Technical Report Data Science Competition

LOGIKA UI 2025

Implementasi Model EVA-02 dalam Klasifikasi Rumah Adat di Indonesia

Disusun oleh:

Evans Kizito

Bryant Farrel Titanius

Franklin Daniel Situmorang

DSC0075

Depok

2025

I. Pendahuluan

1.1 Latar Belakang

Warisan budaya merupakan manifestasi vital dari identitas, sejarah, dan kearifan lokal suatu bangsa. Di Indonesia, rumah adat Nusantara merepresentasikan kekayaan filosofis yang bervariasi di setiap daerah. Namun, pelestarian warisan arsitektur ini menghadapi tantangan serius. UNESCO dan berbagai studi literatur menyoroti bahwa globalisasi dan modernisasi sering kali menggerus pengetahuan lokal generasi muda terhadap warisan budaya mereka sendiri (Fiorucci et al., 2020). Tanpa upaya dokumentasi dan preservasi yang sistematis, risiko hilangnya identitas visual arsitektur Nusantara semakin meningkat.

Dalam satu dekade terakhir, paradigma pelestarian budaya telah bergeser menuju *Digital Heritage*, di mana teknologi informasi digunakan untuk mendokumentasikan dan merestorasi aset budaya. Secara khusus, perkembangan *Deep Learning* dalam bidang *Computer Vision* telah membuka peluang baru untuk klasifikasi dan pengenalan pola arsitektur secara otomatis. Penelitian terdahulu umumnya mengandalkan *Convolutional Neural Networks* (CNN) seperti *ResNet* atau *EfficientNet* untuk tugas klasifikasi gambar. Namun, perkembangan terkini menunjukkan bahwa arsitektur berbasis mekanisme *Attention*, yaitu *Vision-Transformers* (ViT), mampu menangkap dependensi global pada gambar dengan lebih baik dibandingkan CNN konvensional, terutama pada dataset dengan variasi visual yang kompleks (Dosovitskiy et al., 2020).

Kompetisi Data Science Competition (DSC) LOGIKA UI 2025 mengangkat permasalahan ini melalui studi kasus "Klasifikasi Rumah Adat Nusantara". Tantangan utama dalam dataset ini meliputi variasi pencahayaan, sudut pandang (pose), oklusi, serta ketidakseimbangan kelas (*class imbalance*) antar kategori rumah adat (*Balinese*, *Batak*, *Dayak*, *Javanese*, *Minangkabau*). Ketidakseimbangan ini menuntut penerapan strategi pembelajaran yang tidak hanya akurat secara global, tetapi juga adil terhadap kelas minoritas.

Merespons tantangan tersebut, penelitian ini memfokuskan pada implementasi dan evaluasi model EVA-02. EVA-02 merupakan pengembangan mutakhir dari arsitektur ViT yang dirancang untuk merekonstruksi fitur visual yang kuat melalui *masked image modeling*, yang terbukti unggul dalam berbagai tolok ukur *computer vision* (Fang et al., 2023). Laporan ini mendokumentasikan pendekatan teknis, mulai dari pra-pemrosesan data hingga evaluasi model, untuk menghasilkan sistem klasifikasi yang presisi dan berkontribusi pada

digitalisasi warisan budaya Indonesia.

1.2 Tujuan dan Manfaat

Penelitian ini dirancang dengan tujuan dan manfaat yang terukur sebagai berikut:

1.2.1 Tujuan Penelitian

1. Mengembangkan sistem klasifikasi gambar otomatis berbasis *Deep Learning* yang mampu mengidentifikasi lima kategori rumah adat Nusantara dengan tingkat generalisasi yang tinggi.

2. Mengevaluasi efektivitas arsitektur EVA-02 dalam menangani variabilitas visual dan kompleksitas fitur pada gambar arsitektur tradisional.

3. Mengimplementasikan strategi optimasi model, termasuk teknik augmentasi data dan penyesuaian fungsi *loss*, untuk memaksimalkan metrik *Macro F1-Score* pada dataset dengan distribusi kelas yang tidak seimbang (*imbalanced dataset*).

1.2.2 Manfaat Penelitian

Kontribusi Teoretis: Memberikan bukti empiris mengenai performa arsitektur berbasis *Transformer* (EVA-02) dalam domain pelestarian budaya digital, memperkaya literatur mengenai penerapan AI untuk *Digital Heritage*.

Kontribusi Praktis: Menghasilkan model yang dapat diintegrasikan ke dalam platform edukasi atau pariwisata digital untuk membantu masyarakat mengenali dan mempelajari arsitektur Nusantara secara interaktif.

Pengembangan Teknis: Menjadi referensi studi kasus dalam penanganan *imbalanced multi-class classification* menggunakan teknik *state-of-the-art* dalam kompetisi sains data.

1.3 Batasan Penelitian

Untuk memastikan kedalaman analisis dan fokus pembahasan, penelitian ini dibatasi pada ruang lingkup sebagai berikut:

1. **Sumber Data:** Penelitian menggunakan dataset resmi dari Data Science Competition (DSC) LOGIKA UI 2025, yang terdiri dari 1752 gambar latih dan 444

gambar uji yang terbagi ke dalam lima kelas: *Balinese, Batak, Dayak, Javanese,* dan *Minangkabau*.

2. **Metode Arsitektur:** Fokus utama eksperimen pada babak penyisihan adalah implementasi varian model EVA-02.

3. **Ruang Lingkup Masalah:** Masalah dibatasi pada *single-label multi-class image classification*. Penelitian ini tidak mencakup deteksi lokasi objek (*bounding box*) atau segmentasi semantik bangunan.

