KERJA PRAKTIK - EF234603

Pembuatan Model *Image Captioning* Menggunakan *Deep Learning*

Institut Teknologi Sepuluh Nopember
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
Periode: 30 Juli 2024 – 30 Oktober 2024

Oleh:

Muhammad Hafidh Rosyadi 5025211013

Pembimbing Jurusan
Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.
Pembimbing Lapangan
Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

Pembuatan Model *Image Captioning* Menggunakan *Deep Learning*

Institut Teknologi Sepuluh Nopember
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
Periode: 30 Juli 2024 – 30 Oktober 2024

Oleh:

Muhammad Hafidh Rosyadi 5025211013

Pembimbing Jurusan
Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.

Pembimbing Lapangan
Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

LEMBAR PENGESAHAN
KERJA PRAKTIK

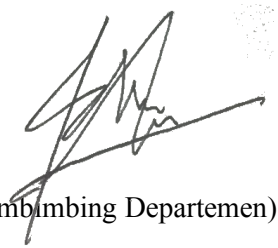
Pembuatan Model Image Captioning Menggunakan Deep Learning

Oleh:

Muhammad Hafidh Rosyadi 5025211013

Disetujui oleh Pembimbing Kerja Praktik:

1. Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.
   NIP. 198412102014042003



(Pembimbing Departemen)

2. Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.
   NIP. 1987202012004

(Pembimbing Lapangan)

Pembuatan Model Image Captioning Menggunakan Deep Learning

Nama Mahasiswa	: Muhammad Hafidh Rosyadi
NRP	: 5025211013
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.
Pembimbing Lapangan	: Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.

ABSTRAK

Image Captioning merupakan teknologi yang menghubungkan pemrosesan citra dengan pemodelan bahasa alami untuk menghasilkan deskripsi teks yang sesuai dengan isi gambar. Meskipun berbagai model telah dikembangkan untuk tugas ini, masih ada tantangan dalam memilih arsitektur fitur ekstraksi yang tepat, serta mengoptimalkan parameter *preprocessing* dan arsitektur model yang digunakan. Penelitian ini bertujuan untuk mengevaluasi berbagai faktor yang memengaruhi kinerja model *Image Captioning*, termasuk arsitektur fitur ekstraksi, penggunaan *padding* dalam *preprocessing* gambar, dan jumlah layer pada model *decoder*. Pengujian dilakukan menggunakan dataset Flickr8k dan dioptimalkan dengan metode *Grid Search* untuk memperoleh kombinasi parameter terbaik. Hasil penelitian menunjukkan bahwa arsitektur fitur ekstraksi berpengaruh signifikan terhadap performa model, dengan model VGG mencatat skor BLEU tertinggi sebesar 0,3656, sedangkan MobileNet menunjukkan stabilitas terbaik dengan nilai rata-rata BLEU sebesar 0,2226 dan ROUGE sebesar 0,0483. Selain itu, eksperimen penggunaan padding dalam *preprocessing* gambar menunjukkan bahwa model tanpa *padding* memiliki kinerja lebih baik

dibandingkan dengan model yang menggunakan *padding*, dengan MobileNet mencatatkan rata-rata BLEU sebesar 0,2583 tanpa *padding* dibandingkan 0,1870 dengan *padding*. Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan teknik *image captioning* dengan memperhatikan aspek penting dalam pemilihan arsitektur serta optimasi parameter, untuk mencapai deskripsi gambar yang lebih akurat dan relevan.

Kata Kunci ***: Image Captioning, Deep Learning, Transformer***

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Pembuatan Model *Image Captioning* Menggunakan *Deep Learning*.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Ibu Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 30 Oktober 2024

Muhammad Hafidh Rosyadi

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam era digital saat ini, gambar telah menjadi salah satu bentuk media yang paling banyak digunakan, baik di televisi, internet, berita, maupun berbagai platform lainnya. Namun, sebagian besar gambar yang tersedia tidak disertai dengan deskripsi teks yang jelas, sehingga dapat menyulitkan aksesibilitas bagi individu dengan keterbatasan visual dan membatasi kemampuan mesin dalam memahami isi gambar. Dilain sisi meskipun manusia mampu menginterpretasikan gambar secara intuitif, pemrosesan gambar oleh mesin masih menjadi tantangan besar dalam bidang kecerdasan buatan.

Salah satu solusi yang banyak dikembangkan untuk mengatasi tantangan ini adalah *image captioning*, yaitu proses menghasilkan deskripsi teks secara otomatis berdasarkan konten visual dalam gambar. *Image captioning* menghubungkan dua bidang utama kecerdasan buatan, yaitu *computer vision* dan natural language processing (NLP), sehingga memungkinkan sistem komputer mengenali dan memahami suatu gambar dalam bentuk bahasa alami. Teknologi ini memiliki berbagai manfaat, termasuk dalam pendidikan, pelacakan video, analisis sentimen, dan peningkatan aksesibilitas bagi penyandang tunanetra.

Berbagai penelitian telah dilakukan untuk mengembangkan model *image captioning* yang lebih akurat. Xu et al. [1] memperkenalkan model berbasis *attention* yang memungkinkan model fokus pada bagian gambar yang lebih relevan saat menghasilkan teks deskripsi. Penelitian Zhang et al. [2] lebih lanjut mengembangkan mekanisme visual attention berbasis fully

convolutional network (FCN)-LSTM untuk meningkatkan akurasi deskripsi gambar. Selain itu, dengan munculnya arsitektur *transformer*, model *image captioning* semakin berkembang pesat. Castro et al. [3] menunjukkan bahwa penggunaan *transformer* dengan *visual attention* dapat meningkatkan efisiensi dan akurasi dalam menghasilkan deskripsi gambar, dengan nilai BLEU-4 mencapai 34,44.

Meskipun berbagai pendekatan telah dikembangkan, masih terdapat beberapa keterbatasan dalam penelitian sebelumnya. Pertama, banyak penelitian masih terbatas pada satu jenis *feature extractor*, seperti ResNet atau VGG, tanpa adanya perbandingan menyeluruh terhadap arsitektur lain yang mungkin lebih efisien. Kedua, sebagian besar evaluasi hanya mengandalkan metrik BLEU, yang meskipun populer, belum cukup untuk menilai kesesuaian semantik dan struktur sintaksis dari deskripsi yang dihasilkan. Ketiga, beberapa penelitian hanya menguji model mereka pada satu dataset tertentu, sehingga belum ada analisis yang cukup luas mengenai performa model di berbagai dataset.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk melakukan perbandingan komprehensif antara berbagai *feature extractor* seperti MobileNet, VGG, dan ResNet dalam tugas image captioning. Model yang dikembangkan akan dievaluasi menggunakan metrik BLEU dan ROUGE pada dataset COCO dan FLICKR8k, untuk memberikan pemahaman yang lebih mendalam tentang efektivitas berbagai metode dalam menghasilkan deskripsi gambar yang akurat dan kontekstual. Dengan demikian, penelitian ini tidak hanya memberikan wawasan mengenai kinerja model, tetapi juga berkontribusi dalam pengembangan sistem *image captioning* yang lebih optimal dan efisien.

1.2. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana pengaruh pemilihan arsitektur CNN pra-latih (Inception, MobileNet, VGG, ResNet) sebagai *feature extractor* terhadap skor BLEU dan Rouge pada model *image captioning*?
2. Apakah penggunaan *padding* dibandingkan *resizing* dengan *stretching* dalam *preprocessing* gambar memberikan dampak signifikan terhadap skor BLEU dan Rouge pada model *image captioning*?
3. Bagaimana variasi jumlah lapisan decoder (3–6 lapisan) memengaruhi kemampuan model dalam menghasilkan caption pada dataset Flickr8k, terutama terkait skor BLEU dan ROUGE?

1.3. Batasan Masalah

1. Penelitian hanya membandingkan performa feature extractor Inception, MobileNet, VGG, dan ResNet.
2. Percobaan preprocessing dibatasi pada dua teknik, yaitu padding (dengan latar belakang hitam) dan resizing dengan stretching (ubah ukuran dengan distorsi aspek rasio).
3. Eksperimen pada jumlah layer decoder hanya mencakup 3–6 layer.
4. Evaluasi model dilakukan pada dataset COCO dan FLICKR8k dengan format teks berbahasa Inggris, dan menggunakan metrik BLEU dan Rouge.

1.4. Tujuan

Tujuan dari penelitian ini adalah untuk memenuhi kewajiban akademik sebesar 4 SKS serta mengembangkan model *image captioning* yang mampu menghasilkan deskripsi otomatis pada gambar. Penelitian ini bertujuan untuk mengevaluasi dan mengembangkan model *deep learning* seperti MobileNet, VGG, dan ResNet dalam menghasilkan deskripsi gambar yang akurat dan relevan. Model yang dikembangkan akan diuji menggunakan metrik evaluasi seperti ROUGE dan BLEU untuk menilai kualitas deskripsi yang dihasilkan. Dengan penelitian ini, diharapkan dapat memberikan kontribusi dalam meningkatkan kemampuan *computer vision* dan *natural language processing*.

1.5. Manfaat

Manfaat dari penelitian ini antara lain:

1. Memberikan kontribusi dalam pengembangan teknologi *image captioning* yang dapat diaplikasikan pada berbagai sektor seperti pendidikan, media, dan layanan bagi penyandang disabilitas.
2. Mendukung individu dengan gangguan penglihatan melalui deskripsi otomatis gambar, sehingga meningkatkan aksesibilitas dan inklusi dalam penggunaan teknologi visual.
3. Meningkatkan penelitian di bidang *computer vision* dan *natural language processing* di Indonesia dengan mengaplikasikan model *deep learning* seperti MobileNet, VGG, dan ResNet untuk menghasilkan deskripsi gambar yang akurat.

1.6. Lokasi dan Waktu Kerja Praktik

Sehubungan dengan pelaksanaan penelitian bersama dosen dari satu jurusan, kerja praktik ini dilaksanakan di Lab MCI meskipun dengan sistem Work From Anywhere (WFA).

Adapun kerja praktik dimulai pada tanggal tanggal 30 Juli 2024 – 30 Oktober 2024.

1.7. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.7.1. Perumusan Masalah

Tahap awal penelitian dimulai dengan mengidentifikasi permasalahan utama, yaitu bagaimana cara kerja deskripsi otomatis pada gambar menggunakan model *deep learning* yang sudah ada pada *website "*tensorflow.org". Penentuan ini dilakukan melalui diskusi dengan Bu Shintami, yang menyampaikan model tidak perlu membuat dari awal, tetapi hanya perlu memodifikasi model yang sudah ada yaitu mobilenet dengan model lainnya seperti VGG, dan ResNet. Kajian literatur juga dilakukan untuk memahami pendekatan dan model yang relevan, seperti MobileNet, VGG, dan ResNet, yang telah digunakan dalam penelitian sebelumnya. Hal ini bertujuan agar kami dapat memahami model yang akan dibuat dengan lebih baik.

1.7.2. Studi Literatur

Pada tahap ini, studi literatur dilakukan untuk memahami berbagai pendekatan dan teknologi dalam *image captioning*. Literatur yang dikaji mencakup penggunaan model *deep learning* untuk menghasilkan deskripsi gambar, penerapan metrik evaluasi seperti ROUGE dan BLEU, serta referensi dari penelitian sebelumnya mengenai pengembangan model yang efisien untuk tugas *computer vision*.

1.7.3. Analisis dan Perancangan Sistem

Proses analisis dilakukan dengan mencoba keefektifan model dengan 2 dataset yaitu dataset COCO dan Flicker8 gambar yang akan digunakan sebagai *input* model. Dataset diambil dari website *"*tensorflow.org" dan dari bu Shintami sendiri untuk mencocokkan gambar dengan deskripsi yang relevan. Selanjutnya, sistem dirancang dengan memanfaatkan model *deep learning* yang sesuai.

Pemilihan model dilakukan berdasarkan kriteria seperti efisiensi, akurasi, dan kemampuan dalam memahami fitur visual.

1.7.4. Implementasi Sistem

Pada tahap implementasi, model *deep learning* seperti MobileNet, VGG, dan ResNet diterapkan untuk menghasilkan deskripsi gambar. Model ini dilatih menggunakan dataset yang telah diproses sebelumnya. Implementasi dilakukan dengan memanfaatkan framework *deep learning* seperti TensorFlow.

1.7.5. Pengujian dan Evaluasi

Tahap ini bertujuan untuk mengukur performa model yang dikembangkan. Model dievaluasi menggunakan metrik ROUGE dan BLEU untuk menilai kualitas deskripsi yang dihasilkan. Pengujian dilakukan untuk membandingkan kinerja model yang berbeda (MobileNet, VGG, ResNet) dalam menghasilkan deskripsi otomatis.

1.7.6. Kesimpulan dan Saran

Setelah pengujian selesai, hasil dari penelitian ini dianalisis untuk menarik kesimpulan. Kesimpulan mencakup performa terbaik dari model yang diuji serta rekomendasi untuk penelitian lebih lanjut. Saran diberikan untuk pengembangan teknologi *image captioning* ke depannya, termasuk kemungkinan integrasi dengan teknologi lain seperti perangkat bantu untuk tunanetra.

1.8. Sistematika Laporan

Pada laporan ini meliputi bagian bagian sebagai berikut

A. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

B. Bab II Profil Perusahaan

Bab ini berisi gambaran umum Laboratorium Manajemen Cerdas Informasi Teknik Informatika ITS mulai dari profil, lokasi perusahaan.

C. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

D. Bab IV Analisis dan Perancangan Arsitektur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

E. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

F. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

G. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil Laboratorium Manajemen Cerdas Informasi (MCI)

Laboratorium Profil Laboratorium Manajemen Cerdas Informasi merupakan salah satu laboratorium di jurusan Teknik Informatika ITS, Laboratorium ini berfokus pada pengembangan keahlian di bidang analisis, sintesis, dan evaluasi proses bisnis serta sistem informasi dalam sistem Enterprise. Lulusan yang dihasilkan diharapkan mampu mengimplementasikan rekayasa pengetahuan dalam aplikasi praktis, melakukan investigasi dan pengujian, serta mengevaluasi kematangan dan kesesuaian prosedur standar serta tata kelola teknologi informasi. Selain itu, laboratorium ini juga mencakup pengelolaan proyek dan sumber daya manusia, serta merancang dan mengimplementasikan solusi berbasis data terdistribusi dan teknologi Big Data.

2.2. Logo Perusahaan

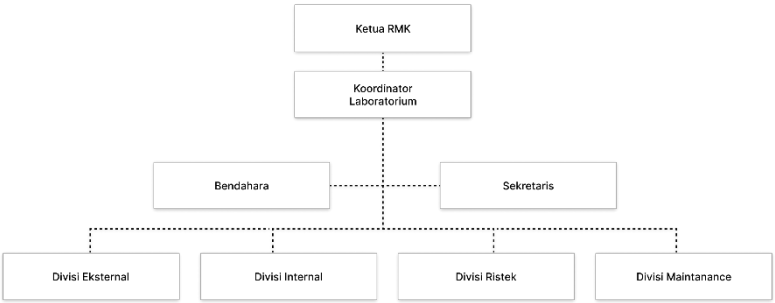
Berikut merupakan gambar logo Laboratorium Manajemen Cerdas Informasi (MCI) Teknik Informatika ITS yang dapat dilihat pada Gambar 2.1.

