IMPLEMENTASI ARSITEKTUR *TRANSFORMER* PADA *IMAGE CAPTIONING* DENGAN BAHASA INDONESIA

Disusun Oleh:

N a m a	: Umar Abdul Aziz Al-Faruq
NIM	: 17523215

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI ARSITEKTUR *TRANSFORMER* PADA *IMAGE CAPTIONING* DENGAN BAHASA INDONESIA

TUGAS AKHIR

Disusun Oleh:

N a m a : Umar Abdul Aziz Al-Faruq
NIM : 17523215

Yogyakarta, 20 Juli 2021

Pembimbing,

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

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI ARSITEKTUR *TRANSFORMER* PADA *IMAGE CAPTIONING* DENGAN BAHASA INDONESIA

TUGAS AKHIR

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

Yogyakarta, 20 Juli 2021

Tim Penguji

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

Anggota 1

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

Anggota 2

Dr. Syarif Hidayat, S.Kom., MIT. _____

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 : Umar Abdul Aziz Al-Faruq

NIM : 17523215

Tugas akhir dengan judul:

IMPLEMENTASI ARSITEKTUR *TRANSFORMER* PADA *IMAGE CAPTIONING* DENGAN BAHASA INDONESIA

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian 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 apapun.

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

Yogyakarta, 20 Juli 2021

(Umar Abdul Aziz Al-Faruq)

HALAMAN PERSEMBAHAN

Assalamu'alaikum Warahmatullahi Wabarakatuh Alhamdulillahirobbil'alamin, puji syukur kepada Allah SWT berkat izin, karunia dan inayah-Nya yang telah memberikan kemudahan, kelancaran serta keberkahan selama proses pembuatan hingga penyelesaian Tugas Akhir ini tepat pada waktunya. Semoga keberhasilan ini menjadi satu langkah awal untuk masa depanku dalam meraih cita-cita dan menjadi seseorang yang hebat dan bermanfaat, Allahumma Aamiin.

Terimakasih untuk kedua orang tua saya, Umi dan Abi yang telah memberikan kasih sayang dan segalanya kepada saya, proses yang sudah saya lalui selama ini tidak akan berhasil jika tidak ada dukungan dari Umi dan Abi. Terimakasih juga telah memberikan kesempatan kepada saya untuk menuntut ilmu di UII, semoga ilmu yang saya dapatkan bermanfaat dan nilai-nilai dari UII setelah saya lulus dapat saya wujudkan di kehidupan saya.

Terima kasih untuk Bapak Dhomas Hatta Fudholi, S.T, M.Eng, Ph.D., selaku pembimbing, Ketua Program Studi Informatika dan Para Dosen Informatika yang selalu membimbing dan mengajarkan ilmu-ilmu pengetahuan hingga pelajaran hidup yang sangat berharga untuk bekal kehidupan yang akan mendatang.

Untuk teman-temanku terutama anak-anak HBS (Ardi, Amin, Akbar, Mulia, Rama, Jela, Rafi, Afif, Fazma, Umar, Alif, Axel, Gozy, Andri, Teguh, Dika, Nopal, Ivan dan lainnya) yang saya banggakan, terimakasih atas segalanya kisah kehidupan pada masa perkuliahan selama empat tahun ini semoga kita semua dapat mencapai cita-cita yang kita inginkan dan semoga setelah lulus dari perkuliahan silaturahmi tetap harus kita jaga dengan baik.

Semoga Allah mengganti kebaikan-kebaikan yang kalian berikan dengan kebaikan yang lebih lagi. Semoga diberikan kelancaran, kemudahan, kebarokahan, kesehatan dan kebahagiaan untuk kita semua. Aamiin.

HALAMAN MOTO

“Your growth is your responsibility” – Iqbal Farabi

“If you don’t think you can be strong enough to be untouchable, make companions instead” – Yoo Joonghyuk

KATA PENGANTAR

Alhamdulillah segala puji bagi Allah SWT yang telah memberikan Rahmat dan Hidayah-Nya serta berbagai kemudahan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Shalaway serta salam tak lupa pula penulis ucapkan kepada Nabi Muhamad SAW, beserta keluarga, sahabat dan para pengikutnya hingga akhir zaman.

Skripsi berjudul "Implementasi Arsitektur *Transformer* pada *Image Captioning* dengan Bahasa Indonesia" yang disusun untuk memenuhi persyaratan guna mendapatkan gelar Sarjana Komputer (S.Kom) pada Program Studi Informatika di Universitas Islam Indonesia.

Selama proses penyusunan skripsi ini, penulis mendapatkan banyak bimbingan, masukan motivasi serta dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan banyak terima kasih kepada:

1. Kedua orang tua penulis, Umi dan Abi yang senantiasa mendo'akan, memberikan perhatian dan kasih saying kepada penulis serta memberikan motivasi dan dukungan agar penulis dapat menyelesaikan kuliah dengan baik serta menjadi orang yang sukses dan bermanfaat di masa depan.
2. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Dosen Pembimbing yang telah memberikan banyak saran, arahan dan masukan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
4. Seluruh Dosen dan Staff Karyawan Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang telah memberikan banyak ilmu, dukungan dan bantuan selama masa perkuliahan.
5. Teman-teman Harapan Bangsa (Arap, Ardhy, Nopal, Akbar, Fardika, Dafa) dan yang lainnya tak bisa penulis sebutkan satu per satu, terima kasih atas dukungan dan waktu kebersamaanya selama masa perkuliahan yang kemudian menjadikan Jogja semakin dirindu-rindukan. Semoga kesuksesan selalu mengiringi kalian semua.
6. Semua pihak yang telah membantu penulis dengan penuh keikhlasan, yang tidak dapat disebutkan satu persatu, terima kasih atas segala bantuan yang telah diberikan kepada penulis.

Pada akhirnya, penulis mengharapkan semoga skripsi ini dapat bermanfaat bagi penulis dan semua pihak yang berkenan menelaah di kemudian hari. Semoga Allah SWT memberikan

limpahan rahmat, karunia dan balasan yang lebih baik atas kebaikan semua pihak yang secara langsung maupun tidak langsung membantu terwujudnya skripsi ini, Aamiin ya Rabbal alamin.

Yogyakarta, 20 Juli 2021

(Umar Abdul Aziz Al-Faruq)

SARI

Penelitian image captioning untuk menghasilkan deskripsi yang baik pada gambar dalam Bahasa Inggris banyak dilakukan. Sedikit penelitian yang ditemukan mengenai *image captioning* untuk menghasilkan deskripsi gambar dalam Bahasa Indonesia. Penelitian dengan Bahasa Indonesia semuanya menggunakan model *sequence-to-sequence* dan *attention mechanism*. Kedua model telah memberikan hasil yang baik namun terdapat kekurangan yang krusial. Model *seq2seq* memberikan performa yang buruk saat berhadapan dengan kalimat panjang sedangkan *attention mechanism* memakan banyak *resource* karena mengandalkan RNN. Terinspirasi oleh keberhasilan arsitektur *Transformer* dalam *machine translation*, penelitian ini akan berfokus dalam mengembangkan arsitektur *Transformer* untuk diimplementasikan pada *image captioning* Bahasa Indonesia. Dibandingkan penelitian *image captioning* Bahasa Indonesia yang sudah ada, arsitektur *Transformer* bisa melakukan komputasi paralel sehingga mampu mengakselerasi proses *training* karena *Transformer* hanya menggunakan *attention mechanism* tanpa mengandalkan RNN. Penelitian ini menggunakan dataset MS COCO 2014 yang sudah diterjemahkan ke dalam Bahasa Indonesia. Penelitian ini mendapatkan skor rata-rata BLEU-{1,2,3,4} sebesar {78.05, 68.21, 61.89, 52.09}, skor tersebut menunjukkan bahwa arsitektur *Transformer* melampaui hasil matrik evaluasi BLEU yang didapatkan oleh kedua model sebelumnya secara signifikan.

Kata kunci: *Image Captioning Bahasa Indonesia, Attention Mechanism, Transformer*.

GLOSARIUM

Artificial Intelligence	AI adalah bagian dari ilmu computer. Program ini menggunakan bahasa yang sama dengan sistem konvensional, tetapi dengan logika yang berbeda.
Deep learning	DL adalah bagian dari metodologi *machine learning* yang menggunakan jaringan syaraf tiruan. ML hanya berfokus pada cara computer belajar tanpa terprogram secara spesifik.
Machine *learning*	ML merupakan salah satu bagian dari kecerdasan buatan yang membuat mesin dapat melakukan inferensi terhadap data yang diberikan
Inference	Inferensi adalah proses menggunakan model yang sudah dilatih untuk membuat prediksi terhadap data yang belum terlihat, dalam hal ini adalah data *validation*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Image captioning adalah kemampuan mendeskripsikan isi sebuah gambar dalam bentuk kalimat (S. He, W.Liao et al., 2020). Kemampuan ini memerlukan metode dari dua bidang *artificial intelligence*, yaitu *computer vision* untuk memahami isi gambar yang diberikan dan *natural language processing* untuk mengubah isi pada gambar menjadi bentuk kalimat. Karena kemajuan pada kedua bidang tersebut, membuat *image captioning* mendapat banyak perhatian beberapa tahun terakhir. Karena kemajuan tersebut *image captioning* mulai mengimplemntasikan metode-metode dari *NLP*, salah satunya *attention mechanism*.

Terinspirasi dari pengembangan *machine translation*, *attention mechanism* telah banyak digunakan dalam *framework encoder decoder* untuk *imcage captioning* dan mencapai hasil yang memuaskan (L. Huang, W. Wang et al., 2019). Framework ini menggunakan CNN (*Convolutional Neural Network*) + RNN (*Recurrent Neural Network*), di mana CNN berperan sebagai *image encoder* yang mengekstrak *region-based* fitur visual dari input gambar dan RNN berperan sebagai *caption decoder* untuk menghasilkan kalimat (J. Yu, J. Li, Z. Yu et al., 2019).

Dengan perkembangan *machine translation*, munculah arsitektur baru, yaitu *transformer* (A. Vaswani et al., 2017). Arsitektur ini menggunakan *self-attention mechanism* dan telah menjadi *state-of-the-art* pada bidang NLP (*Natural Language Processing*). *Transformer* mengakselerasi proses pelatihan dan menggunakan *self-attention* untuk menarik dependensi global antara input dan output yang berbeda. Pada *machine transtalation*, *transformer* bisa melakukan *training* secara signifikan lebih cepat daripada arsitektur yang menggunakan lapisan *recurrent* atau *convolution*.

Perkembangan ini membuat model *image captioning* mulai mengimplementasikan arsitektur *transformer*, seperti *Meshed-Memory Transformer for Image Captioning* (M. Cornia, M. Stefanini et al., 2020), *Boosted Transformer for Image Captioning* (J. Li, P. Yao et al., 2019), dan *Image Captioning through Image Transformer* (S. He, W.Liao et al., 2020). Implementasi tersebut memberikan hasil yang memuaskan dalam *image captioning* berbahasa Inggris dengan mendapatkan skor matrik evaluasi yang tinggi. Karena keberhasilan penelitian tersebut, penulis akan mengimplementasikan arsitektur *transformer* ke *image captioning* dalam Bahasa Indonesia.

Pada penelitian *image captioning* dalam Bahasa Indonesia, penulis mendapatkan tiga penelitian, yaitu Keterangan Gambar Otomatis Berbahasa Indonesia dengan CNN dan LSTM (Amiin, 2021), *Generating image description on Indonesian language using convolutional neural network and gated recurrent unit* (Nugraha, 2019), dan *Image Captioning with Attention for Smart Local Tourism using EfficientNet* (Fudholi, 2021). Ketiga penelitian yang didapatkan menggunakan metode *CNN-LSTM/GRU* dan *attention mechanism*. Dari penelitian yang didapatkan tidak ada penlitian yang menggunakan arsitektur *transformer*.

Berangkat dari latar belakang tersebut, Penelitian ini akan menggunakan arsitektur *transformer* sebagai basis model image captioning untuk menghasilkan deskripsi gambar dalam Bahasa Indonesia. Hasil deskripsi diharapkan memiliki kalimat sealami mungkin seperti bahasa manusia. Evaluasi metrik yang akan digunakan untuk penelitian ini adalah BLEU, untuk menilai seberapa bagus deskripsi yang dihasilkan. Penelitian ini diharapkan mendapatkan hasil yang melampaui hasil penelitian *image captioning* dalam Bahasa Indonesia yang sudah ada.

1.2 Rumusan Masalah

Transformer telah menjadi arsitektur pengembangan model Image Captioning yang terbukti lebih baik daripada arsitektur RNN. Oleh karena itu, ada kebutuhan untuk mengambangkan model Image Captioning bahasa Indonesia dengan menggunakan Arsitektur ini. Kebutuhan ini juga didukung oleh studi literatur yang dilakukan karena belum adanya penelitian *Image Captioning* bahasa Indonesia dengan *transformer*. Model Image Captioning bahasa Indonesia akan dicoba untuk dikembangkan dengan hyperparameter tuning untuk mendapatkan model terbaik. Evaluasi model dan pengembangan aplikasi juga akan dilakukan untuk memberikan bukti konseptual dari model yang dikembangkan.

1.3 Batasan Masalah

Untuk memperoleh gambaran yang lebih jelas, maka diberikan batasan-batasan sebagai berikut:

a. Menggunakan dataset MSCOCO 2014 yang diterjemahkan ke dalam Bahasa Indonesia.
b. Gambar pada dataset berupa gambar foto kamera bukan gambar kartun ataupun lukisan.
c. Menggunakan dua model ekstraksi fitur gambar, yaitu *InceptionV3* dan *EfficientNet*.
d. Matrik evaluasi yang digunakan hanya BLEU.

