Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

Terbit : 12 Januari 2025

Paradigma Klasifikasi Ragam Seni Lukis Berbasis *Convolutional Neural Network* (CNN) Dengan MobileNetV2 Dan Implementasi Pada Postman Melalui Flask Api

[1]Ratu Nurmalika, [2*] Makmun, [3]Bambang Yulianto, [4]Ichsani Mursidah, [5]Dhian Sweetania, [6]Puji Sularsih
[1]Prodi Teknik Sipil, Fakultas Teknik Sipil dan Perencanaan,
Universitas Gunadarma, Jakarta, Indonesia
[2,4,5,6] Prodi Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi,
Universitas Gunadarma, Jakarta, Indonesia
[3]Prodi Teknik Industri, Fakultas Teknik Industri,
Universitas Gunadarma, Jakarta, Indonesia

[1]nurmalikaratu@staff.gunadarma.ac.id, [2*]makmun@staff.gunadarma.ac.id,
[3]bambang_yulianto@staff.gunadarma.ac.id, [4]mursidah@staff.gunadarma.ac.id,
[5]dhian_sweetania@staff.gunadarma.ac.id, [6]puji@staff.gunadarma.ac.id

*Korespondensi: makmun@staff.gunadarma.ac.id

ABSTRAK

Tujuan dari penelitian ini adalah untuk membuat model klasifikasi *genre* seni lukis menggunakan *Convolutional Neural Network* (CNN) dengan arsitektur MobileNetV2. MobileNetV2 dipilih karena kemampuannya untuk bekerja dengan baik meskipun digunakan pada perangkat dengan daya komputasi terbatas. Data yang digunakan dalam penelitian ini mencakup berbagai genre seni lukis yang didapatkan secara gratis melalui situs Kaggle. yang kemudian melakukan *Data Preprocessing dan Augmentation*. Setelah model dilatih, langkah implementasi dilakukan menggunakan framework web Flask, yang berbasis Python. Ini memungkinkan API untuk diakses melalui Postman. API ini memungkinkan pengguna mengunggah karya seni dan menerima prediksi genre sebagai tanggapan. Hasil penelitian menunjukkan bahwa model yang dikembangkan dapat mengklasifikasikan genre seni lukis dengan akurasi 82% dan kehilangan 0.4, dan bahwa *Application Program Interface* (API) yang dibangun dapat berfungsi dengan baik untuk menyediakan layanan prediksi. Diharapkan bahwa penerapan ini akan memberikan kontribusi yang signifikan dalam bidang analisis seni dan aplikasi teknologi dalam seni lukis, serta memberikan alat yang bermanfaat bagi kurator, seniman, dan peneliti seni.

Kata Kunci: *Convolutional Neural Network* (CNN), *Deep Learning*, Klasifikasi Seni Lukis, MobileNetV2, TensorFlow

PENDAHULUAN

Berbagai bidang, termasuk seni lukis, telah mengalami perubahan besar sebagai akibat dari kemajuan teknologi yang terjadi di era komputer dan internet saat ini. Seni lukis adalah salah satu cara ekspresi manusia yang telah ada sejak zaman prasejarah, dan saat ini dapat dikombinasikan dengan teknologi kontemporer untuk menghasilkan

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

karya-karya yang lebih beragam dan inovatif. Pengenalan citra (*image recognition*) melalui Machine Learning (ML), terutama model *Convolutional Neural Network* (CNN), adalah salah satu teknologi yang dapat digunakan.

Klasifikasi gambar merupakan salah satu aplikasi dari pembelajaran mesin yang memiliki potensi besar dalam bidang seni lukis. Salah satu metode yang terbukti efektif dalam klasifikasi gambar adalah penggunaan CNN. MobileNetV2, sebagai salah satu arsitektur CNN yang ringan dan efisien, sangat cocok digunakan dalam pengembangan model klasifikasi gambar seni lukis. MobileNetV2 dirancang untuk perangkat dengan daya komputasi terbatas, namun tetap mempertahankan akurasi yang tinggi dalam pengenalan gambar.

Untuk melihat hasil dari prediksi yang dibuat oleh model, penelitian ini terbatas pada integrasi model ke antarmuka. Tujuan dari penelitian ini adalah untuk membuat model ML yang dapat diterapkan yang dapat memprediksi klasifikasi genre seni lukis dengan tingkat akurasi lebih dari 80%. Tujuannya adalah untuk membuat model ini dapat dengan mudah membedakan dan memberikan prediksi klasifikasi genre untuk setiap gambar seni lukis yang diberikan.

TINJAUAN PUSTAKA

Artificial Intelligence

Artificial intelligence (AI) merupakan sebuah disiplin ilmu komputer yang memiliki tujuan untuk mengembangkan sistem-sistem yang pada umumnya memerlukan sebuah bentuk supervisi dari manusia, yang dimana kecerdasan dan rasionalitas manusia tidak dapat digantikan. AI mencakup berbagai aplikasi yang luas, mulai dari pemrosesan bahasa alami, pengenalan gambar, pengambilan keputusan, hingga permainan strategi (Russell & Norvig, 2016).

