MULTICLASS IMAGE CLASSIFICATION: A COMPARATIVE STUDY OF A SELF MADE MODEL AND EFFICIENTNET-B0

Calista Lianardi - 2702325880[1)], Ririn Saprina Kadang - 2702315715[1)]
[1)]Data Science, School of Computer Science, Universitas Bina Nusantara,

Pendahuluan

Bencana alam seperti gempa bumi, kebakaran di wilayah perkotaan, tanah longsor, dan banjir merupakan fenomena yang sering terjadi di berbagai belahan dunia. Dampaknya tidak hanya mencakup kerugian material dan infrastruktur, tetapi juga dapat mengakibatkan korban jiwa dalam jumlah besar. Dalam situasi darurat seperti ini, kemampuan untuk mengenali jenis bencana secara cepat dan akurat menjadi sangat krusial guna mendukung respons yang tepat dari tim penyelamat maupun pihak berwenang.

Di era digital, gambar dan foto yang diambil oleh masyarakat, media, maupun perangkat pemantau menjadi sumber informasi utama dalam menggambarkan kondisi bencana secara visual. Oleh karena itu, pengembangan sistem berbasis kecerdasan buatan (*Artificial Intelligence/AI*) yang mampu mengklasifikasikan jenis bencana dari gambar secara otomatis memiliki nilai strategis yang tinggi. Sistem semacam ini dapat mempercepat proses identifikasi bencana, mengurangi beban kerja manual, dan membantu dalam pengambilan keputusan yang lebih cepat serta tepat sasaran.

Perkembangan teknologi *deep learning* memungkinkan dilakukannya *multiclass image classification*, yakni proses mengidentifikasi kelas dari sebuah gambar yang dapat termasuk ke dalam lebih dari dua kategori. Dengan melatih model deep learning menggunakan dataset yang merepresentasikan berbagai jenis bencana alam, sistem dapat belajar mengenali pola visual khas dari masing-masing kategori bencana.

Proyek ini menggunakan *Disaster Images Dataset* dari platform Kaggle, yang berisi kumpulan gambar dari empat kategori bencana alam, yaitu:

1. Earthquake - gambar yang menampilkan dampak dari gempa bumi seperti bangunan roboh, keretakan tanah, atau kehancuran infrastruktur.
2. Urban Fire - gambar yang menunjukkan kebakaran yang terjadi di lingkungan perkotaan.
3. Land Slide - gambar tanah longsor, lereng bukit yang runtuh, jalan yang tertimbun, atau rumah-rumah yang hancur karena longsoran.

4. Water Disaster - gambar bencana yang berhubungan dengan air, seperti banjir atau genangan air yang tinggi.

Tujuan utama proyek ini adalah membangun sebuah model klasifikasi berbasis deep learning yang mampu mengidentifikasi kategori bencana dari gambar secara otomatis dan akurat. Pendekatan ini diharapkan dapat berkontribusi dalam sistem pendeteksian dini (*early warning system*), serta mendukung upaya mitigasi dan penanganan bencana secara lebih efisien.

Ruang Lingkup Proyek

Untuk mencapai tujuan tersebut, proyek ini mencakup tahapan-tahapan berikut:

1. Analisis Data (*Data Analysis*)
   Melakukan eksplorasi awal untuk memahami karakteristik data, distribusi kelas, serta potensi tantangan dalam proses klasifikasi.

2. Persiapan Data (*Data Preparation*)
   Mencakup proses pembersihan data, menghitung image weight, pembagian dataset (train, validation, test), serta normalisasi data agar sesuai dengan kebutuhan model.

3. Pemodelan dan Eksperimen (*Modeling and Experimentation*)
   Membangun dan melatih model *deep learning* dengan arsitektur yang sesuai, seperti *Convolutional Neural Network* (CNN), untuk melakukan klasifikasi gambar.

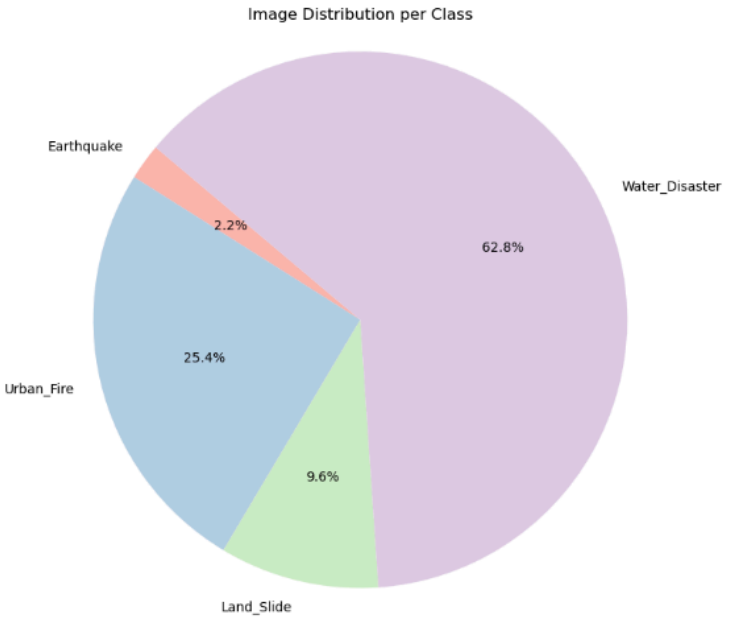
4. Evaluasi Model (*Evaluation*)
   Mengukur performa model menggunakan metrik evaluasi seperti akurasi, presisi, recall, dan F1-score, serta analisis confusion matrix.

5. Dokumentasi (*Documentation*)
   Menyusun laporan lengkap terkait proses pengembangan model, temuan penting, hasil evaluasi, serta potensi pengembangan di masa depan.