1.4 Tujuan Penelitian

Mengimplemantasikan arsitektur *Transformer* pada *image captioning* berbahasa Indonesia guna mendapatkan hasil matrik evaluasi BLEU yang melampaui penelitian sebelumnya.

1.5 Manfaat Penelitian

Diharapkan penelitian ini dapat bermanfaat:

a. Untuk meningkatkan pengetahuan dan kemampuan penulis mengenai *artificial intelligence*.
b. Dapat digunakan sebagai acuan untuk penelitian berikutnya yang memiliki kesamaan dengan penelitian ini.

1.6 Sistematika Penulisan

Sistematika yang dibuat pada tugas akhir ini akan dibagi dalam lima bagian, yaitu:

a. BAB I PENDAHULUAN

   Dalam bab ini membahas mengenai latar belakang penelitian, perumusan masalah, batasan masalah, tujuan dan manfaat penelitian yang merupakan gambaran menyeluruh dari penelitian skripsi ini.

b. BAB II LANDASAN TEORI

   Dalam bab ini membahas mengenai berbagai teori yang mendasari analisis permasalahan yang berhubungan dengan pembahasan.

c. BAB III METODE PENELITIAN

   Bab ini berisi pembahasan atau pemaparan metode yang penulis gunakan dalam pencarian data maupun perancangan sistem yang dilakukan pada penelitian.

d. BAB IV HASIL DAN PEMBAHASAN

   Bab ini membahasa mengenai hasil dan pembahasan rancangan pembuatan *image captioning* dengan menggunakan arsitektur *transformer*.

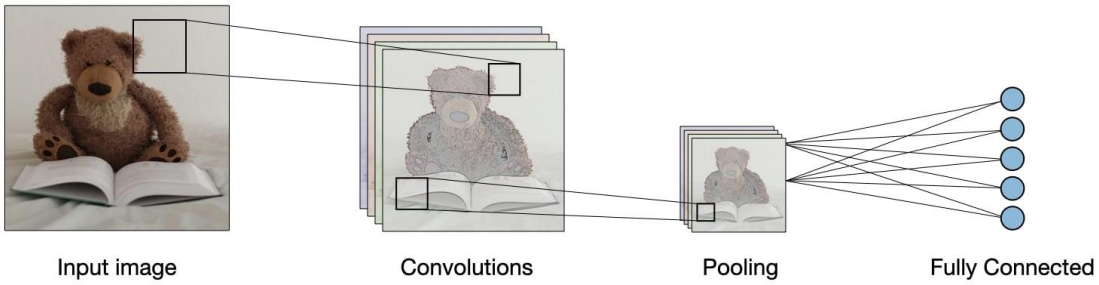
e. BAB V PENUTUP

Bab ini berisi kesimpulan dari hasil pembahasan seluruh bab serta saran-saran yang kiranya dapat diperhatikan dan dipertimbangkan untuk penelitian dimasa mendatang.

BAB II
LANDASAN TEORI

2.1 Convolutional Neural Networks

Convolutional Neural Network yang juga dikenal sebagai CNN adalah jenis *neural network* yang umumnya terdiri dari beberapa lapisan, yaitu lapisan *convolutional*, lapisan *pooling*, dan lapisan *fully-connected* (S. Albawi et al., 2018). Pada Gambar 2.2 merupakan contoh arsitektur CNN yang diterapkan pada gambar boneka beruang.



Gambar 2.1 Contoh arsitektur *Convolutional Neural Networks* yang disederhanakan
Sumber: Afshine (2019)

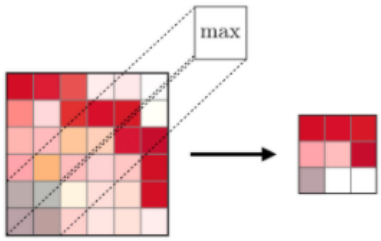
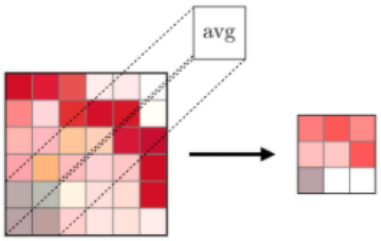
2.1.1 Jenis Lapisan

Lapisan Konvolusi

Menggunakan filter yang melakukan operasi *convolution* saat memindai input sehubungan dengan dimensinya. Hyperparameternya mencakup ukuran filter F dan langkah S. O yang dihasilkan dinamakan *feature map* atau *activation map*. Tujuan dari operasi ini adalah untuk mengekstrak *high-level features* seperti tepi, warna, orientasi gradient, dsb.

Pooling

Operasi downsampling, diterapkan setelah lapisan konvolusi untuk mengurangi kompleksitas untuk lapisan selanjutnya. Selain itu berguna untuk mengekstrak *dominant features* yang melakukan beberapa invariansi spasial. *Pooling* ada dua jenis, yaitu *max pooling* dan *average pooling*. Penjelasan kedua jenis ini dapat dilihat pada Gambar 2.3.

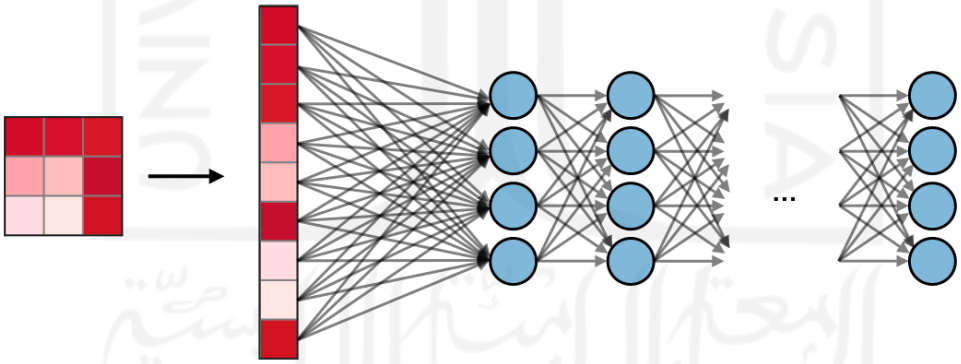
Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

Gambar 2.2 Dua jenis *pooling* pada *Convolutional Neural Network*

Sumber: Afshine (2019)

Fully-Connected

Beroperasi pada input yang diratakan di mana setiap input terhubung ke seluruh neuron. Lapisan ini ditemukan pada akhir arsitektur CNN dan bisa digunakan untuk mengoptimalkan tujuan seperti skor kelas. Ilustrasi dari *fully-connected* dapat dilihat pada Gambar 2.4.



Gambar 2.3 Lapisan *fully-connected*

Sumber: Afshine (2019)

2.2 InceptionV3

Inception-v3 adalah arsitektur *Convolutional Neural Network* dari keluarga *Inception* yang membuat beberapa perbaikan termasuk menggunakan *Label Smoothing*, *Factorizing Convolution*, dan *Auxiliary* Classifier tambahan untuk menyebarkan informasi label ke bawah

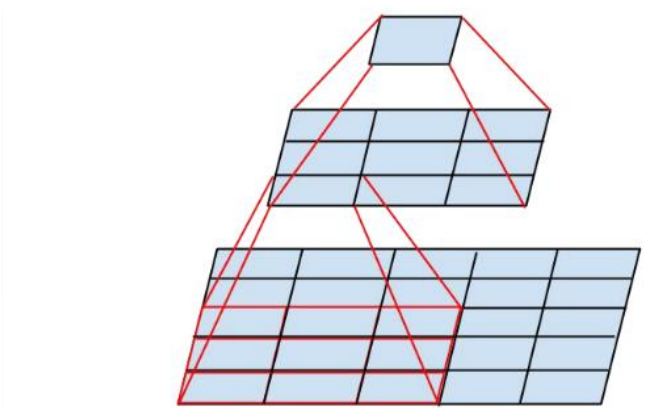
jaringan (bersamaan dengan penggunaan *batch normalization* untuk lapisan di *sidehead*) (C. Szegedy, V. Vanhoucke et al., 2016).

2.2.1 *Factorizing Convolution*

Tujuan dari *factorizing convolution* adalah untuk mengurangi jumlah koneksi atau parameter tanpa menurunkan efisiensi jaringan.

Factorizing into Smaller Convolution

Dua *konvolusi* 3 x 3 menggantikan satu konvolusi 5 x 5, seperti pada Gambar 2.6.



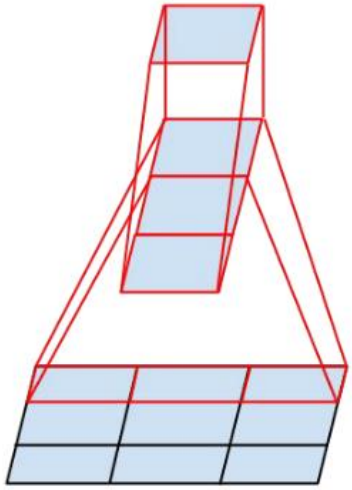
Gambar 2.4 Dua konvolusi 3 x 3 menggantikan satu konvolusi 5 x 5

Sumber: Szegedy (2016)

Dengan menggunakan 1 lapisan filter 5×5, maka jumlah parameter = $5 \times 5 = 25$. Dengan menggunakan 2 lapisan filter 3×3, maka jumlah parameter = $3 \times 3 + 3 \times 3 = 18$. Jumlah parameter berkurang sebesar 28%.

Factorization into Asymmetric Convolutions

Satu konvolusi 3×1 diikuti dengan satu konvolusi 1×3 menggantikan satu konvolusi 3×3, seperti pada Gambar 2.7.

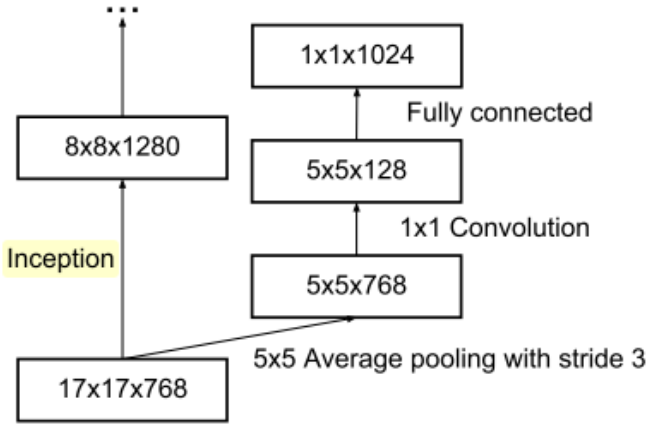


Gambar 2.5 Satu konvolusi 3x1 diikuti dengan satu konvolusi 1x3 menggantikan satu konvolusi 3x3

Sumber: Szegedy (2016)

2.2.2 *Auxiliary Classifier*

Hanya 1 *auxiliary classifier* yang digunakan di atas lapisan 17×17 terakhir, dari pada menggunakan 2 *auxiliary classifier*. Tujuan dari *auxiliary classifier* adalah digunakan sebagai *regularizer*. Pada Gambar 2.8 merupakan penyederhanaan dari *Auxiliary Classifier*.

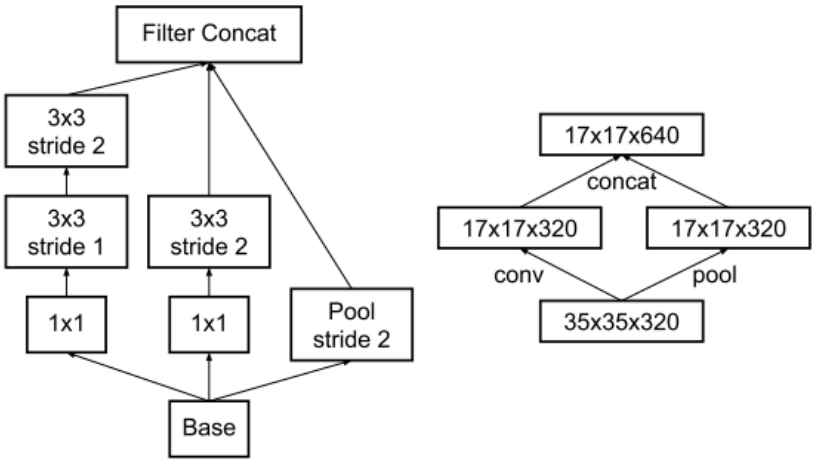


Gambar 2.6 Auxiliary classifier yang digunakan di atas lapisan 17x17 terakhir

Sumber: Szegedy (2016)

2.2.3 *Efficient Grid Size Reduction*

Gambar 2.9 adalah modul *Inception* yang mengurangi ukuran *grid* bersamaan dengan memperluas *filter banks*.

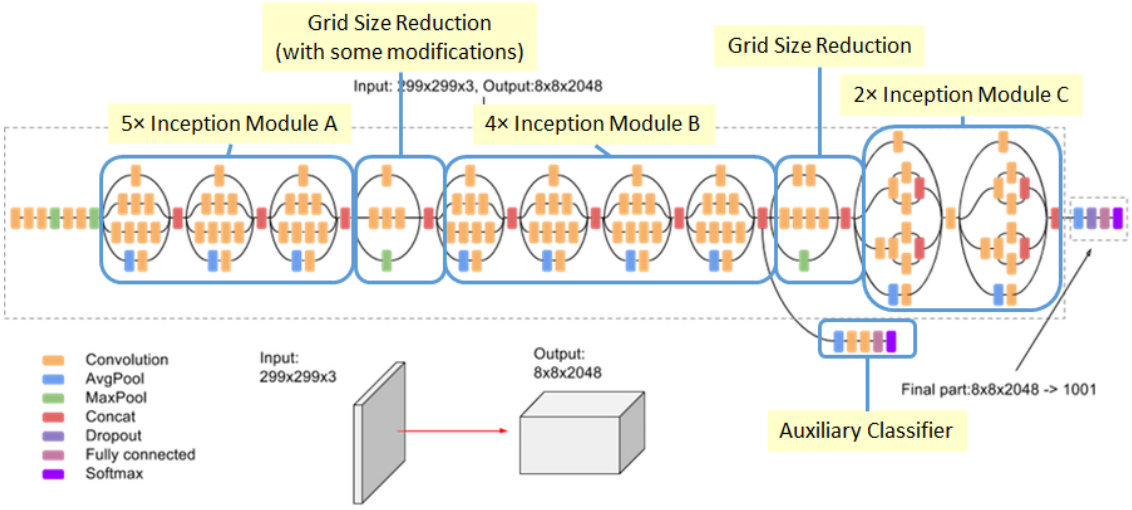


Gambar 2.7 Detail arsitektur (kiri) efficient grid size reduction (kanan)

Sumber: Szegedy (2016)

2.2.4 Arsitektur *InceptionV3*

Dari penjelasan-penjelasan sebelumnya, kemudian disatukan ke dalam sebuah arsitektur yang utuh seperti yang ditunjukkan pada Gambar 2.10.