Machine Learning

Machine Learning (ML) merupakan cabang atau tingkatan yang lebih spesifik dari AI yang hanya berfokus pada pengembangan algoritma yang dapat memungkinkan sebuah komputer untuk belajar dari dan membuat sebuah prediksi berdasarkan data-data yang diberikan kepadanya. ML memanfaatkan metode-metode statistik untuk menemukan pola yang umum dalam data dan melakukan generalisasi temuan ini pada data yang baru. Metode-metode ini termasuk regresi, klasifikasi, klastering, dan metode lainnya yang digunakan untuk menganalisis data (Goodfellow, Bengio, & Courville, 2016).

Deep Learning

Cabang turunan dari AI yang lebih berfokus pada penggunaan jaringan saraf tiruan berlapis-lapis atau *deep neural networks* untuk memodelkan dan memahami data yang diberikan dengan sebuah representasi yang sangat kompleks dalam bentuk bobot. Teknik deep learning berhasil menghasilkan beberapa kemajuan signifikan dalam berbagai aplikasi seperti pengenalan gambar, pemrosesan bahasa alami, dan pengenalan suara yang secara otomatis mengekstraksi fitur dari data mentah, membuatnya sangat efektif untuk tugas-tugas yang membutuhkan analisis data yang kompleks dan non-linear. Dalam *deep learning*, konsep penggunaan data *training* dan data validation memiliki kesamaan dengan machine learning secara umum, tetapi dengan beberapa perbedaan yang mencolok karena kompleksitas dan kedalaman model yang digunakan.

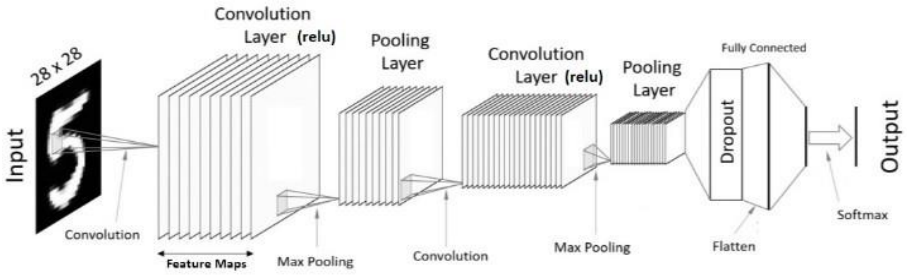
Data training dalam *deep learning* adalah subset dari dataset yang digunakan untuk melatih jaringan saraf dalam, di mana model yang terdiri dari banyak lapisan (*layer*) belajar

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v13i2.14457 p-ISSN : 2089-9424

untuk mengenali pola dan fitur dari data input. Setelah proses pelatihan, data validation digunakan untuk mengevaluasi kinerja model. Data ini tidak digunakan selama pelatihan dan berfungsi sebagai pengukur independen untuk menguji kemampuan model dalam menggeneralisasi data baru. Dalam *deep learning*, teknik seperti *early stopping*, *regularization*, dan *dropout* sering diterapkan untuk mengatasi *overfitting* selama pelatihan model *deep learning*.

***Convolutional Neural Network* (CNN)**

CNN adalah jenis jaringan saraf tiruan yang dirancang untuk memproses data dalam bentuk grid, seperti citra. CNN menggunakan lapisan konvolusi untuk mengekstraksi fitur penting dari gambar, seperti tepi, tekstur, dan pola. Teknik ini sangat efektif untuk tugas-tugas pengenalan gambar dan telah digunakan secara luas dalam berbagai aplikasi, termasuk seni lukis (Krizhevsky et al., 2012).



Gambar 1. Convolutional layer

TensorFlow.

Kegunaan *TensorFlow* Dalam *Deep Learning*; 1) Pelatihan Model, *TensorFlow* digunakan untuk membangun dan melatih berbagai jenis model pembelajaran mesin, dari regresi linear sederhana hingga jaringan saraf dalam (*deep neural networks*). 2) Pemrosesan Gambar: *TensorFlow* sering digunakan dalam tugas *computer vision* seperti klasifikasi gambar, deteksi objek, dan segmentasi gambar. 3) Pemrosesan Bahasa Alami (NLP): *TensorFlow* digunakan untuk tugas NLP seperti analisis sentimen, penerjemahan mesin, dan chatbot. 4) Sistem Rekomendasi: *TensorFlow* membantu dalam pengembangan sistem rekomendasi yang mempersonalisasi konten bagi pengguna berdasarkan preferensi dan perilaku mereka.

METODE PENELITIAN

Pada metode penelitian ini dilakukan tahapan yang dapat dijelaskan sebagai berikut:

1. Pengumpulan Data
   a. Mengunduh Dataset: Dataset seni lukis diunduh dari *web* Kaggle.
   b. Menyimpan Dataset: Dataset yang telah diunduh disimpan di Google Drive untuk memudahkan akses dari Google Colab.
2. Preprocessing Data
   a. Mengakses Dataset: Dataset diambil dari Google Drive menggunakan Google Colab dan dimasukkan ke dalam variabel untuk pemrosesan lebih lanjut.
   b. Data Augmentation: Melakukan augmentasi data untuk memperbanyak dan memperkaya variasi data latih, seperti rotasi, flipping, dan zooming.
   c. Preprocessing: Melakukan normalisasi gambar agar memiliki nilai antara 0 dan 1.
3. Pelatihan Model
   a. Arsitektur Model: Menggunakan MobileNetV2 sebagai arsitektur dasar, dengan tambahan lapisan sebagai berikut:
      1) GlobalAveragePooling2D**:** Untuk mereduksi dimensi data.