Hasil Analisa Sederhana

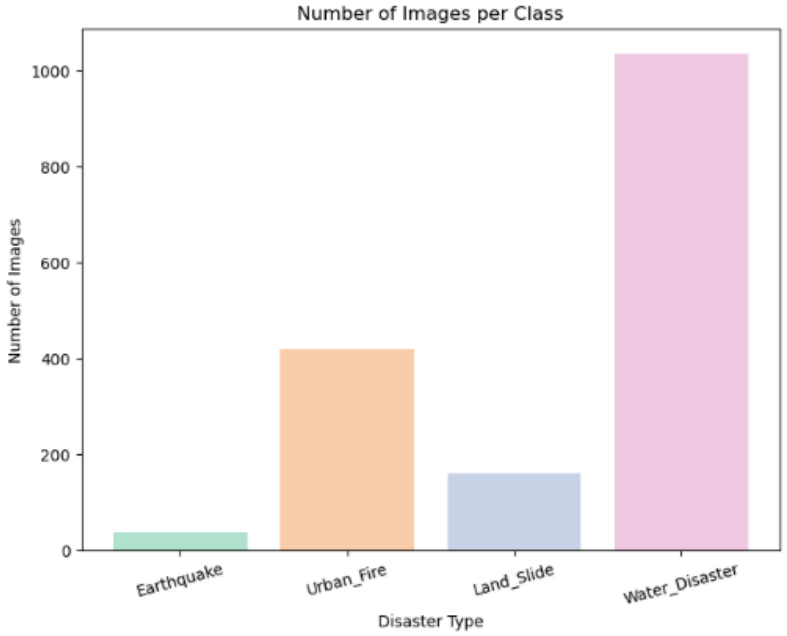
Analisis awal dilakukan untuk memahami struktur dan karakteristik dataset sebelum melanjutkan ke tahap pemodelan. Proses ini melibatkan beberapa pendekatan eksploratif seperti visualisasi distribusi gambar per kelas, pemeriksaan ukuran gambar, analisis statistik warna, serta histogram intensitas warna RGB untuk masing-masing kelas.

Distribusi gambar divisualisasikan menggunakan pie chart dan histogram, yang menunjukkan bahwa kelas *Water_Disaster* mendominasi dengan proporsi sekitar 62.8% dari keseluruhan dataset. Sementara itu, *Urban_Fire* berada di posisi kedua dengan 25.4%, *Land_Slide* sekitar 9.6%, dan *Earthquake* menjadi yang paling sedikit dengan hanya 2.2%. Ketimpangan distribusi ini menunjukkan adanya masalah ketidakseimbangan kelas (*class imbalance*), yang dapat mempengaruhi performa model klasifikasi.



Gambar 1.1 Diagram distribusi image per kelas

Untuk mengatasi ketimpangan ini, dilakukan pendekatan menggunakan *image weight* sebagai alternatif dari augmentasi data. Hal ini didasarkan pada fakta bahwa kelas *Water_Disaster* memiliki lebih dari 1000 gambar, *Urban_Fire* sekitar 400, sementara *Land_Slide* kurang dari 200 dan *Earthquake* bahkan tidak mencapai 100 gambar. Penerapan image weight memungkinkan model memperlakukan kelas minoritas dengan perhatian lebih tanpa harus menambah data sintetis yang mungkin mengurangi kualitas representasi data.



Gambar 1.2 Diagram batang jumlah image per kelas

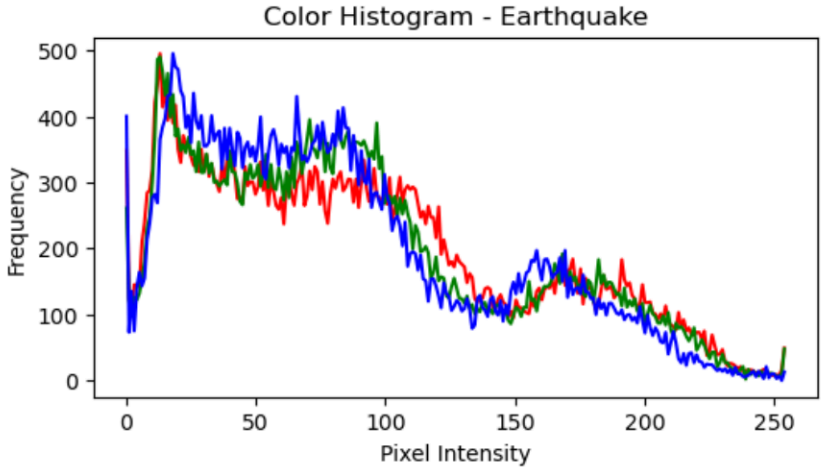
Sampling acak dilakukan untuk mengambil satu gambar per kelas.
Gambar-gambar ini digunakan sebagai representasi visual untuk memahami karakteristik umum masing-masing kategori. Selain itu, ukuran gambar dianalisis untuk melihat sebaran dimensi, dengan hasil sebagai berikut:

Rata-rata lebar gambar: 495 piksel
Rata-rata tinggi gambar: 425.5 piksel
Lebar terkecil: 161 piksel
Lebar terbesar: 1280 piksel
Tinggi terkecil: 120 piksel
Tinggi terbesar: 1350 piksel