Gambar 2.1 Logo Lab MCI

2.3. Lokasi

Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111.

2.4. Struktur Perusahaan



Gambar 2.2 Struktur Organisasi Lab MCI

Struktur organisasi lab MCI terdiri dari ketua RMK, koordinator laboratorium, bendahara, sekretaris, divisi eksternal, divisi internal, divisi ristek, dan divisi maintenance yang dapat dilihat pada Gambar 2.2. Daftar anggota lab MCI 2023 adalah sebagai berikut:

Dosen anggota:

1. Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D. (Kepala Lab)
2. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

3. Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.
4. Nurul Fajrin Ariyani, S.Kom., M.Sc.
5. Abdul Munif, S.Kom., M.Sc.
6. Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D
7. Kelly Rossa Sungkono, S.Kom., M.Kom.

Teknisi:

1. Gayuh Adi Rustanto, S.Kom.

Mahasiswa admin:

1. Andika Laksana Putra
2. Hanun Shaka Puspa
3. Akmal Ariq Romadhon
4. Fathin Muhashibi Putra
5. Fadilla Rizky Nurhidayah
6. Dimas Fadilah Akbar

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Penelitian terkait

Perkembangan teknologi berbasis *computer vision* dan *natural language processing* (NLP) telah membuka jalan bagi berbagai aplikasi praktis. Salah satu pendekatan yang semakin populer adalah penerapan *visual attention* dalam model-model pembelajaran mesin. *Visual attention* adalah mekanisme yang memungkinkan model untuk memfokuskan perhatian pada bagian-bagian gambar yang relevan, sehingga meningkatkan akurasi dalam memproses dan menginterpretasikan gambar. Dalam konteks ini, *visual attention* telah terbukti efektif dalam berbagai tugas *computer vision*, seperti deteksi objek dan segmentasi gambar, karena model dapat lebih cerdas dalam memilih fitur-fitur yang penting.

Salah satu penelitian penting dalam bidang ini dilakukan oleh Xu et al. [1] memperkenalkan model berbasis *attention* yang secara otomatis dapat mempelajari untuk menggambarkan konten gambar. Dalam penelitiannya, model ini dilatih dengan dua pendekatan, yaitu deterministik menggunakan teknik *backpropagation* standar, serta stokastik dengan memaksimalkan batas bawah variasional. Model ini juga mampu secara visual memfokuskan perhatiannya pada objek-objek penting di gambar saat menghasilkan kata-kata yang sesuai dalam urutan *output*. Hasilnya, penelitian ini menunjukkan peningkatan yang signifikan dalam performa pada dataset benchmark seperti Flickr8k, Flickr30k, dan MS COCO, memberikan kontribusi besar bagi pengembangan teknik visual attention.

Seiring dengan kemajuan mekanisme attention, teknologi ini telah diadaptasi secara luas dalam *image captioning*, yaitu tugas untuk menghasilkan deskripsi berbasis teks dari gambar. Dalam *image captioning*, visual attention memainkan peran penting dengan membantu model untuk fokus pada elemen-elemen gambar yang relevan selama proses pembangkitan teks. Dengan cara ini, model dapat menghasilkan kalimat yang lebih informatif dan kontekstual. Penelitian sebelumnya, seperti yang dilakukan oleh Zhang et al [2], mengusulkan metode visual attention baru yang berbasis pada kerangka kerja fully convolutional network (FCN)-LSTM. Model ini mengatasi keterbatasan mekanisme attention sebelumnya yang hanya dapat memetakan fitur visual pada grid dengan resolusi tetap dan kurang dapat menangani area luas seperti langit atau pantai. Sugiman mengusulkan penggunaan label grid-wise (segmentasi semantik) yang memungkinkan representasi visual dari grid-cell terkait satu sama lain, menciptakan mekanisme perhatian visual yang lebih terperinci dan terarah secara semantik, yang menghasilkan deskripsi gambar yang lebih akurat, lengkap, dan beragam.

Tidak hanya sampai situ saja penelitian dalam bidang *image captioning* terus berkembang, dengan teknologi yang digunakan semakin canggih. Salah satu kemajuan terbaru adalah penggunaan model transformer, yang lebih unggul dibandingkan dengan model *deep learning* tradisional seperti LSTM, RNN, dan lainnya. Keunggulan utama transformer terletak pada kemampuannya untuk melakukan perhitungan attention secara parallel, yang memungkinkan pemrosesan data yang lebih cepat dan efisien. Beberapa penelitian, seperti yang dilakukan oleh Castro et al. [3], menunjukkan bahwa penggunaan visual attention dalam arsitektur *encoder-decoder* berbasis *transformer* memberikan dampak

signifikan terhadap efisiensi tugas *captioning* gambar. Abigail meneliti pengaruh konfigurasi hyperparameter yang berbeda, termasuk fungsi loss dan optimizers berbasis gradient, serta membandingkan performa berbagai arsitektur konvolusional. Hasil eksperimen mereka menunjukkan bahwa transformer, terutama model Data-efficient Image Transformer (DeiT), memberikan peningkatan yang signifikan dibandingkan dengan model konvolusional lainnya, dengan nilai BLEU-4 mencapai 34,44, mempertegas potensi transformer dalam menghasilkan kualitas *captioning* yang optimal.

Sejarah perkembangan *image captioning* menunjukkan bahwa model tidak hanya memerlukan *feature extractor* yang handal, tetapi juga kemampuan untuk menangkap hubungan kompleks antara fitur visual dan semantik dalam gambar. Salah satu tantangan utama dalam meningkatkan performa model *image captioning* adalah pemilihan *feature extractor* yang tepat. Meskipun banyak penelitian sebelumnya telah mengembangkan berbagai arsitektur untuk tujuan ini, sebagian besar masih terbatas pada pengujian dengan sejumlah kecil dataset atau hanya mengandalkan evaluasi berbasis metrik yang terbatas, sehingga belum memberikan gambaran menyeluruh tentang efektivitasnya.

Maka dari itu penelitian ini bertujuan untuk mengisi celah tersebut dengan melakukan perbandingan mendalam antara berbagai *feature extractor* seperti VGG dan MobileNet secara menyeluruh pada dataset berbeda, seperti COCO dan FLICKR8 dan mengevaluasi model dengan metrik yang lebih luas, termasuk BLEU dan ROUGE score. Dengan demikian, penelitian ini tidak hanya memberikan wawasan tentang performa model, tetapi juga menyediakan pendekatan baru dalam mengevaluasi kualitas *caption* secara lebih komprehensif.

3.2. Landasan Teori

Image captioning adalah tugas yang menggabungkan bidang visi komputer (computer vision) dan pemrosesan bahasa alami (natural language processing) untuk menghasilkan *caption* secara otomatis dari konten visual. Untuk dapat membangun sistem *image captioning* yang efektif, diperlukan pemahaman terhadap konsep-konsep dasar yang menjadi fondasi dari teknologi tersebut. Dalam konteks penelitian ini, pendekatan yang digunakan berbasiskan *Deep Learning* dengan arsitektur *Transformer*. Oleh karena itu, sebelum membahas secara teknis metode yang digunakan, bagian ini akan menguraikan beberapa teori yang relevan, mulai dari konsep dasar Deep Learning, mekanisme Visual Attention yang meniru cara manusia memfokuskan perhatian pada informasi penting, arsitektur Transformer yang menjadi tulang punggung pemrosesan urutan data teks, hingga metode evaluasi seperti BLEU dan ROUGE yang digunakan untuk menilai kesesuaian hasil caption dengan deskripsi referensi.

3.2.1. Deep Learning

Berdasarkan Sarker [4], *Deep Learning* adalah sebuah teknologi berbasis ANN atau *Artificial Neural Network*, yang menggunakan *layer-layer* untuk merepresentasikan abstraksi data untuk membuat model komputasi. Pada setiap *layer* dari sebuah model *Deep Learning*, terdapat sekumpulan *node* yang biasa disebut *neuron.* Setiap *neuron* terdiri atas *input, weight, summation function, activation function*, dan *output*. Sekumpulan *neuron* pada sebuah *layer* akan menerima masukan dari *layer* sebelumnya untuk dikomputasikan pada masing-masing *neuron* sehingga dapat diteruskan ke *layer* berikutnya.

Teknologi *Deep Learning* semakin banyak digunakan dikarenakan beberapa perbedaan dari *machine learning.* Perbedaan

utama dari kedua teknologi ini terletak pada tahap ekstraksi fitur. Pada *Machine Learning*, ekstraksi fitur dilakukan sebelum pelatihan model. Sementara *Deep Learning*, ekstraksi fitur serta pelatihan model pada data yang diberikan dilakukan secara otomatis oleh *layer-layer* pembentuk model tersebut.

3.2.2. Visual Attention

Dalam kehidupan sehari-hari, saat kita membuka mata, terdapat begitu banyak informasi yang dapat kita peroleh. Namun, untuk memahami informasi tersebut, penglihatan kita memerlukan kemampuan untuk memisahkan informasi yang penting dari yang tidak penting, seperti membedakan informasi yang relevan dari yang tidak relevan. Menurut Carrasco [5], Attention merupakan proses seleksi, dimana seleksi ini diperlukan untuk membatasi kapasitas pada proses visual sehingga dapat memprioritaskan beberapa aspek informasi sambil mengabaikan yang lain dengan berfokus pada lokasi atau aspek tertentu dari pemandangan visual.

3.2.3. Transformer

Dalam beberapa decade terakhir, pada tugas dalam bidang Computer Vision (CV) sudah didominasi oleh Convolutional Neural Networks (CNNs). Yang mana hal ini sangalah berbeda dengan Transformer, yang mulai dikenal dalam pengolahan bahasa alami dan kini juga diterapkan dalam bidang CV, membawa pendekatan baru dengan kemampuan untuk menangani hubungan yang lebih kompleks dalam data visual. Berdasarkan Han et al. [6], Transformer didasarkan pada mekanisme self-attention, yang memungkinkan model untuk memberikan perhatian lebih signifikan pada bagian-bagian penting dalam data. Selain itu, Transformer banyak digunakan dalam pemrosesan bahasa alami (NLP), seperti yang terlihat pada model BERT dan GPT-3.

3.2.4. BLEU (Bilingual Evaluation Understudy)

Dalam membuat model *captioning*, proses penyusunan kata-kata menjadi kalimat yang tepat sangat penting. Oleh karena itu, evaluasi menjadi langkah krusial untuk memastikan bahwa *caption* yang dihasilkan sesuai dengan struktur yang benar, seperti hubungan antara Subjek, Predikat, dan Objek dalam bahasa Inggris. Menurut Wolk et al. [7], BLEU score dikembangkan untuk mengukur kesamaan antara *output* model dan terjemahan manusia. Dengan demikian, BLEU score dirancang untuk menghitung sejauh mana hasil terjemahan dari Statistical Machine Translation (SMT) mendekati terjemahan yang diberikan oleh manusia karena memungkinkan berbeda dalam penggunaan kata, urutan kata, dan panjang frasa.

3.2.5. ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Selain BLEU yang digunakan untuk mengukur kesamaan *output* terjemahan teks oleh model dengan terjemahan manusia, terdapat juga metrik ROUGE yang dirancang untuk mengevaluasi kualitas ringkasan teks yang dihasilkan oleh model. Menurut Lin [8], ROUGE score adalah metrik evaluasi yang menghitung kesamaan antara unit teks, seperti kata atau frasa, yang muncul pada ringkasan kandidat dan ringkasan referensi. Dengan menggunakan pendekatan berbasis recall, ROUGE dapat mengidentifikasi sejauh mana ringkasan model mencakup informasi penting dari ringkasan referensi.

BAB IV
ANALISIS DAN PERANCANGAN ARSITEKTUR SISTEM

4.1. Analisis Sistem

Bab ini menjelaskan tahapan dalam merancang dan membangun sistem otomatis untuk deskripsi gambar (image captioning) yang mengadopsi model yang dikembangkan menggunakan TensorFlow. Model ini terinspirasi oleh penelitian Xu et al [1] dengan modifikasi pada komponen decoder, menggunakan arsitektur Transformer-decoder berlapis dua. Tujuan dari sistem ini adalah untuk menghasilkan deskripsi teks otomatis dari gambar dengan memanfaatkan teknik *attention* yang ada pada arsitektur Transformer. Proses perancangan dan analisis sistem dibagi menjadi dua bagian utama, yaitu Definisi Umum Sistem dan Analisis Kebutuhan Sistem.

4.1.1. Definisi Umum Aplikasi

Sistem yang dikembangkan bertujuan untuk menghasilkan deskripsi teks otomatis untuk gambar menggunakan teknologi *deep learning*. Dalam hal ini, arsitektur model menggabungkan fitur visual yang diekstraksi dari gambar dan memanfaatkan model Transformer-decoder untuk menghasilkan teks deskriptif. Secara garis besar, alur proses sistem ini terdiri dari beberapa tahap sebagai berikut:

1. Ekstraksi Fitur Gambar: Fitur dari gambar akan diekstraksi menggunakan model yang sudah dilatih, seperti MobileNet, VGG, atau ResNet. Fitur ini digunakan untuk mengidentifikasi elemen-elemen penting dalam gambar yang diperlukan untuk menghasilkan deskripsi yang akurat.
2. Transformasi Fitur Gambar ke dalam Model Transformer: Fitur gambar yang telah diekstraksi akan menjadi *input* bagi lapisan cross-attention dalam Transformer-decoder. Lapisan

ini memungkinkan model untuk memfokuskan perhatian pada bagian-bagian gambar yang relevan selama proses pembuatan deskripsi.

3. Proses *Captioning*: Pada tahap ini, model akan menghasilkan deskripsi gambar secara bertahap menggunakan mekanisme self-attention dan cross-attention. Proses ini memungkinkan model untuk memahami hubungan antar kata serta menggambarkan gambar dengan cara yang lebih alami dan kontekstual.

4.1.2. Analisis Kebutuhan

Kebutuhan perangkat lunak mencakup berbagai komponen penting yang mendukung pembangunan dan pelatihan model, sedangkan kebutuhan perangkat keras memfasilitasi proses komputasi dan penyimpanan data. Kebutuhan-kebutuhan ini saling melengkapi dalam mendukung pengembangan sistem secara keseluruhan. Secara umum, kebutuhan perangkat lunak dibagi menjadi beberapa aspek utama, seperti kerangka kerja (framework) deep learning, bahasa pemrograman yang digunakan, dan metrik evaluasi model. Sementara itu, kebutuhan perangkat keras meliputi elemen-elemen seperti GPU, memori, dan penyimpanan. Semua komponen ini perlu dirancang dan dipenuhi agar proses pembangunan, pelatihan, dan penerapan model dapat berjalan lancar.

a. Kebutuhan Perangkat Lunak:

Sistem ini menggunakan TensorFlow dan Keras sebagai kerangka kerja utama dalam pengembangan arsitektur *deep learning*. Kedua kerangka kerja tersebut menyediakan berbagai operasi yang diperlukan dalam proses pembangunan dan pelatihan model, khususnya model Transformer. TensorFlow dan Keras mendukung komputasi numerik berskala besar serta memungkinkan optimalisasi model secara efisien. Ketersediaan dokumentasi dan komunitas pengguna yang luas turut mendukung kemudahan dalam pengembangan sistem.