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v13i2.14457 p-ISSN : 2089-9424

2) Flatten**:** Untuk meratakan output dari lapisan sebelumnya.
3) Dense: Dengan 128 neuron dan aktivasi ReLU untuk lapisan fully connected.
4) Dropout: Dengan nilai 0.2 untuk mencegah overfitting.
5) Dense: Lapisan output dengan 7 neuron dan aktivasi softmax untuk klasifikasi.

b. Melatih Model: Model dilatih menggunakan data yang telah diproses dengan konfigurasi hyperparameter berikut:
1) Loss Function: categorical_crossentropy
2) Optimizer: Adam dengan learning rate 0.001
3) Metrics: accuracy
4) Steps per Epoch: train_generator.samples // batch_size
5) Epochs: 1000
6) Validation Data: validation_generator
7) Validation Steps: validation_generator.samples // batch_size
8) Callbacks: callback

9) Evaluasi Model
a. *Stop Loss Function*: Menggunakan fungsi stop loss untuk memberhentikan pelatihan saat model mencapai akurasi tertentu atau ketika loss tidak mengalami perbaikan.
b. Visualisasi Performa: Menggunakan Pandas dan Matplotlib untuk membuat grafik yang menunjukkan perkembangan akurasi dan loss selama proses pelatihan.

HASIL DAN PEMBAHASAN

1. Pengumpulan Data (*Data Collection*)

Proses pertama yang dilakukan dalam merancang sebuah model deep learning adalah menemukan sebuah dataset yang dapat digunakan sesuai dengan kebutuhan. Tempat yang populer di kalangan data science untuk keperluan dataset salah satunya adalah Kaggle. Di dalam model, akan digunakan 2 sumber data untuk gambar seni lukisan yang dapat digunakan, dimana keduanya dapat digunakan secara bebas. Sumber pertama merupakan sebuah dataset komprehensif yang berisikan tentang sejumlah genre dari seni lukis dengan nama WikiArt Art Movements/Styles yang diciptakan oleh Sivar Azadi, seperti yang dapat dilihat pada tabel 1. Dataset pertama secara keseluruhan memiliki ukuran secara total sebanyak 29 Gigabyte.

Tabel 1. Dataset WikiArt Art Movements/Styles

Judul Genre	Jumlah Data
Academic Art	1305 Gambar
Art Nouveau	3035 Gambar
Baroque	5312 Gambar
Expressionism	2607 Gambar
Japanese Art	2235 Gambar
Neoclassicsm	3115 Gambar
Primitivsm	1324 Gambar
Realism	5373 Gambar
Renaissance	6192 Gambar
Rococo	2521 Gambar
Judul Genre	Jumlah Data
Romanticsm	6813 Gambar
Symbolism	1510 Gambar
Western Medieval	1158 Gambar

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

Sumber dataset yang kedua adalah dataset gambar seni lukis yang diciptakan oleh pengguna bernama Bryanb, yang dimana dataset tersebut bernama Abstract Art Gallery. Dataset ini memiliki ukuran sebesar 727 Megabyte. Sama dengan dataset pertama, dataset ini bebas untuk digunakan bagi pengguna yang mengunduhnya. Dataset ini hanya berisikan satu jenis genre yang dimuatnya yaitu Abstract sesuai dengan namanya, namun terdapat 2 direktori terpisah yang dimilikinya dengan jumlah data yang berbeda.

Tabel 2. Dataset Abstract Art Gallery

Judul Genre	Direktori	Jumlah Data
Abstract	1	2872 Gambar
	2	90 Gambar

Data-data yang telah dikumpulkan kemudian disatukan dan dimasukkan kedalam direktori yang telah dibuat di dalam Google Drive agar dapat dengan mudah melakukan integrasi dengan Tensorflow. Dataset yang telah terkumpul berjumlah 45.462 gambar yang tersebar pada 14 jenis *genre* yang berbeda-beda.

Tabel 3. Keseluruhan dataset dalam Google Drive

Judul Genre	Jumlah Data
Abstract	2962 Gambar
Academic Art	1305 Gambar
Art Nouveau	3035 Gambar
Baroque	5312 Gambar
Expressionism	2607 Gambar
Japanese Art	2235 Gambar
Neoclassicsm	3115 Gambar
Primitivsm	1324 Gambar
Realism	5373 Gambar
Renaissance	6192 Gambar
Rococo	2521 Gambar
Romanticsm	6813 Gambar
Symbolism	1510 Gambar
Western Medieval	1158 Gambar

Jika ingin menggunakan seluruh data yang terdapat pada Google Drive ini maka dari sini dapat melanjutkan ke dalam tahap selanjutnya, namun penulis memiliki sumber daya komputer yang terbatas sehingga memutuskan untuk hanya menggunakan 7 jenis genre. Dataset yang digunakan sesuai dengan tabel 4.