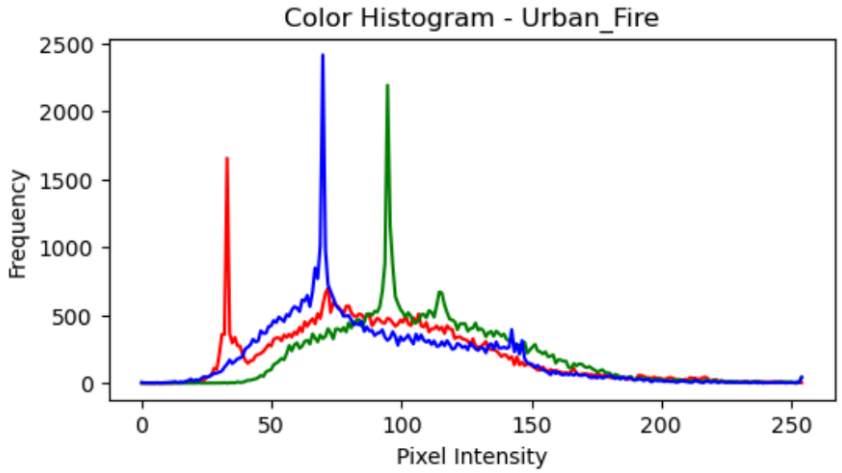
Temuan ini menunjukkan bahwa dataset memiliki variasi ukuran gambar yang cukup besar. Hal ini penting untuk diperhatikan karena ukuran yang tidak konsisten dapat mempengaruhi efisiensi dan akurasi proses pelatihan model. Oleh karena itu, diperlukan proses *resizing* gambar ke dimensi tetap sebelum dimasukkan ke dalam model.

Selanjutnya, plot intensitas warna RGB untuk satu sampel dari tiap kelas diplotkan untuk melihat distribusi nilai piksel dan frekuensinya. Dari grafik tersebut, terlihat bahwa setiap kelas memiliki distribusi warna yang berbeda, yang mencerminkan ciri khas visual dari masing-masing bencana. Misalnya, *Urban_Fire* menunjukkan puncak intensitas tinggi di saluran merah karena dominasi warna api, sedangkan *Water_Disaster* cenderung memiliki intensitas

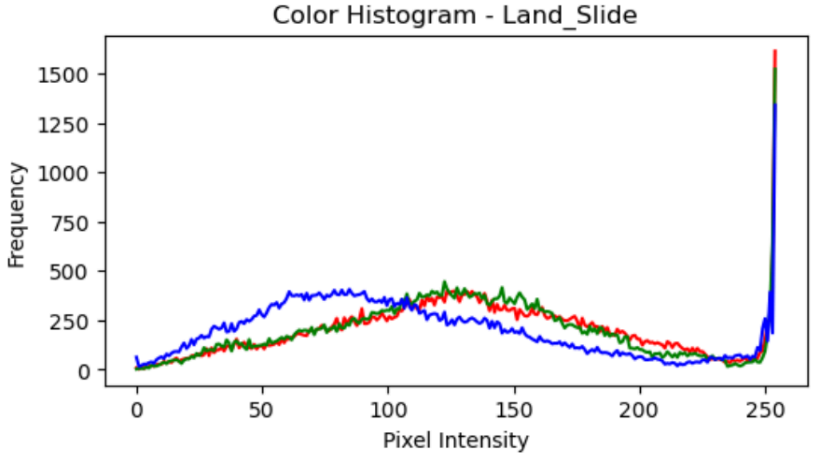
tinggi pada saluran biru. Analisis ini memberikan gambaran awal bahwa perbedaan distribusi warna dapat menjadi fitur penting yang membantu model membedakan antar kelas.



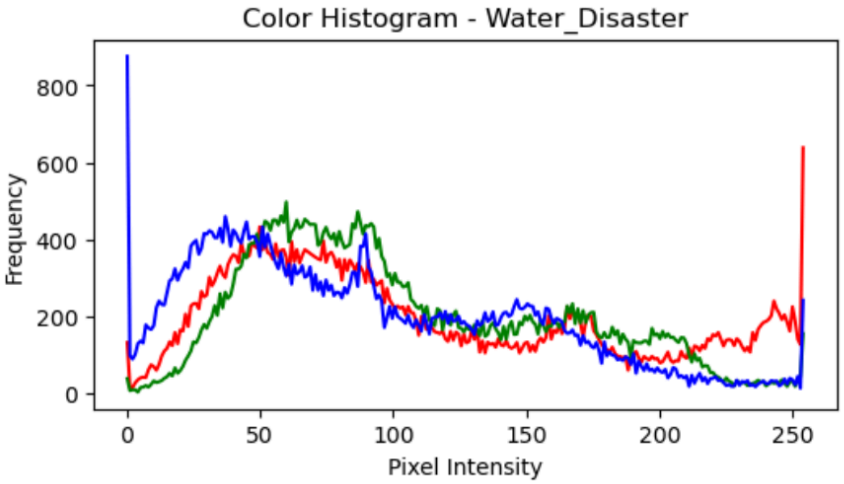
Gambar 1.3 Histogram distribusi RGB untuk kelas *Earthquake*



Gambar 1.4 Histogram distribusi RGB untuk kelas *Urban_Fire*



Gambar 1.5 Histogram distribusi RGB untuk kelas *Land_Slide*



Gambar 1.6 Histogram distribusi RGB untuk kelas *Water_Disaster*

Selain histogram, dilakukan juga perhitungan nilai rata-rata (mean) dan standar deviasi (standard deviation) dari tiap saluran warna RGB untuk setiap kelas dengan hasil sebagai berikut

Classes	**Mean RGB**	**Std**
Earthquake	[104.37, 101.52, 97.63]	[54.66, 53.25, 52.93]
Urban_Fire	[115.73, 95.35, 83.34]	[62.25, 54.68, 49.60]
Land_Slide	[114.83, 111.81, 103.04]	[55.50, 53.39, 55.08]
Water_Disaster	[126.20, 125.81, 118.33]	[54.27, 52.28, 55.53]