Bahasa pemrograman yang digunakan dalam pengembangan sistem ini adalah Python, yang dipilih berdasarkan dukungannya

terhadap pustaka-pustaka pembelajaran mesin dan analisis data. Python memiliki ekosistem pustaka yang sangat lengkap, seperti NumPy dan Pandas, yang memudahkan pengolahan data numerik dan manipulasi data tabular. Selain itu, Python kompatibel dengan berbagai pustaka *deep learning*, termasuk TensorFlow dan Keras. Sintaksis yang sederhana dan tingkat keterbacaan kode yang tinggi menjadikan Python efisien dalam pengembangan sistem berbasis kecerdasan buatan. Dengan demikian, Python merupakan bahasa pemrograman yang tepat untuk digunakan dalam sistem ini.

Evaluasi performa model dilakukan dengan menggunakan metrik-metrik yang sesuai untuk tugas *image captioning*. Metrik *accuracy* dan *loss* digunakan untuk mengevaluasi performa pelatihan model dalam aspek klasifikasi. Metrik *accuracy* mengukur proporsi prediksi yang benar, sedangkan *loss* menunjukkan seberapa jauh hasil prediksi dari target yang diharapkan. Selain itu, kualitas deskripsi teks yang dihasilkan oleh model dievaluasi menggunakan metrik BLEU dan ROUGE. Penggunaan kedua jenis metrik ini bertujuan untuk memperoleh evaluasi yang komprehensif terhadap aspek kuantitatif dan kualitatif dari sistem.

b. Kebutuhan Perangkat Keras:

Pelatihan model Transformer dalam sistem ini memerlukan sumber daya komputasi yang tinggi, khususnya pada sisi unit pemroses grafis (*Graphics Processing Unit*). Oleh karena itu, diperlukan GPU yang mendukung arsitektur CUDA, seperti NVIDIA GTX 1650, untuk mempercepat proses pelatihan. Selain menggunakan GPU lokal, sistem ini juga memanfaatkan layanan GPU berbasis cloud, yakni L4 GPU yang tersedia pada platform Google Colab. Penggunaan GPU dengan spesifikasi tersebut mampu mempercepat proses pelatihan model dan meningkatkan efisiensi pengolahan data. Hal ini penting untuk memastikan sistem dapat dilatih dalam waktu yang lebih singkat dan dengan performa optimal.

Dalam hal kapasitas memori, sistem ini memerlukan RAM minimal sebesar 16 GB untuk menunjang proses pelatihan model dalam skala besar. RAM yang cukup memungkinkan sistem untuk memproses data dalam ukuran besar serta mengelola pelatihan model dengan ukuran *batch* yang tinggi. Penggunaan memori yang memadai juga mengurangi risiko terjadinya *bottleneck* selama pelatihan. Dukungan layanan komputasi seperti Google Colab turut memberikan fleksibilitas dalam penggunaan memori tambahan. Dengan demikian, kebutuhan memori menjadi aspek penting yang tidak dapat diabaikan dalam implementasi sistem ini.

Selain aspek komputasi dan memori, kapasitas penyimpanan juga menjadi komponen penting dalam sistem. Sistem memerlukan ruang penyimpanan minimal sebesar 100 GB yang digunakan untuk menyimpan dataset serta hasil model yang telah dilatih. Dataset yang digunakan dalam pelatihan umumnya memiliki ukuran yang besar, sehingga penyimpanan yang mencukupi sangat diperlukan. Selain itu, model hasil pelatihan dan berbagai hasil evaluasi juga memerlukan ruang yang cukup agar dapat diakses dan digunakan ulang dengan mudah. Oleh sebab itu, pemilihan media penyimpanan dengan kapasitas memadai merupakan bagian integral dari infrastruktur sistem.

4.2. Perancangan Infrastruktur Sistem

Bagian memfokuskan perancangan infrastruktur sistem pada pengembangan dan integrasi komponen-komponen model untuk mendukung tugas image captioning. Model yang dikembangkan menggunakan arsitektur encoder-decoder berbasis transformer, di mana encoder berupa jaringan CNN (misalnya MobileNet, VGG, ResNet) yang mengekstraksi fitur visual dari citra, sedangkan decoder merupakan model bahasa berbasis transformer yang menghasilkan deskripsi teks. Dalam arsitektur yang dirancang, fitur yang diperoleh dari CNN (vektor embedding) kemudian disesuaikan dengan token yang dihasilkan dari teks caption dataset oleh komponen *decoder* untuk menghasilkan *caption* yang sesuai.

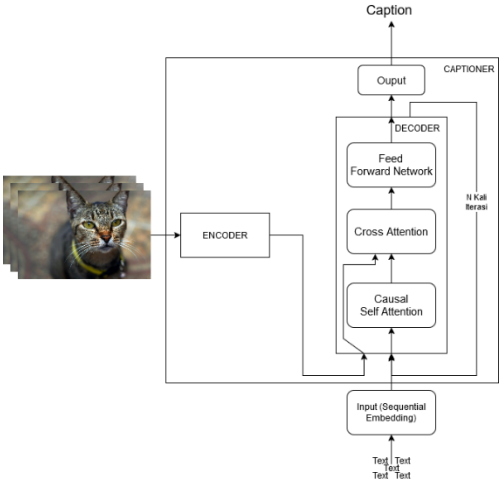
4.2.1. Arsitektur Sistem

Arsitektur sistem *image captioning* yang digunakan pada penelitian ini menggunakan model yang sudah dibuat oleh TensorFlow yang mana model ini terinpirasi dari Xu et al [1] dengan modifikasi pada komponen *decoder* menggunakan 2-*layer Transformer-decoder*. Desain arsitektur *Image Captioner* pada penelitian ini secara umum kami bagi menjadi 3 bagian yang dijelaskan sebagai berikut:

1. Ekstraksi Fitur Gambar (*Encoder*)

Pada tahap ekstraksi fitur gambar, kami menggunakan empat model ekstraksi fitur yang berbeda, yaitu MobileNet, VGG,ResNet, dan inception. Masing-masing model ini dipilih untuk mengidentifikasi fitur yang relevan dari gambar *input*. Gambar yang diterima akan diproses oleh salah satu model ekstraksi fitur ini untuk menghasilkan representasi fitur yang akan digunakan oleh sistem *Captioner*. Setelah dilakukan evaluasi, model yang memberikan hasil terbaik akan dipilih untuk tahap selanjutnya.

2. *Transformer-decoder*



Gambar 4.1. Arsitektur Model Deep Learning

Transformer-decoder berfungsi untuk menghasilkan teks sebagai *output* berdasarkan fitur gambar yang telah diproses oleh encoder sebelumnya. Fokus utama dari *transformer-decoder* adalah membangun sistem yang mampu menghasilkan teks deskriptif untuk gambar (image captioning) atau aplikasi lain yang membutuhkan generasi teks dari representasi non-teks. Seperti yang terlihat pada Gambar 4.1 *Transformer-decoder* sendiri terdiri dari tiga komponen utama, yaitu *Input*, *Decoder*, dan *Output*.

a. Input

Pada bagian *input*, token yang diterima dari model diproses menjadi representasi vektor yang dapat dimengerti oleh model Transformer. Token *embedding* digunakan untuk mengonversi ID token menjadi vektor numerik, sementara *positional encoding* ditambahkan untuk memberikan informasi posisi setiap token dalam urutan *input*. Proses ini memastikan bahwa urutan dan posisi dalam *input* gambar dapat dipahami oleh model. Selain itu, *masking* diterapkan untuk memastikan bahwa *padding* tidak mempengaruhi hasil proses.

Seiring dengan pemrosesan token, pada bagian *Input*, *SeqEmbedding* berfungsi untuk menggabungkan informasi token dan posisi dalam representasi vektor yang terintegrasi. Pada tahap ini, pertama-tama dilakukan lookup *embedding* untuk setiap ID token dan posisi token dalam urutan *input*. Token *embedding* digunakan untuk mengonversi ID token menjadi vektor numerik, sementara *positional encoding* memberikan informasi posisi untuk setiap token. Kedua *embedding* ini kemudian dijumlahkan untuk membentuk representasi kombinasi token dan posisi yang akan diproses lebih lanjut. Penggunaan *mask*_zero=True diterapkan untuk memastikan bahwa *padding* tidak mempengaruhi hasil perhitungan.

b. Decoder

Decoder merupakan bagian inti dari *Transformer-decoder* yang bertugas untuk memproses informasi yang diterima dan menghasilkan prediksi *output*. Pada bagian ini, terdapat beberapa komponen yang bekerja secara bersamaan:

- Causal Self Attention (CSA), mekanisme yang memastikan setiap posisi *output* dalam sebuah urutan hanya dapat memperhatikan token-token yang berada di posisi sebelumnya maupun dirinya sendiri. Hal ini dicapai dengan menerapkan sebuah *causal mask* pada mekanisme *attention* yang membatasi perhatian model hanya pada token yang tidak berada di masa depan. Secara matematis, perhatian dihitung menggunakan fungsi:

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

  di mana Q, K, dan V masing-masing adalah matriks *query*, *key*, dan *value*, dan d_k adalah dimensi dari key. Dalam CSA, sebelum diterapkan *softmax*, *mask* M ditambahkan ke skor perhatian sehingga posisi masa depan diatur ke nilai negatif tak terhingga, yaitu:

  $$Attention(Q, K, V) = softmax\left(\frac{QK^{\top}}{\sqrt{d_k}} + M\right)V \qquad ....(3.2)$$

  Mask ini memiliki nilai $-\infty$ di atas diagonal utama dan nol di bawahnya, memastikan bahwa setiap token hanya dapat memperhatikan token yang berada di sebelah kiri atau sama dengan posisinya. Dengan demikian, mekanisme ini menjaga sifat "causal" dari model, mencegah "mengintip" token yang berada di masa depan selama proses pelatihan dan inferensi.
- Cross Attention, mekanisme yang memungkinkan model untuk menghubungkan informasi yang diperoleh dari gambar *input* (melalui encoder gambar) dengan urutan token yang sedang dihasilkan dalam *decoder*. Ini membantu *decoder* untuk menghasilkan teks yang lebih relevan dengan gambar. Mekanisme ini juga menggunakan *layer MultiHeadAttention* sama seperti CSA, namun menggunakan *query* berupa hasil dari CSA (konteks dari teks) dan *value* berupa fitur yang telah diekstrak pada bagian *encoder*.e
- Feed Forward Network (FFN), setelah melalui proses *attention*, Feed Forward Network diterapkan untuk memproses setiap token secara independen menggunakan

lapisan transformasi non-linear. FFN membantu model dalam mempelajari representasi yang lebih kompleks dan meningkatkan kemampuan model dalam menghasilkan teks yang lebih akurat.

c. *Output*

Pada bagian *Output*, bagian yang digunakan untuk menghasilkan skor atau *logits* bagi setiap token dalam kosakata model. *Layer* ini tidak hanya sekadar memetakan *input* menjadi skor, tetapi juga menambahkan bias berdasarkan distribusi token dalam dataset. Bias ini dihitung melalui fungsi *adapt*, yang menganalisis frekuensi kemunculan setiap token dalam data pelatihan. Fungsi *adapt* menghitung distribusi probabilitas dari token, menormalkannya, dan menetapkan bias untuk setiap token agar mencerminkan seberapa sering token tersebut muncul dalam data. Token yang dilarang, seperti token tidak dikenal ([UNK]), diberi bias negatif yang sangat besar agar tidak dipilih dalam hasil akhir. Dengan pendekatan ini, TokenOutput memungkinkan model menghasilkan distribusi token yang lebih realistis dan relevan dengan data, mengurangi kesalahan prediksi pada token yang jarang muncul atau tidak valid.

3. *Captioner*

Pada bagian ini semua bagian model sebelumnya disatukan menjadi satu kesatuan model yang utuh untuk melakukan training maupun *captioning*. Tujuan utamanya adalah untuk menghasilkan deskripsi berupa teks dari gambar yang diberikan. Proses ini dimulai dengan mengekstraksi fitur gambar (img_*feature*s) dan menginisialisasi daftar token *output* dengan token [START]. Fitur gambar yang telah diekstraksi kemudian dipasangkan dengan token yang ada untuk diteruskan ke model. Model kemudian menghasilkan daftar *logits*, yang akan digunakan untuk memilih token berikutnya berdasarkan probabilitas yang dihitung. Proses ini diulang hingga model menghasilkan token [END], yang menandakan akhir dari *caption* yang dihasilkan. Dengan demikian, model ini dapat secara otomatis menghasilkan deskripsi yang relevan dan tepat untuk gambar yang diberikan.

4.2.2. Data *Preparation*

Penelitian ini menggunakan dua dataset yang berbeda, yaitu dataset Flickr8k dan COCO. Dataset Flickr8k diperoleh dari situs web TensorFlow, sementara dataset COCO diperoleh dari Ibu Shintami. Dataset pertama (Flickr8k) berisi pasangan nama foto dan *caption*, dengan contoh berikut: "1000268201_693b08cb0e.jpg#0 A child in a pink dress is climbing up a set of stairs in an entryway." Setiap gambar dalam dataset ini dilengkapi dengan lima *caption*, sehingga total baris data mencapai 40.460 ribu baris. Sementara itu, dataset kedua (COCO) juga berisi pasangan nama foto dan *caption*, dengan contoh berikut: "http://images.cocodataset.org/train2017/000000081942.jpg, 'a street scene with a traffic light, an awning, a poster, and a tall telephone pole.'" Pada dataset COCO, setiap foto hanya memiliki satu *caption*, dan jumlah total baris data mencapai 174.295 ribu baris.

Pada tahap awal, kami diarahkan untuk melatih model menggunakan dataset Flickr8k terlebih dahulu guna mengevaluasi performa model. Selanjutnya, pada percobaan berikutnya, model diuji menggunakan dataset COCO yang diberikan oleh Ibu Shintami. Setelah eksperimen dengan kedua dataset ini, Ibu Shintami menyarankan untuk menggunakan dataset COCO yang telah diberikan. Setelah model selesai dilatih, tahap selanjutnya adalah pengujian menggunakan dataset uji yang berbeda, yaitu dataset yang tersedia di Kaggle dengan nama "220k-GPT4Vision Image *Captions*." Tujuan dari eksperimen ini adalah untuk mengevaluasi apakah *caption* yang dihasilkan oleh model sesuai dengan struktur kalimat yang terdapat pada dataset uji atau masih jauh berbeda.

4.2.3. *Text tokenization*

Setelah data siap, tahap selanjutnya adalah *preprocessing* data yang diperlukan untuk mempersiapkan *caption* agar dapat diproses oleh model. Karena tujuan dari model yang akan dikembangkan adalah menghasilkan *caption*, yang pada dasarnya merupakan

sebuah kalimat yang terdiri dari kata-kata, maka penting untuk mengonversi kata-kata dalam *caption* pada dataset pelatihan menjadi token. Tokenisasi ini memungkinkan model untuk memproses teks dalam format yang dapat dipahami dan digunakan untuk menghasilkan *caption* baru. Proses ini dimulai dengan menggunakan *layer TextVectorization* untuk mengubah teks menjadi urutan bilangan bulat. Langkah pertama adalah dengan menggunakan metode *adapt* untuk mengiterasi seluruh *caption* dalam dataset, membagi *caption* menjadi kata-kata, dan kemudian membangun kosa kata yang mencakup kata-kata yang paling sering muncul. Setelah kosa kata terbentuk, setiap kata dalam *caption* akan dipetakan ke dalam indeks kosa kata tersebut, sehingga menghasilkan urutan token. Semua urutan token ini kemudian dipadatkan hingga panjang yang telah ditentukan, dalam hal ini 50 token. Selain itu, proses ini juga mencakup pembuatan pemetaan antara kata dan indeks serta sebaliknya, yang berguna untuk memvisualisasikan hasil dan mendekodekan *caption* yang dihasilkan oleh model. Dengan cara ini, *caption* dalam dataset pelatihan dapat dipersiapkan untuk digunakan dalam pelatihan model *captioning*.

