampilan dan memanggil model

Gambar 4.24 merupakan tampilan dari *user interface* yang telah dibuat. Tampilan dibuat dengan tombol "*Browse files*" untuk mengunggah gambar. Gambar baru yang telah diunggah akan menggantikan gambar awal pria memegang tongkat bisbol. Tombol "*Generate captions*!" di bawah gambar digunakan untuk menampilkan hasil prediksi *caption*. Pada bagian kiri, terdapat keterangan jenis gambar yang dapat diproses oleh model yang telah dibuat.



.Gambar 4.24 Tampilan *User Interface*

BAB V
KESIMPULAN

5.1 Kesimpulan

Kesimpulan dari penelitian ini di antaranya:

a. Penelitian menghasilkan dua model dengan dua ekstraksi fitur yang berbeda yaitu InceptionV3 (Model 1) dan VGG16 (Model 2). Kedua model tersebut menggunakan LSTM sebagai *decoder*. Setiap model terbukti mampu untuk menghasilkan *caption* berbahasa Indonesia sesuai dengan gambar.
b. Hasil evaluasi pada Model 1 dengan menggunakan skor BLEU menghasilkan skor BLEU-1,2,3,4 {34.93, 22.55, 14.75, 10.15} dan pada Model 2 menghasilkan skor BLEU-1,2,3,4 {22.02, 14.29, 9.41, 6.28}. Dari hasil skor BLEU yang telah didapat dapat dilihat bahwa Model 1 memiliki rata-rata skor BLEU yang lebih baik dari Model 2. Selain itu kedua model yang dibangun dengan LSTM memiliki sedikit keunggulan dari model yang dibangun dengan GRU, hasil yang didapat model GRU-InceptionV3 pada BLEU-1,2,3,4 adalah {27.03, 16.33, 9.39, 5.78} dan untuk GRU-VGG16 pada BLEU-1,2,3,4 {20.93, 13.93, 9.68, 6.73}.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, penulis berharap penelitian ini dapat dikembangkan oleh peneliti-peneliti lain. Selain itu penulis memiliki beberapa saran untuk penelitian berikutnya:

a. Menggunakan topik yang lebih spesifik pada pemilihan gambar, misalnya pada *domain* wisata, medis, lalu-lintas, olahraga, dll.
b. Memperbanyak *caption* untuk satu gambar dan tidak hanya dilengkapi satu *caption* saja.
c. Penelitian berikutnya diharapkan mampu memberikan penjelasan spesifik dari gambar seperti nama orang ataupun nama kendaraan dalam gambar.
d. Menambah jumlah *dataset* agar menghasilkan performa lebih baik.

DAFTAR PUSTAKA

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU), (1), 2–8. Retrieved from http://arxiv.org/abs/1803.08375

Al-muzaini, H. A., Al-yahya, T. N., & Benhidour, H. (2018). Automatic Arabic image captioning using RNN-LSTM-based language model and CNN. *International Journal of Advanced Computer Science and Applications*, *9*(6), 67–73. https://doi.org/10.14569/IJACSA.2018.090610

Bibi, I., Akhunzada, A., Malik, J., Iqbal, J., Mussaddiq, A., & Kim, S. (2020). A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware. *IEEE Access*, *8*, 129600–129612. https://doi.org/10.1109/ACCESS.2020.3009819

Brownlee, J. (2017). Deep Learning for Natural Language Processing Develop Deep Learning Models for Natural Language in Python. *Machine Learning Mastery*, 414. Retrieved from http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf

Chung, H., & Shin, K. S. (2018). Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability (Switzerland)*, *10*(10). https://doi.org/10.3390/su10103765

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 1–9. Retrieved from http://arxiv.org/abs/1412.3555

Eka Putra, W. S. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, *5*(1). https://doi.org/10.12962/j23373539.v5i1.15696