Gambar 2.8 Arsitektur *InceptionV3*

Sumber: Tsang (2018)

2.2.5 *Label Smoothing as Regularization*

Tujuan dari *label smoothing* adalah mencegah *logit* paling besar menjadi lebih besar dari yang lainnya.

$$q' = (1-\epsilon)\delta_{k,y} + \frac{\epsilon}{K} \quad (2.1)$$

Di mana $\varepsilon = 0.1$ yang merupakan sebuah *hyperparameter* dan $K = 1000$ yang merupakan jumlah kelas.

2.3 EfficientNet

EfficientNet adalah arsitektur *Convolutional Neural Network* dan metode penskalaan yang secara seragam menskalakan semua dimensi kedalaman/lebar/resolusi menggunakan *compound coefficient*. Tidak seperti praktik konvensional yang bebas menskalakan factor-faktor ini, metode penskalaan *EfficientNet* secara seragam menskalakan lebar, kedalaman, dan resolusi jaringan dengan koefisien penskalaan tetap. *EfficientNet* menggunakan koefisien gabungan untuk menskalakan lebar, kedalaman, dan resolusi jaringan secara seragam yang berprinsip (M. Tan, Q. Le, 2019).

Metode penskalaan gabungan dibenarkan oleh intuisi bahwa jika input gambar lebih besar, maka jaringan membutuhkan lebih banyak jaringan untuk meningkatkan bidang reseptif dan lebih banyak *channels* untuk menangkap pola berbutir halus pada gambar yang lebih besar (M. Tan, Q. Le, 2019).

2.3.1 *Compund Model Scaling*

Formulasi Masalah

Lapisan ConvNet sering dibagi ke dalam banyak tahap dan semua lapisan di setiap tahapnya membagikan arsitektur yang sama. Karenanya formula *ConvNet* bisa didefinisakan sebagai.

$$\mathcal{N} = repeat(i = 1 \dots s)\mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i\rangle}\right) \quad (2.2)$$

Di mana $\mathcal{F}_i^{L_i}$ menunjukkan lapisan F_i diulang L_i kali dalam tahap i, $\langle H_i, W_i, C_i\rangle$ menunjukkan bentuk input tensor X dari lapisan (M. Tan, Q. Le, 2019).

Tidak *seperti* design *ConvNet* biasa yang Sebagian besar berfokus mencari lapisan arsitektur terbaik, penskalaan model mencoba memperluas panjang (L_i), lebar (C_i), dan/atau resolusi (H_i, W_i) jaringan tanpa mengubah F_i pada jaringan dasar. Dengan memperbaiki F_i, penskalaan model menyederhankan masalah desain untuk sumber daya konstrain baru. Target

EfficientNet adalah untuk memaksimalkan akurasi model untuk setiap kendala sumber daya yang diberikan, yang mana bisa diformulasikan sebagai:

$$\max_{d,w,r} Accuracy\big(\mathcal{N}(d,w,r)\big) \quad (2.3)$$

$$\mathcal{N}(d,w,r) = repeat(i = 1 \dots s)\mathcal{F}_i^{d\cdot L_i}\big(X_{\langle r\cdot H_i, r\cdot W_i, w\cdot C_i\rangle}\big) \quad (2.4)$$

Di mana w, d, r adalah koefisien untuk penskalaan lebar, kedalaman, dan resolusi jaringan (M. Tan, Q. Le, 2019).

Compound Scaling

EfficientNet mengusulkan teknik penskalaan sederhana namun sangat efektif yang menggunakan *compound coefficient* Φ untuk menskalakan lebar, kedalaman, dan resolusi jaringan secara seragam.

$$depth: d = \alpha^{\phi} \quad (2.5)$$

$$width: w = \beta^{\phi} \quad (2.6)$$

$$resolution: r = \gamma^{\phi} \quad (2.7)$$

$$s.t. \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \quad (2.8)$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \quad (2.9)$$

Di mana α, β, γ adalah konstan yang bisa ditentukan dengan *small grid search*. Secara intuitif, Φ adalah koefisien yang ditentukan pengguna yang menontrol berapa banyak sumber daya yang tersedia sedangkan α, β, γ menentukan cara menetapkan sumber day aini masing-masing ke kedalaman, lebar, dan resolusi jaringan (M. Tan, Q. Le, 2019).

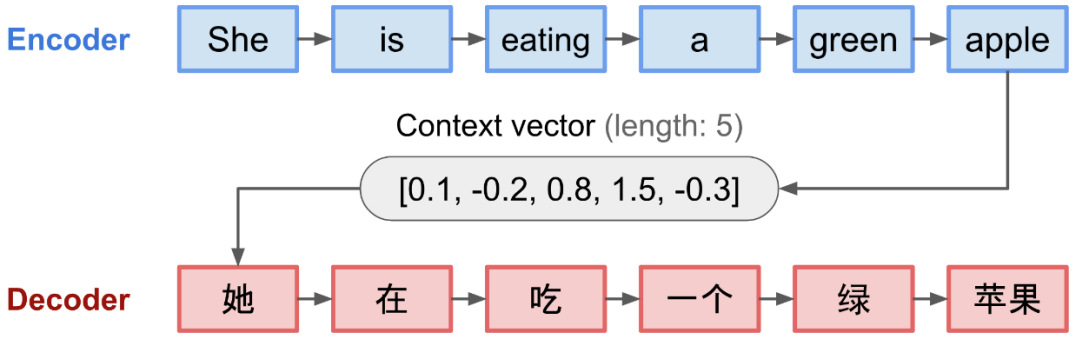
2.4 Model *Sequence to Sequence*

Diperkenalkan pertama kali pada tahun 2014 oleh Google. Model ini menggunakan metode *multilayered Long Short-Term Memory (LSTM)* untuk memetakan urutan input ke vektor konteks dengan dimensi yang tetap dan menggunakan *deep LSTM* lain untuk men-*decode* urutan target dari vektor konteks (Sutskever et al., 2014). Model ini bertujuan untuk memetakan *fixed-length input* dengan *fixed-length output* di mana panjang dari input dan output mungking berbeda (Kostadinov, 2019).

Model seq2seq ini menggunakan arsitektur *endocer-decoder*, terdiri dari:

a. *Encoder* memproses urutan input dan memadatkan informasi ke dalam vektor konteks (*sentence embedding*) dengan Panjang urutan yang tetap.
b. *Decoder* diinisialisasi dengan vector untuk memancarkan output yang diubah. Pekerjaan awal hanya menggunakan bagian akhir dari jaringan *encoder* sebagai bagian awal *decoder*.

Encoder dan *decoder* adalah RNN (*Recurrent Neural Network*), contohnya menggunakan LSTM atau GRU seperti pada Gambar 2.11 (Weng, 2018).



Gambar 2.9 *Encoder-decoder sequence to sequence model, menerjemahkan kalimat "she is eating a green apple" to Chinese*
Sumber: Lilian (2018)

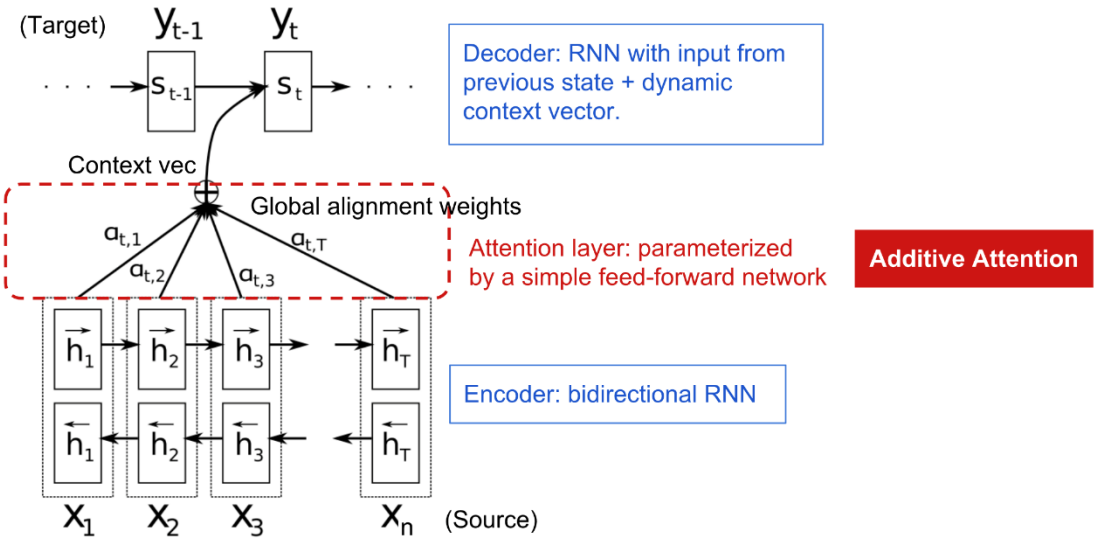
Kelemahan dari model seq2seq ini adalah ketidakmampuannya mengingat kalimat panjang. Sering kali model ini melupakan bagian pertama setelah selesai memproses seluruh masukan. Untuk mengatasi masalah ini lahirlah *Attention Mechanism* (Bahdanau et al., 2015).

2.5 *Attention Mechanism*

Attention adalah *interface* yang menghubungkan *encoder* dan *decoder* yang memberikan *decoder* dengan informasi penting dari setiap bagian tersembunyi *encoder*. Dengan *framework* ini, model bisa secara selektif focus pada bagian-bagian penting dari urutan input dan mempelajari hubungan diantaranya. Ini membantu model mengatasi secara efisien dengan input kalimat yang panjang (Chauhan, 2021).

Attention mechanism lahir untuk membantu mengingat sumber kalimat yang panjang dalam *neural machine translation* (NMT). Daripada membangun satu vector konteks dari bagian akhir milik *encoder*, *attention* membuat jalan pintas antara vector konteks dan seluruh

sumber input. Penyesuaian antara sumber dan target dipelajari dan dikendalikan oleh vektor konteks (Weng, 2018). Contoh penerapan *attention mechanism* menggunakan model *encoder* dan *decoder* dapat dillihat pada Gambar 2.12.



Gambar 2.10 *Model encoder-decoder dengan additive attention mechanism*

Sumber: Lilian (2018)

"*Neural Machine Translation by Jointly Learning to Align and Translate"* (Bahdanau et al., 2015). Makalah ini yang mendasari makalah terkenal "*Attention is All You Need"* (Vaswani et al., 2017) dengan *transformer* yang merevolusi era *deep learning* yang membawa konsep memproses kata secara paralel daripada memprosesnya secara berurutan.

2.5.1 Kategori *Attention Mechanism*

Dengan bantuan *attention,* ketergantungan antara urutan sumber dan target tidak dibatasi oleh jarak antara lagi. *Attention* memberikan peningkatan besar dalam bidang *natural language processing* dan bidang *computer vision* (Xu et al. 2015) dan peneliti mulai mengekplorasi bentuk lain dari *attention mechanism* (Luong, et al., 2015; Britz, et al., 2017; Vaswani et al., 2017). Pada Tabel 2.2 menunjukkan beberapa nama yang menggunakan *attention mechanism*, sedangkan pada Tabel 2.3 menerangkan kategori *attention mechanism* dengan lingkup yang lebih luas.

Tabel 2.1 Beberapa *attention mechanism* yang popular

Nama	Fungsi skor keselarasan	Citasi
Content-base attention	$score(s_t, h_i) = cosine[s_t, h_i]$	Graves2014
Additive	$score(s_t, h_i) = v_a tanh(W_a[s_t; h_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = softmax(W_a s_t)$	Luong2015
General	$score(s_t, h_i) = s_t^T W_a h_i$	Luong2015
Dot-Product	$score(s_t, h_i) = s_t h_i$	Luong2015
Scaled Dot-Product	$score(s_t, h_i) = \frac{s_t h_i}{\sqrt{n}}$	Vaswani2017

Tabel 2.2 Kategori *attention mechanism* yang lebih luas

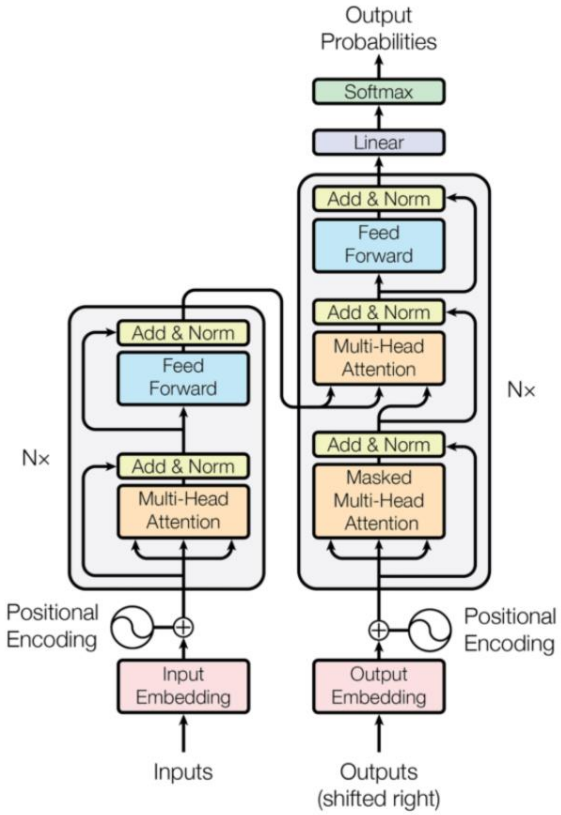
Nama	Definisi	Citasi
Self-Attention	Menghubungkan posisi yang berbeda dari urutan input yang sama. Secara teori *self-attention* bisa mengadopsi semua skor fungsi pada Tabel 1, tapi hanya mengganti urutan target dengan urutan input yang sama	Cheng2016
Global/Soft	*Attention* ditempatkan ke seluruh ruang input	Xu2015
Local/Hard	*Attention* ditempatkan ke sebagian ruang input	Xu2015; Luong2015

2.5.2 *Self Attention*

Self-attention, dikenal juga sebagai *intra-attention*, adalah *attention mechanism* yang menghubungkan posisi berbeda dari satu urutan untuk menghitung representasi dari urutan yang sama. Ini telah ditunjukan sangat bermanfaat dalam *machine reading*, *abstractive summarization*, atau *image captioning*.