Tabel 4. Dataset jenis genre yang digunakan dalam Google Drive

Judul Genre	Jumlah Data
Abstract	2962 Gambar
Expressionism	2607 Gambar
Neoclassicsm	3115 Gambar
Primitivsm	1324 Gambar
Realism	5373 Gambar
Romanticsm	6813 Gambar
Symbolism	1510 Gambar

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

2. Pra-pemrosesan Data (*Data Preprocessing*)

Semua proses dari *preprocessing* dilakukan di dalam Google Colab dan data yang telah disimpan di dalam Google Drive dapat ditarik dari dalam Google Colab dengan menggunakan library Python yang bernama Drive.

```
from google.colab import drive
drive.mount('/content/drive')
```

Gambar 2. Library *drive* dan *mounting*

Dengan *library drive*, memungkinkan untuk melakukan proses yang dinamakan "*mounting*" yang dimana akun Google yang terhubung dengan *Google Colab* juga akan menghubungkan dan membuka akses dengan *Google Drive* yang dimiliki oleh akun tersebut, sehingga memungkinkan untuk melakukan perubahan seperti menghapus, menambahkan, dan mengubah file yang terdapat didalam *Google Drive* pemilik akun tersebut melalui Google Colab. Setelah terhubung, barulah pemrosesan data atau data preprocessing dapat dikerjakan. Jika dilihat dalam tabel 4, database yang termuat berjumlah tidak rata, beberapa database memiliki jumlah yang lebih sedikit dari yang lainnya. Maka perlu disamaratakan jumlah data yang digunakan pada setiap folder. Penulis memutuskan bahwa 1000 data per jenis genre merupakan jumlah yang mumpuni untuk melakukan training. Hal tersebut dapat dilakukan di dalam Google Colab dengan script otomatis seperti pada gambar 3 karena telah terhubung pada Google Drive.

```
import os
import shutil

# Define base directory
base_dir = '/content/drive/MyDrive/Dataset/DatasetV2/Dataset/7Class_Split1000/Training'

# List of categories
categories = [
    'Abstract', 'Expressionism','Neoclassicism', 'Primitivism', 'Realism','Romanticism','Symbolism'
]

# Define the number of images to select from each category
num_images_per_category = 1000

# Directory to store the sampled images
sampled_dir = '/content/drive/MyDrive/Dataset/DatasetV2/WikiArt Art Movements Styles/Sampled_Arts_1000'
os.makedirs(sampled_dir, exist_ok=True)

# Loop through categories and select 1000 images from each
for category in categories:
    # Construct the path for each category
    category_path = os.path.join(base_dir, category)

    # Check if the path exists to avoid errors
    if os.path.exists(category_path):
        # Create a directory for the sampled images in the sampled directory
        sampled_category_dir = os.path.join(sampled_dir, category)
        os.makedirs(sampled_category_dir, exist_ok=True)

        # List all files in the category directory
        all_files = os.listdir(category_path)

        # Select the first 1000 files (images)
        selected_files = all_files[:num_images_per_category]

        # Copy the selected files to the sampled category directory
        for file_name in selected_files:
            source_file = os.path.join(category_path, file_name)
            destination_file = os.path.join(sampled_category_dir, file_name)
            shutil.copy(source_file, destination_file)

        print(f"Selected 1000 images from {category}.")
    else:
        print(f"Category path does not exist: {category_path}")
```

Gambar 3. Kode untuk membagi gambar sesuai jumlah tertentu

Kode ini bertujuan untuk menyalin 1000 gambar dari setiap kategori seni dalam dataset ke direktori baru untuk penggunaan lebih lanjut. Pertama, kode mendefinisikan direktori dasar di mana dataset asli berada dan membuat daftar kategori seni yang akan diproses. Selanjutnya, ditentukan bahwa 1000 gambar akan disalin dari masing-masing kategori. Kode kemudian membuat direktori tujuan untuk menyimpan gambar yang telah disalin. Melalui sebuah loop, kode ini memeriksa keberadaan direktori untuk setiap kategori, membuat direktori tujuan untuk gambar yang disalin jika belum ada, dan mengumpulkan semua file dalam direktori kategori tersebut. Selanjutnya, kode memilih

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

1000 file pertama dari daftar dan menyalinnya ke direktori tujuan. Proses ini diulangi untuk setiap kategori, dan jika direktori kategori tidak ada, pesan kesalahan akan dicetak. Dijalankannya kode tersebut akan memakan waktu dan hasil yang didapatkan adalah penambahan direktori pada Google Drive yang berisikan direktori jenis genre dari data gambar seni lukis yang masing masing berisi 1000 gambar seperti pada tabel 5.