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari *Image Captioning*, yang kami gunakan. Implementasi ini terbagi menjadi 4 bagian yaitu, pemuatan data, pembuatan model *image captioning*, pelatihan model, dan evaluasi hasil *caption*.

5.1. Pemuatan Data

Pada penelitian ini akan digunakan dua jenis dataset yaitu flickr8k dan COCO. Dataset flickr8k digunakan memiliki jumlah data sekitar 8000 gambar. Di lain sisi, dataset COCO memiliki sekitar 100 ribu gambar. Pada dataset flickr8k pemuatan gambar secara sequensial masih mungkin dilakukan. Namun, pada dataset COCO yang berukuran cukup besar perlu adanya pendekatan khusus untuk pemuatan data kedalam model pelatihan.

5.1.1. Pemuatan dataset flickr8k

Dataset ini terdiri dari sekitar 8.000 gambar, di mana masing-masing gambar memiliki lima deskripsi teks (caption). Oleh karena itu, proses pengunduhan secara berurutan (sekuensial) terhadap gambar masih memungkinkan dilakukan. Dataset ini dikenal sebagai Flickr8k dan digunakan secara luas untuk tugas image captioning dalam bidang computer vision dan natural language processing. Dataset ini juga mencakup berkas teks yang berisi pasangan nama file gambar dan caption-nya, yang selanjutnya dipisahkan menjadi data pelatihan (training) dan data pengujian (testing). Pembagian ini bertujuan untuk melatih model pada data pelatihan dan menguji performa model secara objektif pada data yang belum pernah dilihat sebelumnya.

Kode Semu 1 Pemuatan data flickr8k

```
1. FUNGSI flickr8k(path='flickr8k'):
```

```
2.      UBAH `path` menjadi objek Path menggunakan
`pathlib.Path`
3.
4.      JIKA jumlah file di dalam `path` kurang dari 16197:
5.          UNDUH dataset gambar dari URL berikut:
6.              origin =
'https://github.com/jbrownlee/Datasets/releases/download/Flick
r8k/Flickr8k_Dataset.zip'
7.          EKSTRAK file ke direktori `path`
8.
9.          UNDUH dataset teks dari URL berikut:
10.             origin =
'https://github.com/jbrownlee/Datasets/releases/download/Flick
r8k/Flickr8k_text.zip'
11.         EKSTRAK file ke direktori `path`
12.
13.     BACA file `Flickr8k.token.txt` dari `path` dan PECAH
menjadi baris
14.     PECAH setiap baris menjadi pasangan (filename,
caption)
15.     HAPUS bagian `#` dari filename
16.
17.     INISIALISASI dictionary `cap_dict` untuk menyimpan
keterangan berdasarkan filename
18.     UNTUK setiap pasangan (filename, caption):
19.         TAMBAHKAN caption ke dalam `cap_dict`
berdasarkan filename
20.
21.     BACA file `Flickr_8k.trainImages.txt` dari `path` untuk
daftar file training
22.     BUAT `train_captions` dengan:
23.         (Path ke file gambar, daftar keterangan untuk file
tersebut dari `cap_dict`)
24.
```

```
25.    BACA file `Flickr_8k.testImages.txt` dari `path` untuk
daftar file testing
26.    BUAT `test_captions` dengan:
27.        (Path ke file gambar, daftar keterangan untuk file
tersebut dari `cap_dict`)
28.
29.    BUAT dataset TensorFlow `train_ds` dari
`train_captions`
30.    BUAT dataset TensorFlow `test_ds` dari `test_captions`
31.
32.    KEMBALIKAN `train_ds` dan `test_ds`
```

5.1.2. Pemuatan dataset COCO

Dataset COCO ini terbagi menjadi dua bagian yaitu *train* dan *test*. Masing – masing baris data dari dataset ini berisi tautan menuju gambar dan teks *caption*. Pada setiap bagian data tersebut memiliki lebih dari 100 ribu baris data. Oleh karena itu, untuk pemuatan data gambar tidak memungkinkan untuk dimuat langsung dari tautan yang tersedia, sehingga diperlukan pengunduhan gambar terlebih dahulu dan kemudian dilakukan pemuatan kedalam model. Pengunduhan data gambar dapat dipercepat dengan melakukan *Multithreading*, yaitu dengan memaksimalkan kinerja komputer dan internet untuk mengunduh gambar secara paralel. Berikut kode semu pengunduhan dataset COCO, dan pemuatan data:

Kode Semu 2 Pengunduhan dan pemuatan daset COCO

```
1.  FUNGSI download_image(url, filename):
2.      BUAT direktori tujuan jika belum ada
3.      UNDUH gambar dari URL dan simpan secara lokal di
`filename`
4.      KEMBALIKAN status keberhasilan atau pesan kesalahan
```

5. **FUNGSI** download_img_concurrent(dir_path, df, start_index=0, end_index=None, max_workers=20):
6. JIKA `end_index` tidak diberikan ATAU lebih besar dari panjang DataFrame:
7. TETAPKAN `end_index` ke panjang DataFrame
8.
9. PILIH subset DataFrame berdasarkan indeks `start_index` dan `end_index`
10. IKAT `dir_path` ke fungsi `download_image_with_progress` menggunakan `partial`
11. JALANKAN unduhan paralel untuk setiap URL menggunakan `ThreadPoolExecutor`
12. TAMPILKAN progress menggunakan `tqdm`
13. CETAK "All downloads completed."

14. **FUNGSI** coco(train_csv='train_split_2.csv', test_csv='z_test_final_2.csv'):
15. BACA file CSV `train_csv` dan `test_csv` ke DataFrame
16. GRUPKAN caption berdasarkan URL untuk data train dan test
17. BUAT list tuple (path gambar, daftar caption) untuk data train
18. BUAT list tuple (path gambar, daftar caption) untuk data test
19. KONVERSI list tuple ke dataset TensorFlow
20. KEMBALIKAN dataset TensorFlow untuk train dan test

21. **FUNGSI** download_image_with_progress(url, dir_path):
22. EKSTRAK nama file dari `url`
23. TENTUKAN path penyimpanan `save_as` dengan `dir_path` dan nama file
24. PANGGIL `download_image` dengan URL dan path `save_as`
25. KEMBALIKAN status unduhan

26. **PROGRAM UTAMA**:
27. PANGGIL `download_img_concurrent` dengan path direktori, DataFrame, dan parameter lainnya
28. PANGGIL `coco` untuk memuat dataset caption train dan test

5.2. Pembuatan Model *Image Captioning*

Arsitektur *Image Captioning* yang kami gunakan terdiri dari tiga komponen utama, yaitu *Encoder*, *Transformer-Decoder*, dan *Captioner* yang berfungsi menggabungkan seluruh proses. Encoder bertugas mengubah gambar menjadi representasi fitur menggunakan arsitektur CNN yang telah dilatih sebelumnya (pretrained). Representasi fitur ini kemudian menjadi masukan bagi *Transformer-Decoder* untuk menghasilkan urutan kata yang sesuai. *Transformer-Decoder* mencocokkan fitur gambar dengan teks yang tersedia dalam dataset untuk membentuk deskripsi yang relevan. Seluruh proses ini dikendalikan oleh *Captioner* yang mengoordinasikan kerja *Encoder* dan *Decoder* secara terintegrasi.

5.2.1. Encoder pada Model

Bagian ini berguna untuk mengubah gambar menjadi representasi vektor, sehingga dapat dicocokkan dengan teksnya. Hal tersebut dapat dilakukan dengan model pra-latih gambar dari arsitektur – arsitektur pengolahan gambar yang tersedia secara umum, seperti ResNet, MobileNet, dan VGG. Penggunaan model pra-latih tersebut tidak hanya menghasilkan representasi vektor biasa, namun juga berguna untuk mewakili gambar.

Kode Semu 3 *Encoder* gambar

1. TETAPKAN konstanta `IMAGE_SHAPE` ke (224, 224, 3) untuk mengubah seluruh bentuk image ke ukuran tersebut
2. FUNGSI Encode(image_path):
3. BACA file gambar dari `image_path` menggunakan `tf.io.read_file`

4. DEKODE file gambar menjadi format JPEG dengan 3 kanal menggunakan `tf.io.decode_jpeg`
5. UBAH ukuran gambar menjadi `IMAGE_SHAPE[:-1]` menggunakan `tf.image.resize`
6. KEMBALIKAN gambar yang sudah didekode dan diubah ukurannya

5.2.2. *Transformer-Decoder* pada Model

Pada bagian ini, model memiliki tiga komponen utama, yaitu Input (*Embedding Teks*), Decoder, dan Output. Proses *embedding* dilakukan tidak hanya pada setiap token, tetapi juga pada posisi masing-masing token dalam teks melalui *positional encoding*. *Decoder* terdiri dari tiga bagian utama, yaitu *Causal Self Attention* untuk memahami konteks dalam urutan teks, *Cross Attention* untuk menghubungkan informasi dari fitur gambar, dan *Feed Forward* Network untuk transformasi lanjutan. *Output* dari model berupa skor logits pada setiap token, yang mencerminkan probabilitas masing-masing kata dalam kosakata. Skor logits ini kemudian digunakan untuk melakukan klasifikasi multikelas dalam membentuk kalimat deskripsi gambar.

A. Input (Embedding Teks)

Pada bagian ini kode mendefinisikan kelas SeqEmbedding yang merupakan lapisan *embedding* khusus untuk teks dalam arsitektur *transformer*. Kelas ini menggabungkan dua jenis *embedding*, yaitu *token embedding* dan *positional encoding*, untuk merepresentasikan urutan kata dalam teks. *Token embedding* dilakukan menggunakan *Embedding layer* berdasarkan ukuran kosakata (vocab_size), sementara *positional embedding* dilakukan berdasarkan panjang maksimum sekuens (max_length) agar model dapat memahami urutan kata. Dalam metode *call*, *token embedding* diterapkan pada input, kemudian posisi setiap token dihitung menggunakan tf.range, dan *positional encoding* ditambahkan ke

token embedding menggunakan *layer Add*. Hasil akhir dari penjumlahan kedua *embedding* ini dikembalikan sebagai representasi teks yang siap digunakan oleh bagian *decoder* dalam model.

Kode Semu 4 *Encoder* gambar

```
1. DEFINISIKAN kelas `SeqEmbedding` yang diturunkan dari `tf.keras.layers.Layer`
2.
3. KONSTRUKTOR `__init__(vocab_size, max_length, depth)`:
4.     PANGGIL konstruktor superclass
5.     INISIALISASI lapisan `pos_embedding` dengan:
6.         `Embedding` dengan `input_dim=max_length`, `output_dim=depth`
7.     INISIALISASI lapisan `token_embedding` dengan:
8.         `Embedding` dengan `input_dim=vocab_size`, `output_dim=depth`, `mask_zero=True`
9.     INISIALISASI lapisan `add` dengan:
10.        `Add`
11.
12. FUNGSI `call(seq)`:
13.    TERAPKAN `token_embedding` pada `seq` untuk menghasilkan token embedding
14.    HITUNG posisi menggunakan `tf.range` untuk panjang sekuens
15.    TAMBAHKAN dimensi baru pada posisi untuk kompatibilitas dengan embedding
16.    TERAPKAN `pos_embedding` pada posisi untuk menghasilkan posisi embedding
17.    GABUNGKAN token embedding dan posisi embedding menggunakan `add`
18.    KEMBALIKAN hasil penjumlahan
```

B. Decoder pada Model

Pada bagian ini kode mendefinisikan komponen-komponen utama dari Decoder dalam arsitektur Image Captioning, yang terdiri dari Causal Self Attention, Cross Attention, dan Feed Forward Network. Kelas CausalSelfAttention menggunakan mekanisme attention satu arah (causal) untuk memastikan bahwa prediksi setiap token hanya dipengaruhi oleh token-token sebelumnya, dilengkapi dengan normalisasi dan penjumlahan residu. Kelas CrossAttention memungkinkan Decoder memanfaatkan informasi dari fitur gambar yang dihasilkan oleh Encoder dengan mencocokkan urutan teks dengan representasi visual, sambil menyimpan skor attention untuk interpretasi. Sementara itu, kelas FeedForward menerapkan transformasi non-linear berlapis pada hasil attention guna memperkuat representasi fitur, dengan dropout dan normalisasi untuk meningkatkan generalisasi. Ketiga komponen ini digabungkan dalam kelas DecoderLayer, yang menerima input berupa pasangan urutan teks dan fitur gambar, kemudian mengalirkan data melalui self-attention, cross-attention, dan feedforward secara bertahap untuk menghasilkan keluaran akhir dari Decoder.

Kode Semu 4 *Decoder* Teks dan gambar

```
1. DEFINISIKAN kelas `CausalSelfAttention` yang
diturunkan dari `tf.keras.layers.Layer`
2.   KONSTRUKTOR `__init__(**kwargs)`:
3.     PANGGIL konstruktor superclass
4.     INISIALISASI `mha` dengan
`MultiHeadAttention(**kwargs)`
5.     INISIALISASI `add` dengan `Add()`
6.     INISIALISASI `layernorm` dengan
`LayerNormalization()`
7.
8.   FUNGSI `call(x)`:
9.     HITUNG `attn` dengan `mha(query=x, value=x,
use_causal_mask=True)`
```

```
10.    PERBARUI `x` dengan `add([x, attn])`
11.    KEMBALIKAN `layernorm(x)`
12.
13. DEFINISIKAN kelas `CrossAttention` yang diturunkan
dari `tf.keras.layers.Layer`
14.  KONSTRUKTOR `__init__(**kwargs)`:
15.    PANGGIL konstruktor superclass
16.    INISIALISASI `mha` dengan
`MultiHeadAttention(**kwargs)`
17.    INISIALISASI `add` dengan `Add()`
18.    INISIALISASI `layernorm` dengan
`LayerNormalization()`
19.
20.  FUNGSI `call(x, y, **kwargs)`:
21.    HITUNG `attn, attention_scores` dengan `mha(query=x,
value=y, return_attention_scores=True)`
22.    SIMPAN `attention_scores` ke
`self.last_attention_scores`
23.    PERBARUI `x` dengan `add([x, attn])`
24.    KEMBALIKAN `layernorm(x)`
25.
26. DEFINISIKAN kelas `FeedForward` yang diturunkan dari
`tf.keras.layers.Layer`
27.  KONSTRUKTOR `__init__(units, dropout_rate=0.1)`:
28.    PANGGIL konstruktor superclass
29.    INISIALISASI `seq` dengan `Sequential` yang berisi:
30.      `Dense(units=2*units, activation='relu')`
31.      `Dense(units=units)`
32.      `Dropout(rate=dropout_rate)`
33.    INISIALISASI `layernorm` dengan
`LayerNormalization()`
34.
35.  FUNGSI `call(x)`:
36.    PERBARUI `x` dengan `x + seq(x)`
37.    KEMBALIKAN `layernorm(x)`
```

```
38.
39. DEFINISIKAN kelas `DecoderLayer` yang diturunkan dari `tf.keras.layers.Layer`
40.   KONSTRUKTOR `__init__(units, num_heads=1, dropout_rate=0.1)`:
41.     PANGGIL konstruktor superclass
42.     INISIALISASI `lstm` dengan `LSTM(units, return_sequences=True, return_state=True)`
43.     INISIALISASI `self_attention` dengan `CausalSelfAttention(num_heads=num_heads, key_dim=units, dropout=dropout_rate)`
44.     INISIALISASI `cross_attention` dengan `CrossAttention(num_heads=num_heads, key_dim=units, dropout=dropout_rate)`
45.     INISIALISASI `ff` dengan `FeedForward(units=units, dropout_rate=dropout_rate)`
46.
47.   FUNGSI `call(inputs, training=False)`:
48.     PECAH `inputs` menjadi `in_seq` dan `out_seq`
49.     PERBARUI `out_seq` dengan `self_attention(out_seq)`
50.     PERBARUI `out_seq` dengan `cross_attention(out_seq, in_seq)`
51.     SIMPAN `cross_attention.last_attention_scores` ke `self.last_attention_scores`
52.     PERBARUI `out_seq` dengan `ff(out_seq)`
53.     KEMBALIKAN `out_seq`
```

C. Output pada Model

Kode ini mendefinisikan bagian output dari model Image Captioning yang bertugas menghasilkan skor logits untuk setiap token dalam proses klasifikasi multikelas. Kelas TokenOutput mewarisi dari tf.keras.layers.Layer dan menggunakan layer Dense untuk memproyeksikan output Decoder ke dalam dimensi ukuran kosakata. Token-token tertentu seperti [UNK], [START], dan

token kosong dapat dikecualikan melalui parameter banned_tokens, yang diproses dalam fungsi adapt() untuk menghitung distribusi token dari dataset dan menghasilkan bias log-probabilitas yang akan ditambahkan ke output logits. Bias ini disesuaikan untuk menghindari pemilihan token yang tidak diinginkan dan membantu model belajar distribusi token yang lebih natural. Pada akhirnya, fungsi call() akan menjumlahkan output dari layer Dense dengan bias tersebut untuk memberikan skor akhir kepada setiap token.