2.6 *Transformer*

"*Attention is All You Need*" (Vaswani et al., 2017) adalah salah satu makalah paling berpengaruh. Makalah ini memperkenalkan arsitektur baru yang disebut *Transformer*. Arsitektur ini sepenuhnya dibangun di atas *self-attention mechanism* tanpa mengandalkan *Recurrent Network* (GRU, LSTM). *Transformer* adalah arsitektur untuk mengubah satu urutan ke urutan lain dengan bantuan *encoder* dan *decoder*. Model dari arsitektur *transformer* dapat dilihat pada Gambar 2.13.



Gambar 2.11 Transformer – model arsitektur

Sumber: Vaswani et al (2017)

Komponen utama dari *transformer* adalah unit dari *multi-head self-attention mechanism*. *Transformer* melihat representasi *encoder* dari input sebagai pasangan *key-value*, (K, V), *n* dimensi. dalam konteks NMT, keduanya adalah *encoder hidden states*. Dalam *decoder*, output sebelumnya dikompresi ke dalam sebuah *query* (Q dari *m* dimensi) dan output teksnya dihasilkan oleh pemetaan query dan *keys* dan *values* ini (Weng, 2018).

Transformer mengadopsi *scaled dot-product attention*: outputnya adalah jumlah nilai yang dibobotkan, di mana bobot yang ditetapkan untuk setiap nilai ditentukan oleh *dot-product* dari *query* dengan semua *keys*:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.10)$$

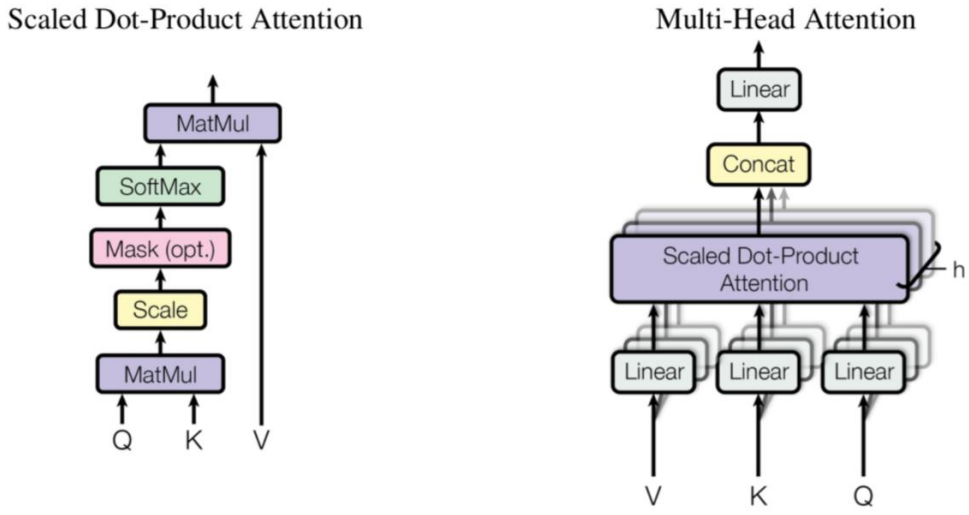
2.6.1 *Multi-Head Self-Attention* dan *Scaled Dot-Product Attention*

Multi-head mechanism berjalan melalui *scaled dot-product attention* berkali-kali dalam parallel. Output independent *attention* hanya digabungkan dan diubah secara linier menjadi dimensi yang diharapkan (Weng, 2018). Pada Gambar 2.14 merupakan ilustrasi mengenai *scaled dot-product attention* dan *multi-head attention*.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (2.11)$$

$$wherehead_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (2.12)$$

Di mana $W_i^Q, W_i^K, W_i^V, dan\ W^O$ adalah matrik parameter untuk dipelajari.

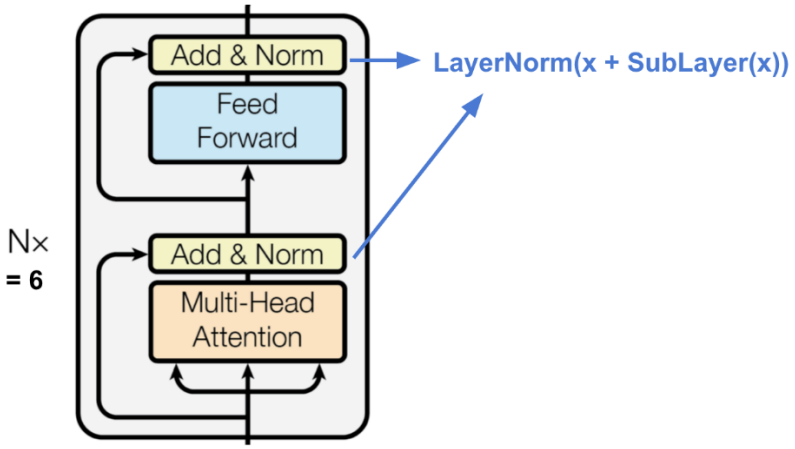


Gambar 2.12 (kiri) Scaled Dot-Product Attention. (kanan) Multi-Head Attention
Sumber: Vaswani et al (2017)

2.6.2 *Encoder*

Encoder seperti pada Gambar 2.5 menghasilkan representasi *attention-based* dengan kemampuan menemukan potongan informasi dari konteks yang memiliki potensi besar.

a. Tumpukan N=6 lapisan identik
b. Setiap lapisan memiliki *multi-head self-attention layer* dan *fully connected feed-forward network*
c. Setiap sub-lapisan mengadopsi koneksi residual dan lapisan *normalization.*

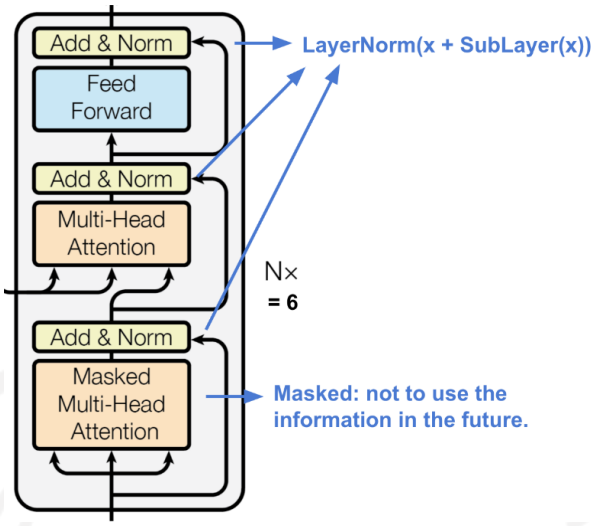


Gambar 2.13 *Encoder transformer*
Sumber: Vaswani et al (2017)

2.6.3 *Decoder*

Decoder bisa mengambil dari representasi yang di *encode*. Gambar 2.6 menjelaskan struktur *decoder* dalam bentuk diagram alur.

a. Tumpukan N=6 lapisan identik
b. Setiap lapisan memiliki dua sub-lapisan dari *multi-head attention mechanism* dan satu sub-lapisan dari *fully connected feed-forward network*
c. Sama seperti *encoder*, setiap sub-layer mengadopsi koneksi residual dan lapisan *normalization.*



Gambar 2.14 *Decoder transformer*

Sumber: Vaswani et al (2017)

2.7 Matrik Evaluasi

Belajar untuk secara otomatis menghasilkan kalimat untuk mendeskripsikan isi konten sebuah gambar dipertimbangkan sebagai bagian tugas yang krusial dalam *image captioning*. Evaluasi model *image captioning* umumnya dikerjakan menggunakan matrik evaluasi seperti BLEU, METEOR, ROUGE, CIDEr, atau SPICE (Yin, 2018). Dalam perhitungannya evaluasi matrik tersebut menghitung kata yang tumpeng tindih antara kalimat prediksi dengan kalimat referensi. Pada penelitian ini penulis hanya menggunakan matrik evaluasi BLEU, karena matrik ini yang paling banyak digunakan selama lebih dari 15 tahun terakhir.

2.7.1 BLEU

Skor BLEU (*Bilingual Evaluation Understudy*) adalah algoritma pencocokan string yang menyediakan metrik kualitas dasar untuk peneliti dan pengembang *machine translation* (MT). BLEU merupakan metrik penilaian kualitas *machine translation* yang paling banyak digunakan selama lebih dari 15 tahun terakhir. Meskipun Sebagian besar memahami bahwa metrik BLEU memiliki banyak kekurangan, metrik ini terus menjadi metrik utama untuk mengukur output sistem *machine translation* hingga hari ini, di masa kejayaan *Neural Machine Translation* (NMT) (V. Kirti, 2019).

Skor BLEU hanya mencerminkan bagaimana kinerja sistem pada rangkaian kalimat sumber dan terjemahan yang dipilih untuk pengujian. Karena terjemahan yang dipilih untuk setiap segmen mungkin bukan satu-satunya yang benar, seringkali terjemahan yang baik dapat

dinilai buruk. Akibatnya, skor tidak selalu mencerminkan potensi kinerja yang sebenarnya dari sistem, terutama pada konten yang berbeda dari materi tes tertentu (V. Kirti, 2019).

Cara Kerja BLEU

Untuk melakukan pengukuran BLEU diperlukan hal-hal.

a. Satu atau lebih terjemahan referensi. Ini harus menjadi data yang belum digunakan dalam mebangun sistem (data *training*) dan idealnya tidak diketahui oleh pengembang sistem *machine translation*
b. Umumnya direkomendasikan bahwa 1000 kalimat atau lebih yang digunakan untuk mendapatkan pengukuran yang bermakna. Sampel yang terlalu kecil dapat mempengaruhi skor secara signifikan hanya dengan beberapa kalimat yang cocok atau tidak cocok
c. Output terjemahan otomatis dari dataset sumber yang sama persis
d. Utilitas pengukuran yang melakukan perbandingan dan perhitungan skor.

Metrik BLEU menilai terjemahan pada skala 0 hingga 1. Semakin mendekati 1 skor kalimat uji, semakin tumpeng tindih dengan terjemahan referensi manusia maka semakin baik sistemnya. Skor BLEU sering dinyatakan dalam skala 1 hingga 100 untuk menyederhanakan komunikasi (V. Kirti, 2019).

Output *machine translation* akan menilai 1 hanya jika identik dengan termahan referensi manusia. Tapi bahkan dua terjemahan manusia yang kompeten dari materi yang sama mungkin hanya mendapatkan skor 0.6 hingga 0.7 karena cenderung menggunakan kosakata dan ungkapan yang berbeda (V. Kirti, 2019).

2.8 Studi Literatur Sejenis

Dalam penelitian ini, penulis menggunakan literatur penelitian sejenis dengan penelitian yang sudah ada sebelumnya. Hal ini dimaksudkan untuk membandingkan studi literatur tersebut. Literatur penelitian dilakukan pada 3 makalah penelitian *image captioning* dengan Bahasa Inggris dan 2 makalah penelitian *image captioning* dengan Bahasa Indonesia. Makalah yang didapatkan menggunakan dataset MSCOCO dan Flickr30K. Tabel 2.4 memberikan sedikit penjelasan mengenai metode apa yang dipakai dan hasil dari penelitian tersebut.