Tabel 5. Dataset yang telah disamaratakan

Judul Genre	Jumlah Data
Abstract	1000 Gambar
Expressionism	1000 Gambar
Neoclassicsm	1000 Gambar
Primitivsm	1000 Gambar
Realism	1000 Gambar
Romanticsm	1000 Gambar
Symbolism	1000 Gambar

Data Augmentasi dan Normalisasi, direktori dataset training dan validasi yang telah dibuat dan dimasukkan kedalam variabel untuk dipanggil maka model dapat menerima data tersebut sebagai bahan untuk training. Namun jika model menerima dataset dengan mentah maka hasil dari model tidak akan maksimal dan akan memakan waktu training yang lama, untuk itu harus dilakukan Data Augmentation dan Normalisasi seperti pada gambar 4.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the paths for training and validation directories
train_dir = '/content/drive/MyDrive/Dataset/DatasetV2/Dataset/7Class_Split1000/Training'
val_dir = '/content/drive/MyDrive/Dataset/DatasetV2/Dataset/7Class_Split1000/Validation'

# Define image dimensions and batch size
img_width, img_height = 224, 224
batch_size = 32

# Data augmentation configuration for training
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Data augmentation for validation (only rescaling)
val_datagen = ImageDataGenerator(rescale=1./255)

# Training data generator
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)

# Validation data generator
validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
```

Gambar 4. Kode untuk Data Augmentation dan Normalisasi

Kode tersebut bertujuan untuk menyiapkan generator data gambar dengan augmentasi untuk pelatihan dan validasi model menggunakan TensorFlow dan Keras. Pertama, jalur direktori untuk data pelatihan dan validasi didefinisikan, diikuti dengan menetapkan dimensi gambar dan ukuran batch. Konfigurasi augmentasi data untuk pelatihan dilakukan dengan menggunakan ImageDataGenerator, yang mencakup operasi seperti penskalaan ulang nilai piksel gambar, rotasi, pergeseran lebar dan tinggi, pemotongan, zoom, dan flipping horizontal untuk meningkatkan keragaman data pelatihan. Sebaliknya, augmentasi data untuk validasi hanya melibatkan penskalaan

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

ulang nilai piksel gambar. Selanjutnya, generator data untuk pelatihan dibuat menggunakan metode flow_from_directory, yang mengarahkan generator ke direktori pelatihan, menetapkan ukuran target gambar, ukuran batch, dan mode kelas sebagai categorical. Hal yang sama dilakukan untuk data validasi. Generator ini akan menghasilkan batch gambar yang di-augmentasi selama proses pelatihan dan evaluasi model, membantu meningkatkan generalisasi model dengan menyediakan data yang lebih bervariasi.

3. Pelatihan Model

Model yang digunakan dalam penelitian ini merupakan sebuah model yang memanfaatkan model raksasa yang telah dikembangkan oleh Google bernama MobileNetV2, yang dimana model tersebut memiliki lapisan yang besar pada teknik CNN. Dengan alasan tersebut MobileNetV2 merupakan model bantuan yang tepat untuk pengenalan citra. MobileNetV2 tersedia dalam bentuk library, sehingga yang harus dilakukan adalah memanggil library tersebut untuk menggunakannya. Dalam model penulis MobileNetV2 dijadikan sebagai lapisan dasar atau base layer yang dapat dilakukan dengan cara seperti pada gambar 5.

```
# Load MobileNetV2 as base model
base_model = tf.keras.applications.MobileNetV2(input_shape=(img_width, img_height, 3),
                                               include_top=False,
                                               weights='imagenet')
base_model.trainable = False  # Freeze the base model
```

Gambar 5. Kode untuk MobileNetV2

```
# Define the model
model = tf.keras.Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')  # Final output layer with 2 classes
])
```

Gambar 6. Kode model deep learning

Kode tersebut menerapkan pendekatan transfer learning dengan menggunakan MobileNetV2 sebagai model dasar. MobileNetV2 telah dilatih terlebih dahulu pada dataset ImageNet, sehingga mampu mengenali pola visual umum. Saat kode dijalankan, base_model MobileNetV2 diinisialisasi dengan bobot yang sudah terlatih dan kemudian dibekukan, sehingga tidak mengalami penyesuaian bobot selama pelatihan model selanjutnya.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Func  (None, 7, 7, 1280)        2257984
 tional)

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 flatten (Flatten)           (None, 1280)              0

 dense (Dense)               (None, 128)               163968

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 7)                 903

=================================================================
Total params: 2422855 (9.24 MB)
Trainable params: 164871 (644.03 KB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

Gambar 7. Model Klasifikasi

Layer-layer tambahan ditambahkan secara berurutan di atas base_model untuk menyesuaikan model sesuai kebutuhan. GlobalAveragePooling2D digunakan untuk mereduksi dimensi spasial keluaran base_model menjadi satu vektor rata-rata global. Selanjutnya, vektor tersebut diratakan menjadi satu dimensi menggunakan Flatten.

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v13i2.14457 p-ISSN : 2089-9424

Setelah itu, ada dua *layer Dense* tambahan: Dense(128, activation='relu') untuk mengekstraksi fitur lebih lanjut dengan 128 unit dan aktivasi ReLU, yang diikuti oleh Dropout(0.2) untuk mengurangi risiko *overfitting* dengan cara secara acak menghilangkan 20% unit selama pelatihan. Terakhir, layer Dense dengan 7 unit dan aktivasi softmax digunakan sebagai layer *output*, yang menghasilkan distribusi probabilitas untuk 7 kelas yang berbeda. Model ini siap untuk dilatih dengan data yang sesuai dan dievaluasi untuk tugas klasifikasi gambar dengan 7 kelas target. Loss dan Optimisasi, model yang telah dibuat kemudian dilakukan kompilasi dengan fungsi loss dan optimisasi. Hal tersebut dapat dilakukan dengan menggunakan kode yang tertera pada gambar 8.