4. **Metrik Evaluasi:** Kinerja model diukur secara kuantitatif menggunakan Macro F1-Score sebagai metrik primer untuk mengakomodasi ketidakseimbangan data, serta *Accuracy* sebagai metrik sekunder.

II. Landasan Teori

2.1 *Vision-Transformer*

Sebelum hadirnya ***Transformer***, pemrosesan data sekuensial bergantung pada model rekuren seperti RNN dan LSTM yang memproses token secara berurutan, sehingga sulit menangkap *long-range dependencies* dan tidak efisien untuk diparalelisasi. CNN memang dapat diterapkan pada data sekuensial, namun tetap terbatas pada konteks lokal dan membutuhkan kedalaman besar untuk memahami struktur global. Arsitektur *Transformer* memecahkan dua masalah tersebut menggunakan mekanisme ***self-attention***, yang sepenuhnya berbasis operasi matriks, memungkinkan setiap token mengakses seluruh token lain dalam satu langkah komputasi. Dengan proyeksi

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \tag{1}$$

skor perhatian dihitung melalui

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right), \tag{2}$$

dan digunakan untuk menghasilkan

$$\text{Attention}(X) = AV. \tag{3}$$

Self-attention memungkinkan paralelisasi penuh dan menangkap hubungan global secara efisien.

Keunggulan *Transformer* kemudian diterapkan ke bidang *computer vision* melalui ***Vision-Transformer*** (ViT), yang memperlakukan gambar sebagai urutan token. Gambar $X \in \mathbb{R}^{H\times W\times C}$ dibagi menjadi patch berukuran $P \times P$ dengan jumlah

$$N = \frac{HW}{P^2}, \tag{4}$$

dan setiap patch diubah menjadi embedding berdimensi D melalui

$$x_i = W_e\text{vec}(p_i) + b_e. \tag{5}$$

Kemudian, *class token* dan *positional embedding* ditambahkan sehingga terbentuk

$$Z_0 = [x_{cls} \mid x_1 \mid \cdots \mid x_N] + E, \tag{6}$$

yang menjadi masukan untuk *Transformer encoder*. *Encoder* terdiri atas ***Multi-Head Self-Attention*** (MHSA) yang menangkap interaksi antar patch dan ***Feed-Forward Network*** (FFN) yang memperkaya representasi setiap token:

$$\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2, \tag{7}$$

keduanya dilengkapi *residual connection* dan *LayerNorm* untuk stabilitas pelatihan.

Setelah melalui beberapa blok *encoder*, *class token* terakhir h_{cls} menyimpan representasi global gambar dan digunakan oleh *classification head* linear

$$\hat{y} = h_{cls}W_c + b_c, \qquad p = \text{softmax}(\hat{y}). \tag{8}$$

Walaupun ViT menangkap konteks global sejak awal dan mudah diskalakan, arsitektur ini

tidak memiliki *local inductive bias* seperti CNN sehingga pola dasar harus dipelajari dari data besar. Selain itu, kompleksitas $O(n^2)$ dari *self-attention* membuat ViT mahal pada gambar beresolusi tinggi dan lebih sensitif terhadap pemilihan *hyperparameter*.

2.2 Arsitektur Model EVA-02

Meskipun *Vision-Transformer* (ViT) berhasil menggantikan CNN pada masalah *computer vision* yang berskala besar, ViT standar masih memiliki beberapa kekurangan. Beberapa di antaranya meliputi pembahasan pada subbab sebelumnya, seperti

- performa yang sangat bergantung pada data dan pretraining berskala besar,
- ketidakefisienan komputasi terutama pada resolusi tinggi, dan
- sensitivitas terhadap regularisasi dan konfigurasi training.

Untuk mengatasi keterbatasan tersebut, dikembangkanlah model EVA-02. EVA-02 merupakan generasi kedua model EVA yang secara signifikan meningkatkan kualitas representasi visual ViT melalui *redesign pipeline pretraining*, *distillation*, dan optimisasi arsitektur. EVA-02 dibangun untuk mencapai performa tertinggi pada *ImageNet* dan dataset *downstream*, sekaligus mempertahankan efisiensi dan stabilitas training.

Secara garis besar, EVA-02 bukan sekadar ViT yang diperbesar; EVA-02 adalah ViT yang dibentuk ulang dengan prinsip *high-quality large-scale pretraining*, *aggressive distillation* ke arah *teacher model* yang sangat kuat, optimisasi *hyperparameter* dan regularisasi yang dirancang khusus, dan stabilitas training jangka panjang. Hal inilah yang membuat EVA-02 menjadi salah satu *backbone* terbaik untuk tugas klasifikasi, deteksi, serta *foundation modeling*.

Secara arsitektur, EVA-02 masih mengikuti struktur ViT, yaitu *patch embedding*, *positional embedding*, *class token*, dan *Transformer encoder*. Perbedaannya bukan pada struktur dasar, tetapi pada cara model di-*training*, modifikasi kecil pada blok *Transformer*, dan skala *pretraining*-nya. Secara umum, pengembangan paling berdampak yang dilakukan pada model EVA-02 dapat dibagi menjadi tiga:

1. ***Masked Image Modeling* (MIM) yang lebih agresif.** EVA-02 memakai strategi MIM dengan *mask ratio* sangat tinggi (75–90%). Karena ViT tidak menggunakan

MIM, maka EVA-02 akan jauh lebih efisien data dan lebih paham struktur visual dasar.

2. ***Distillation* besar-besaran dari *teacher* berskala besar.** EVA-02 dilatih dengan *distillation* dari *teacher* seperti BEiT-G atau EVA-CLIP. Hal ini berbeda dengan ViT yang hanya belajar langsung dari label (murni *supervised*). Dengan demikian, representasi *class token* EVA-02 jauh lebih tepat dan stabil.