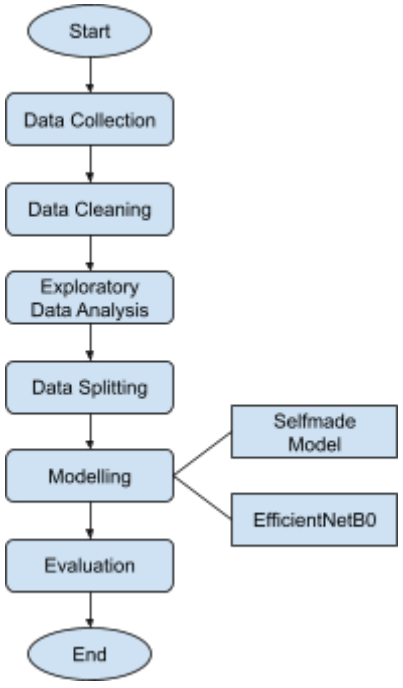
Tabel 1. Mean & Std RBG tiap kelas

Perbedaan nilai mean dan standar deviasi ini mengindikasikan bahwa setiap kelas memiliki pola distribusi warna yang unik. Misalnya, Urban_Fire memiliki dominasi warna merah yang lebih tinggi, sementara Water_Disaster cenderung memiliki warna yang lebih terang dan biru. Nilai-nilai ini juga dapat digunakan dalam proses normalisasi data agar model lebih stabil selama proses pelatihan.

Terakhir, dilakukan analisis terhadap bentuk (shape) atau dimensi file gambar pada setiap kelas. Dengan memahami bentuk atau ukuran gambar, kita dapat mengatur parameter input layer pada model agar sesuai dan seragam. Hal ini krusial dalam model CNN, karena perbedaan bentuk input dapat menyebabkan error atau hasil yang tidak konsisten. Analisis bentuk ini juga membantu dalam merancang pipeline preprocessing yang optimal.

Metode

Penelitian ini terdiri atas enam tahapan proses terstruktur untuk membangun model klasifikasi citra multi kelas berbasis *deep learning*. Proses dimulai dari pengumpulan dataset yang bersumber dari Kaggle, kemudian dilanjutkan dengan tahap pembersihan data untuk memastikan hanya file gambar yang valid yang digunakan dalam proses pemodelan. Setelah itu, dilakukan analisis eksploratif guna memahami karakteristik data, seperti distribusi kelas dan potensi ketidakseimbangan data. Setelah pembersihan dan analisis selesai, data dibagi menjadi data pelatihan, pengujian, dan validasi untuk memungkinkan pelatihan dan evaluasi model pada kumpulan data terpisah. Model kemudian dibangun dan dilatih menggunakan arsitektur *Convolutional Neural Network* (CNN), dan pada akhirnya dievaluasi dengan menggunakan berbagai metrik performa untuk mengukur tingkat akurasi dan efektivitas klasifikasi. Setiap tahapan dalam metode ini disusun agar menghasilkan model yang tidak hanya akurat, namun juga dapat diandalkan dalam mengidentifikasi jenis bencana alam secara otomatis berdasarkan citra.



Gambar 1. Diagram Alur Metodologi

I. Pengumpulan Data (*Data Collection*)

Dataset yang digunakan dalam penelitian ini diperoleh dari platform *Kaggle*, yaitu *Disaster Images Dataset*. Dataset ini berisi kumpulan gambar dari empat kategori bencana alam, yakni *Earthquake, Urban Fire, Land Slide, dan Water Disaster.* Gambar-gambar ini mencerminkan situasi bencana nyata dan akan digunakan sebagai data input untuk membangun model klasifikasi citra multikelas.

II. Pembersihan dan Seleksi Data (*Data Cleaning*)

Sebelum dilakukan proses analisis dan pelatihan model, dilakukan tahap pembersihan data. Proses ini dilakukan dengan menggunakan modul *pathlib* untuk membaca *path* gambar serta *imghdr* untuk memverifikasi format file gambar. Hanya file dengan format PNG, JPG, JPEG, BMP, dan GIF yang dipertahankan, karena model hanya dapat membaca dan memproses gambar dalam format valid. File yang tidak sesuai format tersebut dihapus dari dataset untuk mencegah error saat pemrosesan selanjutnya. Langkah ini berperan penting dalam mencegah error ketika memuat data ke dalam *data generator* serta memastikan model hanya menerima input yang sesuai.

III. Analisis Data (*Exploratory Data Analysis*)

Setelah data dibersihkan, dilakukan analisis eksploratif terhadap data. Analisis ini mencakup visualisasi distribusi gambar untuk setiap kategori bencana, identifikasi jumlah gambar per kelas, serta pemetaan potensi ketidakseimbangan data. Visualisasi seperti bar chart digunakan untuk menampilkan jumlah gambar per kategori, yang membantu dalam memahami struktur dataset dan menjadi dasar untuk penyesuaian pada proses pelatihan seperti penggunaan *image weight*.

IV. Pemisahan Data (*Data Splitting*)

Dataset kemudian dibagi menjadi tiga subset: train, validation, dan test, dengan rasio 70:15:15. Dimana train set digunakan untuk melatih model dengan tujuan mengenali pola umum dari gambar, validation set digunakan untuk memantau kinerja model saat pelatihan, mengatur parameter, dan mencegah overfitting, dan test set digunakan untuk mengevaluasi kinerja akhir model pada data yang tidak pernah dilihat sebelumnya. Penggunaan validation set sangat penting dalam pengembangan model deep learning, karena jika hanya menggunakan train dan test, proses tuning bisa menjadi bias dan menurunkan kemampuan generalisasi. Selain itu, image weight dihitung berdasarkan proporsi kelas dalam train set untuk mengimbangi ketidakseimbangan data. Ini dilakukan dengan memberikan bobot lebih tinggi pada kelas minoritas selama pelatihan, agar model tidak terlalu memprioritaskan kelas mayoritas.