Tabel 2.3 Perbandingan studi literatur sejenis

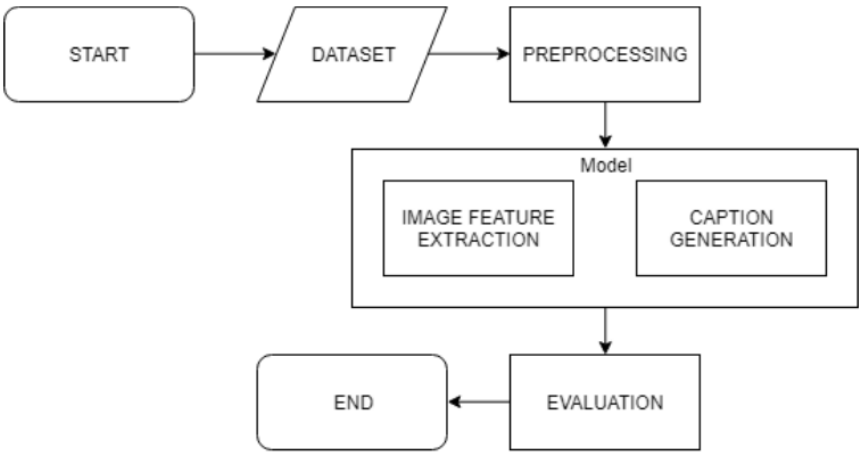
Penulis	**Judul**	**Skor BLEU validation**			
		1	**2**	**3**	**4**
Marcella et al (2020)	*Meshed-Memory Transformer for Image Captioning*	81.6	66.4	51.8	38.7
Ting Yao et al (2019)	*Hierarchy Parsing for Image Captioning*	81.6	66.2	51.5	39.3
Jun Yu et al (2019)	*Multimodal Transformer with Multi-View Visual Representation for Image Captioning*	81.7	66.8	52.4	40.4
Nugraha et al (2019)	*Generating Image Description on Indonesian Language using Convolutional Neural Network and Gated Recurrent Unit*	36.7	17.8	6.7	2.0
Amiin et al (2020)	*Keterangan Gambar Otomatis Berbahasa Indonesia dengan CNN dan LSTM*	29.0	47.0	56.0	60.0
Dhomas et al (2021)	*Image Captioning with Attention for Smart Local Tourism using EfficientNet*	24.51			

Berdasarkan studi literatur yang dipaparkan dalam Tabel 2.4, penlitian ini menggunakan hasil studi literatur tersebut sebagai referensi untuk pengembangan penelitian skripsi ini. Salah satu referensi yang akan diambil adalah matrik evaluasi, seperti BLUE yang didapatkan dari hasil inferensi yang dilakukan menggunakan data *validation* pada literatur tersebut. Matrik evaluasi tersebut akan dijadikan sebagai acuan untuk penelitian ini apakah output dari penelitian ini berhasil atau tidak dan apakah penelitian ini bisa mendekati hasil dari penelitian dari studi literatur yang dilakukan.

BAB III
METODOLOGI PENELITIAN

3.1 Desain Sistem

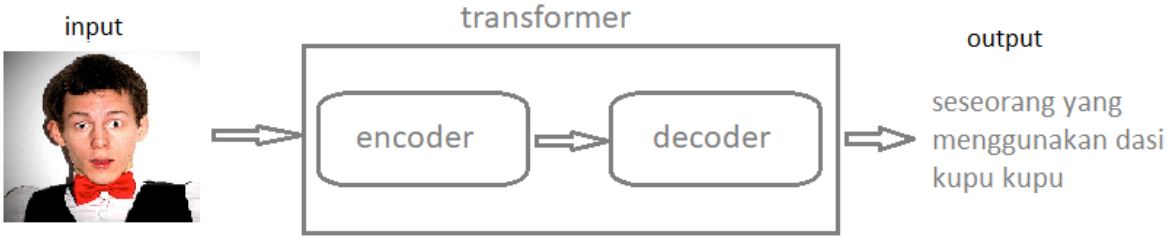
Terdapat empat proses utama dalam metodologi penelitian ini, yaitu *preprocessing* gambar dan *preprocessing* kalimat, *image feature extraction*, *caption generation*, dan *evaluation*. Desain sistem tersebut bisa dilihat pada Gambar 3.1.



Gambar 3.1 Desain sistem *image captioning*

Sumber: Nugraha et al (2019)

Gambar 3.2 merupakan ilustrasi yang di-*oversimplified* mengenai bagaimana alur proses *image captioning* bekerja. Dimulai input berupa gambar kemudian diteruskan ke *encoder* untuk pemrosesan gambar lalu diberikan ke *decoder* untuk memprediksi gambar yang diproses sebelumnya. Hasil dari *decoder* berupa keluaran kalimat yang mendeskripsikan gambar yang diinput dengan struktur kalimat yang sesuai dengan kaidah bahasa.



Gambar 3.2 Desain sistem *image captioning*

3.2 Tools

Untuk membangun model dalam penelitian ini, penulis menggunakan bahasa pemrograman *Python* karena kesederhanaan dan konsistensi, akses ke banyak *library* dan *frameworks* untuk *artificial intelligence* dan *machine learning*, fleksibilitas dan memiliki komunitas yang luas. Mengimplementasikan AI dan ML memerlukan banyak waktu. Penting untuk memiliki lingkungan yang terstruktur dan teruji dengan baik untuk memungkinkan pengembang muncul dengan solusi terbaiknya.

Untuk mengurangi waktu pengembangan dalam penelitian ini, penulis menggunakan sejumlah *Python libraries* dan *framework*. *Python*, dengan tumpukan teknologinya yang kaya, memiliki kumpulan *libraries* untuk *artificial intelligence* dan *machine learning*. Beberapa *libraries* yang penulis gunakan dalam penelitian ini adalah *TensorFlow*, *Keras*, *Numpy*, *Pandas*, *Matplotlib*, dan *NLTK*.

3.3 Dataset

Menggunakan dataset MS COCO 2014 yang diterjemahkan ke dalam Bahasa Indonesia menggunakan Google Translate untuk digunakan dalam penelitian ini. Dataset ini memilki 82.783 gambar dengan 413.915 kalimat dalam *training*, 40.504 gambar dengan 202.520 kalimat dalam *validation*, dan 40.775 gambar dengan 379.249 kalimat dalam *testing*. Setiap gambar dideskripsikan dengan minimal 5 kalimat.

3.3.1 Pengumpulan Dataset

Dataset menggunakan MSCOCO captions yang diunduh dari situs cocodataset. Menurut makalah (X. Chen, H. Fang et al., 2015) gambar yang digunakan pada dataset diambil dari Microsoft COCO (T. Lin, M. Maire et al., 2014). Gambar-gambar tersebut dibagi menjadi tiga bagian, yaitu *training*, *validation*, dan *testing*. Gambar-gambar tersebut diperoleh dengan mencari pasangan dari 80 kategori objek dan berbagai tipe kejadian pada situs web Flickr. Tujuan dari proses pengumpulan gambar MS COCO adalah untuk mengumpulkan gambar berisi banyak objek dalam konteks aslinya. Karena kerumitan visual Sebagian besar gambar dalam *dataset*, hal ini memberikan tantangan yang menarik dan sulit untuk *image captioning*.

Untuk menghasilkan *dataset* dari *image captions*, jumlah *training*, *validation*, dan *testing* yang sama digunakan seperti dalam dataset MS COCO aslinya. Dua dataset diambil. Dataset pertama MS COCO c5 memiliki lima teks referensi untuk setiap gambar dalam dataset *training*, *validation*, dan *testing* MS COCO. Dataset kedua MS COCO c40 memiliki 40 kalimat

referensi untuk 5000 gambar yang dipilih secara acak dari dataset *testing* MS COCO. MS COCO c40 dibuat karena banyak metrik evaluasi otomatis mencapai korelasi yang lebih tinggi dengan penilaian manusia saat diberikan lebih banyak kalimat referensi (R. Vedantam et al., 2014).

Proses mengumpulkan teks menggunakan *Amazon's Mechanical Turk* (AMT) yang diperoleh dari gambar Flickr. Setiap teks yang dibuat juga menggunakan subjek manusia dengan aturan yang telah ditetapkan. Aturan tersebut antara lain.

a. Mendeskripsikan semua bagian penting pada gambar
b. Kalimat tidak diawali dengan "There is"
c. Jangan mendeskripsikan detail yang tidak penting
d. Jangan menggambarkan hal-hal yang mungkin terjadi di masa depan atau masa lalu
e. Jangan menggambarkan apa yang mungkin orang katakan
f. Jangan memberikan nama yang sesuai
g. Panjang kalimat minimal 8 kata.

Jumlah kalimat yang dikumpulkan adalah 413.915 kalimat untuk 82.783 gambar pada *training*, 202.520 kalimat untuk 40.504 gambar pada *validation*, dan 379.249 kalimat untuk 40.775 gambar pada *testing* yang termasuk 179.189 untuk MS COCO c5 dan 200.060 untuk MS COCO c40. Untuk setiap gambar pada *testing*, diperoleh satu tambahan kalimat untuk menghitung skor dari kinerja manusia untuk dibandingkan dengan skor dari mesin pembuat kalimat. Jumlah total kalimat yang diperoleh adalah 1.026.459.

3.4 Praproses Gambar

3.4.1 Ekstraksi Fitur

Ekstraksi fitur gambar pada gambar menggunakan *InceptionV3* dan *EfficientNet* yang merupakan *pretrained* pada *imagenet* untuk mengklasifikasikan setiap gambar. Dari *pretrained* ini akan mengekstrak fitur dari lapis konvolusi terakhir.

Pertama, gambar diubah sesuai dengan format dari *InceptionV3* dan *EfficientNet* dengan.

a. Mengubah ukuran gambar menjadi 299×299
b. Melakukan normalisasi gambar sehingga gambar memiliki pixel dengan jarak antara -1 hingga 1, yang mana menyesuaikan format gambar yang digunakan untuk melatih *InceptionV3* dan *EfficientNet*.

Kemudian, membuat model keras tensorflow di mana lapisan outputnya adalah lapisan konvolusi terakhir dalam model *InceptionV3* dan *EfficientNet*. Bentuk output dari masing-masing model lapisan ini adalah $8 \times 8 \times 2048$ dan $7 \times 7 \times 1028$.

Terakhir, melakukan *preprocess* untuk setiap gambar dengan *InceptionV3* dan *EfficientNet* dan cache outputnya ke disk. Meng-cache output dalam RAM akan lebih cepat tetapi juga membutuhkan banyak memori, memerlukan 8 * 8 * 2048 float tiap gambar.

3.5 Praproses Kalimat

Praproses kalimat (A. Nugraha et al., 2019) yang dilakukan ada 6 tahapan.

a. *Lower Case*

Mengubah semua kalimat menjadi kalimat berhuruf kecil. Proses ini bertujuan untuk memudahkan perhitungan jumlah kata unik dalam dataset.

b. *Mark Caption*

Menambahkan token “<start>” pada awal kalimat dan token “<end>” pada akhir kalimat. Ini dilakukan supaya sistem bisa mulai menghasilkan kalimat saat diberikan token “<start>” dan berhenti menghasilkan kalimat saat menemui token “<end>”.

c. *Tokenizing*

Memisahkan kalimat menjadi kata yang individu. Ini dilakukan untuk memberikan kosa kata dari seluruh kata unik dalam dataset.

d. *Text to Sequence*

Mengubah kata ke dalam indeks kata berurutan. Proses ini akan menghasilkan vector yang menunjukan urutan kata dalam bentuk kalimat.

e. *Pad Sequence*

Memberikan angka 0 apabila panjang dari vector urutan kata kurang dari yang ditentukan dan menghapus akhir vector jika panjang melebihi dari yang ditentukan.

f. *Shift Sequence*

Menggeser vector urutan kata sebanyak satu langkah sehingga model mampu belajar menghasilkan kata berikutnya.

3.6 Vocabulary

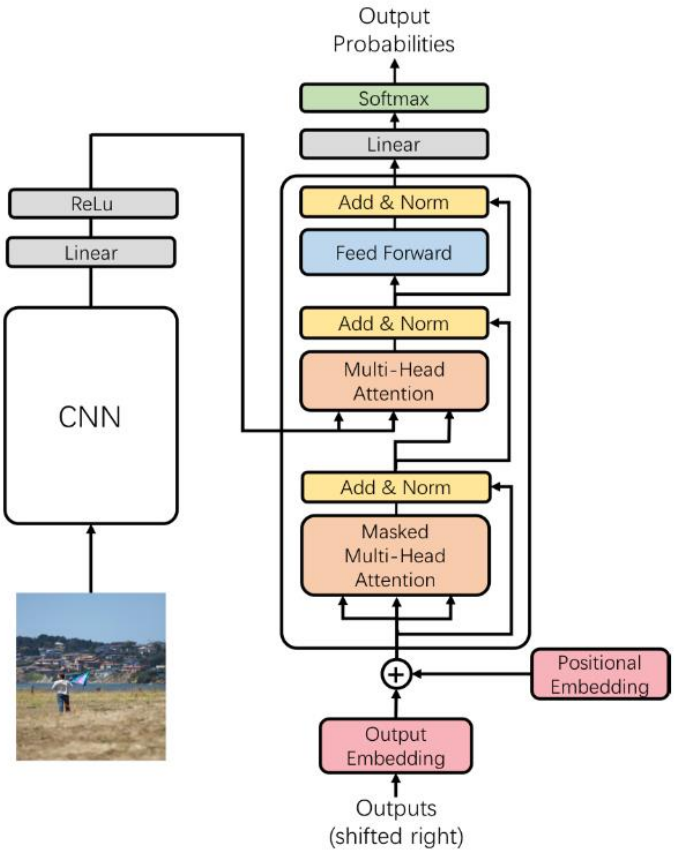
Vocabulary didapatkan dari praproses kalimat saat melakukan *tokenizing*. *Vocabulary* ini digunakan untuk membentuk kalimat yang terstruktur yang mengikuti kaidah bahasa pada saat

melakukan prediksi *image captioning*. Karena keterbatasan sumber daya yang digunakan, *vocabulary* yang diambil hanya 10000 kata paling banyak mucul pada dataset.

3.7 Arsitektur

Transformer mengikuti keseluruhan arsitektur ini menggunakan *self-attention* dan *point-wise* yang ditumpuk, lapisan *full connected* untuk *encoder* dan *decoder* seperti yang ditunjukan pada Gambar 3.3.

a. Urutan sumber dan target keduanya melewati lapisan *embedding* terlebih dahulu untuk membuat data dengan dimensi yang sama $d_{model} = 512$.
b. Untuk mempertahankan informasi posisi, sinusoid-wave-based positional encoding diterapkan dan dijumlahkan dengan output embedding.
c. Lapisan *softmax* dan *linear* ditambahkan ke output *decoder* akhir



Gambar 3.3 Arsitektur yang digunakan pada penelitian ini

Sumber: Xinxin (2018)

3.7.1 *Encoder* dan *Decoder*

***Encoder*:** *encoder* tersusun dari $N = 4$ tumpukan lapisan yang sama. Setiap lapisan memiliki dua sub-lapisan. Lapisan pertama adalah *multi-head self-attention mechanism*, dan lapisan kedua adalah *position-wise fully connected feed-forward network*. Menggunakan koneksi sisa antara sub-lapisan, diikuti dengan normalisasi lapisan (Vaswani et al., 2017).