3. **Optimisasi *attention* dan FFN untuk stabilitas skala besar.** EVA-02 menambahkan sejumlah perubahan yang ringan namun penting, seperti *Q/K normalization*, penyesuaian faktor *scaling*, pembesaran FFN, dan juga normalisasi pre-*LayerNorm*. Seluruh hal ini memungkinkan EVA-02 untuk dilatih pada parameter ratusan juta atau bahkan miliaran, tanpa terjadi *exploding* atau *vanishing gradients*.

Dengan pengembangan ini, tentu EVA-02 akan menjadi model yang lebih baik untuk *computer vision* dibanding dengan ViT. Keunggulan EVA-02 dibanding dengan ViT yaitu:

- Akurasi yang lebih tinggi akibat adanya *distillation* dan MIM.
- Representasi fitur jauh lebih stabil, karena menggunakan *class token* yang lebih informatif untuk klasifikasi.
- Efisien data akibat adanya *pre-training* yang kuat.
- Model EVA-02 juga lebih mudah di-*fine-tune* karena *backbone*-nya sudah matang sehingga sensitivitas terhadap *hyperparameter* menjadi lebih rendah.

2.3 *Loss Function*

Loss function digunakan untuk mengukur seberapa jauh prediksi model dari label sebenarnya. Pada klasifikasi *multiclass*, dua *loss function* yang paling umum digunakan adalah ***Cross-Entropy Loss*** dan ***Focal Loss***, terutama ketika model bekerja dengan arsitektur modern seperti ViT dan EVA-02.

Cross-Entropy Loss menghitung divergensi antara distribusi prediksi model dan distribusi label *ground-truth*. Diberikan logits model $z \in \mathbb{R}^C$ dan label *one-hot* y, probabilitas

dihitung menggunakan *softmax*

$$\hat{p}_c = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}}. \tag{9}$$

Cross-Entropy Loss kemudian didefinisikan sebagai

$$\mathcal{L}_{CE} = -\sum_{c=1}^{C} y_c \log(\hat{p}_c). \tag{10}$$

Jika label yang benar adalah kelas t, maka bentuknya menjadi

$$\mathcal{L}_{CE} = -\log(\hat{p}_t). \tag{11}$$

Cross-Entropy Loss bekerja dengan memaksimalkan probabilitas yang diberikan model kepada kelas benar, sehingga sangat umum digunakan sebagai *baseline* dalam tugas klasifikasi.

Meskipun demikian, *Cross-Entropy Loss* memiliki kelemahan fatal, dimana ia kurang efektif dalam menangani data yang *imbalanced*. Hal ini karena model akan mudah mengabaikan kelas minoritas, dan *overconfident* terhadap kelas mayoritas. *Focal Loss* adalah salah satu *loss function* yang dapat menangani masalah tersebut. Ide dari *Focal Loss* ini adalah mengurangi kontribusi sampel yang mudah, sehingga model fokus pada sampel sulit. *Focal Loss* didefinisikan sebagai

$$\mathcal{L}_F = -(1 - \hat{p}_t)^{\gamma} \log(\hat{p}_t). \tag{12}$$

dimana $\gamma \geq 0$ adalah *focusing parameter*. Umumnya, digunakan $\gamma = 1$ atau $\gamma = 2$, tetapi *tuning* pada parameter ini juga dapat dilakukan. Dengan *loss function* tersebut, jika suatu sampel mudah yaitu sampel dengan $\hat{p}_t$ besar, maka $(1 - \hat{p}_t)^{\gamma}$ mengecil, dan akibatnya $\mathcal{L}_F$ mengecil. Sebaliknya, untuk sampel sulit yaitu sampel dengan $\hat{p}_t$ kecil, $(1 - \hat{p}_t)^{\gamma}$ akan membesar, dan akibatnya $\mathcal{L}_F$ juga membesar.

2.4 Metrik Evaluasi

Metrik evaluasi yang digunakan cukup beragam. Sebagai fokus utama, digunakan metrik evaluasi ***Macro F1-Score***:

$$\text{Macro F1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2\cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}. \tag{13}$$

dimana

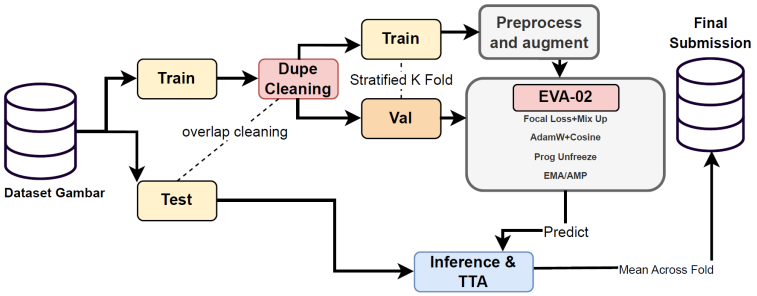
$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \quad \text{dan} \quad \text{Recall}_i = \frac{TP_i}{TP_i + FN_i}. \tag{14}$$

Selain metrik *Macro F1-Score*, kami juga menggunakan metrik ***Accuracy*** sebagai evaluasi tambahan. *Accuracy* diberikan oleh

$$\text{Accuracy} = \frac{\#\{\text{Prediksi benar}\}}{N}. \tag{15}$$

III. Metodologi

3.1 Alur Kerja Penelitian



Gambar 1. *Pipeline* Pelatihan (*Training*)

3.2 Deskripsi Data

Dataset terdiri dari 1752 gambar latih (*train*) dan 444 gambar uji (*test*) yang terbagi ke dalam lima kelas: *Balinese*, *Batak*, *Dayak*, *Javanese*, dan *Minangkabau*.