V. Pelatihan Model (*Modelling*)

Pada tahap pelatihan, dilakukan dua pendekatan arsitektural untuk keperluan perbandingan. Pertama, dibangun arsitektur CNN sederhana dari awal sebagai baseline, terdiri atas beberapa lapisan konvolusi dan pooling untuk mengekstraksi fitur, diikuti dengan lapisan dense untuk

klasifikasi akhir. Kedua, digunakan *EfficientNet-B0*, model *pretrained* yang terkenal karena efisiensi dan akurasinya dalam klasifikasi gambar. Model ini dimodifikasi pada bagian output layer agar sesuai dengan jumlah kelas dalam dataset. Seluruh pelatihan dilakukan tanpa augmentasi data, dengan mempertimbangkan *image weight* hasil EDA untuk mengatasi ketidakseimbangan kelas.

VI. Evaluasi Model (*Evaluation*)

Setelah pelatihan selesai, performa model dievaluasi menggunakan data pengujian. Evaluasi utama menggunakan metrik akurasi untuk mengetahui tingkat keberhasilan klasifikasi gambar ke dalam kelas yang tepat. Selain itu, confusion matrix digunakan untuk melihat detail kesalahan klasifikasi pada masing-masing kelas. Dengan membandingkan performa CNN buatan sendiri dan EfficientNet-B0, dapat ditentukan model mana yang lebih optimal untuk diterapkan dalam konteks klasifikasi gambar bencana alam.

Hasi dan Pembahasan

Eksperimen ini dilakukan untuk membandingkan performa dua model CNN buatan sendiri dan satu model pretrained EfficientNet-B0 pada tugas klasifikasi citra bencana alam dengan jumlah data yang relatif kecil dan distribusi kelas yang tidak seimbang. Evaluasi dilakukan menggunakan metrik akurasi dan analisis confusion matrix. Tujuannya adalah untuk menilai sejauh mana arsitektur dan strategi training mempengaruhi performa klasifikasi.

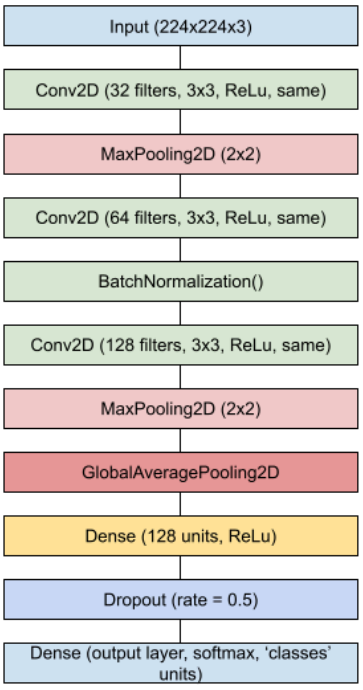
I. Model Rancangan Pribadi (CLRKModel)

Model pertama yang dirancang secara mandiri, CLRKModel, menggunakan tiga lapisan konvolusi berturut-turut dengan konfigurasi kernel (3,3) dan padding 'same', serta diaktifkan dengan fungsi aktivasi ReLU. Pemilihan ReLU (*Rectified Linear Unit*) bertujuan untuk menghindari masalah vanishing gradient dan mempercepat proses konvergensi selama pelatihan.

Setiap dua lapisan konvolusi disertai dengan *MaxPooling2D*, yang berfungsi untuk menurunkan dimensi spasial citra dan mengurangi jumlah parameter, sekaligus memperkenalkan translational invariance. Pooling dilakukan dengan ukuran (2,2) sebagai konfigurasi standar yang cukup efisien untuk ekstraksi fitur dasar. Kemudian, *BatchNormalization* ditambahkan pada middle layer untuk menstabilkan distribusi aktivasi dan mempercepat proses pelatihan.

Alih-alih menggunakan Flatten, model menggunakan *GlobalAveragePooling2D,* yang dipilih karena ukuran dataset yang sangat kecil pada kelas dengan ukuran data terkecil(< 100 data). *Global Average Pooling* cenderung lebih robust terhadap overfitting karena menghasilkan parameter yang jauh lebih sedikit dibandingkan Flatten.

Setelah proses penyetaraan fitur, model memiliki satu Dense layer dengan 128 neuron yang diikuti oleh Dropout sebesar 0.5. Besarnya dropout dipilih sebagai bentuk regularisasi untuk mencegah co-adaptation antar neuron dan memperkecil risiko overfitting. Lapisan terakhir menggunakan fungsi aktivasi softmax karena permasalahan ini merupakan klasifikasi multi-kelas.