Kode Semu 5 *Output* pada Model

```
1. DEFINISIKAN kelas `TokenOutput` yang diturunkan dari `tf.keras.layers.Layer`
2. KONSTRUKTOR `__init__(tokenizer, banned_tokens=('', '[UNK]', '[START]'), **kwargs)`:
3.     PANGGIL konstruktor superclass
4.     INISIALISASI `dense` dengan `Dense(units=tokenizer.vocabulary_size(), **kwargs)`
5.     SIMPAN `tokenizer` ke `self.tokenizer`
6.     SIMPAN `banned_tokens` ke `self.banned_tokens`
7.     INISIALISASI `bias` dengan `None`
8.
9.   FUNGSI `adapt(ds)`:
10.    INISIALISASI `counts` sebagai `collections.Counter()`
11.    BUAT `vocab_dict` dari `tokenizer.get_vocabulary()`
12.    UNTUK setiap `tokens` dalam `tqdm.tqdm(ds)`:
13.      PERBARUI `counts` dengan `tokens.numpy().flatten()`
14.    INISIALISASI `counts_arr` dengan `np.zeros(shape=(tokenizer.vocabulary_size(),))`
15.    PERBARUI `counts_arr` dengan nilai dari `counts`
16.    UNTUK setiap `token` dalam `banned_tokens`:
17.      SET `counts_arr[vocab_dict[token]]` ke 0
18.    HITUNG `total` sebagai `counts_arr.sum()`
19.    HITUNG `p` sebagai `counts_arr/total`
20.    SET `p[counts_arr==0]` ke 1.0
21.    HITUNG `log_p` sebagai `np.log(p)`
```

```
    HITUNG `entropy` sebagai `-(log_p*p).sum()`
    CETAK `Uniform entropy` dan `Marginal entropy`
    SIMPAN `log_p` ke `self.bias`
    SET `self.bias[counts_arr==0]` ke -1e9

FUNGSI `call(x)`:
    HITUNG `x` sebagai `dense(x)`
    KEMBALIKAN `x + self.bias`
```

5.2.3. *Captioner pada Model*

Kode ini menyatukan seluruh komponen model image captioning ke dalam satu kelas bernama Captioner, yang diturunkan dari tf.keras.Model, sehingga model yang telah dilatih dapat menghasilkan caption otomatis dari gambar input. Di dalam konstruktor, Captioner menginisialisasi tokenizer, feature extractor, embedding untuk urutan token, beberapa layer decoder, dan output layer. Metode call() menangani alur data dari gambar dan teks melalui ekstraksi fitur, embedding, hingga decoding dan prediksi token. Selain itu, terdapat metode simple_gen() yang berfungsi untuk menghasilkan caption secara otomatis berdasarkan gambar dengan memulai dari token [START] dan melakukan prediksi token demi token hingga mencapai token [END], dengan opsi pengaturan temperatur untuk mengontrol tingkat kerandoman prediksi. Kombinasi ini memungkinkan proses inference caption secara end-to-end dari input gambar.

Kode Semu 6 *Captioner* pada Model

```
1.  DEFINISIKAN kelas `Captioner` yang diturunkan dari
`tf.keras.Model`
2.  METODE KELAS `add_method(cls, fun)`:
3.      SET `fun` sebagai metode kelas dengan `setattr`
4.      KEMBALIKAN `fun`
5.
```

6. KONSTRUKTOR `__init__(tokenizer, feature_extractor, output_layer, num_layers=10, units=256, max_length=50, num_heads=1, dropout_rate=0.1)`:
7. PANGGIL konstruktor superclass
8. SIMPAN `feature_extractor` ke `self.feature_extractor`
9. SIMPAN `tokenizer` ke `self.tokenizer`
10. INISIALISASI `word_to_index` dengan `StringLookup` dari `tokenizer.get_vocabulary()`
11. INISIALISASI `index_to_word` dengan `StringLookup` dari `tokenizer.get_vocabulary()`, `invert=True`
12. INISIALISASI `seq_embedding` dengan `SeqEmbedding` menggunakan `vocab_size`, `depth`, `max_length`
13. INISIALISASI `decoder_layers` sebagai list dari `DecoderLayer` untuk `num_layers` dengan `units`, `num_heads`, `dropout_rate`
14. SIMPAN `output_layer` ke `self.output_layer`
15.
16. METODE `call(self, inputs)`:
17. PECAH `inputs` menjadi `image` dan `txt`
18. JIKA `image.shape[-1] == 3`, TERAPKAN `feature_extractor` pada `image`
19. RATAKAN `image` dengan `einops.rearrange(image, 'b h w c -> b (h w) c')`
20. JIKA `txt.dtype == tf.string`, TERAPKAN `tokenizer` pada `txt`
21. TERAPKAN `seq_embedding` pada `txt`
22. UNTUK setiap `dec_layer` dalam `decoder_layers`:
23. PERBARUI `txt` dengan `dec_layer(inputs=(image, txt))`
24. TERAPKAN `output_layer` pada `txt`
25. KEMBALIKAN `txt`
26.
27. METODE `simple_gen(self, image, temperature=1)`:

```
28.  INISIALISASI `initial` dengan
`word_to_index([['[START]']])`
29.  HITUNG `img_features` dengan
`feature_extractor(image[tf.newaxis, ...])`
30.  SET `tokens` ke `initial`
31.  UNTUK `n` dalam range 50:
32.    HITUNG `preds` dengan `self((img_features,
tokens)).numpy()`
33.    AMBIL `preds` pada indeks terakhir
34.    JIKA `temperature == 0`, HITUNG `next` dengan
`tf.argmax(preds, axis=-1)[:, tf.newaxis]`
35.    LAIN, HITUNG `next` dengan
`tf.random.categorical(preds/temperature, num_samples=1)`
36.    PERBARUI `tokens` dengan `tf.concat([tokens, next],
axis=1)`
37.    JIKA `next[0](citation_0) == word_to_index('[END]')`,
BREAK
38.  KONVERSI `tokens[0, 1:-1]` menjadi `words` dengan
`index_to_word`
39.  GABUNG `words` menjadi `result` dengan
`tf.strings.reduce_join`
40.  KEMBALIKAN `result.numpy().decode()`
```

5.3 Pelatihan Model

Bagian kode ini mendefinisikan dua fungsi penting untuk proses pelatihan model captioning, yaitu masked_loss dan masked_acc, yang dirancang untuk hanya menghitung loss dan akurasi pada elemen-elemen yang valid, dengan mengabaikan token padding atau label nol. Fungsi masked_loss menghitung sparse softmax cross-entropy loss dan menerapkan mask untuk mengecualikan label yang tidak valid serta nilai loss yang ekstrem, kemudian menghitung rata-rata loss hanya pada elemen yang ter-mask. Sementara itu, fungsi masked_acc menghitung akurasi prediksi dengan membandingkan hasil argmax dari prediksi

dengan label yang dikonversi ke tipe yang sesuai, dan menggunakan mask untuk menghitung rata-rata akurasi hanya pada elemen yang relevan. Pendekatan ini memastikan bahwa pelatihan model lebih akurat dan stabil dengan hanya mempertimbangkan data yang bermakna.

Kode Semu 7 Penggunaan *Loss* dan Akurasi untuk Pelatihan model

```
1. DEFINISIKAN fungsi `masked_loss(labels, preds)`
2. HITUNG `loss` dengan `tf.nn.sparse_softmax_cross_entropy_with_logits(labels, preds)`
3. BUAT `mask` dengan kondisi `(labels != 0) & (loss < 1e8)`
4. KONVERSI `mask` ke tipe data `loss` dengan `tf.cast(mask, loss.dtype)`
5. KALIKAN `loss` dengan `mask`
6. HITUNG `loss` sebagai `tf.reduce_sum(loss) / tf.reduce_sum(mask)`
7. KEMBALIKAN `loss`
8. DEFINISIKAN fungsi `masked_acc(labels, preds)`
9. BUAT `mask` dengan `tf.cast(labels != 0, tf.float32)`
10. HITUNG `preds` sebagai `tf.argmax(preds, axis=-1)`
11. KONVERSI `labels` ke `tf.int64`
12. HITUNG `match` sebagai `tf.cast(preds == labels, mask.dtype)`
13. HITUNG `acc` sebagai `tf.reduce_sum(match * mask) / tf.reduce_sum(mask)`
14. KEMBALIKAN `acc`
```

Pelatihan model dilakukan dengan pendekatan grid search guna mengevaluasi berbagai skenario percobaan yang melibatkan penggunaan padding pada gambar, pemanfaatan model pra-latih, dan variasi parameter pada bagian decoder. Grid search digunakan untuk menguji kombinasi hyperparameter seperti jumlah unit, dropout rate, jumlah layer dan head pada decoder, serta learning rate. Proses dimulai dengan impor modul yang diperlukan dan

inisialisasi parameter pelatihan. Kemudian, setiap kombinasi parameter dijalankan secara iteratif menggunakan itertools.product dan tqdm untuk memantau progres. Dua model dibentuk dalam setiap iterasi: satu tanpa padding dan satu dengan padding. Keduanya dikompilasi dan dilatih menggunakan dataset yang sesuai. Setelah pelatihan, model dievaluasi menggunakan skor BLEU dan ROUGE baik pada data pelatihan maupun pengujian. Hasil dari tiap eksperimen dicatat dalam bentuk DataFrame dan disimpan sebagai file CSV untuk dianalisis lebih lanjut.

Kode Semu 8 Pelatihan model dengan menggunakan *Grid Search*

```
1. IMPORT `itertools`
2. IMPORT `pandas` sebagai `pd`
3. INISIALISASI `results` sebagai list kosong
4. SET `EPOCHS` ke 100
5. SET `STEPS_PER_EPOCH` ke 100
6. SET `VALIDATION_STEPS` ke 20
7. INISIALISASI `param_grid` sebagai dictionary dengan:
8.     'units': [128, 256]
9.     'dropout_rate': [0.3, 0.4]
10.    'num_layers': [3, 4, 5, 6]
11.    'num_heads': [3, 4, 5, 6]
12.    'learning_rate': [1e-4]
13. SET `results_dir` ke 'record_inception'
14. GENERATE semua kombinasi hyperparameter dengan
`itertools.product` dari `param_grid`
15. SIMPAN hasilnya ke `combinations`

16. UNTUK setiap `idx`, `(units, dropout_rate, num_layers,
num_heads, learning_rate)` dalam `tqdm(combinations,
desc="Grid Search")`:
17.    INISIALISASI `results` sebagai list kosong
18.    SIMPAN `output_layer` ke `temp_output_layer`
19.    SIMPAN `output_layer_padding` ke
`temp_output_layer_padding`
```

20. CETAK informasi pelatihan model dengan parameter saat ini
21.
22. DEFINISIKAN `model` sebagai `Captioner` dengan parameter saat ini
23. DEFINISIKAN `model_padding` sebagai `Captioner` dengan parameter saat ini dan `output_layer_padding`
24.
25. INISIALISASI `optimizer` dengan `tf.keras.optimizers.Adam(learning_rate=1e-4)`
26. KOMPILE `model` dengan `optimizer`, `masked_loss`, dan `masked_acc`
27. KOMPILE `model_padding` dengan `optimizer`, `masked_loss`, dan `masked_acc`
28.
29. LATIH `model` dengan `fit` pada `train_ds_1`, dengan `steps_per_epoch`, `validation_data`, `validation_steps`, `epochs`, `callbacks`, `verbose=0`
30. LATIH `model_padding` dengan `fit` pada `train_ds_1_padding`, dengan `steps_per_epoch`, `validation_data`, `validation_steps`, `epochs`, `callbacks`, `verbose=0`
31.
32. CETAK "Evaluating BLEU and ROUGE scores..."
33. HITUNG `bleu_train`, `bleu_test`, `rouge_train`, `rouge_test` dengan `bleu_score_test` dan `rouge_score_test` untuk `model`
34. HITUNG `bleu_train_padding`, `bleu_test_padding`, `rouge_train_padding`, `rouge_test_padding` dengan `bleu_score_test` dan `rouge_score_test` untuk `model_padding`
35.
36. CETAK skor BLEU dan ROUGE untuk data pelatihan dan pengujian
37.

```
38.  TAMBAHKAN hasil ke `results` untuk `no_padding`
39.  TAMBAHKAN hasil ke `results` untuk `padding`
40.
41.  BUAT `df` sebagai `pd.DataFrame` dari `results`
42.  SIMPAN `df` ke `csv` dengan nama `csv_filename` di
`results_dir`
```

5.4 Evaluasi Hasil dari *Caption*

Evaluasi dilakukan dengan menggunakan BLEU score 1 dan 2 serta ROUGE-2. Nilai BLEU dihitung menggunakan fungsi *corpus_bleu* dari pustaka NLTK. Baik nilai BLEU maupun ROUGE diperoleh dengan membandingkan hasil generasi model dengan caption label yang tersedia dalam dataset. Untuk mewakili setiap skenario evaluasi dalam proses Grid Search, dilakukan perhitungan rata-rata nilai dari seluruh data dalam dataset. Pendekatan ini memberikan gambaran umum mengenai performa model secara keseluruhan.