```
# Compile the model
model.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(learning_rate=0.001),
    metrics=['accuracy']
)
```

Gambar 8. Kode kompilasi Loss dan Optimisasi

Kode tersebut bertujuan untuk mengkompilasi model yang telah dibangun sebelumnya. Pada tahap ini, model dikonfigurasi untuk proses pelatihan dengan spesifikasi; *Loss Function*: Menggunakan categorical crossentropy sebagai fungsi kerugian (*loss function*). Optimizer: Menggunakan Adam optimizer dengan *learning rate* sebesar 0.001. *Metrics*: Model akan dievaluasi selama pelatihan berdasarkan metrik akurasi (*accuracy*), yang mengukur seberapa sering prediksi model cocok dengan label yang sebenarnya.

Training, model yang telah dibuat dan dataset yang telah di bagi memungkinkan pelatihan model dengan cara menggunakan kode seperti pada gambar 9.

```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=1000,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size,
    callbacks=[callback]
)
```

Gambar 9. Kode menjalankan training model

Gambar 9 menunjukan bahwa Kode tersebut adalah penggunaan metode fit() pada objek model untuk melatih model deep learning. Berikut adalah penjelasan langkah demi langkah dari kode tersebut:

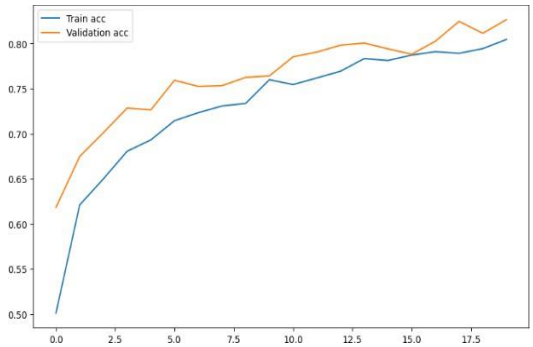
a. train_generator: Merupakan generator atau objek iterator yang digunakan untuk memuat data pelatihan (train data). Generator ini menghasilkan batch-batch data secara bertahap selama proses pelatihan.
b. steps_per_epoch: Jumlah langkah yang akan dijalankan pada setiap epoch. Nilai ini dihitung sebagai jumlah sampel data pelatihan (train_generator.samples) dibagi dengan ukuran batch (batch_size).
c. epochs: Jumlah epoch atau siklus pelatihan yang akan dieksekusi. Dalam contoh tersebut, pelatihan akan berlangsung selama 1000 epoch.
d. validation_data: Generator atau objek iterator yang digunakan untuk memuat data validasi. Data ini digunakan untuk mengevaluasi model setelah setiap epoch pelatihan selesai.

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

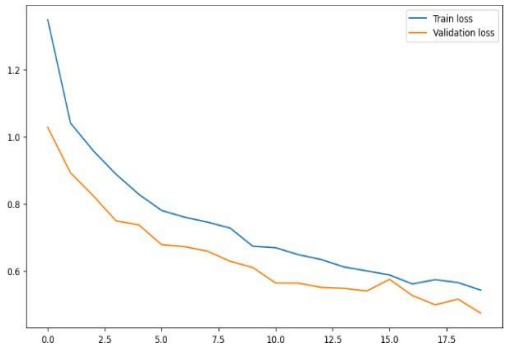
e. validation_steps: Jumlah langkah yang akan dijalankan pada setiap epoch validasi. Nilai ini dihitung sebagai jumlah sampel data validasi (validation_generator.samples) dibagi dengan ukuran batch (batch_size).
f. callbacks: Sebuah list yang berisi objek callback yang akan dipanggil selama pelatihan model. Dalam kode tersebut, *callback* yang digunakan adalah *AccuracyThresholdCallback* yang telah didefinisikan sebelumnya. *Callback* ini bertujuan untuk memberhentikan pelatihan jika akurasi baik pada data pelatihan maupun validasi telah mencapai atau melebihi 80%.

4. Evaluasi Model

Hasil yang didapatkan merupakan sebuah model yang dapat melakukan klasifikasi gambar seni lukis berdasarkan jenis genrenya. Akurasi dan loss dari model dapat diperiksa dengan menggunakan library matplotlib. Masing-masing dari akurasi dan *loss* dibagi menjadi 2 kategori, yaitu untuk training dan validasi.



Gambar 10. Akurasi training dan validasi model



Gambar 11. Loss training dan validasi

Hasil dari gambar 10 dan 11 menunjukan bahwa model telah mencapai tingkat akurasi sekitar 82% dan loss 0.4. Angka tersebut merupakan hasil yang dinilai cukup oleh penulis untuk menggunakannya dan tidak melanjutkan optimisasi atau perubahan lebih lanjut.

API dan Implementasi, model yang sudah dilatih disimpan dan diunduh menjadi file dengan ekstensi .h5 yang kemudian dimasukkan ke dalam VS Code untuk dilakukan integrasi dengan *web microframework Flask* untuk membuat sebuah API yang dapat dihubungkan dengan Postman.

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v13i2.14457 p-ISSN : 2089-9424

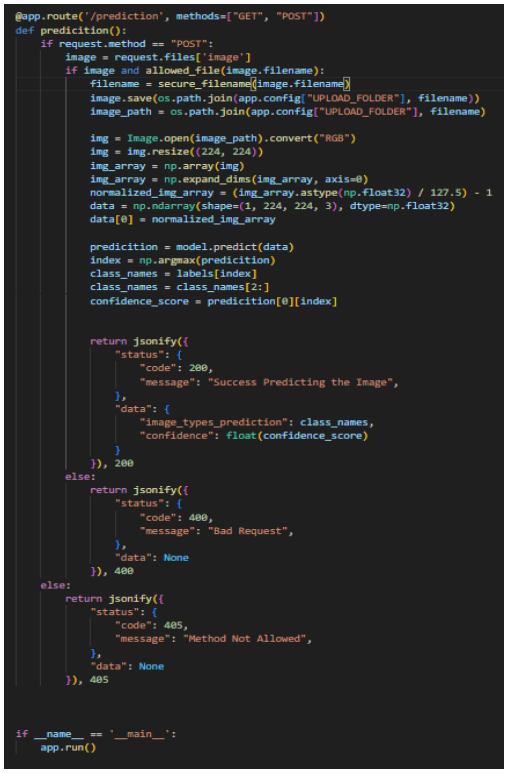
```
app = Flask(__name__)
app.config["ALLOWED_EXTENSIONS"] = set(['png', 'jpg', 'jpeg'])
app.config["UPLOAD_FOLDER"] = "static/uploads/"