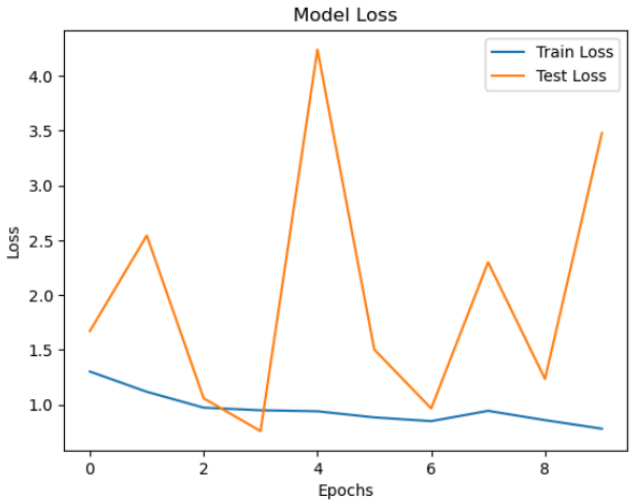


Gambar 2.1 Diagram Arsitektur CLRKMmodel

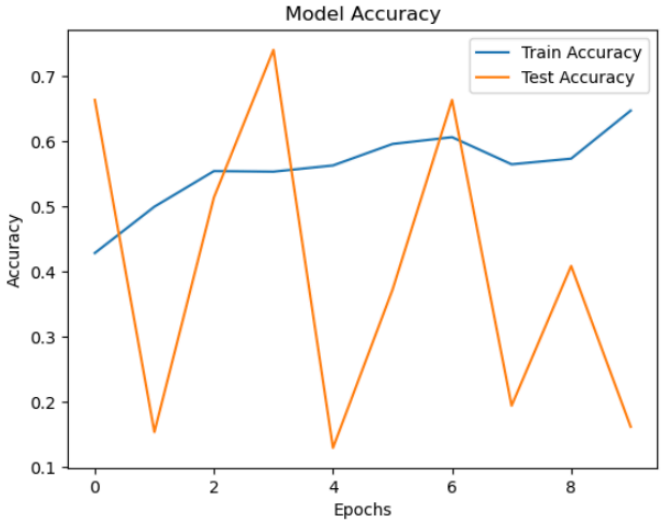
Model ini dilatih selama 10 epoch dengan batch size sebesar 32, menggunakan fungsi loss sparse_categorical_crossentropy. Fungsi loss ini dipilih karena data label yang digunakan berbentuk integer, bukan one-hot encoded.

Namun, hasil pelatihan menunjukkan bahwa model mengalami **overfitting yang cukup parah**, ditandai dengan tren akurasi *train* yang meningkat drastis sementara akurasi *test* stagnan atau bahkan menurun, serta fluktuasi

tajam pada kurva loss dan accuracy. Hal ini diperkuat dengan hasil confusion matrix yang menunjukkan bahwa model hanya fokus mengenali sebagian kecil kelas (terutama prediksi kelas 3 ke 0 sebanyak 122 kali) dan accuracy sebesar 0.09.



Gambar 2.2 Diagram Train Test Loss CLRKModel



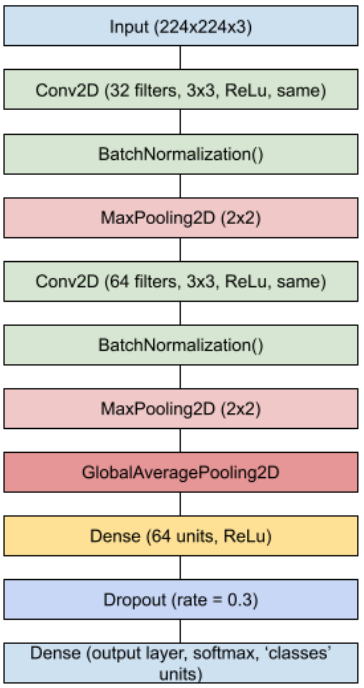
Gambar 2.3 Diagram Train Test Accuracy CLRKModel

II. Perbaikan Model Rancangan Pribadi (CLRKModelV2)

Berdasarkan hasil tersebut, dilakukan perbaikan terhadap arsitektur awal dengan merancang model baru bernama CLRKModelV2. Model ini merupakan versi penyederhanaan dari model sebelumnya, dengan hanya menggunakan dua lapisan konvolusi serta menambahkan Batch Normalization yang diharapkan dapat mengurangi internal covariate shift dan mempercepat konvergensi, dan MaxPooling yang dapa mengurangi resolusi fitur secara efisien, mencegah overfitting, dan menstabilkan proses pelatihan setelah setiap lapisan konvolusi. Penambahan ini

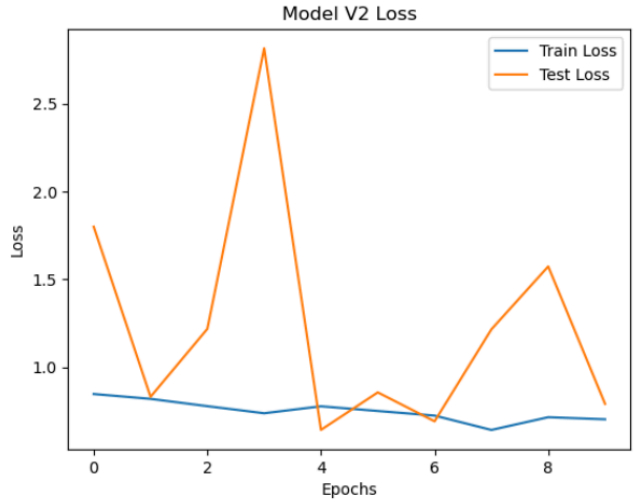
bertujuan untuk menstabilkan distribusi aktivasi selama pelatihan, mempercepat konvergensi, serta mengurangi sensitivitas terhadap inisialisasi parameter.

Setelah proses ekstraksi fitur, digunakan kembali *GlobalAveragePooling2D* untuk menjaga jumlah parameter tetap rendah. Jumlah neuron pada *Dense* layer juga dikurangi menjadi 64 unit agar sesuai dengan representasi fitur dari dua lapisan konvolusi. Dropout juga dikurangi menjadi 0.3 agar tetap memberikan efek regularisasi namun tidak menghambat pembelajaran terlalu drastis.