Fudholi, D. H., Windiatmoko, Y., Afrianto, N., Susanto, P. E., Suyuti, M., Hidayatullah, A. F., & Rahmadi, R. (2021). Image Captioning with Attention for Smart Local Tourism using EfficientNet. *IOP Conference Series: Materials Science and Engineering*, *1077*(1), 012038. https://doi.org/10.1088/1757-899x/1077/1/012038

Gollapudi, S. (2019). *Deep Learning for Computer Vision*. *Learn Computer Vision Using OpenCV*. https://doi.org/10.1007/978-1-4842-4261-2_3

Gu, J., Wang, G., Cai, J., & Chen, T. (2017). An Empirical Study of Language CNN for Image Captioning. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 1231–1240. https://doi.org/10.1109/ICCV.2017.138

Hidayatullah, A. F. (2021). Membuat Aplikasi Web Sains Data dengan Mudah Menggunakan Streamlit - Informatika UII. Retrieved September 11, 2021, from https://informatics.uii.ac.id/2021/03/15/streamlit-membuat-aplikasi-web-sains-data/

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8693 LNCS*(PART 5), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Liu, S., Bai, L., Hu, Y., & Wang, H. (2018). Image Captioning Based on Deep Neural Networks. *MATEC Web of Conferences*, *232*, 1–7. https://doi.org/10.1051/matecconf/201823201052

Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F., & Zhang, Y. (2018). Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. *Remote Sensing*, *10*(7). https://doi.org/10.3390/rs10071119

Manaswi, N. K. (2018). *Deep Learning with Applications Using Python*. *Deep Learning with Applications Using Python*. https://doi.org/10.1007/978-1-4842-3516-4

Nugraha, A. A., Arifianto, A., & Suyanto. (2019). Generating image description on Indonesian language using convolutional neural network and gated recurrent unit. *2019 7th International Conference on Information and Communication Technology, ICoICT 2019*, 1–6. https://doi.org/10.1109/ICoICT.2019.8835370

Nursikuwagus, A., Munir, R., & Khodra, M. L. (2020). Image Captioning menurut Scientific Revolution Kuhn dan Popper, *10*(1994), 110–121. https://doi.org/10.34010/jamika.v10i2

Pa Pa Aung, S., Pa Pa, W., & Nwe, T. L. (2020). Automatic Myanmar Image Captioning using CNN and LSTM-Based Language Model. *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, (May), 139–143. Retrieved from https://www.aclweb.org/anthology/2020.sltu-1.19

Papineni, K., Roukos, S., Ward, T., Zhu, W., & Heights, Y. (2001). IBM Research Report Bleu : a Method for Automatic Evaluation of Machine Translation. *Science*, *22176*(February), 1–10. https://doi.org/10.3115/1073083.1073135

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. *Advances in Neural Information Processing Systems*, (Nips), 2360–2368.

Putra, J. W. G. (2019). Pengenalan Konsep Pembelajaran Mesin dan Deep Learning. *Computational Linguistics and Natural Language Processing Laboratory*, *4*, 1–235. Retrieved from https://www.researchgate.net/publication/323700644

Rohim, A., Sari, Y. A., & Tibyani. (2019). Convolution neural network (cnn) untuk pengklasifikasian citra makanan tradisional. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(7), 7038–7042. Retrieved from http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5851/2789

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Tan, J. H., Chan, C. S., & Chuah, J. H. (2019). Image Captioning with Sparse Recurrent Neural Network, (October). Retrieved from http://arxiv.org/abs/1908.10797

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., … Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *32nd International Conference on Machine Learning, ICML 2015*, *3*, 2048–2057.

You, Q., Jin, H., Wang, Z., Fang, C., Luo, J., Ave, P., & Ca, S. J. (2016). Image Captioning with Semantic Attention. *Rbi*. https://doi.org/10.1039/c6tc00314a

Zeng, A., Sun, X., Huang, F., Liu, M., Xu, Q., & Lin, S. (2020). SRNet: Improving Generalization in 3D Human Pose Estimation with a Split-and-Recombine Approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12359 LNCS*(July), 507–523. https://doi.org/10.1007/978-3-030-58568-6_30

LAMPIRAN