Decoder**:** *decoder* juga tersusun dari $N = 4$ tumpukan lapisan yang sama. Sebagai tambahan ke dua sub-lapisan dalam tiap lapisan *encoder*, *decoder* memasukkan sub-lapisan ketiga, yang menjalankan *multi-head attention* di atas output tumpukan *encoder*. Sama halnya dengan encoder, menggunakan koneksi sisa antara sub-lapisan, diikuti dengan lapisan normalisasi (Vaswani et al., 2017).

3.7.2 *Attention*

Fungsi *attention* bisa digambarkan sebagai pemetaan kueri dan sekumpulan pasangan *key-value* ke output, di mana kueri, kunci, nilai dan output adalah semuanya vector. Output dihitung sebagai bobot penjumlah dari nilainya, dimana bobot tersebut diberikan ke tiap nilai yang dihitung dengan fungsi kesesuaian dari kueri dengan kunci yang sesuai.

Scaled Dot-Product Attention

Input *attention* ini adalah kueri dan kunci dari dimensi d_k dan nilai dari dimensi d_v. Prakteknya, fungsi *attention* menghittung sekumpulan kueri secara bersamaan, ke dalam matrik Q, kunci dan nilainya juga dikemas bersama ke dalam matrik-matrik *K* dan *V*. Gambar 3.3 (kiri) memberikan penjelasan sederhana dalam bentuk diagram.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (3.1)$$

Fungsi attention yang paling umum digunakan adalah *additive attention* dan *dot-product attention*. *Dot-product attention* indentik dengan algoritma ini, kecuali pada bagian factor penskalaan. *Additive attention* menghitung fungsi kesamaan menggunakan *feed-forward network* dengan satu lapisan tersembunyi (Vaswani et al., 2017).

Multi-Head Attention

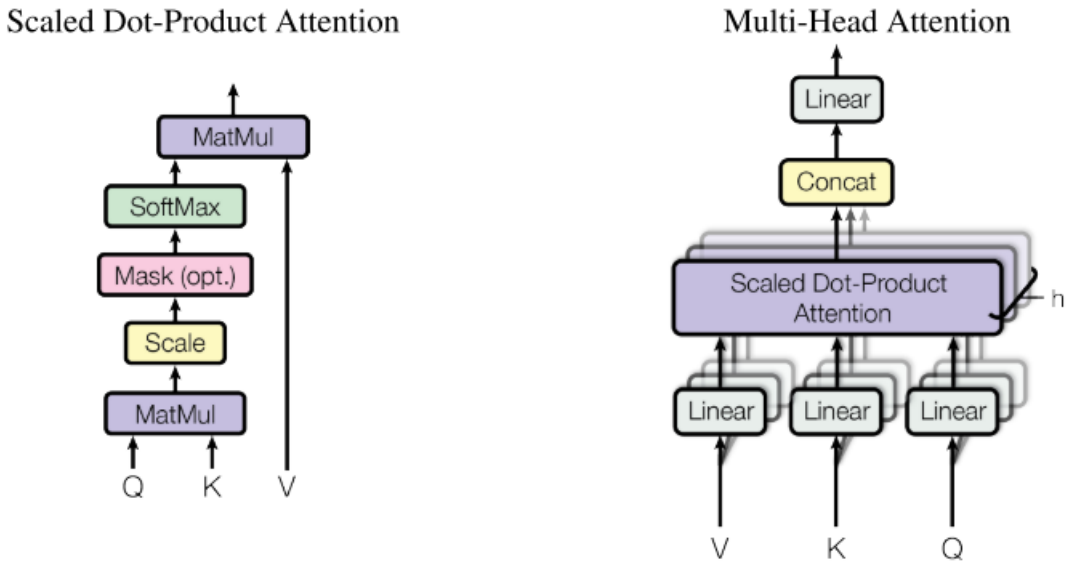
Multi-head attention memungkinkan model untuk bersamaan memperhatikan informasi dari sub-ruang representasi yang berbeda di posisi yang berbeda. Pada Gambar 3.3 (kanan) memberikan gambaran mengenai lapisan-lapisan yang dimiliki oleh *multi-head attention*.

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O \quad (3.2)$$

$$wherehead_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3.3)$$

Di mana proyeksinya adalah matrik parameter $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}, W_i^K \in \mathbb{R}^{d_{model}\times d_k}, W_i^V \in \mathbb{R}^{d_{model}\times d_k}\ dan\ W^O \in \mathbb{R}^{hd_v\times d_{model}}$.

Dalam penelitian ini peneliti menggunakan $h = 4, 6, 8$ paralel lapisan *attention*, atau *heads*. Untuk setiap lapisan peneliti menggunakan $d_k = d_v = d_{model}/h = 64$. Karena untuk mengurangi dimensi setiap *head*, total komputasi yang dikeluarkan sama dengan *single-head attention* dengan *full dimensionality* (Vaswani et al., 2017).



Gambar 3.4 (kiri) Scaled Dot-Product Attention (kanan) Multi-Head Attention
Sumber: Vaswani et al (2017)

Penerapan Attention pada Model

Transformer menggunakan *multi-head attention* dalam tiga cara yang berbeda (Vaswani et al., 2017).

a. Dalam lapisan "*encoder-decoder attention*", kueri berasal dari lapisan *decoder* sebelumnya, dan kunci memori dan nilai berasal dari output *encoder*. Ini meniru ciri khas *encoder-decoder attention mechanism* dalam model *sequence-to-sequence*.
b. Dalam encoder terdapat lapisan self-attention. Dalam lapisan self-attention semua kunci, nilai dan kueri berasal dari tempat yang sama, dalam hal ini, output dari lapisan sebelumnya dalam encoder.
c. Lapisan self-attention dalam encoder memungkinkan setiap posisi dalam decoder menempati semua posisi dalam decoder. Dalam scaled dot-product attention, semua nilai dalam input softmax yang sesuai ditutup.

3.7.3 *Position-wise Feed-Forward Networks*

Tambahan ke sub-lapisan *attention*, setiap lapisan dalam *endocer* dan *decoder* memiliki *fully connected feed-forward network*, yang diterapkan ke setiap posisi secara terpisan dan identic. Ini memiliki dua transformasi linier dengan aktivasi ReLU diantaranya (Vaswani et al., 2017).

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (3.4)$$

3.7.4 *Embedding* dan *Softmax*

Sama dengan model transduksi urutan lainnya, menggunakan *embeddings* yang dipelajari untuk mengonversi token input dan token output ke vector dimensi d_{model}. Selain itu juga menggunakan transformasi linier dan fungsi *softmax* untuk mengonversi output *decoder* ke probabilitas token selanjutnya yang diprediksi. Dalam model ini, membagikan bobot matrik yang sama diantara dua lapisan *embedding* dan transformasi linier *pre-softmax* (Vaswani et al., 2017).

3.7.5 *Positional Encoding*

Karena model tidak memiliki *recurrence* dan *convolution*, supaya model dapat menggunakan urutan, perlu memasukkan beberapa informasi mengenai posisi relative dan absolut dari token dalam urutan. Di akhir, ditambahkan "*positional encodings*" ke input *embendding* di bagian bawah tumpukan *encoder* dan *decoder*. *Positional encoding* memiliki

dimensi yang sama seperti *embedding*, sehingga keduanya bisa dijumlahkan (Vaswani et al., 2017).

Model ini menggunakan fungsi sin dan cos dari frekuensi yang berbeda.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \quad (3.5)$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \quad (3.6)$$

3.8 Training

3.8.1 Data *Training*

Dataset yang digunakan untuk *training*, yaitu 10000 gambar dengan tiap gambar memiliki 5 kalimat referensi. Dataset ini dibagi menjadi dua bagian, yaitu dataset *training* dan dataset *validation*. Perbandingan pembagian untuk datasetnya 8:2, 8000 gambar dengan 40000 kalimat referensi sebagai dataset *training* dan 2000 gambar dengan 10000 kalimat referensi sebagai dataset *validation*.

3.8.2 *Optimizer*

Penelitian ini menggunakan *Adam optimizer* dengan $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$. *Learning rate* diubah-ubah selama proses *training* berdasarkan rumus.

$$lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \quad (3.7)$$

Ini sesuai dengan peningkatan *learning rate* secara linier untuk langkah *training warmup_steps* pertama, dan penurunan *learning rate* setelahnya secara proporsional dengan akar kuadrat terbalik dari *step_num*. penelitian ini menggunakan *warmup_steps = 4000*.

3.8.3 *Regularization*

Residual Dropout Peneliti menerapkan *dropout* ke output setiap sub-lapisan, sebelum *dropout* ditambahkan ke sub-lapisan input dan dinormalisasikan. Sebagai tambahan, peneliti menerapkan *dropout* ke penjumlahan *embeddings* dan *positional encodings* dalam tumpukan *encoder* dan *decoder*. Untuk model awal, peneliti menggunakan nilai $P_{drop} = 0.1$.

3.9 Evaluasi Kalimat

3.9.1 BLEU

BLEU dihitung menggunakan beberapa presisi modifikasi *n-gram* (M. Lei, 2019). Secara khusus,

$$BP = \begin{cases} 1 & if\ c > r \\ exp\left(1 - \frac{r}{c}\right) & if\ c \leq r \end{cases} \quad (3.8\ )$$

Di mana c adalah jumlah *unigram* (panjang) dalam semua kalimat kandidat, dan r merupakan panjang kecocokan terbaik untuk setiap kalimat kandidat dalam korpus. Di sini panjang kecocokan terbaik adalah panjang kalimat referensi terdekat dengan kalimat kandidat. Tidak sulit untuk menemukan bahwa BLEU selalu bernilai antara 0 dan 1. Hal ini karena $BP, w_n, dan\ p_n$ selalu diantara 0 dan 1.

$$BLEU\ =\ BP \cdot exp\left(\sum_{n=1}^{N} w_n\ log\ p_n\right) \quad (3.9)$$

Di mana p_n adalah presisi yang dimodifikasi untuk *n-gram*, w_n adalah bobo tantara 0 dan 1 untuk $log\ p_n$ dan $\sum_{n=1}^{N} w_n = 1$, dan BP adalah pinalti untuk menghukung *machine translation* yang pendek. Dalam penelitian ini $N = 4$, nilai tersebut diambil berdasarkan studi literatur yang sudah dilakukan, semua penelitian terdahulu menggunakan nilai tersebut untuk menghitung skor BLEU. Nilai tersebut merupakan konvensi dari makalah penelitian BLEU dan menjadi nilai *default* pada *library* yang digunakan pada penelitian ini.

Untuk menghitung skor BLEU, penelitian ini menggunakan bantuan *library python* yang dinamakan *nltk*. Dalam *library* tersebut terdapat fungsi *sentence_bleu* yang digunakan untuk mendapatkan skor BLEU dari 1 hingga 4. Untuk menghindari kalimat pendek terkena pinalti, penelitian ini menggunakan fungsi *smoothing*.

3.10 Membangun *User Interface*

Penulis menggunakan *Streamlit library python* untuk membangun aplikasi web yang memungkinkan pengguna untuk mengunggah gambar berformat dan menerima hasil dari prediksi *image captioning*.

Penggunaan *library* ini sangatlah mudah. Apa yang penulis lakukan adalah.