def allowed_file(filename):
    return '.' in filename and \
    filename.split('.', 1)[1] in app.config["ALLOWED_EXTENSIONS"]


model = load_model("modeltesting.h5", compile=False)
with open("labels7class.txt", "r") as file:
    labels = file.read().splitlines()


@app.route('/')
def index():
    return jsonify({
        "status": {
            "code": 200,
            "message": "Success fetching the API",
        },
        "data": None
    }), 200
```

Gambar 12. Kode endpoint Flask bagian 1



```
@app.route('/prediction', methods=["GET", "POST"])
def predicition():
    if request.method == "POST":
        image = request.files['image']
        if image and allowed_file(image.filename):
            filename = secure_filename(image.filename)
            image.save(os.path.join(app.config["UPLOAD_FOLDER"], filename))
            image_path = os.path.join(app.config["UPLOAD_FOLDER"], filename)

            img = Image.open(image_path).convert("RGB")
            img = img.resize((224, 224))
            img_array = np.array(img)
            img_array = np.expand_dims(img_array, axis=0)
            normalized_img_array = (img_array.astype(np.float32) / 127.5) - 1
            data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
            data[0] = normalized_img_array

            predicition = model.predict(data)
            index = np.argmax(predicition)
            class_names = labels[index]
            class_names = class_names[2:]
            confidence_score = predicition[0][index]


            return jsonify({
                "status": {
                    "code": 200,
                    "message": "Success Predicting the Image",
                },
                "data": {
                    "image_types_prediction": class_names,
                    "confidence": float(confidence_score)
                }
            }), 200
        else:
            return jsonify({
                "status": {
                    "code": 400,
                    "message": "Bad Request",
                },
                "data": None
            }), 400
    else:
        return jsonify({
            "status": {
                "code": 405,
                "message": "Method Not Allowed",
            },
            "data": None
        }), 405