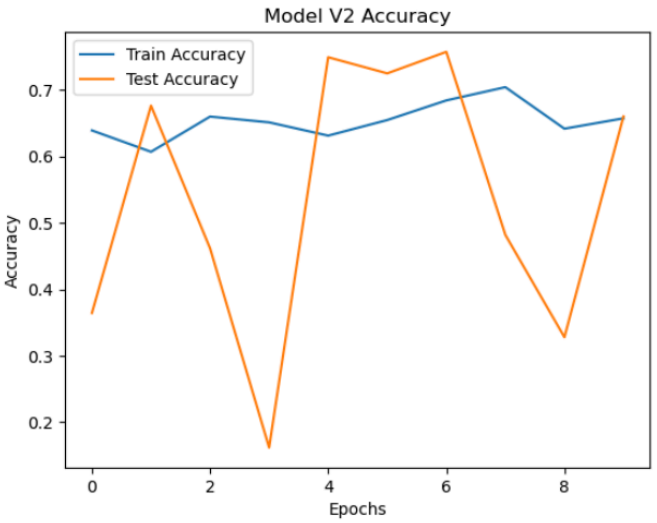


Gambar 2.4 Diagram Arsitektur CLRKMmodelV2

Hasil pelatihan menunjukkan sedikit perbaikan dibanding model sebelumnya: akurasi meningkat hingga 0.26, dan confusion matrix memperlihatkan hasil prediksi lebih menyebar ke kelas yang benar (misalnya prediksi terbanyak 3–2 sebanyak 155). Namun, grafik performa masih menunjukkan fluktuasi dan kecenderungan overfitting meskipun lebih menunjukan kestabilan dibandingkan model pertama.



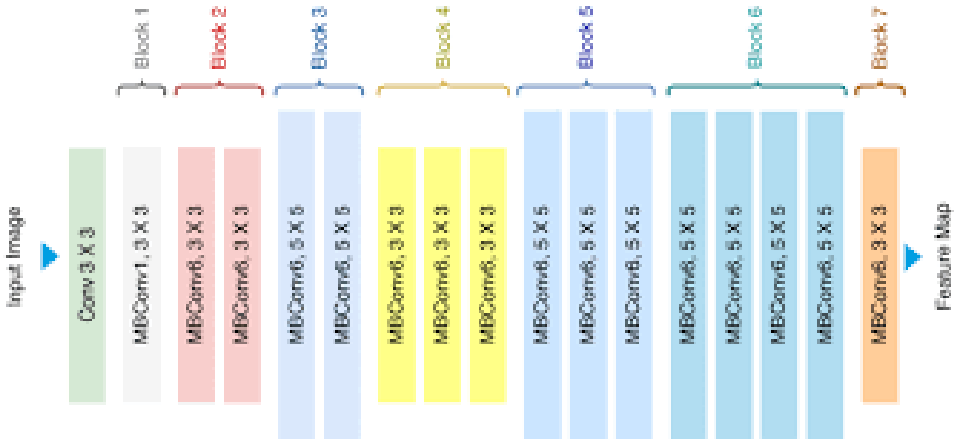
Gambar 2.6 Diagram Train Test Loss CLRKModelV2



Gambar 2.7 Diagram Train Test Accuracy CLRKModelV2

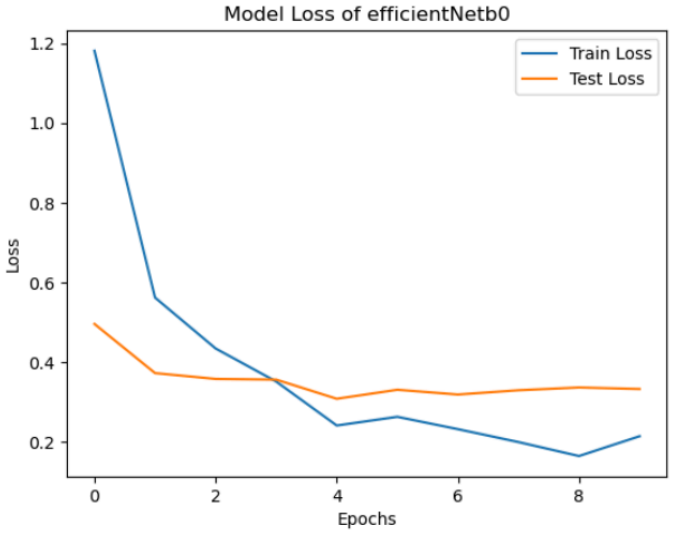
III. Komparasi Dengan Pre-trained Model (EfficientNet-B0)

Sebagai baseline model pembanding, digunakan EfficientNet-B0, sebuah model pretrained convolutional neural network yang telah dioptimalkan melalui compound scaling. Model ini memiliki sekitar 4 juta parameter, namun menunjukkan hasil yang sangat stabil dan efektif dalam pembelajaran.

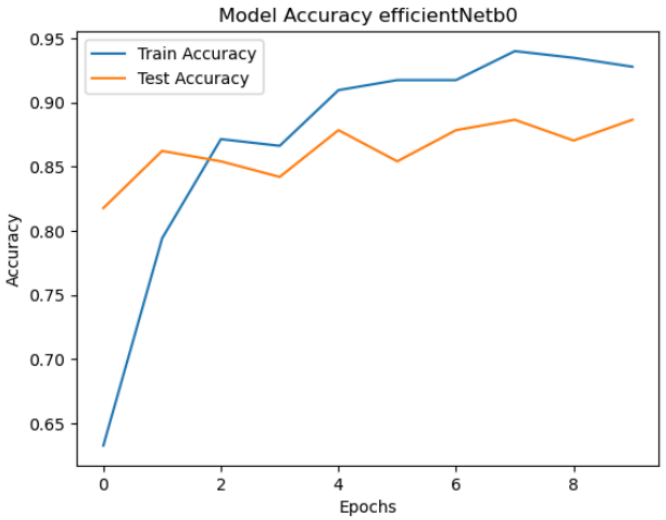


Gambar 2.8 Diagram Arsitektur EfficientNetB0

Dengan train loss yang menurun secara konsisten, test accuracy yang stabil antara 85% - 90% dan train accuracy yang meningkat dan konvergen secara halus sejak epoch ke-4. Membuktikan stabilitasnya dalam training model



