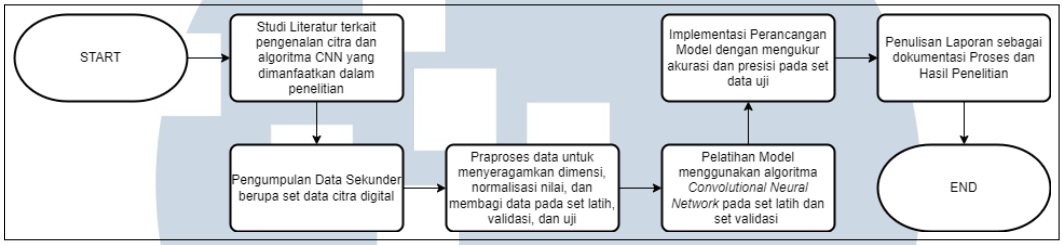
BAB 3
METODOLOGI PENELITIAN

Gambar 3.1 mengilustrasikan metodologi penelitian yang diterapkan pada penelitian ini.



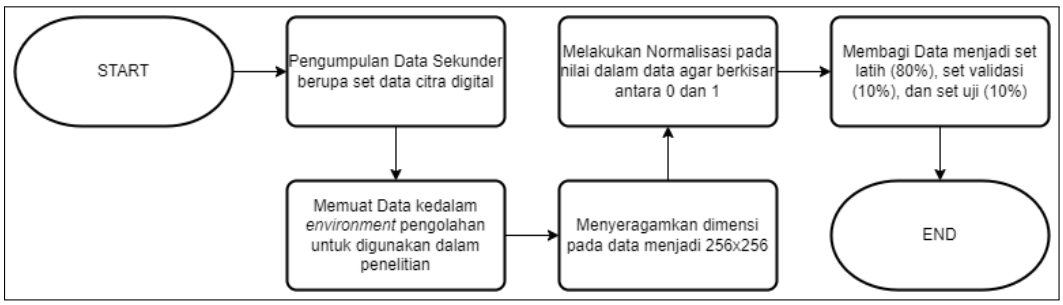
Gambar 3.1. Flowchart metodologi penelitian

3.1 Studi Literatur

Tahap studi literatur dilakukan dalam rangka meningkatkan wawasan dengan cara menelusuri materi terutama terkait konsep CNN dan *Image identification* yang diperlukan untuk melakukan proses penelitian.

3.2 Pengumpulan dan Praproses Data

Sumber data yang digunakan dalam penelitian ini adalah data sekunder berupa dataset *AI and Human Art Classification* yang disediakan pada repositori publik. Dataset berisi 10.330 citra hasil generasi kecerdasan buatan dan 8.288 citra karya manusia. Gambar 3.2 merupakan diagram alur yang memvisualisasikan proses pengumpulan dan praproses data.



Gambar 3.2. Flowchart pengumpulan dan praproses data

Data yang telah dikumpulkan dimuat dengan label yang sudah ditetapkan, data kemudian diseragamkan dengan ukuran 256x256 piksel berentang warna 0 sampai dengan 255 dalam tiga dimensi warna standar yakni merah, hijau, dan biru. Data juga dinormalisasi agar warna citra berkisar antara nol dan satu untuk meningkatkan efisiensi dalam pelatihan model terhadap data. Pada Gambar 3.1, pemuatan data, penyeragaman dimensi, serta normalisasi data diilustrasikan.

```
size = 256
channel = 3
data = keras.utils.image_dataset_from_directory(
            "/kaggle/input/ai_art_classification/train",
            color_mode=("grayscale" if channel == 1 else "rgb"),
            image_size=(size, size), label_mode="categorical",
            class_names=["NON_AI_GENERATED", "AI_GENERATED"])
data = data.map(lambda x, y: (x/255, y))
```

Kode 3.1: Pemuatan dan Praproses Data

Pemuatan Data dilakukan dengan fungsi *image_dataset_from_directory* yang memanfaatkan *pipeline* yang disediakan *library* Keras untuk menciptakan sebuah Dataset Tensorflow yang memuat dataset yang dibagi ke dalam *batch* yang mengelompokkan citra secara acak. Dalam penelitian ini, ukuran *batch* yang digunakan adalah ukuran bawaan pada *pipeline* tersebut yakni 32. Gambar 3.3 menampilkan beberapa citra pada dataset yang telah dipraproses.

Gambar 3.3. Contoh data pada dataset

Data yang sudah dimuat dan diseragamkan kemudian dibagi menjadi tiga set: set latih, set validasi, serta set uji dengan proporsi masing-masing 80%, 10%, dan 10% yang sudah digunakan dalam penelitian sebelumnya [38]. Set validasi digunakan untuk melakukan evaluasi model selama proses pelatihan guna menentukan *hyperparameter* yang paling mendekati optimal pada model [39]. Pada penelitian ini, set validasi digunakan untuk menentukan titik *Early Stopping* dalam menghindari *overfitting*. Proses tersebut dilakukan dengan membagi jumlah *batch* data berdasarkan proporsi set data yang diilustrasikan pada Gambar 3.2.

```
print(f"{len(data)} batches")
train_size = round(len(data)*.8)
val_size = round(len(data)*.1)
test_size = round(len(data)*.1)
if(train_size+val_size+test_size < len(data)):
    train_size += len(data) % (train_size+val_size+test_size)

val = data.take(val_size)
test = data.skip(val_size).take(test_size)
train = data.skip(val_size+test_size).take(train_size)
print(f"Train: {train_size} batches")
print(f"Validation: {val_size} batches")
print(f"Test: {test_size} batches")
```

Kode 3.2: Pembagian Data ke Set Latih, Validasi, dan Uji

Melalui kode tersebut, diketahui bahwa citra dimuat dalam 582 *batch*. Kode tersebut telah membagi dataset yang telah dimuat menjadi data latih sebesar 466 *batch*, data validasi sebesar 58 *batch*, dan data uji sebesar 58 *batch*.