3.3 *Pre-Processing* dan Augmentasi

3.3.1 Pembersihan Berbasis *pHash* (*Near-Duplicate Removal*)

Kami menerapkan *perceptual hashing* (*pHash*) berukuran 16×16 (256-bit) pada seluruh *image training* untuk menangkap kemiripan visual yang stabil terhadap rotasi/kompresi ringan. Dua gambar dinyatakan *near-duplicate* bila jarak Hamming antar-*hash* memenuhi $d_H \leq 8$. Kami membentuk klaster *near-duplicate* menggunakan struktur *union–find*. Untuk klaster homogen (semua anggota berlabel kelas yang sama), satu gambar dengan area piksel terbesar dipertahankan sebagai perwakilan demi menjaga kualitas visual, sedangkan sisanya dihapus sebagai duplikat dalam-kelas. Pendekatan ini mengurangi redundansi tanpa mengorbankan keragaman bentuk arsitektur yang relevan.

3.3.2 Resolusi Duplikasi Lintas-Kelas

Apabila suatu klaster memuat lebih dari satu label (misalnya sebuah rumah yang sama muncul di folder *Minangkabau* dan *Batak*), kasus tersebut diperlakukan sebagai *cross-class conflict*. Untuk menjaga kemurnian sinyal kelas dan menghindari bias akibat pelabelan ambigu, kami memilih strategi pembersihan konservatif dengan menghapus seluruh anggota klaster konflik (`RESOLVE_MODE = drop_conflicts`). Pendekatan ini lebih aman dibanding *majority relabeling* pada dataset kecil.

3.3.3 Pencegahan Kebocoran *Train–Test*

Selain pembersihan di *train*, kami mengecek tumpang tindih *near-duplicate* antara *train* dan *test* memakai ambang Hamming yang sama. Setiap gambar latih yang berjarak $d_H \leq 8$ terhadap gambar uji dihapus dari *train*. Praktik ini mencegah *leakage* sehingga metrik validasi dan inferensi tidak terinflasi oleh kemiripan visual yang tidak sah.

3.3.4 Contoh Penanganan *Near-Duplicate* dan *Noise*

Perceptual hashing (*pHash*) kami gunakan untuk mendeteksi kemiripan visual; pasangan dengan jarak Hamming $d_H \leq 8$ dianggap *near-duplicate*. Gambar 2 memperlihatkan tiga kasus dan keputusan kurasi: (i) *train-test* ***leakage*** (hapus dari *train*), (ii) **duplikat lintas-kelas ambigu** (hapus keduanya), dan (iii) **duplikat lintas-kelas dengan kelas benar jelas** (hapus yang salah, pertahankan yang benar).

(a) Train–test *near-duplicate* ($d_H \leq 8$) — hapus dari *train*.

(b) Lintas-kelas ambigu — hapus keduanya.

(c) Lintas-kelas jelas — keep yang benar, hapus yang salah.

Gambar 2. Contoh penanganan *near-duplicate/noise* berbasis *pHash*.

3.3.5 Ringkasan Eksekusi Pembersihan

Setelah eksekusi, diperoleh hasil pembersihan: `single_class_rep` sebanyak 1627 (*image* perwakilan yang dipertahankan), `single_class_duplicate` sebanyak 26 (duplikat dalam-kelas yang dihapus), `cross_class_conflict` sebanyak 68 (seluruh anggota klaster lintas-kelas yang dihapus), dan terakhir terdapat `near_duplicate_with_test` sebanyak 31 (*image* latih yang dihapus karena beririsan dengan *test*). Daftar akhir *image* yang digunakan setelah pembersihan disimpan ke `data_clean/train_list.csv` (1627 baris) untuk data *training* dan `data_clean/test_list.csv` (444 baris) untuk data *test*. Seluruh keputusan (`keep/remove`) beserta alasan dan ID klaster terdokumentasi di `data_clean/clean_report.csv`, sehingga proses dapat diaudit ulang secara deterministik.

3.3.6 Penyesuaian Ukuran dan Normalisasi

Pada pelatihan, gambar ditransformasikan ke resolusi 448×448 dengan menggunakan augmentasi `RandomResizedCrop` dengan rentang skala $(0.8, 1.0)$ dan rasio aspek $(0.9, 1.1)$ untuk menambah variasi *framing* tanpa merusak morfologi struktur atap, tiang, dan siluet fasad. Seluruh gambar kemudian dinormalisasi menggunakan statistik *ImageNet*, yaitu

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \quad \boldsymbol{\mu} = [0.485, 0.456, 0.406],\ \boldsymbol{\sigma} = [0.229, 0.224, 0.225].$$

Pada validasi dan inferensi, gambar hanya diubah ukurannya menjadi 448×448 serta dinormalisasi agar distribusi piksel konsisten.

3.3.7 Augmentasi Geometrik dan Fotometrik

Rangkaian augmentasi *Albumentations* yang digunakan bersifat moderat namun efektif: `HorizontalFlip` (p=0.5) dan `RandomRotate90` (p=0.3) menambah variasi orientasi; `ShiftScaleRotate` (batas geser 0.05, skala 0.10, rotasi $15°$, p=0.5) memperkaya komposisi *frame*; `ColorJitter` (kecerahan/kontras/saturasi 0.2, *hue* 0.1, p=0.7) memodelkan perubahan pencahayaan. Kemudian, augmentasi `OneOf{ GaussianBlur, MotionBlur }` (p=0.3) mensimulasikan *defocus*/gerak; dan `CoarseDropout` (hingga 8 lubang berukuran maksimal 24×24 piksel, p=0.4) membuat model tangguh terhadap occlusion kecil. Komposisi ini dirancang agar memperluas keragaman tanpa mengubah ciri bentuk struktural yang membedakan kelas.