Kode Semu 9 Evaluasi model

```
1.  IMPORT `SmoothingFunction`, `corpus_bleu`,
`sentence_bleu` dari `nltk.translate.bleu_score`
2.  DEFINISIKAN fungsi `avg(lst)`
3.  KEMBALIKAN `sum(lst) / len(lst)`
4.  DEFINISIKAN fungsi `bleu_score_test(dataset, model,
load_img)`
5.  INISIALISASI `recap` sebagai list kosong
6.  UNTUK setiap `key` dalam `dataset.keys()`:
7.    INISIALISASI `pred` sebagai list berisi `dataset[key]`
8.    MUAT `image` dengan `load_img(key)`
9.    INISIALISASI `score_bleu` sebagai list kosong
10.   UNTUK setiap `t` dalam `(0.0, 0.5, 1.0)`:
11.     HASILKAN `result` dengan `model.simple_gen(image,
temperature=t)`
12.     KONVERSI `result` menjadi list kata
```

```
13.      INISIALISASI `cc` dengan `SmoothingFunction()`
14.      COBA:
15.          TAMBAHKAN skor BLEU ke `score_bleu` dengan
`corpus_bleu`
16.      KECUALI:
17.        CETAK pesan kesalahan
18.        TAMBAHKAN 0 ke `score_bleu`
19.    TAMBAHKAN rata-rata `score_bleu` ke `recap`
20.   KEMBALIKAN rata-rata `recap`
21. DEFINISIKAN fungsi `rouge_score_test(dataset, model,
load_img, n=2)`
22.   INISIALISASI `recap` sebagai list kosong
23.   UNTUK setiap `key` dalam `dataset.keys()[:10]`:
24.    INISIALISASI `pred` sebagai `dataset[key]`
25.    MUAT `image` dengan `load_img(key)`
26.    INISIALISASI `score_rouge` sebagai list kosong
27.    UNTUK setiap `t` dalam `(0.0, 0.5, 1.0)`:
28.      HASILKAN `result` dengan `model.simple_gen(image,
temperature=t)`
29.      COBA:
30.        HITUNG `rouge_score` dengan `rouge_n(pred, result,
n)`
31.      KECUALI:
32.        CETAK pesan kesalahan
33.        SET `rouge_score` ke 0
34.      TAMBAHKAN `rouge_score` ke `score_rouge`
35.    TAMBAHKAN rata-rata `score_rouge` ke `recap`
36.   KEMBALIKAN rata-rata `recap`
```

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini membahas secara mendalam tahap uji coba terhadap model *Image Captioning* yang telah diimplementasikan. Melalui pengujian ini, diharapkan dapat diperoleh wawasan penting yang akan mendukung pengembangan lebih lanjut dari model *Image Captioning*, baik dari segi akurasi maupun efisiensi. Proses uji coba akan meliputi evaluasi kinerja model menggunakan metrik BLEU-1-2 dan Rouge-2 serta analisis mendetail terhadap hasil yang diperoleh. Dengan demikian, bab ini bertujuan untuk memberikan gambaran komprehensif mengenai efektivitas model *Image Captioning* serta langkah-langkah strategis yang perlu diambil untuk meningkatkan kemampuannya dalam menghasilkan deskripsi gambar yang akurat dan relevan.

6.1. Skenario Pengujian

Pengujian dilakukan dengan memantau nilai BLEU Score dan ROUGE Score pada data uji dari dataset Flickr8k guna memperoleh gambaran yang lebih komprehensif mengenai pengaruh berbagai kombinasi skenario terhadap performa model. Eksperimen dilakukan menggunakan pendekatan Grid Search dengan kombinasi parameter sebagai berikut: units (jumlah unit pada *feedforward layer*) yaitu [128, 256], dropout_rate yaitu [0.3, 0.4], num_layers (jumlah pengulangan layer *decoding*) yaitu [3, 4, 5, 6], num_heads (jumlah head pada *multi-head attention*) yaitu [3, 4, 5, 6], dan learning_rate sebesar [1e-4]. Pendekatan ini bertujuan untuk memahami lebih dalam bagaimana masing-masing konfigurasi parameter memengaruhi kinerja model yang telah diimplementasikan. Skenario yang akan diamati meliputi:

a. Pengaruh Arsitektur *Feature Extraction* terhadap Kinerja Model

Pada skenario ini, kami menguji pengaruh berbagai arsitektur fitur ekstraksi terhadap kinerja model yang diimplementasikan. Kami menggunakan fitur ekstraktor berupa model pra-latih dari dataset ImageNet yang berbasis Convolutional Neural Network (CNN). Model-model yang digunakan dalam pengujian ini meliputi Inception, MobileNet, *Visual Geometry Group* (VGG), dan ResNet. Dengan menguji berbagai arsitektur ini, kami bertujuan untuk memahami bagaimana perbedaan dalam fitur ekstraksi dapat mempengaruhi hasil akhir dari model yang dikembangkan.

b. Pengaruh Penggunaan *Padding* terhadap *Preprocessing* Data Gambar

Pada bagian ini, kami mengeksplorasi pengaruh penggunaan *padding* dalam *preprocessing* data gambar. Ketika melakukan transformasi gambar ke ukuran yang sesuai dengan model, sering kali dimensi gambar asli berbeda dengan dimensi yang dibutuhkan oleh model. Untuk mengatasi masalah ini, kami melakukan percobaan dengan menambahkan *padding* atau dengan merentangkan gambar agar sesuai dengan dimensi model. Melalui pendekatan ini, kami bertujuan untuk memahami bagaimana metode penyesuaian ukuran gambar dapat mempengaruhi kinerja dan akurasi model.

c. Pengaruh jumlah *layer* pada *Decoder* model

Pada bagian ini, kami meneliti pengaruh jumlah *layer* pada model *decoder* dalam konteks *image captioning*. Jumlah *layer* pada *decoder* berperan penting dalam tingkat kompleksitas model dan kemampuannya untuk menghasilkan *caption* yang akurat. Kami melakukan eksperimen dengan variasi penggunaan 3 hingga 6 *layer decoder* untuk menganalisis dampaknya terhadap performa model. Penambahan jumlah *layer* dapat meningkatkan kapasitas model dalam mempelajari pola yang lebih kompleks, namun juga

berpotensi meningkatkan risiko *overfitting* dan peningkatan biaya komputasi.

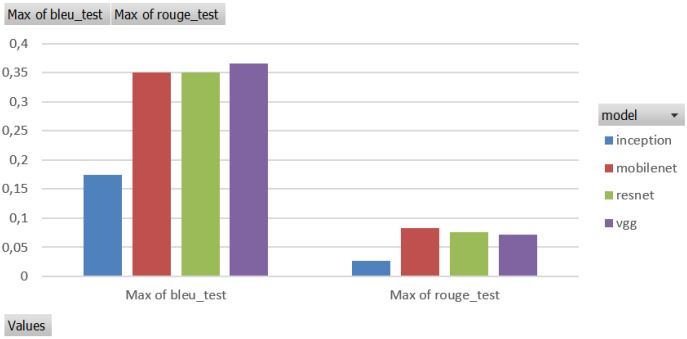
Setelah menemukan skenario terbaik, pelatihan model *image captioning* selanjutnya akan dilakukan pada dataset COCO. Langkah ini bertujuan untuk menilai apakah model yang telah dioptimasi untuk dataset Flickr8k juga mampu memberikan performa yang memuaskan pada dataset yang lebih besar dan kompleks seperti COCO. Proses ini tidak hanya menguji skalabilitas dan adaptabilitas model, tetapi juga memberikan wawasan lebih dalam tentang keunikan dan tantangan yang mungkin dihadapi saat model diterapkan pada berbagai jenis data.

6.2. Pembahasan Hasil Eksperimen

Pada bagian ini, dibahas hasil eksperimen dari setiap skenario percobaan yang telah dilaksanakan. Setiap skenario diuji melalui proses pelatihan (training) dan pengujian (testing) pada data Flickr8k (testing menggunakan 10 data gambar). Hasil percobaan kemudian dianalisis untuk mengevaluasi pengaruh variasi skenario terhadap skor ROUGE dan BLEU. Skenario dengan skor ROUGE dan BLEU tertinggi pada data testing diidentifikasi sebagai skenario optimal. Selanjutnya, skenario optimal tersebut digunakan untuk pembangkitan caption pada dataset COCO.

6.2.1. Hasil Pengaruh Arsitektur Ekstraksi Fitur

Skenario ini data hasil *Grid Search* akan dilakukan analisis untuk penggunaan fitur ekstraksi. Untuk mewakili perbandingan setiap fitur ekstraksi digunakan fungsi agregasi berupa nilai maksimal dan nilai rata – rata. Hal tersebut dilakukan untuk mengetahui potensi dan stabilitas dari model.



Gambar 6.1 Nilai Maksimal Pada Hasil Percobaan Ekstraksi Fitur

Gambar 6.1 diperoleh berdasarkan konfigurasi dibawah, nilai bleu dan rouge yang digunakan adalah bleu_test dan rouge_test (tertanda tebal).

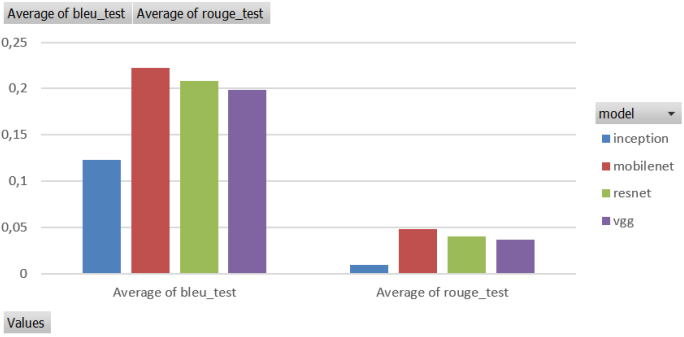
Tabel 6.1 Nilai Maksimal pada Hasil Percobaan Ekstraksi Fitur dan konfigurasinya terurut.

model	jenis	units	dropout	Layer	head	LR	bleu_train	bleu_test	rouge_train	rouge_test
inception	stretching	256	0,3	5	6	0,0001	0,089	**0,175**	0,0085	0,011
mobilenet	stretching	256	0,4	5	5	0,0001	0,311	**0,35**	0,0974	0,075
resnet	stretching	128	0,4	6	3	0,0001	0,23	**0,351**	0,0789	0,067
vgg	stretching	256	0,4	4	3	0,0001	0,379	**0,366**	0,0915	**0,07**
inception	stretching	128	0,3	3	6	0,0001	0,111	0,1128	0,0272	**0,03**
mobilenet	stretching	128	0,4	6	4	0,0001	0,406	0,3135	0,1052	**0,08**
resnet	padding	256	0,3	3	3	0,0001	0,366	0,2827	0,0717	**0,08**
vgg	stretching	256	0,4	4	3	0,0001	0,379	**0,366**	0,0915	**0,07**

Dari Gambar 6.1, terlihat bahwa VGG mencatatkan skor tertinggi untuk BLEU sebesar 0,3656, diikuti oleh ResNet dengan 0,3507, dan MobileNet dengan 0,3501. Meskipun perbedaan antara ResNet dan MobileNet cukup kecil, VGG menunjukkan sedikit keunggulan dalam menghasilkan deskripsi yang paling mendekati *ground truth*. Sebaliknya, Inception memiliki performa terendah

dengan nilai BLEU maksimum sebesar 0,1749, menunjukkan bahwa fitur yang diekstraksi olehnya mungkin kurang relevan untuk tugas *image captioning* ini.

Pada metrik ROUGE, MobileNet justru unggul dengan nilai maksimum tertinggi sebesar 0,0828, sedikit lebih tinggi dari ResNet (0,0755) dan VGG (0,0724). Inception kembali menunjukkan performa terendah dengan nilai maksimum 0,0275. Kinerja MobileNet yang lebih baik pada ROUGE mencerminkan kemampuannya untuk menghasilkan deskripsi dengan relevansi kata yang lebih baik terhadap *ground truth*, meskipun struktur sintaksis atau konteksnya mungkin tidak sekuat yang diukur melalui BLEU.



Gambar 6.2 Nilai Rerata pada Hasil Percobaan Ekstraksi Fitur

Pada hasil nilai rata-rata yang tersedia pada Gambar 6.2 MobileNet menunjukkan keunggulan yang lebih konsisten. MobileNet memiliki rata-rata BLEU tertinggi sebesar 0,2226, diikuti oleh ResNet dengan 0,2079, dan VGG dengan 0,1988. Inception tetap berada di posisi terakhir dengan rata-rata BLEU 0,1234. Pola yang sama juga terlihat pada metrik ROUGE, di mana MobileNet mencatat rata-rata tertinggi sebesar 0,0483, diikuti oleh ResNet (0,0406) dan VGG (0,0370), dengan Inception kembali menjadi yang terendah pada nilai rata-rata ROUGE sebesar 0,0100.

Performa MobileNet yang konsisten pada kedua metrik ini menunjukkan kestabilannya di berbagai skenario eksperimen.

Secara keseluruhan, bahwa VGG dan ResNet memiliki potensi performa tinggi dalam skenario tertentu, terutama ketika performa maksimum menjadi fokus utama. Namun, MobileNet menawarkan keunggulan dalam stabilitas dan efisiensi, menjadikannya pilihan yang lebih andal untuk eksperimen yang membutuhkan konsistensi hasil di seluruh percobaan. Di sisi lain, Inception tampaknya kurang cocok untuk tugas *image captioning* tanpa optimasi lebih lanjut, mengingat performanya yang jauh di bawah model lainnya pada semua metrik.

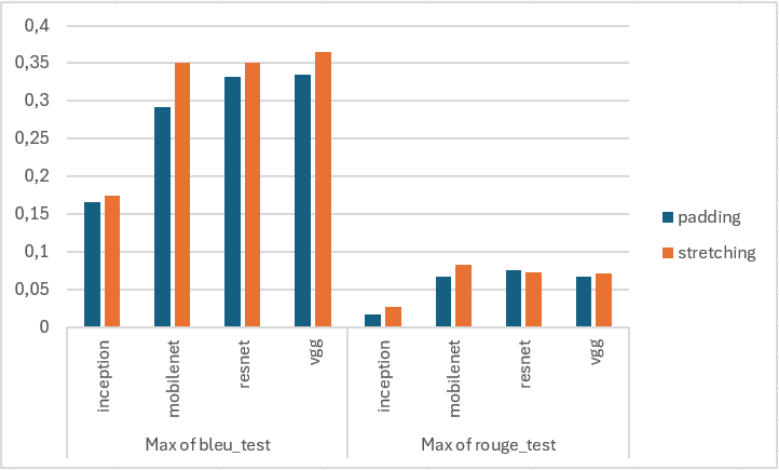
6.2.2. Hasil Pengaruh Penggunaan *Padding*

Skenario ini muncul karena adanya ukuran khusus pada setiap model fitur ekstraktor. Di lain sisi, gambar yang akan digunakan baik untuk pelatihan dan validasi memiliki ukuran dan dimensi yang dinamis.

Gambar 6.3 Bentuk Transformasi Gambar dengan stretching (kiri) dan dengan Padding(kanan)

Maka dari itu skenario ini muncul untuk melihat pengaruh pra-pemrosesan gambar yang berbeda pada saat transformasi data memengaruhi hasil dari tugas *image captioning*. Hasil pra pemrosesan itu dapat dilihat pada Gambar 6.3, dimana dalam kasus tersebut dibutuhkan dimensi gambar berukuran 300×300 piksel, namun gambar yang tersedia berupa gambar *portrait* vertikal.