1. Mengunggah gambar
2. Memprediksi kalimat untuk gambar yang diunggah
3. Menampilkan hasil prediksi.

Langkah-langkah yang diperlukan untuk membangun UI:

1. Dalam aplikasi ini, satu elemen kunci untuk memungkinkan pengguna mengunggah gambar sehingga model bisa membuat prediksi, dan hal tersebut bisa dilakukan dengan memangginl fungsi 'file_uploader'. Gambar 3.4 adalah kode penuh untuk mengunggah gambar. Fungsi "*upload_image()*" adalah fungsi yang memberikan akses kepada pengguna supaya dapat mengunggah gambar yang diinginkan untuk melakukan prediksi. Kode pada baris 3 merupakan tempat untuk mengunggah gambar dari pengguna dan menyimpannya ke dalam variable dan kode pada baris 8 berfungsi untuk menampil gambar yang pengguna unggah.

```
1   def upload_image():
2       args = { 'sunset' : 'sunset.jpeg' }

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

5       image1 = Image.open(img_open)
6       rgb_im = image1.convert('RGB')
7       rgb_im.save("saved_image.jpeg")

8       st.image(image1,use_column_width=True,caption="Your image")
```

Gambar 3.5 Kode untuk mengunggah gambar dan menampilkannya

2. Untuk bisa memprediksi gambar yang sudah diunggah, perlu menggunakan model yang sudah disimpan pada saat proses *training*. Gambar 3.5 menunjukkan kode yang dibutuhkan. Fungsi "*load_model()*" adalah fungsi untuk menggunakan model yang akan digunakan oleh penulis. Kode pada baris 3-7 merupakan *hypermarater* yang digunakan. Kode pada baris 9-16 masing-masing, memanggil model untuk ekstraksi fitur gambar sebelum gambar diberikan ke *encoder*, mendefinisikan *transformer* pada bagian *encoder*, mendefinisikan *transformer* pada bagian *decoder*. Kode baris 20-24 mendefinisikan fungsi *loss* yang digunakan, yaitu *Sparse Categorical Crossetropy* dan mengkompile model *image captioning*. Pada baris 25 penulis menunggah *pretrained* model yang sudah di *training*.

```
def load_model():
    # Dimension for the image embeddings and token embeddings
    EMBED_DIM = 512
    # Number of self-attention heads
    NUM_HEADS = 4
    # Per-layer units in the feed-forward network
    FF_DIM = 512

    cnn_model = get_cnn_model()
    encoder = TransformerEncoderBlock(
        embed_dim=EMBED_DIM, dense_dim=FF_DIM, num_heads=NUM_HEADS
    )
    decoder = TransformerDecoderBlock(
        embed_dim=EMBED_DIM, ff_dim=FF_DIM, num_heads=NUM_HEADS
    )
    caption_model = ImageCaptioningModel(
        cnn_model=cnn_model, encoder=encoder, decoder=decoder
    )
    # Define the loss function
    cross_entropy = keras.losses.SparseCategoricalCrossentropy(
        from_logits=True, reduction="none"
    )
    # Compile the model

    caption_model.compile(optimizer=keras.optimizers.Adam(), loss=cross_entropy
)
    caption_model.load_weights("model_efficient4/modelEfficient")

    return caption_model
```

Gambar 3.6 Kode untuk menggunakan model

3. Setelah membuat modelnya, gambar yang diunggah perlu diubah ukurannya menjadi 299×299. Gambar 3.6 merupakan kode untuk mengubah ukuran gambarnya. Fungsi ”*read_image()*” digunakan untuk mengubah ukuran gambar yang diunggah oleh pengguna menjadi ukuran 299×299. Hal ini dilakukan untuk menyesuaikan format yang diperlukan untuk melakukan ekstraksi fitur gambar pada model CNN.

```
def read_image(img_path, size=(299,299)):
    img = tf.io.read_file(img_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, size)
    img = tf.image.convert_image_dtype(img, tf.float32)
    return img
```

Gambar 3.7 Kode untuk mengubah ukuran gambar

4. Supaya hasil prediksi memiliki hasil yang bagus, perlu membuat pemetaan kata dan *vocab* atau korpus supaya kalimat prediksi memiliki struktur kata yang baik sesuai dengan bahasa manusia. Gambar 3.7 merupakan kode untuk melakukan hal tersebut. Fungsi “*word_map()*” berfungsi untuk mengubah *string* menjadi urutan *integer* di mana setiap *integer* mewakili indeks sebuah kata pada korpus atau *vocab*. Pada kode

baris 11, penulis menggunakan lapisan "*TextVectorization*" untuk melakukan vektorisasi data. Penulis juga menggunakan skema *custom standardization*.

```
1   def word_map():
2       # Vocabulary size
3       VOCAB_SIZE = 10000
4       # Fixed length allowed for any sequence
5       SEQ_LENGTH = 20
6
7       text_data_path = "textDataEfficient4.pickle"
8       with open(text_data_path, "rb") as f:
9           text_data = pickle.load(f)
10
11      vectorization = TextVectorization(
12          max_tokens=VOCAB_SIZE,
13          output_mode="int",
14          output_sequence_length=SEQ_LENGTH,
15          standardize=custom_standardization,
16      )
17      vectorization.adapt(text_data)
18
19      vocab_path = "vocabEfficient4.pickle"
20      with open(vocab_path, "rb") as f:
21          vocab = pickle.load(f)
22
23      index_lookup = dict(zip(range(len(vocab)), vocab))
24      max_decoded_sentence_length = SEQ_LENGTH - 1
25
26      return vectorization, index_lookup, max_decoded_sentence_length
```

Gambar 3.8 Kode untuk pemetaan kata dan membuat korpus

5. Setelah melakukan mengunggah model, mengubah ukuran gambar, dan pemetaan kata. Selanjutnya membuat fungsi "*generate_caption*" sesuai pada Gambar 3.8. Kode baris 2 memanggil fungsi pemetaan kata yang kemudian disimpan dalam variable. Kode baris 5-6 membaca gambar yang diunggah oleh pengguna yang diubah menjadi *numpy*. Baris 9-10 memberikan gambar ke model CNN untuk dilakukan ekstraksi fitur gambar. Baris 13 memberikan hasil ekstraksi ke *encoder*, kemudian pada baris 16 dari *encoder* diberikan ke *decoder* untuk menghasilkan kalimat dan pada baris 29 merupakan hasil prediksi yang sudah berbentuk sebuah kalimat terstruktur.

```
1   def generate_caption(image, caption_model):
2       vectorization, index_lookup, max_decoded_sentence_length = word_map()
3
4       # Read the image from the disk
5       img = read_image(image)
6       img = img.numpy().astype(np.uint8)
7
8       # Pass the image to the CNN
9       img = tf.expand_dims(img, 0)
10      img = caption_model.cnn_model(img)
11
12      # Pass the image features to the Transformer encoder
13      encoded_img = caption_model.encoder(img, training=False)
14
15      # Generate the caption using the Transformer decoder
16      decoded_caption = "<start> "
17      for i in range(max_decoded_sentence_length):
18          tokenized_caption = vectorization([decoded_caption])[:, :-1]
19          mask = tf.math.not_equal(tokenized_caption, 0)
20          predictions = caption_model.decoder(
21              tokenized_caption, encoded_img, training=False, mask=mask
22          )
23          sampled_token_index = np.argmax(predictions[0, i, :])
24          sampled_token = index_lookup[sampled_token_index]
25          if sampled_token == " <end>":
26              break
27          decoded_caption += " " + sampled_token
28
29      predicted   =   decoded_caption.replace("<start>   ",   "").replace("   <end>",
"").strip()
30
31      return predicted
```