3.3.8 Regularisasi Berbasis Label: *MixUp* dan *Label Smoothing*

Selama pelatihan, kami menggunakan *MixUp* dengan parameter awal $\alpha_0 = 0.4$ yang diturunkan linier terhadap *epoch* ($\alpha_t = \alpha_0\,(1 - \frac{t}{T})$) agar intensitas pencampuran mengecil mendekati akhir pelatihan. Untuk sepasang contoh $(\mathbf{x}_i, y_i)$ dan $(\mathbf{x}_j, y_j)$, kami membentuk

$$\lambda \sim \mathrm{Beta}(\alpha_t, \alpha_t), \quad \tilde{\mathbf{x}} = \lambda\,\mathbf{x}_i + (1-\lambda)\,\mathbf{x}_j, \quad \tilde{\mathbf{y}} = \lambda\,\mathbf{y}_i + (1-\lambda)\,\mathbf{y}_j,$$

dan meminimalkan $\mathcal{L} = \lambda\,\ell(f(\tilde{\mathbf{x}}), y_i) + (1-\lambda)\,\ell(f(\tilde{\mathbf{x}}), y_j)$. Selain itu, kami menerapkan *label smoothing* dengan $\varepsilon = 0.1$; target halus untuk kelas k ditulis sebagai

$$y_k^{\mathrm{LS}} = \begin{cases} 1-\varepsilon, & k = y, \\ \varepsilon/(C-1), & k \neq y, \end{cases}$$

sehingga *cross-entropy* menjadi $\mathcal{L}_{\mathrm{LS}} = -\sum_{k=1}^{C} y_k^{\mathrm{LS}} \log p_k$. Kombinasi *MixUp* dan *label smoothing* menekan *overconfidence*, memperhalus batas keputusan, dan membuat model lebih tahan terhadap sisa *label noise* yang mungkin lolos dari pembersihan.

3.3.9 Penanganan Ketidakseimbangan Kelas saat *Sampling*

Ketidakseimbangan kelas ditangani pada tingkat pembentukan *mini-batch* menggunakan `WeightedRandomSampler`. Probabilitas pemilihan sampel kelas c berbanding terbalik dengan frekuensi kelas tersebut, yaitu $w_c \propto \frac{1}{n_c}$. Dengan demikian, kelas minoritas memperoleh eksposur yang memadai tanpa perlu melakukan duplikasi berkas, dan proses ini berjalan selaras dengan *loss* yang peka terhadap ketimpangan.

3.3.10 Augmentasi pada Saat Uji (*Test-Time Augmentation*)

Pada inferensi, kami menggunakan TTA sederhana yang aman, yaitu transformasi `Resize` menjadi ukuran 448×448 diikuti dengan transformasi normalisasi `Normalize`, sesuai dengan sub-subbab 3.3.6.

3.3.11 Reprodusibilitas dan Jejak Eksekusi

Seluruh eksperimen dijalankan dengan pengacakan terkendali (*controlled randomization* dengan `seed` yang tetap) dan organisasi berkas yang konsisten. Himpunan gambar hasil pembersihan ditempatkan pada `data_clean`, sedangkan jejak keputusan disediakan melalui `data_clean/clean_report.csv`. Daftar akhir contoh latih dan uji yang dipakai model masing-masing tersimpan di `data_clean/train_list.csv` dan `data_clean/test_list.csv` untuk memudahkan replikasi dan kolaborasi.

3.4 Metodologi Implementasi dan Pelatihan Model

Setelah dilakukan *pre-processing* pada data dan diperoleh data yang lebih baik digunakan dalam pelatihan, selanjutnya barulah dilakukan pelatihan pada model. Pelatihan model ini mengikuti *pipeline* berikut.

3.4.1 *Cross-Validation* (*Stratified K-Fold*)

Mengingat jumlah data yang terbatas dan distribusi kelas yang tidak seimbang, kami menggunakan `StratifiedKFold` dengan $K = 5$. Strategi ini membagi data latih menjadi 5 *fold* (lipatan), di mana setiap *fold* menjaga proporsi kelas yang sama dengan dataset aslinya. Model dilatih sebanyak 5 kali, di mana pada setiap iterasi, satu *fold* digunakan sebagai data validasi dan 4 *fold* lainnya sebagai data latih. Penggunaan *seed* yang tetap (misal, `seed=42`) memastikan pembagian *fold* ini bersifat *reproducible*.

3.4.2 Arsitektur Model

Model utama yang digunakan adalah EVA-02. Secara khusus, varian dari EVA-02 yang digunakan adalah `eva02_large_patch14_448`, yaitu *backbone* ViT yang sudah melalui *pre-training* MIM pada ImageNet-22K. Untuk mengimplementasikan model ini, digunakan library `timm`.

3.4.3 *Loss Function*, *Optimizer*, dan *Scheduler*

Model dilatih menggunakan *loss function Focal Loss*. *Loss function* ini dipilih karena dataset tidak seimbang (*imbalanced data*) dan *Focal Loss* efektif dalam mengurangi dominasi kelas mayoritas dengan memberikan penalti lebih besar pada sampel yang sulit, sebagaimana dijelaskan pada subbab 2.3.

Selanjutnya, ***optimizer*** yang digunakan adalah `AdamW`, yang umum dipakai untuk arsitektur *Transformer* karena stabil untuk training model besar.