Terlihat pada sebelah kiri gambar ditarik kesamping untuk memenuhi perbedaan dimensi sedangkan pada bagian kanan gambar, gambar diberi *padding* berwarna hitam untuk menyesuaikan.



Gambar 6.4 Nilai Maksimal pada Hasil Percobaan Ekstraksi Fitur

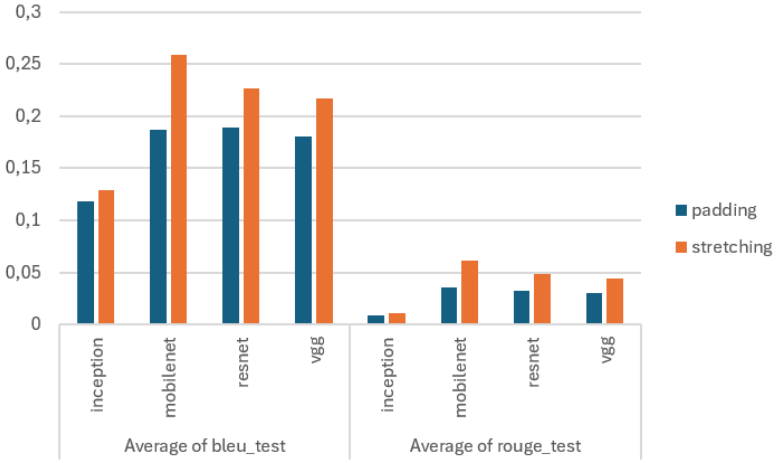
Gambar 6.4 diperoleh diperoleh berdasarkan konfigurasi dibawah, nilai bleu dan rouge yang digunakan adalah bleu_test dan rouge_test (tertanda tebal).

Tabel 6.2 Nilai Maksimal pada Hasil Percobaan Jenis *Preprocessing* dan Konfigurasinya Terurut.

model	jenis	units	dropout	Layer	Head	LR	bleu train	bleu test	rouge train	rouge test
inception	padding	256	0,3	4	6	0,0001	0,125582	**0,166**	0,004786	0,010098
inception	stretching	256	0,3	5	6	0,0001	0,089434	**0,17493**	0,008507	0,011266
mobilenet	padding	256	0,3	3	4	0,0001	0,338162	**0,2914**	0,079724	0,06311
mobilenet	stretching	256	0,4	5	5	0,0001	0,311394	**0,35013**	0,097432	0,075164
resnet	padding	128	0,3	3	4	0,0001	0,300422	**0,33184**	0,072508	0,058026
resnet	stretching	128	0,4	6	3	0,0001	0,229577	**0,35073**	0,078913	0,066773
vgg	padding	128	0,3	3	3	0,0001	0,293547	**0,33447**	0,062449	0,054724
vgg	stretching	256	0,4	4	3	0,0001	0,379422	**0,36559**	0,091456	**0,07236**

inception	padding	128	0,3	3	3	0,0001	0,127332	0,122469	0,011719	**0,01676**
inception	stretching	128	0,3	3	6	0,0001	0,110799	0,112816	0,027247	**0,02747**
mobilenet	padding	256	0,3	5	4	0,0001	0,291471	0,251772	0,086031	**0,06737**
mobilenet	stretching	128	0,4	6	4	0,0001	0,40605	0,313549	0,10521	**0,08277**
resnet	padding	256	0,3	3	3	0,0001	0,366073	0,282704	0,071697	**0,0755**
resnet	stretching	128	0,3	4	5	0,0001	0,324401	0,2248	0,102511	**0,07279**
vgg	padding	128	0,4	3	4	0,0001	0,329408	0,257096	0,088047	**0,06694**
vgg	stretching	256	0,4	4	3	0,0001	0,379422	**0,36559**	0,091456	**0,07236**

Pada hasil nilai Maksimal yang tersedia pada Gambar 6.4, terlihat bahwa tanpa *padding*, semua *feature extractor* mencapai performa terbaiknya. VGG mencatatkan nilai BLEU tertinggi sebesar 0,3656 dibandingkan dengan skenario *padding* yang hanya mencapai 0,3345. Hal serupa juga terjadi pada MobileNet (0,3501 tanpa *padding* vs. 0,2914 dengan *padding*) dan ResNet (0,3507 tanpa *padding* vs. 0,3318 dengan *padding*). Inception juga menunjukkan performa maksimum BLEU yang lebih baik tanpa *padding*, meskipun selisihnya kecil (0,1749 tanpa *padding* vs. 0,1660 dengan *padding*). Pada metrik ROUGE, tren serupa terjadi, di mana nilai maksimum tanpa *padding* lebih tinggi untuk MobileNet (0,0828) dan VGG (0,0724). Namun, ResNet adalah satu-satunya *feature extractor* yang memiliki nilai maksimum ROUGE lebih tinggi pada skenario *padding* (0,0755) dibandingkan tanpa *padding* (0,0728), menunjukkan potensi unik dalam menangani data yang diproses dengan *padding*.



Gambar 6.5 Nilai Rerata pada Hasil Percobaan Ekstraksi Fitur

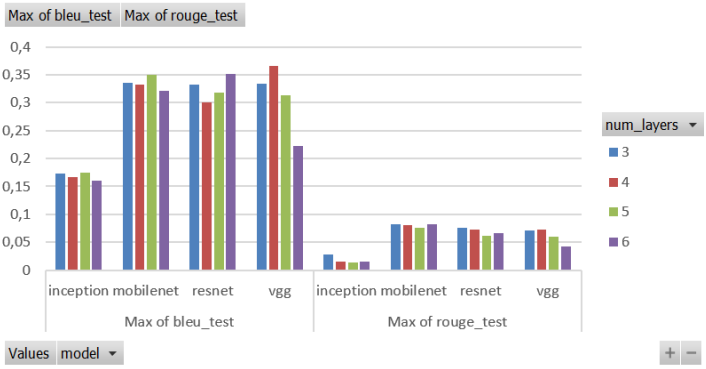
Pada hasil nilai rerata pada Gambar 6.5, performa tanpa *padding* juga unggul secara konsisten. Pada metrik BLEU, MobileNet memiliki rata-rata tertinggi tanpa *padding* sebesar 0,2583, diikuti oleh ResNet (0,2270) dan VGG (0,2169). Sebaliknya, dengan *padding*, rata-rata BLEU MobileNet turun menjadi 0,1870, menunjukkan bahwa *padding* dapat memengaruhi efisiensi dan akurasi dalam ekstraksi fitur. Tren serupa juga terjadi pada metrik ROUGE, di mana MobileNet mencatat rata-rata tertinggi tanpa *padding* sebesar 0,0612, sementara ResNet dan VGG memiliki performa yang serupa dengan penurunan akibat padding.

Secara keseluruhan, hasil percobaan menunjukkan bahwa skenario tanpa *padding* memberikan hasil yang lebih baik untuk sebagian besar *feature extractor*, baik dari nilai maksimum maupun rata-rata. Hal ini menunjukkan bahwa *padding* dapat mengurangi efisiensi representasi fitur, terutama pada arsitektur seperti MobileNet dan VGG. Namun, ResNet menunjukkan

fleksibilitas yang lebih baik dalam menangani skenario dengan *padding*, terutama pada metrik ROUGE, yang dapat menjadikannya pilihan menarik untuk dataset atau aplikasi yang memerlukan *padding*. Pendekatan strategis yang menghilangkan atau meminimalkan penggunaan *padding* dapat meningkatkan performa secara signifikan, khususnya untuk model seperti MobileNet dan VGG yang lebih sensitif terhadap skenario ini.

6.2.3. Hasil Pengaruh Jumlah *Layer* pada *Decoder* model

Pada bagian ini, untuk mewakili setiap skenario digunakan fungsi agregasi berupa nilai maksimal. Hal tersebut dilakukan untuk memperoleh potensi dari pembuatan model dalam konteks nilai BLEU dan ROUGE.



Gambar 6.6 Nilai Maksimal pada Hasil Percobaan Penggunaan Layer

Gambar 6.6 diperoleh berdasarkan konfigurasi Tabel 6.3, nilai bleu dan rouge yang digunakan adalah bleu_test dan rouge_test (tertanda tebal).

Tabel 6.3 Nilai Maksimal pada Hasil Percobaan Penggunaan Layer dan konfigurasi secara Terurut

Layer	model	jenis	units	dropout	head	LR	bleu_train	bleu_test	rouge_train	rouge_test
3	inception	stretching	128	0,4	3	1E-04	0,127888	**0,17247**	0,030257	0,021889
4	inception	stretching	256	0,3	3	1E-04	0,073715	**0,16181**	0,0061738	0,012464
5	inception	stretching	256	0,3	6	1E-04	0,089434	**0,17493**	0,0085067	0,011266
6	inception	padding	128	0,3	5	1E-04	0,102722	**0,16005**	0,0072432	0,011188
3	mobilenet	stretching	128	0,3	5	1E-04	0,431968	**0,3352**	0,0992941	0,068041
4	mobilenet	stretching	256	0,4	6	1E-04	0,271729	**0,33162**	0,0971673	**0,08119**
5	mobilenet	stretching	256	0,4	5	1E-04	0,311394	**0,35013**	0,0974316	**0,07516**
6	mobilenet	stretching	256	0,3	6	1E-04	0,357804	**0,32101**	0,1177315	0,071099
3	resnet	padding	128	0,3	4	1E-04	0,300422	**0,33184**	0,0725078	0,058026
4	resnet	stretching	256	0,3	6	1E-04	0,352804	**0,30108**	0,0949106	0,061641
5	resnet	stretching	256	0,4	3	1E-04	0,308384	**0,31088**	0,1061832	0,037759
6	resnet	stretching	128	0,4	3	1E-04	0,229577	**0,35073**	0,0789128	0,066773
3	vgg	padding	128	0,3	3	1E-04	0,293547	0,334473	0,062449	0,054724
4	vgg	stretching	256	0,4	3	1E-04	0,379422	**0,36559**	0,0914556	**0,07236**
5	vgg	stretching	128	0,4	4	1E-04	0,329682	**0,31287**	0,0713703	0,049432
6	vgg	stretching	256	0,4	4	1E-04	0,116254	**0,22225**	0,0186146	0,015272
3	inception	stretching	128	0,3	6	1E-04	0,110799	0,112816	0,0272466	**0,02747**
4	inception	padding	128	0,3	3	1E-04	0,093572	0,094954	0,0064619	**0,01521**
5	inception	stretching	128	0,4	3	1E-04	0,084674	0,143686	0,0044206	**0,01358**
6	inception	stretching	256	0,4	5	1E-04	0,084938	0,152619	0,0085849	**0,01614**
3	mobilenet	stretching	128	0,3	4	1E-04	0,345993	0,296793	0,1104534	**0,08168**

4	mobilenet	stretching	256	0,4	6	1E-04	0,271729	**0,33162**	0,0971673	**0,08119**
5	mobilenet	stretching	256	0,4	5	1E-04	0,311394	**0,35013**	0,0974316	**0,07516**
6	mobilenet	stretching	128	0,4	4	1E-04	0,40605	0,313549	0,1052097	**0,08277**
3	resnet	padding	256	0,3	3	1E-04	0,366073	0,282704	0,0716967	**0,0755**
4	resnet	stretching	128	0,3	5	1E-04	0,324401	0,2248	0,1025112	**0,07279**
5	resnet	stretching	256	0,3	6	1E-04	0,303529	0,254583	0,0974327	**0,06135**
6	resnet	stretching	128	0,4	3	1E-04	0,229577	0,350729	0,0789128	**0,06677**
3	vgg	stretching	256	0,3	5	1E-04	0,35553	**0,31193**	0,0905873	**0,07047**
4	vgg	stretching	256	0,4	3	1E-04	0,379422	**0,36559**	0,0914556	**0,07236**
5	vgg	stretching	256	0,4	4	1E-04	0,300706	0,166603	0,0832107	**0,05923**
6	vgg	padding	128	0,3	4	1E-04	0,164872	0,165915	0,0143	**0,04254**

Dapat dilihat pada Gambar 6.6, terlihat bahwa jumlah *layer* pada model arsitektur berpengaruh terhadap performa metrik BLEU dan ROUGE, meskipun pola pengaruhnya bervariasi tergantung pada jenis model yang digunakan. MobileNet menunjukkan konsistensi yang relatif baik pada kedua metrik, dengan skor BLEU tertinggi (0,350) dicapai pada 5 *layer*, sementara skor ROUGE tertinggi (0,083) muncul pada 6 *layer*. Hal ini mengindikasikan bahwa penambahan *layer* hingga batas tertentu dapat meningkatkan kemampuan model dalam menghasilkan teks yang presisi (BLEU) sekaligus menjaga relevansi konten (ROUGE). Namun, penurunan skor BLEU MobileNet pada 6 *layer* (0,321) menunjukkan risiko *overfitting* atau kompleksitas berlebih ketika jumlah *layer* melebihi kapasitas optimal.

Di sisi lain, VGG memberikan pola yang unik skor BLEU tertinggi (0,366) dicapai pada 4 *layer*, tetapi mengalami penurunan drastis (0,222) saat *layer* ditambah menjadi 6. Penurunan ini juga diikuti oleh skor ROUGE yang rendah (0,043), mengisyaratkan bahwa arsitektur VGG mungkin kurang efektif dalam menangani kedalaman *layer* yang tinggi untuk tugas *image captioning*. Fenomena ini dapat disebabkan oleh karakteristik VGG yang memiliki parameter sangat banyak, sehingga penambahan *layer* justru memperburuk generalisasi model. Sebaliknya, ResNet menunjukkan peningkatan skor BLEU (0,351) pada 6 *layer*, yang mungkin dimungkinkan berkat mekanisme *skip connection* yang mengurangi masalah *vanishing gradient*, memungkinkan pelatihan lebih stabil pada jaringan yang dalam.

Perbandingan antara model juga mengungkapkan bahwa Inception memiliki skor ROUGE yang secara konsisten lebih rendah dibandingkan model lain (maksimal 0,027),dan skor BLEU-nya (0,175). Disparitas ini menunjukkan bahwa meskipun Inception mampu menghasilkan kata-kata yang sesuai dengan referensi (precision), model ini kurang efektif dalam menangkap keseluruhan konteks atau informasi penting (recall), yang diukur oleh ROUGE. Sementara itu, MobileNet unggul dalam konsistensi kedua metrik, menegaskan bahwa efisiensi arsitektur berbasis *depthwise separable convolution* pada mobilenet tidak hanya mengurangi kompleksitas komputasi tetapi juga menjaga kualitas output.

Hasil ini menekankan bahwa optimalisasi jumlah *layer* harus disesuaikan dengan karakteristik model. Sebagai contoh, VGG mencapai performa terbaik dengan *layer* lebih sedikit, sementara ResNet justru memerlukan kedalaman lebih tinggi. Selain itu, rendahnya skor ROUGE secara umum (di bawah 0,1 pada sebagian

besar kasus) dibandingkan BLEU mengindikasikan bahwa model-model ini cenderung menghasilkan *caption* yang akurat pada level frase (sesuai BLEU), tetapi kurang komprehensif dalam merepresentasikan keseluruhan makna gambar. Hal ini menjadi tantangan untuk percobaan selanjutnya, seperti integrasi mekanisme *attention* atau peningkatan kualitas data pelatihan untuk meningkatkan aspek *recall*.