Gambar 2.9 Diagram Train Test Loss EfficientNet-B0



Gambar 2.10 Diagram Train Test Accurracy EfficientNet-B0

Stabilitas dan performa tinggi dari efficientNet ini kemungkinan terjadi karena compound scaling yang secara seimbang meningkatkan depth, width, dan resolution, yang tidak dimiliki oleh *self architectured model.* Parameter yang lebih banyak juga berkemungkinan menjadikan model ini lebih optimal dan stabil untuk mempelajari data. Dibandingkan dengan pendekatan klasik (yang biasanya hanya menambah depth atau width saja), compound scaling **menjaga keseimbangan antara kapasitas dan efisiensi**.

Selain itu, jumlah parameter yang jauh lebih besar—sekitar 4 juta dibandingkan dengan hanya 100 ribu pada model pertama dan 300 ribu pada model kedua—memberi ruang bagi model untuk mempelajari representasi fitur yang lebih kompleks dan beragam. Ini sangat berperan penting, terutama dalam konteks transfer learning, karena EfficientNet telah dilatih terlebih dahulu pada dataset besar seperti ImageNet, sehingga mampu langsung mengarahkan pembelajaran pada fitur yang relevan meskipun digunakan di dataset kecil seperti data *Earthquake* yang dimiliki.

IV. Analisa Performa dan Potensi Perbaikan

Dua model buatan sendiri menunjukkan performa rendah dibanding EfficientNet karena:

1. Tidak Menggunakan Data Augmentation

   Augmentasi data seperti rotation, zoom, atau horizontal flip dapat memperkaya representasi visual dari data kecil sehingga meningkatkan kemampuan generalisasi. Tanpa augmentasi, model

sulit belajar pola yang bervariasi dan hanya “menghafal” data pelatihan, meskipun menggunakan weighting.

2. Jumlah Data Terlalu Kecil untuk CNN yang Dalam
   CNN membutuhkan jumlah data yang cukup agar mampu belajar representasi fitur dengan baik. Model dengan 100K parameter masih terlalu kompleks bila dibandingkan dengan dataset <100 gambar, terutama tanpa bantuan pretrained weight atau augmentasi.

3. GlobalAveragePooling vs Flatten
   GlobalAveragePooling memang membantu mengurangi jumlah parameter dan menghindari overfitting, namun juga bisa mengorbankan representasi spasial detail jika fitur belum cukup abstrak. Flatten mungkin terlalu overfit, tapi bisa lebih representatif jika jaringan tidak terlalu dalam.

4. BatchNormalization dan MaxPooling Membantu
   Model kedua lebih baik karena adanya BatchNormalization pada tiap conv layer, yang menstabilkan distribusi aktivasi dan mempercepat pelatihan. MaxPooling mengurangi dimensi fitur secara efisien, membuat representasi lebih ringkas.

Dibandingkan dengan EfficientNet, jelas terlihat bahwa arsitektur buatan sendiri belum memiliki keseimbangan struktural dan kapasitas representasi yang cukup untuk menangkap kompleksitas data. Kekuatan EfficientNet juga didukung oleh penggunaan teknik regularisasi lanjutan, seperti *swish activation*, *dropout*, dan *depthwise separable convolution*, yang menjadikannya tidak hanya akurat tetapi juga efisien. Ini menjelaskan mengapa EfficientNet mampu menunjukkan peningkatan akurasi yang stabil dan penurunan loss yang konsisten sepanjang proses pelatihan, sementara kedua model sebelumnya cenderung fluktuatif dan mudah mengalami overfitting.

Kesimpulan

Eksperimen ini menunjukkan bahwa performa klasifikasi citra bencana alam sangat dipengaruhi oleh arsitektur model, strategi pelatihan, serta karakteristik data yang digunakan. Model-model buatan sendiri (CLRKModel dan CLRKModelV2) menunjukkan keterbatasan dalam menangani dataset yang kecil dan tidak seimbang, terutama karena tidak menggunakan augmentasi data,

memiliki kapasitas parameter yang belum optimal, serta kurangnya pretrained knowledge. Meskipun terjadi peningkatan performa pada versi kedua model rancangan, hasilnya masih jauh dari memuaskan, dengan akurasi maksimum hanya sebesar 0.26.

Sebaliknya, model pretrained EfficientNet-B0 mampu memberikan hasil yang jauh lebih stabil dan akurat dengan akurasi test mencapai 85–90%. Hal ini menegaskan keunggulan pendekatan transfer learning, compound scaling, dan regularisasi lanjutan dalam menghadapi keterbatasan data. EfficientNet berhasil mengonversi representasi fitur secara lebih dalam dan efisien karena telah dilatih pada dataset besar, serta memiliki struktur arsitektur yang seimbang antara kompleksitas dan efisiensi.

Dengan demikian, dapat disimpulkan bahwa untuk tugas klasifikasi citra dengan dataset terbatas, penggunaan model pretrained seperti EfficientNet jauh lebih disarankan dibandingkan merancang model dari awal, kecuali jika diiringi dengan teknik augmentasi dan strategi pelatihan yang lebih matang.

Referensi

https://medium.com/geekculture/multiclass-image-classification-dcf9585f2ff9
https://www.tensorflow.org/tutorials/images/classification?hl=id
https://www.researchgate.net/figure/Architecture-of-EfficientNet-B0-with-MBConv-as-Basic-building-blocks_fig3_356981443
https://keras.io/api/applications/
https://keras.io/api/applications/efficientnet/#efficientnetb0-function