Kemudian, ***scheduler*** yang digunakan adalah ***Cosine Annealing Warm Restarts***. Tujuan digunakannya *scheduler* ini adalah untuk memberikan *warmup* di awal, mengatur *learning rate* secara siklik (berulang), dan membantu model keluar dari jebakan titik minimum lokal.

3.4.4 Prosedur Pelatihan

Untuk setiap *fold*, dataset *training* (`train`) dan validasi (`val`) dibangun menggunakan *transform* yang telah didefinisikan masing-masing. Kemudian, model EVA-02

dibuat dan dipindahkan ke device (GPU). Setelah itu, model dilatih selama iterasi sebanyak *epoch* yang ditetapkan (kami menetapkan penggunaan *epoch* sebanyak 30). Setelah setiap *epoch*, dilakukan evaluasi pada data validasi menggunakan metrik evaluasi *Macro F1-Score*. Terakhir, dari setiap *epoch*, model terbaik pada *fold* tersebut kemudian disimpan sebagai `best_eva02_foldX.pth`. Training berlangsung per *fold* sehingga menghasilkan sebanyak 5 model terpisah.

3.4.5 Inferensia

Inferensia dilakukan pada data *test* menggunakan model terbaik dari setiap *fold*. Untuk melakukan inferensia, model-model terbaik yang sudah disimpan di-*load*, kemudian data *test* diproses dengan *transform* untuk inferensia. Hasilnya adalah probabilitas prediksi untuk setiap gambar.

3.4.6 *Test-Time Augmentation* (TTA)

Untuk meningkatkan generalisasi, inferensia juga dapat dilakukan dengan beberapa transformasi TTA, seperti:

- *horizontal flip*
- *rotate*
- *cutout*
- *scaling*

Setiap gambar diprediksi beberapa kali, hasil outputnya dirata-ratakan menghasilkan prediksi final per *fold*.

Meskipun terdapat opsi ini, kami memutuskan untuk tidak menggunakan TTA sama sekali pada *pipeline*. Hal ini karena augmentasi yang berat pada data *test* kurang cocok untuk dilakukan karena data bersifat *real-world*, sehingga seluruh informasi yang diperlukan bisa saja terdapat dalam data.

3.4.7 *Ensembling* Antar *Fold*

Setelah seluruh fold melakukan inferensia, prediksi akhir dilakukan dengan cara merata-ratakan seluruh output probabilitas dari semua fold (*mean fold ensemble*). Metode ini umumnya dapat meningkatkan stabilitas dan menurunkan variansi prediksi.

Probabilitas-probabilitas ini kemudian digunakan untuk klasifikasi, dan dengan demikian diperoleh hasil prediksi kelas.

IV. Analisis Hasil Eksperimen dan Pengujian

Bab ini memaparkan hasil eksperimen yang dilakukan untuk mengevaluasi kinerja metode klasifikasi gambar rumah adat menggunakan arsitektur EVA-02. Seluruh proses pengujian mengacu pada *pipeline* yang telah dirancang pada Bab 3, dan hasilnya dianalisis untuk menilai efektivitas pendekatan yang diusulkan dalam menyelesaikan permasalahan klasifikasi *multiclass* pada dataset yang *imbalanced*. Penyajian hasil mencakup visualisasi, tabel evaluasi, serta interpretasi kualitatif untuk memperlihatkan secara komprehensif bagaimana model berperilaku selama pelatihan maupun saat melakukan prediksi.

4.1 Skenario Eksperimen dan Metrik Evaluasi

4.1.1 Skenario Pengujian

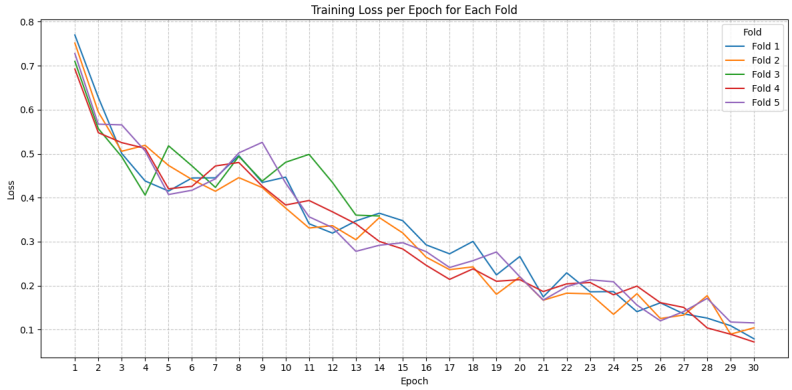
Eksperimen dilakukan secara konsisten pada seluruh *fold*. *Backbone* yang digunakan adalah `eva02_large_patch14_448.mim_in22k_ft_in22k`, dan validasi dilakukan menggunakan *5-Fold Stratified Cross-Validation* agar setiap kelas tetap terdistribusi seimbang pada tiap *fold* meskipun dataset utama tidak seimbang.

Pelatihan berlangsung selama 30 epoch dengan ukuran gambar 448×448 dan *batch size* 14. Strategi ***progressive unfreezing*** diterapkan: pada *epoch* awal, hanya *classification head* yang di-*freeze*, lalu mulai dari *epoch* 8 *backbone* diaktifkan agar dapat beradaptasi dengan data tanpa menyebabkan *catastrophic forgetting*. Semua skenario ini dijaga konsisten agar hasil evaluasi stabil dan dapat dibandingkan.