6.3. Hasil *Captioning* Pada Gambar MSCOCO

Pada bagian ini, dilakukan percobaan pelatihan model *image captioning* menggunakan dataset MSCOCO dengan konfigurasi optimal dari skenario terbaik pada dataset Flickr8k (Terbaik dari skor Rouge maupun BLEU), yakni arsitektur VGG sebagai *encoder*, tanpa padding (stretching), units sejumlah 256, dropout 0,4, *learning rate* sebesar 0,0001, jumlah *layer* sebanyak 4, dan jumlah *head* sebanyak 3. Model kemudian diuji pada subset dataset 220k-GPT4Vision dengan dua nilai *temperature* berbeda (0.05 dan 0.5) untuk mengevaluasi pengaruh variasi generasi terhadap kualitas *caption*. Analisis difokuskan pada dua aspek: kasus dengan skor BLEU tertinggi dan kasus dengan skor ROUGE tertinggi.

Pada skor BLEU terbaik diperoleh data sesuai yang disajikan pada Tabel 6.1 sebagai berikut:

Tabel 6.4 skor BLEU *image captioning* terbaik

url	caption	caption_t_0.05	caption_t_0.5	rouge_t_0.05	rouge_t_0.5	bleu_t_0.05	bleu_t_0.5
000000004495.jpg	a room with a television set on the right side	a room with a bed a wooden headboard a wooden ...	a room with a darkcolored sofa on the left sid...	0.105263	0.448276	0.13041	0.515356

Data diatas diperoleh dari gambar:

Gambar 6.7 Sampel Gambar dengan Nama 000000004495.jpg

Hasil generasi *caption* dari gambar 6.7 dinyatakan pada Tabel 6.2 di bawah:

Tabel 6.5 Hasil perbandingan generasi caption Gambar 6.7

Ground truth caption	**Hasil generasi Caption dengan temperatur 0,05**	**Hasil generasi Caption dengan temperatur 0,5**
a room with a television set on the right side, displaying a dark screen or turned off, and a fabric-upholstered armchair on the left side facing the television.	**a room with a** bed **a wooden headboard a wooden headboard a wooden headboard a wooden headboard** and **a** drawer unit with **a** drawer below the bed **the room has a** cozy and homely ambiance with **a** comfortable and comfortable seating arrangement **the room has a** warm and cozy	**a room with a** darkcolored sofa **on the left side a television set on the right side a television set on the left side** and a sofa **on the right side**

Dari hasil *caption* tersebut, *ground truth caption* mendeskripsikan gambar 000000004495.jpg dengan elemen penting seperti televisi di sebelah kanan dan kursi berpelapis di sebelah kiri. Hasil generasi dengan *temperature* 0,05 menghasilkan *caption* yang repetitif ("wooden headboard" diulang empat kali) dan tidak relevan dengan objek dalam gambar (misal: "bed" dan "drawer unit"), sehingga skor BLEU (0,130) dan ROUGE (0,105) rendah. Sebaliknya, *temperature* 0,5 menghasilkan *caption* yang lebih ringkas dan sesuai konteks ("dark-colored sofa" dan "television set"), meskipun terdapat inkonsistensi spasial ("TV di kiri dan kanan"). Skor BLEU yang lebih tinggi (0,515) menunjukkan bahwa model dengan *temperature* 0,5 lebih baik dalam menangkap *n-gram* yang sesuai dengan referensi, meski ROUGE (0,448) terbatas karena ketidaklengkapan deskripsi. Hal ini mengindikasikan bahwa *temperature* tinggi akan meningkatkan eksplorasi model terhadap distribusi kata, hal tersebut cenderung meningkatkan variasi leksikal namun berisiko mengurangi akurasi spasial. Dilain sisi, *temperature* rendah justru memicu repetisi akibat eksploitasi *token* berprobabilitas tinggi secara berlebihan.

Pada ROUGE skor terbaik diperoleh data sesuai yang disajikan pada Tabel 6.3 sebagai berikut:

Tabel 6.6 Skor rouge *Image Captioning* Terbaik

url	caption	caption_t_0.05	caption_t_0.5	rouge_t_0.05	rouge_t_0.5	bleu_t_0.05	bleu_t_0.5
000000437996.jpg	a baseball game in progress, ...	a baseball game in progress featuring a batter...	a baseball game in progress featuring a batter...	0.528926	0.347107	0.352393	0.280253

Data diatas diperoleh dari gambar:

Gambar 6.8 Sampel Gambar dengan Nama 000000437996.jpg

Hasil generasi *caption* dari gambar 6.8 diatas dinyatakan pada Tabel 6.4 di bawah:

Tabel 6.7 Hasil Perbandingan Generasi Caption Gambar 6.8

Ground truth caption	**Hasil generasi Caption dengan temperatur 0,05**	**Hasil generasi Caption dengan temperatur 0,5**
a baseball game in progress, featuring a batter swinging at a ball, a catcher wearing a dark-colored uniform, and an umpire closely observing the play. The background includes advertisements for FOX SPORTS and other sponsors. The batter is wearing a red uniform, and the catcher is wearing a	**a baseball game in progress** featuring **a batter** swinging at a baseball bat **the catcher is wearing** a mask and a mask and a baseball glove **the batter is wearing a red uniform** and the catcher is wearing a mask and a mask **wearing protective gear** and	**a baseball game in progress** featuring **a batter wearing a red** jersey swinging a baseball bat **the catcher is wearing** a mask and a mask and a catcher **wearing protective gear** the home plate is visible and the catcher is wearing a mask and glove the home plate is also

dark-colored one. The catcher is wearing protective gear, including a mask and mitt. There are several spectators visible in the background, watching the game.		

Dari hasil *caption* tersebut, *ground truth caption* mendeskripsikan gambar 000000437996.jpg sebagai adegan permainan bisbol dengan detail pemain, kostum, dan latar iklan. Generasi dengan *temperature* 0,05 menghasilkan *caption* ("a batter swinging… catcher wearing a mask") yang memperoleh skor ROUGE tertinggi (0,528) karena terjadi kecocokan leksikal tinggi dengan referensi, seperti "batter," "catcher," dan "protective gear." Namun, repetisi frasa ("mask" diulang) menyebabkan skor BLEU (0,352) lebih rendah dibandingkan *temperature* 0,05 pada contoh sebelumnya. Sementara itu, *temperature* 0,5 menghasilkan deskripsi yang lebih beragam ("home plate is visible") tetapi mengandung ketidakrelevanan ("home plate" disebut dua kali) dan inkonsistensi ("catcher wearing a mask and glove" tidak disebutkan dalam referensi), sehingga skor ROUGE (0,347) dan BLEU (0,280) turun signifikan. Hasil ini menunjukkan bahwa *temperature* rendah lebih efektif untuk mempertahankan kata kunci esensial, meski kurang mampu menghasilkan variasi semantik yang kaya.

Secara keseluruhan, model dapat melakukan *captioning* dengan baik, namun pemilihan *temperature* berpengaruh kritis pada *trade-off* antara ketepatan kata (BLEU) dan kelengkapan semantik (ROUGE). *Temperature* rendah (0,05) cenderung menghasilkan *caption* yang konservatif dengan repetisi *token*

berprobabilitas tinggi, sehingga cocok untuk skenario yang memprioritaskan kesesuaian kata kunci. Sebaliknya, *temperature* tinggi (0,5) meningkatkan keberagaman leksikal tetapi berisiko terhadap halusinasi objek atau relasi spasial yang tidak ada, seperti kasus penyebutan "sofa" dan "bed" yang tidak sesuai gambar. Fenomena ini menggarisbawahi kelemahan model Transformer dalam mengelola konteks panjang dan dependensi spasial kompleks, terutama ketika *decoding* dilakukan secara *greedy*.

BAB VII
KESIMPULAN DAN SARAN

7.1 Kesimpulan

1. Pengaruh Pemilihan Arsitektur CNN Pra-Latih (Inception, MobileNet, VGG, ResNet) terhadap Nilai BLEU dan ROUGE:

a. VGG menunjukkan performa terbaik dalam presisi teks dengan BLEU tertinggi (0,3656), tetapi kurang stabil jika dibandingkan MobileNet dan ResNet
b. MobileNet unggul dalam stabilitas performa dengan rata-rata BLEU (0,2226) dan ROUGE (0,0483) tertinggi, cocok untuk aplikasi yang memprioritaskan konsistensi.
c. ResNet memberikan hasil moderat dengan rata rata BLEU 0,2079 dan ROUGE 0,0406.
d. Inception tampaknya kurang cocok untuk tugas *image captioning* tanpa optimasi lebih lanjut, mengingat performanya yang jauh di bawah model lainnya pada metrik BLEU maupun Rouge.

2. Pengaruh penggunaan padding dibandingkan resizing dengan stretching dalam preprocessing gambar memberikan dampak signifikan terhadap skor BLEU dan Rouge pada model *image captioning.*

Penggunaan padding dalam preprocessing gambar tidak memberikan pengaruh yang signifikan pada skor BLEU dan Rouge. Sebaliknya, Hasil percobaan menunjukkan bahwa skenario tanpa *padding* memberikan hasil yang lebih baik untuk sebagian besar *feature extractor*, baik dari nilai maksimum maupun rata-rata. Hal ini menunjukkan bahwa *padding* dapat mengurangi efisiensi representasi fitur, terutama pada arsitektur seperti MobileNet dan VGG. Namun, ResNet menunjukkan fleksibilitas

yang lebih baik dalam menangani skenario dengan *padding*, terutama pada metrik ROUGE, yang dapat menjadikannya pilihan menarik untuk dataset atau aplikasi yang memerlukan *padding*.

3. Pengaruh jumlah lapisan decoder (3–6 lapisan) memengaruhi kemampuan model dalam menghasilkan caption pada dataset Flickr8k, terutama terkait skor BLEU dan ROUGE.

a. Secara umum, Jumlah layer mempengaruhi skor BLEU dan ROUGE, tetapi tidak selalu berbanding lurus dan setiap arsitektur memiliki jumlah layer optimal yang berbeda.
b. MobileNet menunjukkan performa paling konsisten dengan BLEU optimal (0,350) pada 5 layer dan ROUGE optimal (0,083) pada 6 layer.
c. VGG: Performa terbaik pada layer yang lebih sedikit dengan BLEU tertinggi (0,366) pada percobaan 4 layer, namun mengalami penurunan drastis pada percobaan 6 layer. Hal tersebut mengindikasikan bahwa model kurang efektif dalam menangani kedalaman *layer* yang tinggi untuk tugas *image captioning*.
d. ResNet: Menunjukkan peningkatan performa dari *layer* 4 hingga 6 *layer* (BLEU 0,351), hal tersebut mengidikasikan bahwa model dapat dilakukan pelatihan dengan jaringan lebih dalam.
e. Inception: Menunjukkan performa yang lebih rendah dibanding arsitektur lain dengan skor ROUGE maksimal 0,027 dan BLEU 0,175

7.2 Saran

Berdasarkan hasil dan temuan dari penelitian yang telah dilakukan, terdapat beberapa arah yang dapat dijadikan sebagai fokus pengembangan untuk penelitian selanjutnya. Saran-saran ini diharapkan dapat memberikan kontribusi terhadap peningkatan performa model dan memperluas cakupan solusi yang dihasilkan. Berikut adalah dua saran yang dapat dipertimbangkan:

1. Eksplorasi Hybrid *Feature Extractor*

Salah satu arah pengembangan yang potensial adalah eksplorasi terhadap metode hybrid feature extractor, yaitu dengan mengombinasikan fitur dari beberapa arsitektur CNN. Misalnya, arsitektur VGG memiliki keunggulan dalam menangkap detail tekstur, sementara MobileNet lebih unggul dalam efisiensi komputasi karena ukurannya yang ringan. Kombinasi keduanya dapat memberikan representasi fitur visual yang lebih seimbang antara kedalaman informasi dan kecepatan proses. Penggabungan fitur ini dapat dilakukan secara paralel atau bertingkat, tergantung pada arsitektur sistem yang digunakan. Dengan melakukan pendekatan ini, diharapkan kualitas input pada tahap encoding meningkat dan menghasilkan performa yang lebih baik dalam proses captioning atau tugas lainnya yang bergantung pada pemahaman visual.

2. Peningkatan Mekanisme *Decoder*

Saran berikutnya adalah melakukan peningkatan terhadap mekanisme decoder, khususnya pada model berbasis Transformer. Salah satu masalah umum pada decoder adalah kemunculan kata yang berulang atau redundan, terutama pada kondisi temperature rendah dalam proses decoding. Untuk mengatasi hal tersebut, dapat diterapkan teknik anti-repetisi seperti *nucleus sampling* atau pemberian penalti terhadap token yang telah muncul sebelumnya. Selain itu, eksplorasi terhadap arsitektur hybrid pada bagian decoder, seperti menggabungkan Transformer dan LSTM, juga perlu dipertimbangkan. Kombinasi ini diharapkan mampu memanfaatkan kekuatan Transformer dalam pemrosesan paralel dan kekuatan LSTM dalam memahami konteks jangka panjang, sehingga dapat menghasilkan deskripsi yang lebih akurat, alami, dan variatif.

LAMPIRAN

1. Tabel hasil Percobaan dapat diakses di Tabel Hasil Percobaan
2. Tabel Hasil Generasi Caption dengan VGG menggunakan dataset COCO
3. Source Code Image Captioning

DAFTAR PUSTAKA

[1] K. Xu *et al.*, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," *CoRR*, vol. abs/1502.03044, 2015.

[2] Z. Zhang, Q. Wu, Y. Wang, and F. Chen, "High-Quality Image Captioning With Fine-Grained and Semantic-Guided Visual Attention," *IEEE Trans. Multimed.*, vol. 21, no. 7, pp. 1681–1693, 2019, doi: 10.1109/TMM.2018.2888822.

[3] R. Castro, I. Pineda, W. Lim, and M. E. Morocho-Cayamcela, "Deep Learning Approaches Based on Transformer Architectures for Image Captioning Tasks," *IEEE Access*, vol. 10, pp. 33679–33694, 2022, doi: 10.1109/ACCESS.2022.3161428.

[4] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 6, p. 420, 2021, doi: 10.1007/s42979-021-00815-1.

[5] M. Carrasco, "Visual attention: The past 25 years," *Vision Res.*, vol. 51, no. 13, pp. 1484–1525, 2011, doi: https://doi.org/10.1016/j.visres.2011.04.012.

[6] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in Transformer," *CoRR*, vol. abs/2103.00112, 2021.

[7] K. Wolk and K. Marasek, "Enhanced Bilingual Evaluation Understudy," *CoRR*, vol. abs/1509.09088, 2015.

[8] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Annual Meeting of the Association for Computational Linguistics*, 2004.

[9] G. S. Handelman *et al.*, "Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods," *Am. J. Roentgenol.*, vol. 212, no. 1, pp. 38–43, Oct. 2018, doi: 10.2214/AJR.18.20224.

BIODATA PENULIS

Nama	: Muhammad Hafidh Rosyadi
Tempat, Tanggal Lahir	: Gresik, 21 September 2003
Jenis Kelamin	: Laki - Laki
Telepon	: +6289681061300
Email AKADEMIS	: 5025211013@student.its.ac.id
Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 8 (Delapan)