if __name__ == '__main__':
    app.run()
```

Gambar 13. Kode endpoint Flask bagian 2

Melalui gambar 12 dan gambar 13 dilihat bahwa Kode tersebut adalah bagian dari implementasi aplikasi web menggunakan library Flask untuk melakukan prediksi terhadap gambar yang diunggah. Pertama, konfigurasi Flask dilakukan dengan menetapkan ekstensi file yang diperbolehkan untuk diunggah (png, jpg, jpeg) dan folder tempat untuk menyimpan file unggahan. Model untuk prediksi gambar (modeltesting.h5) dimuat menggunakan Keras, dan label kelas yang terkait dimuat dari file teks (labels7class.txt).

Aplikasi memiliki dua route utama:

1. Route '/': Ini adalah *endpoint default* yang mengembalikan respons JSON yang menunjukkan status berhasil dengan kode 200.
2. Route '/prediction': Endpoint ini menerima metode GET dan POST. Saat metode POST dipanggil, aplikasi menerima file gambar dari permintaan, menyimpannya di folder yang ditentukan, lalu melakukan pra-pemrosesan gambar seperti mengubah ukuran menjadi 224x224 piksel, normalisasi, dan pengubahan ke dalam format yang dapat dimasukkan ke dalam model. Prediksi dilakukan dengan memanggil model.predict untuk menghasilkan kelas prediksi dan skor kepercayaan. Hasil prediksi dan skor kepercayaan dikembalikan dalam respons JSON.

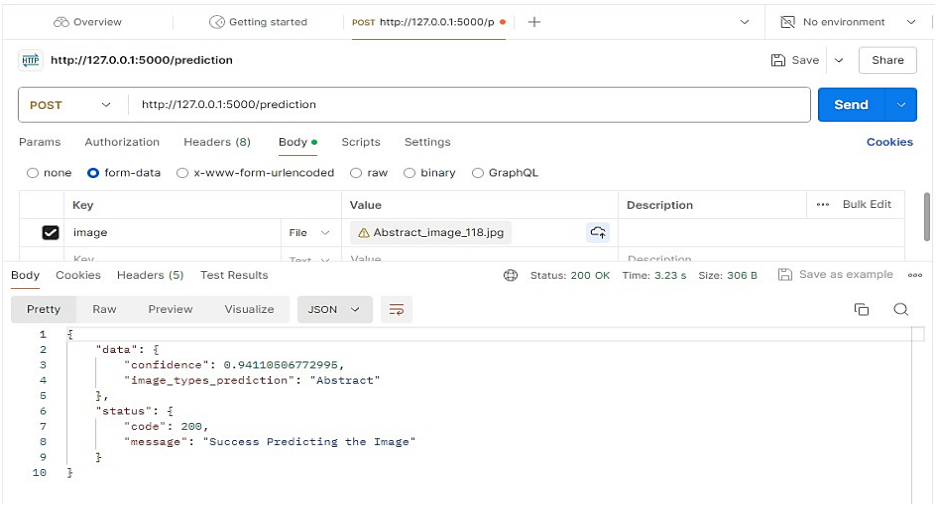
Setiap *route* menghasilkan respons dengan format yang ditentukan, termasuk pesan status yang sesuai dengan hasil dari permintaan tersebut (200 untuk berhasil, 400 untuk kesalahan permintaan, dan 405 untuk metode yang tidak diizinkan). Aplikasi Flask akan berjalan di server lokal jika *script* ini dijalankan sebagai program utama (__name__ == '__main__').

Jurnal Minfo Polgan
Volume 13, Nomor 2, Januari 2025
DOI : https://doi.org/10.33395/jmp.v13i2.14457
e-ISSN : 2797-3298
p-ISSN : 2089-9424

```
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Gambar 14. Output Flask

Mengacu pada gambar 14, ketika kode pada gambar 12 dan gambar 13 dijalankan, output yang akan ditampilkan merupakan sebuah alamat API dimana web sederhana tersebut sedang dijalankan dalam lingkungan lokal komputer yang menjalankannya. Alamat tersebut dapat dimasukkan ke dalam aplikasi Postman untuk melakukan prediksi murni dari model yang telah dibuat.



Gambar 15. Output API pada Postman

Gambar 15 menunjukan bahwa gambar yang dimasukkan berupa gambar seni lukis yang memiliki jenis genre Abstract dengan nama "Abstract_image_118_jpg" dan ketika dikirimkan melalui Postman, model dapat memberikan prediksi yang relatif akurat dan dengan benar memberikan dugaan dari jenis genre yang dimiliki gambar. Dengan tingkat kepercayaan 94%.

KESIMPULAN

Berdasarkan hasil dan pembahasan dari penelitian yang dilakukan berikut merupakan beberapa hal yang dapat disimpulkan dari penelitian ini; 1) Hasil yang didapatkan dari model *deep learning* yang telah dibuat dengan metode *Convolutional Neural Network* (CNN) melebihi 50%, yang dimana akurasi dari training dan validasi melebihi 80% sementara untuk loss berada pada angka 0.4; dan 2) Aplikasi API Flask telah berhasil membuat sebuah alamat yang dapat digunakan pada aplikasi Postman dan berhasil melakukan klasifikasi dari gambar yang diberikan kepada API tersebut.

REFERENSI

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. OSDI, 16, 265-283.

Asthana, A. (2012). Introducing Postman: The only complete API development environment.

Bisong, E. (2019). Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. Apress.

e-ISSN : 2797-3298
p-ISSN : 2089-9424

Brown, A. (2019). Machine Learning in Art: A New Era of Creativity. Art and Technology Journal, 22(1), 23-37.

Cortes, C., & Vapnik, V. (1995). Support-vector Networks. Machine Learning, 20(3), 273-297.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., & Sánchez, C. I. (2017). A Survey on Deep Learning in Medical Image Analysis. Medical Image Analysis, 42, 60-88.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Lutz, M. (2013). Learning Python. O'Reilly Media.

Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A Survey of Deep Learning Techniques for Autonomous Driving. Journal of Field Robotics, 37(3), 362-386.

Grinberg, M. (2014). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T. & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651-666.

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105.

Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach. Pearson.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510-4520.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G. & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.