4.1.2 Metrik Evaluasi

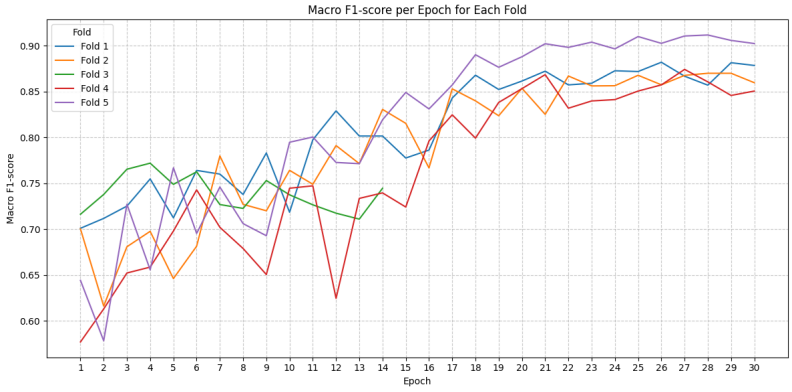
Metrik utama yang digunakan adalah *Macro F1-Score*, yaitu rata-rata *F1-Score* dari seluruh kelas tanpa memberi bobot berdasarkan jumlah sampel. Metrik ini dipilih karena dataset bersifat *imbalanced*, sehingga akurasi saja tidak cukup informatif. Dengan *Macro F1*, performa pada kelas minoritas tetap memiliki peran yang sama dalam penilaian, sehingga evaluasi menjadi lebih adil dan mencerminkan kemampuan model secara keseluruhan.

4.2 Analisis Hasil Proses Pelatihan



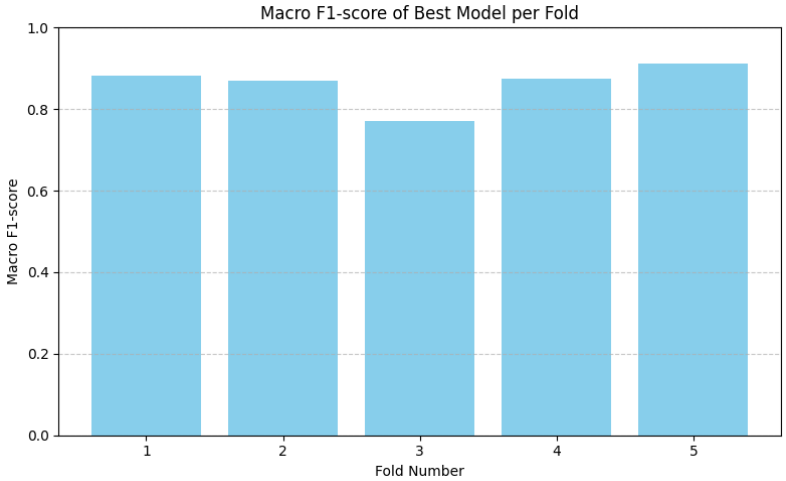
Gambar 3. Grafik Perbandingan *Loss* Terhadap *Epoch* untuk Setiap *Fold*

Gambar 3 menunjukkan pola penurunan yang konsisten, menandakan bahwa model mampu beradaptasi terhadap distribusi data latih. Walau semua *fold* mencapai konvergensi, terlihat adanya variasi. Misalnya, salah satu fold berhenti lebih awal karena *early stopping*, yang mengindikasikan proses belajar yang lebih cepat pada subset data tertentu.



Gambar 4. Grafik Perbandingan *Macro F1-Score* Terhadap *Epoch* untuk Setiap *Fold*

Gambar 4 memberikan gambaran yang lebih jelas mengenai performa pada data validasi. Sebagian besar *fold* menunjukkan tren peningkatan *F1-Score* secara stabil, meskipun beberapa kasus seperti *Fold* 3 mengalami *plateau* lebih cepat atau memiliki nilai akhir yang lebih rendah. Sebaliknya, *Fold* 5 menunjukkan performa konsisten tinggi, merefleksikan sensitivitas model terhadap variasi distribusi data per *fold*.



Gambar 5. Plot *Macro F1-Score* dari Model Terbaik untuk Setiap *Fold*

4.3 Analisis Hasil Evaluasi Model

4.3.1 Performa Keseluruhan

Secara agregat, model menghasilkan *Average Macro F1-Score* = 0.8619. Nilai ini menunjukkan performa yang cukup baik untuk dataset *multiclass* dengan tingkat kemiripan visual yang tinggi. Gambar 5 memperlihatkan variasi kecil antar *fold* (terutama di *fold* 3), namun secara umum model stabil dan tidak menunjukkan indikasi *overfitting* yang ekstrem.

4.3.2 Analisis Klasifikasi

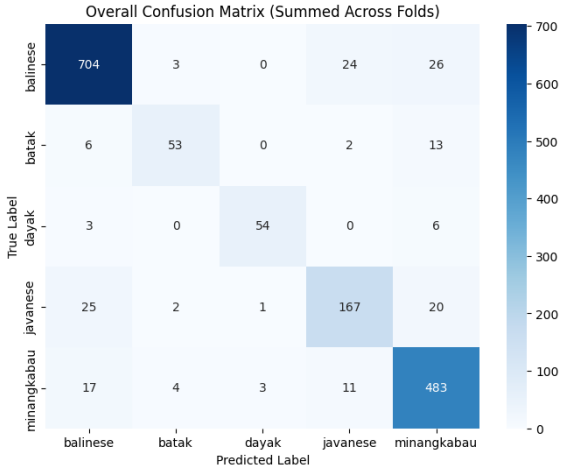
class	precision	recall	f1_score
balinese	0.932	0.932	0.932
batak	0.868	0.718	0.782
dayak	0.934	0.854	0.892
javanese	0.828	0.774	0.798
minangkabau	0.888	0.930	0.906

Gambar 6. *Precision*, *Recall*, dan *F1-Score* untuk Model Terbaik Setiap *Fold*

Gambar 6 menunjukkan bahwa kelas seperti *balinese* dan *minangkabau* memiliki *precision* dan *recall* yang relatif tinggi dan konsisten. Sebaliknya, kelas seperti *batak* dan *javanese* lebih sering mengalami penurunan *F1-score* pada beberapa *fold*, menandakan model lebih sering keliru pada kedua kelas ini. Biasanya, ini terjadi

karena kemiripan elemen arsitektural atau kualitas gambar yang lebih variatif.