Gambar 3.9 Kode untuk melakukan prediksi *image captioning*

6. Terakhir, menampilkan hasil prediksi. Gambar 3.9 kode untuk menampilkan hasil prediksi *image captioning*. Ini adalah kode utama dari semua bagian, tempat memanggil fungsi-fungsi yang sudah didefinisikan sebelumnya. Kode baris 2 memanggil fungsi "*load_model()*" dan menyimpannya ke variable *caption_model*. Kode baris 4-7 menampilkan teks judul dari situs yang dibuat. Kode baris 9 memanggil fungsi "*upload_image()*" sehingga gambar bisa tampil di layer pengguna. Kode baris 11 merupakan tombol untuk melakukan prediksi *image captioning*. Ketika tombol ditekan maka proses prediksi akan berjalan. Setelah selesai hasil dari prediksi *image captioning* tersebut akan ditampilkan ke layer.

```
if __name__ == '__main__':
    caption_model = load_model()

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

    upload_image()

    if st.button('Generate captions!'):
        predicted = generate_caption("saved_image.jpeg", caption_model)

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

Gambar 3.10 Kode untuk menampilkan hasil prediksi *image captioning*

BAB IV
HASIL DAN PEMBAHASAN

4.1 Eksperimen

Pada tahap ini dilakukan beberapa eksperimen dalam proses menghasilkan kalimat deskripsi. Eksperimen yang dilakukan meliputi, 1) proses ekstraksi fitur gambar menggunakan dua *pretrained* arsitektur CNN, yaitu *InceptionV3* dan *EfficientNet*, 2) jumlah *heads* pada *Multi-Head Attention* (4, 6, 8). Pada eksprimen ini penulis membandingkan perbedaan yang terjadi jika menggunakan dua *pretrained* yang berbeda dan menggunakan jumlah *heads* yang berbeda untuk menghasilkan kalimat deskripsi gambar dalam Bahasa Indonesia.

Hyperparameters yang digunakan dalam eksperimen bisa dilihat pada Tabel 4.1. Semua model ekspertimen memiliki *hyperparameters* yang sama kecuali pada bagian NUM_HEADS, nilai yang digunakan berbeda untuk tiap eksperimen.

Penentuan nilai *hyperparameter* beberapa mengikuti nilai *default* dari makalah peneltian "*Attention is All You Need*" (Vaswani, 2017), seperti EMBED_DIM, FF_DIM, NUM_HEADS. Untuk *hyperparameters* lain, penulis menyesuaikan nilainya berdasarkan jumlah dataset (VOCAB_SIZE dan SEQ_LENGTH) dan kemampuan komputasi yang dimiliki (BATCH_SIZE dan EPOCHS). VOCAB_SIZE digunakan untuk membatasi jumlah korpus atau *dictionary* yang diambil untuk membentuk kalimat saat melakukan prediksi *image captioning*.

Tabel 4.1 *Hyperparameters* yang digunakan dalam eksperimen

Hyperparameters	**Nilai**
VOCAB_SIZE	10000
SEQ_LENGTH	20
EMBED_DIM	512
NUM_HEADS	4, 6, 8
FF_DIM	512
BATCH_SIZE	64
EPOCHS	30

4.2 Analisis Output

4.2.1 Hasil Nilai Akurasi dan *Loss*

Ada dua model yang dihasilkan pada penelitian ini, model yang menggunakan *InceptionV3* dan model yang menggunakan *EfficientNet* untuk ekstraksi fitur gambarnya. Kedua model tersebut dilakukan eksperimen dengan menggunakan jumlah lapisan *Multi-Head Self-Attention* yang berbeda, yaitu 4 *heads*, 6 *heads*, 8 *heads*. Kedua model tersebut, dihitung nilai akurasi dan *loss* dengan menggunakan dataset *training* yang berjumlah 8000 gambar dan dataset *validation* yang berjumlah 2000 gambar dan pengujian model dilakukan sebanyak 30 *epochs*. Perbandingan hasil nilai akurasi dan *loss* dari kedua model tersebut dapat dilihat pada Tabel 4.2.

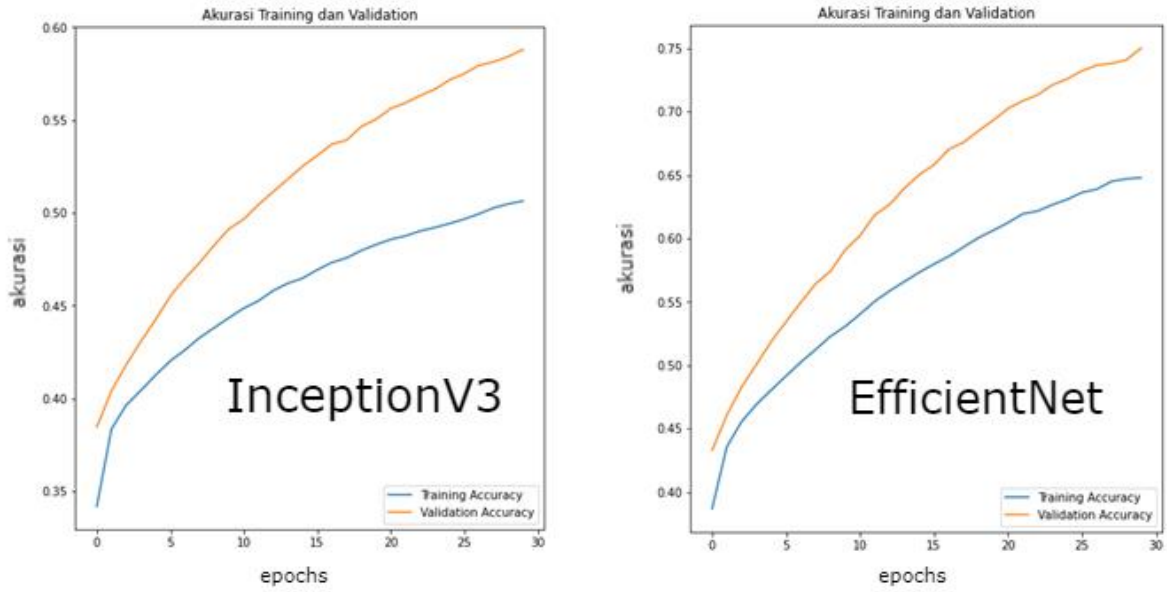
Tabel 4.2 Hasil perbandingan akurasi dan *loss*

	Model *InceptionV3*			**Model *EfficientNet***		
	4 heads	**6 heads**	**8 heads**	**4 heads**	**6 heads**	**8 heads**
Akurasi *training*	0.5093	0.5227	0.5208	0.6501	0.5816	0.5664
Akurasi *validation*	0.5880	0.6062	0.5981	0.7501	0.6646	0.6414
Loss training	10.5757	10.1906	10.3388	6.6967	8.4913	8.9579
Loss validation	8.2392	7.8917	8.1296	4.1696	6.2500	6.7972

Dari Tabel 4.2 menjelaskan secara garis besar model dengan *EfficientNet* memiliki nilai akurasi dan *loss* yang lebih baik dibandingkan dengan *InceptionV3*. Perbedaan nilai akurasi dan *loss* antara *InceptionV3* dengan *EfficientNet* signifikan. Jika dilihat lebih mendalam pada penggunaan jumlah lapisan *Multi-Head Self-Attention* untuk tiap model, didapatkan lapisan dengan jumlah 4 *heads* mendapatkan nilai akurasi paling tinggi dan nilai *loss* paling kecil dibandingkan dengan 6 dan 8 *heads*.

Gambar 4.1 merupakan dua grafik akurasi *training* dan *validation* yang menampilkan dua model ekstraksi fitur gambar berbeda, yaitu *InceptionV3* dan *EfficientNet* dengan masing-masing menggunakan 4 lapisan *Multi-Head Self-Attention*. Grafik tersebut menunjukkan

bahwa akurasi yang dimiliki model dengan *EfficientNet* lebih tinggi dibandingkan akurasi yang dimiliki model dengan *InceptionV3*.



Gambar 4.1 Grafik akurasi *training* dan *validation* dari dua model CNN berbeda

4.2.2 Hasil Skor BLEU

Kalimat referensi dan kalimat prediksi akan dikumpulkan dan dievaluasi secara kolektif menggunakan skor BLEU yang merangkum seberapa dekat kalimat yang dihasilkan dengan kalimat aslinya. Untuk masing-masing skor BLEU ditambahkan bobot yang berbeda. Bobot dari masing-masing skor BLEU:

- BLEU-1 = (1.0, 0, 0, 0)
- BLEU-2 = (0.5, 0.5, 0, 0)
- BLEU-3 = (0.3, 0.3, 0.3, 0)
- BLEU-4 = (0.25, 0.25, 0.25, 0.25)

Pada Tabel 4.3 merupakan rata-rata hasil skor BLEU dari dua Model yang menggunakan model ekstraksi fitur gambar yang berbeda.

Tabel 4.3 Hasil perbandingan rata-rata skor BLEU

BLEU-#	Model *InceptionV3*			Model *EfficientNet*		
	4 heads	6 heads	8 heads	4 heads	6 heads	8 heads
BLEU-1	51.23	51.47	50.83	78.05	54.60	53.43
BLEU-2	38.41	38.11	33.92	68.21	38.97	37.07
BLEU-3	35.44	35.16	27.03	61.89	32.54	30.66
BLEU-4	27.92	27.74	17.87	52.09	23.41	21.57

Skor BLEU untuk ekstraksi fitur gambar model *InceptionV3* dengan tiga *heads* yang berbeda didapatkan model dengan *heads* 4 memiliki skor BLEU tertinggi dibandingkan dengan 6 dan 8 *heads*. Skor BLEU-{1,2,3,4} yang didapatkan 4 *heads*, yaitu {51.23, 38.41, 35.44, 27.92}.

Sedangkan skor BLEU untuk ekstraksi fitur gambar model *EfficientNet* dengan tiga *heads* yang berbeda didapatkan model dengan *heads 4* yang memiliki skor BLEU tertinggi dibandingkan dengan *heads* 6 dan 8. Skor BLEU-{1,2,3,4} yang didapatkan *heads 4*, yaitu {78.05, 68.21, 61.89, 52.09}.

Jika dibandingkan model *InceptionV3* dengan model *EfficientNet* yang digunakan untuk model ekstraksi fitur gambar, model dengan *EfficientNet* jauh lebih baik daripada model dengan *InceptionV3* terutama perbedaan pada *heads 4* pada masing-masing model yang terlihat cukup jauh. Bahkan jika dibandingkan juga dengan *heads* lain, model dengan *EfficientNet* masih tetap unggul dibandingkan model dengan *InceptionV3*.

4.2.3 Hasil Percobaan *Image Captioning*

Pada bagian ini, akan ditampilkan beberapa hasil percobaan *image captioning* yang menggunakan Model dengan *InceptionV3* dan Model dengan *EfficientNet*. Dataset yang digunakan untuk percobaan ini diambil dari dataset *testing* yang berbeda dengan dataset *training* dan dataset *validation*. Dataset ini juga tidak dikenali oleh kedua Model.

1. Model dengan InceptionV3 (4 heads)

Tabel 4.4 menunjukkan hasil prediksi dari *image captioning* yang menggunakan model dengan *InceptionV3* (4 *heads*). Gambar dan hasil prediksi kemudian dibandingkan

secara kulitatif dengan melihat apakah hasil kalimat prediksi sesuai dengan konteks pada gambar.

Tabel 4.4 Hasil prediksi *image captioning* menggunakan model dengan *InceptionV3* (4 *heads*)

Gambar	**Prediksi**	**Kesesuaian**
	orang orang berkumpul di lapangan dengan seekor anjing	Sesuai
	sebuah bus tingkat melaju di jalan saat seorang pria lewat	Sesuai

Gambar	Prediksi	Kesesuaian
	seekor kucing putih dan putih di atas meja	Tidak sesuai
	piring putih dengan topping daging kentang dan sayuran	Sesuai
	seekor jerapah berdiri di lapangan hijau yang rimbun	Tidak sesuai

2. Model dengan EfficientNet (4 heads)

Tabel 4.5 menunjukkan hasil prediksi dari *image captioning* yang menggunakan model dengan *EfficientNet* (4 *heads*). Gambar dan hasil prediksi kemudian dibandingkan secara kulitatif dengan melihat apakah hasil kalimat prediksi sesuai dengan konteks pada gambar.

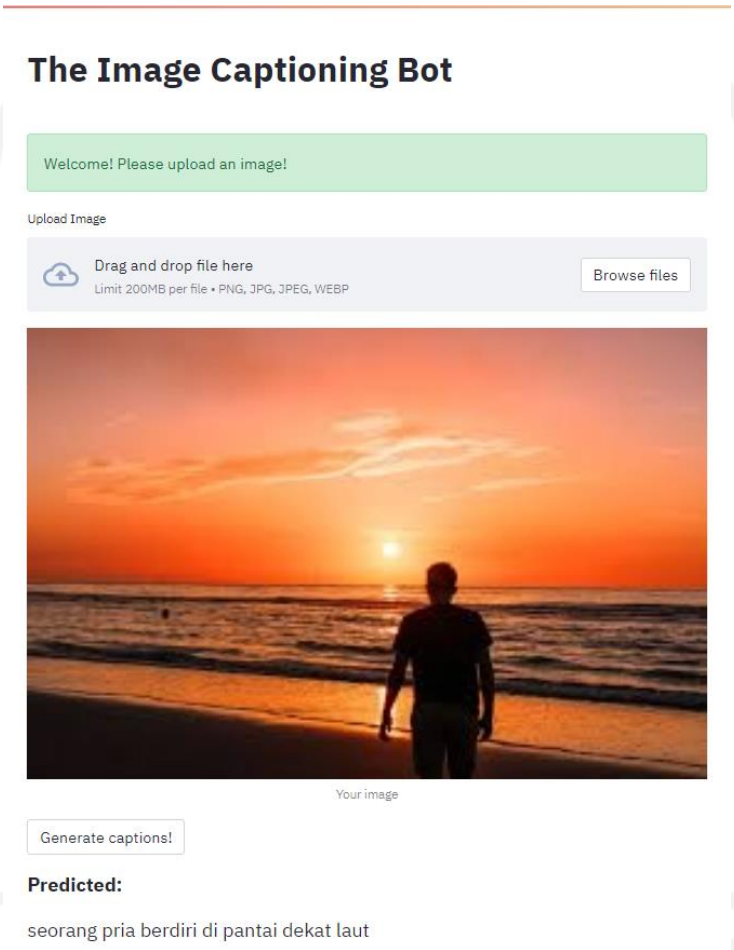
Tabel 4.5 Hasil prediksi *image captioning* menggunakan model dengan *EfficientNet* (4 *heads*)

Gambar	Prediksi	Kesesuaian
	seorang pria dan seorang anjingnya berdiri di lapangan rumput	Tidak sesuai
	jalan yang sibuk dengan lalu lintas pejalan kaki dan beberapa mobil	Sesuai

Gambar	Prediksi	Kesesuaian
	kucing sedang melihat ke layar tv	Sesuai
	sepiring makanan dengan pizza dan sayuran	Tidak sesuai
	jerapah berdiri di dekat pohon di lapangan berumput	Sesuai

4.3 User Interface

Gambar 4.7 merupakan *user interface* dengan prediksinya dalam bentuk aplikasi web yang dibuat menggunakan *streamlit*. *Streamlit* adalah aplikasi *framework open-source* untuk *machine learning* dan *data science*. Pembuatan UI untuk *image captioning* sendiri cukup mudah karena *streamlit* memiliki *library* pada *python* sehingga hal ini membuat pembangunannya mudah dan memiliki UI yang cantik.



Gambar 4.2 *User Interface* untuk prediksi *Image Captioning*

BAB V
PENUTUP

5.1 Kesimpulan

1. Model dengan *EfficientNet* untuk ekstraksi fitur gambar yang menggunakan 4 lapisan *Multi-Head Self-Attention* mendapatkan nilai akurasi paling tinggi, nilai *loss* paling kecil dan skor BLEU paling tinggi.
2. Nilai akurasi dan *loss* pada {*training*, *validation*} masing-masing {0.6501, 0.7501} dan {6.6967, 4.1696} sedangkan skor BLEU-{1,2,3,4} masing-masing {78.05, 68.21, 61.89, 52.09}.
3. Skor akurasi dan *loss* menunjukkan tidak ada *overfitting* karena nilai akurasi *validation* lebih tinggi dibandingkan dengan *training* dan nilai *loss validation* lebih kecil daripada *loss training*.
4. Dari skor BLEU yang didapat, BLEU-1 memiliki skor paling tinggi dikarenakan menggunakan unigram dalam penilaiannya. Skor tersebut juga menunjukkan bahwa hasil dari kalimat prediksi memiliki struktur tata bahasa yang baik dan sesuai dengan kaidah Bahasa.
5. Hasil dari penelitian ini menunjukkan bahwa implementasi arsitektur *transformer* secara signifikan melampaui hasil penelitian *image captioning* dalam Bahasa Indonesia yang sudah ada.

5.2 Saran

Dalam penelitian ini masih terdapat keterbatasan dan ruang untuk peningkatan. Oleh karena itu penulis mengharapkan adanya pengembangan untuk penelitian selanjurnya. Adapun saran yang diberikan oleh penulis untuk penelitian dengan topik ini adalah:

1. Penelitian ini hanya menghasilkan satu hasil prediksi *image captioning* untuk tiap gambar. Untuk penelitian selanjutnya bisa dikembangkan dengan menambah jumlah hasil prediksi *image captioning* untuk tiap gambar menjadi lebih dari satu.
2. Menggunakan model ekstraksi fitur gambar yang memiliki sifat dinamis seperti *Bottom-Up and Top-Down Attention* (P. Anderson, X. He et al., 2018) yang menggunakan model deteksi objek sebagai ekstraksi fiturnya.
3. Mengimplementasikan metode-metode yang digunakan pada penelitian *image captioning* dalam bahasa Inggris ke *image captioning* dalam bahasa Indonesia, seperti

Meshed-Memory Transformer (M. Cornia et al, 2020), *Boosted Transformer* (J. Li et al, 2019), dan *Image Transformer* (S. He et al, 2020).

DAFTAR PUSTAKA

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-Janua*(April 2018), 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Amidi, A., & Amidi, S. (2019). *Cheatsheet: Convolutional Neural Networks*. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *4*(January), 3104–3112.

Amidi, A., & Amidi, S. (2019). *Cheatsheet: Recurrent Neural Networks*. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Kostadinov, S. (2019). *Understanding Encoder-Decoder Sequence to Sequence Model*. Toward Data Science. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

Weng, L. (2018). *Attention? Attention*. Github. https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#whats-wrong-with-seq2seq-model

Chauhan, N. S. (2021). *Attention mechanism in Deep Learning, Explained*. KDnuggets. https://www.kdnuggets.com/2021/01/attention-mechanism-deep-learning-explained.html

Brownlee, J. (2019). *A Gentle Introduction to Computer Vision*. Machine Learning Mastery. https://machinelearningmastery.com/what-is-computer-vision/

Yse, D. L. (2019). *Your Guide to Natural Language Processing (NLP)*. Towards Data Science. https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

Li, J., Yao, P., Guo, L., & Zhang, W. (2019). Boosted Transformer for Image Captioning. *Applied Sciences*, *9*, 3260. https://doi.org/10.3390/app9163260

Cornia, M., Stefanini, M., Baraldi, L., & Cucchiara, R. (2020). *Meshed-Memory Transformer for Image Captioning*. 10575–10584. https://doi.org/10.1109/cvpr42600.2020.01059

He, S., Liao, W., Tavakoli, H. R., Yang, M., Rosenhahn, B., & Pugeault, N. (2020). *Image Captioning through Image Transformer*. http://arxiv.org/abs/2004.14231

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Yu, J., Li, J., Yu, Z., & Huang, Q. (2019). Multimodal transformer with multi-view visual representation for image captioning. *ArXiv*, *14*(8), 1–12. https://doi.org/10.1109/tcsvt.2019.2947482

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollar, P., & Zitnick, C. L. (2015). Microsoft COCO Captions: Data Collection and Evaluation Server. *ArXiv*, 1–7. http://arxiv.org/abs/1504.00325

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & 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

Vedantam, R., Zitnick, C. L., & Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 4566–4575. https://doi.org/10.1109/CVPR.2015.7299087

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, *October 2002*, 311–318. https://doi.org/10.3115/1073083.1073135

C.-Y. Lin. (2004). ROUGE: A Package for Automatic Evaluation of Summaries Chin-Yew. *ACL Workshop*. https://doi.org/10.1253/jcj.34.1213

Denkowski, M., & Lavie, A. (2015). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. *ACL Workshop*, 376–380. https://doi.org/10.3115/v1/w14-3348

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*, 98–103. https://doi.org/10.1109/ICoICT.2019.8835370

Weng, L. (2018). *Attention? Attention*. Github. https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#whats-wrong-with-seq2seq-model

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6077–6086. https://doi.org/10.1109/CVPR.2018.00636

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

Tsang, S.-H. (2018). *Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015*. Medium. https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 10691–10700.

Vashee, K. (2019). *Understanding MT Quality: BLEU Scores*. RWS. https://www.rws.com/blog/understanding-mt-quality-bleu-scores/

Mao, L. (2019). *Bilingual Evaluation Understudy (BLEU)*. Lei Mao's Log Book. https://leimao.github.io/blog/BLEU-Score/

Voykinska, V., Azenkot, S., Wu, S., & Leshed, G. (2016). How blind people interact with visual content on social networking services. *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, *27*, 1584–1595. https://doi.org/10.1145/2818048.2820013

Zhu, X., Li, L., Liu, J., Peng, H., & Niu, X. (2018). Captioning transformer with stacked attention modules. *Applied Sciences (Switzerland)*, *8*(5). https://doi.org/10.3390/app8050739

UNIVERSITAS ISLAM INDONESIA