3.3 Perancangan dan Pelatihan Model

Pada tahap pelatihan, model *Convolutional Neural Network* dibentuk menggunakan *library* Keras. Arsitektur model *Convolutional Neural Network* yang digunakan dalam penelitian ini adalah arsitektur EfficientNetB1 dan Xception. Model EfficientNetB1 memanfaatkan blok *inverted bottleneck* untuk mencapai efisiensi yang lebih baik [40]. EfficientNetB1 memiliki 7,8 juta parameter dan hanya menggunakan 0,7 miliar operasi *floating-point*. Arsitektur dasar pada EfficientNetB1 dapat dilihat pada Tabel 3.1 [40].

Tabel 3.1. Arsitektur EfficientNetB1

Stage i	*Operator* $\hat{F}_i$	*Resolution* $\hat{H}_i \times \hat{W}_i$	*#Channels* $\hat{C}_i$	*#Layers* $\hat{L}$
1	Conv3x3	240x240	32	1
2	MBConv1, k3x3	120x120	16	2
3	MBConv6, k3x3	120x120	24	3
4	MBConv6, k5x5	60x60	40	3
5	MBConv6, k3x3	30x30	80	4
6	MBConv6, k5x5	15x15	112	4
7	MBConv6, k5x5	15x15	192	5
8	MBConv6, k3x3	8x8	320	2
9	Conv1x1 & Pooling & FC	8x8	1280	1

Xception merupakan pengembangan Inception V3 dengan pemanfaatan *depthwise separable convolution* yang memisahkan konvolusi spasial dan konvolusi dimensi. Metode tersebut meningkatkan akurasi dengan memaksimalkan penggunaan parameter yang berjumlah sama dengan Inception V3 [41]. Berbeda dengan EfficientNetB1, Xception unggul pada pemanfaatan lapisan yang lebih sedikit yang memudahkan penyesuaian agar lebih cepat mengenal data [42]. Arsitektur Xception diilustrasikan pada Tabel 3.2 [41].

Tabel 3.2. Arsitektur Xception

Stage i	*Operator* $\hat{F}_i$	*Resolution* $\hat{H}_i \times \hat{W}_i$	*#Channels* $\hat{C}_i$	*#Layers* $\hat{L}$
1	Conv3x3 Conv3x3	299x299 149x149	32 64	1
2	SepConv128, k3x3 Conv1x1 ; Pooling3x3	147x147	128	2 1
3	SepConv, k3x3 Conv1x1 ; Pooling3x3	74x74	256	2 1
4	SepConv, k3x3 Conv1x1 ; Pooling3x3	37x37	728	2 1
5-12	SepConv, k3x3	19x19	728	3
13	SepConv, k3x3 SepConv, k3x3 Conv1x1 ; Pooling3x3	19x19	728 1024	1
14	SepConv, k3x3 SepConv, k3x3	10x10	1536 2048	1
15	Pooling & FC	10x10	2048	1

Pada kedua detail arsitektur, k merupakan kernel. Model dipilih dengan pertimbangan akurasi yang serupa pada dataset ImageNet namun memiliki konfigurasi dan *tradeoff* yang berbeda di mana EfficientNetB1 mengutamakan pengoptimalan pada operasi *floating-point* dan Xception mengutamakan fungsi untuk bisa diterapkan secara luas. EfficientNetB1 memiliki *top-1 accuracy* sebesar 79,1% dan *top-5 accuracy* sebesar 94,4% sedangkan Xception memiliki *top-1 accuracy* sebesar 79,0% dan *top-5 accuracy* sebesar 94,5%. Dalam proses pelatihan model, kedua model tersebut dilatih untuk mempelajari data berdasarkan set latih yang kemudian dievaluasi terhadap set validasi.

3.4 Implementasi Rancangan Model

Tahap implementasi meliputi pengujian model terhadap set uji untuk mengukur akurasi dan presisi dari model yang telah dibangun. Akurasi hasil uji kemudian divisualisasikan dengan *confusion matrix* untuk melihat perbedaan *output* klasifikasi dengan label sebenarnya.

3.5 Penulisan Laporan

Setelah dan sementara aplikasi dibangun, sebuah laporan yang berfungsi sebagai dokumentasi tahapan dan hasil penelitian ditulis.

3.6 Spesifikasi Sistem

Penelitian ini dilaksanakan dengan bantuan komponen perangkat keras dan perangkat lunak yang memiliki spesifikasi sebagai berikut:

3.6.1 Hardware

- Prosesor: Intel(R) Core(TM) i5-1035G1 CPU (4 CPUs) @ 1,00GHz
- RAM: 8GB

3.6.2 Software

- Bahasa Pemrograman: Python
- Sistem Operasi: Microsoft Windows 11 Home Single Language
- IDE: Anaconda, Jupyter Notebook, Kaggle Notebook

Adapun Kaggle Notebook yang digunakan dalam penelitian ini memiliki spesifikasi sebagai berikut:

- GPU: NVIDIA Tesla T4 GPU RAM 16GB
- RAM: 29 GB
- Penyimpanan: 73 GB