4.3.3 Analisis *Confusion Matrix*



Gambar 7. *Confusion Matrix* Gabungan untuk Seluruh *Fold*

Confusion Matrix gabungan dari kelima *fold* pada Gambar 7 memberi gambaran menyeluruh mengenai pola kesalahan model. Diagonal matriks yang kuat pada kelas *balinese* dan *minangkabau* menegaskan kemampuan model mengenali keduanya dengan baik. Pada sisi lain, kesalahan terbesar umumnya terjadi antara *javanese* dan *balinese*, atau pada sebagian sampel *batak*. Hal ini menunjukkan adanya tumpang tindih fitur visual yang membuat model ragu ketika membedakan kedua kelas tersebut. Keraguan ini dapat berasal dari banyak hal, dimulai dari *mislabeling*, sudut yang tidak umum, pencahayaan buruk, atau elemen non-arsitektural. Selain itu, untuk beberapa kelas, karakteristik bangunan yang mirip juga memicu kesalahan. Misalnya, bentuk atap tertentu yang muncul di lebih dari satu budaya.

4.4 Perbandingan dengan Model Lainnya

Tabel 1 di halaman selanjutnya adalah tabel yang membandingkan hasil implementasi model EVA-02 dengan *pipeline training* dan *pre-processing* yang sesuai dengan pembahasan.

Nama Model	Pre-Processing	Public Macro F1	Private Macro F1
EfficientNet Baseline	*False*	0.52312	0.48821
EfficientNet + Transform	*False*	0.74703	0.73107
ConvNeXt Baseline	*False*	0.77987	0.74005
ConvNeXt - Augmented + LDAM	*False*	0.82031	0.82586
Swin-Transformer Baseline	*False*	0.77482	0.69325
EVA-02 Baseline	*False*	0.86177	0.89046
EVA-02 + Pipeline	*False*	0.87174	0.90179
EVA-02 + Pipeline + Pre-Processed	**True**	**0.92937**	**0.98653**

Table 1. Perbandingan Performa Berbagai Model

V. Penutup

5.1 Kesimpulan

Percobaan ini telah berhasil mengembangkan dan mengimplementasikan sistem klasifikasi otomatis untuk lima kategori rumah adat di Indonesia menggunakan arsitektur EVA-02. Model ini terbukti efektif dalam menangani dataset "Klasifikasi Rumah Adat Nusantara" dari DSC LOGIKA UI 2025, yang memiliki tantangan berupa variasi visual, oklusi, dan ketidakseimbangan kelas.

Implementasi strategi yang komprehensif, mulai dari pembersihan data *near-duplicate* berbasis `pHash`, penggunaan *Focal Loss* untuk mengatasi *class imbalance*, dan teknik regularisasi seperti *MixUp* dan *Label Smoothing*, berkontribusi signifikan terhadap robustisitas model.

Dengan menggunakan metodologi *5-Fold Stratified Cross-Validation* dan *mean fold ensembling* pada saat inferensia, *pipeline training* berhasil mencapai *Average Macro F1-Score* sebesar 0.8619. Analisis hasil menunjukkan bahwa model memiliki performa sangat baik pada kelas *balinese* (F1 0.932) dan *minangkabau* (F1 0.906). Namun, tantangan utama yang teridentifikasi adalah adanya kebingungan model (kesalahan klasifikasi) antara kelas yang memiliki kemiripan visual tinggi dan jumlah lebih rendah, terutama antara kelas *javanese* dan *balinese*.

5.2 Rekomendasi

Berdasarkan analisis hasil, berikut adalah rekomendasi untuk pengembangan selanjutnya:

- Mengeksplorasi teknik ***Fine-Grained Visual Classification*** atau **FGVC** untuk meningkatkan kemampuan model dalam membedakan kelas-kelas dengan kemiripan visual tinggi, seperti *javanese* dan *balinese*, dengan fokus pada detail arsitektural yang sulit diidentifikasi.

- Melakukan verifikasi label secara manual untuk mengatasi potensi *label noise* dan ambiguitas data. Proses ini dapat memvalidasi ulang temuan *cross_class_conflict* dan meningkatkan kejelasan sinyal latih antar kelas.

- Mengeksplorasi implementasi *Test-Time Augmentation* (TTA) yang lebih kompleks. *Pipeline* saat ini tidak menggunakan TTA pada inferensi, sehingga penerapan TTA yang tepat berpotensi meningkatkan stabilitas dan robustisitas prediksi akhir.

DAFTAR PUSTAKA

[1] M. Fiorucci, M. Khoroshiltseva, and M. Pontil, "Machine learning for cultural heritage: A survey," *Pattern Recognition Letters*, vol. 133, pp. 102–108, 2020.

[2] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.

[3] Y. Fang et al., "EVA-02: A visual representation for Neon Genesis," *arXiv preprint* arXiv:2303.11331, 2023.

[4] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, Dive Into Deep Learning. Cambridge University Press, 2023.