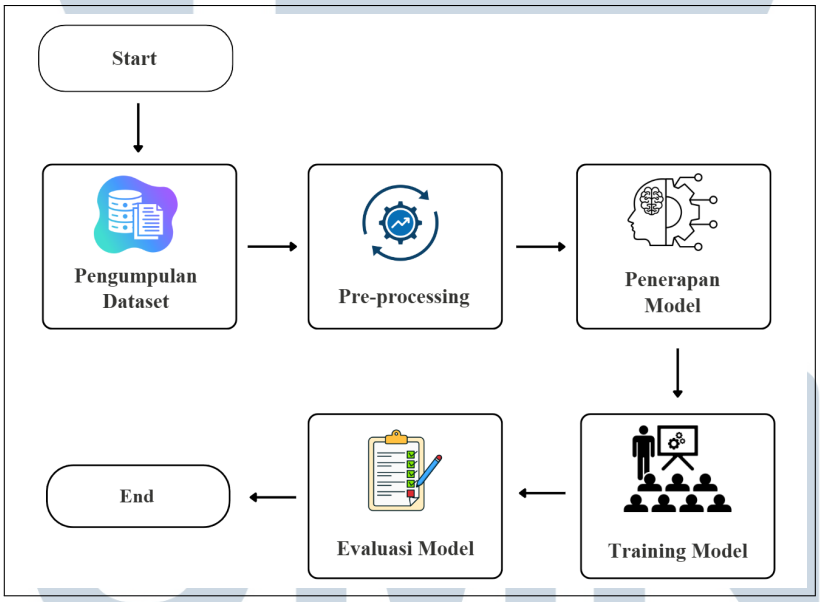
BAB 3
METODE PENELITIAN

3.1 Alur Penelitian

Penelitian ini dilakukan melalui serangkaian tahapan yang sistematis untuk membangun dan mengevaluasi model deteksi gambar deephoax menggunakan pendekatan transfer learning berbasis arsitektur Xception. Alur penelitian disusun mulai dari pengumpulan data hingga evaluasi performa model secara kuantitatif. Setiap tahapan dirancang agar saling terhubung dan mendukung tujuan penelitian, yaitu menghasilkan model klasifikasi yang mampu membedakan gambar asli (real) dan gambar deephoax (fake) secara akurat.



Gambar 3.1. Diagram Alur Penelitian

Tahapan penelitian yang dilakukan seperti yang ditunjukkan pada Gambar 3.1 meliputi pengumpulan dataset, *pre-processing data*, perancangan model berbasis transfer learning, proses pelatihan model, serta evaluasi performa menggunakan berbagai metrik klasifikasi. Alur penelitian ini disajikan dalam bentuk diagram untuk memberikan gambaran yang jelas mengenai proses penelitian dari awal hingga akhir.

3.2 Dataset Penelitian

Dataset merupakan komponen penting dalam penelitian ini karena berperan sebagai sumber utama data yang digunakan untuk melatih, memvalidasi, dan menguji model deteksi gambar deephoax. Dataset yang digunakan terdiri dari gambar wajah asli (real) dan gambar wajah hasil ai-generated (deephoax). Seluruh data berupa gambar (image) yang difokuskan pada area wajah manusia.

3.2.1 Sumber Dataset

Dataset yang digunakan dalam penelitian ini diperoleh dari dua sumber dataset publik yang tersedia pada platform Kaggle, terdiri dari dataset gambar wajah asli dan dataset gambar wajah hasil AI-generated. Dataset gambar wajah asli berasal dari Flickr-Faces-HQ (FFHQ) memiliki variasi yang luas dalam usia, latar belakang, pencahayaan, ekspresi wajah, serta penggunaan aksesori seperti kacamata dan penutup kepala. Gambar-gambar dalam dataset ini dikumpulkan dari Flickr dengan lisensi terbuka. Sementara itu, dataset gambar wajah hasil AI-generated berisi gambar yang dihasilkan menggunakan teknologi kecerdasan buatan, yang secara visual menyerupai wajah manusia asli. Kombinasi kedua dataset ini digunakan untuk merepresentasikan kelas gambar asli dan gambar deephoax dalam penelitian deteksi berbasis transfer learning. Distribusi jumlah data dari masing-masing dataset dapat dilihat pada 3.1 berikut.

Tabel 3.1. Distribusi Dataset Penelitian

Sumber Data	**Gambar Real**	**Gambar Deephoax**	**Total**
Flickr-Faces-HQ (FFHQ)	30.000	–	30.000
AI-Generated Images	–	30.000	30.000
Total	30.000	30.000	60.000

3.2.2 Pembagian Dataset

Dataset yang digunakan dalam penelitian ini dibagi ke dalam tiga bagian, yaitu data training, validation, dan testing, guna memastikan proses pelatihan dan evaluasi model dapat dilakukan secara objektif dan terukur. Data training digunakan untuk melatih model dalam mempelajari pola dan karakteristik visual gambar wajah asli dan deephoax, data validation dimanfaatkan untuk memantau performa model

selama proses pelatihan serta mencegah terjadinya overfitting, sedangkan data testing digunakan untuk mengevaluasi performa akhir model menggunakan data yang belum pernah dilihat sebelumnya. Pembagian dataset penelitian dapat dilihat pada 3.2 berikut.

Tabel 3.2. Pembagian Dataset Penelitian

Subset Utama		**Subset Turunan**		**Deephoax**	**Real**
Training	70% × 60.000 = 42.000	Training	80% × 42.000 = 33.600	16.800	16.800
		Validation	20% × 42.000 = 8.400	4.200	4.200
Testing	30% × 60.000 = 18.000	–	–	9.000	9.000
Total	60.000	–	–	30.000	30.000

3.3 Pre-processing Data

Tahap *pre-processing data* dilakukan untuk menyiapkan dataset gambar sebelum digunakan dalam proses pelatihan dan evaluasi model. Pra-pemrosesan bertujuan untuk menyesuaikan format dan karakteristik data dengan kebutuhan arsitektur Xception, sekaligus meningkatkan kemampuan generalisasi model dalam mendeteksi gambar deephoax. Tahapan pre-processing yang diterapkan dalam penelitian ini meliputi *resize* gambar, normalisasi nilai piksel, serta penerapan *data augmentation* pada data training.

1. Resize Gambar. Seluruh input gambar diubah ukurannya (*resize*) menjadi 299 × 299 piksel. Ukuran ini disesuaikan dengan ukuran input standar yang digunakan oleh arsitektur Xception. Proses *resize* bertujuan untuk menyeragamkan dimensi gambar sehingga dapat diproses secara konsisten oleh model serta menghindari ketidaksesuaian dimensi pada saat pelatihan dan inferensi.

2. Normalisasi. Setelah proses *resize*, dilakukan normalisasi nilai piksel dengan cara menskalakan nilai intensitas piksel dari rentang [0, 255] menjadi rentang

[0, 1]. Normalisasi nilai piksel dilakukan menggunakan metode rescaling dengan membagi setiap nilai piksel dengan konstanta 255, sehingga nilai intensitas gambar berada pada rentang [0,1]. Secara matematis, proses normalisasi dapat dinyatakan sebagai berikut [18].

$$x' = \frac{x}{255} \tag{3.1}$$

di mana x merupakan nilai piksel asli dan x' merupakan nilai piksel hasil normalisasi. Tujuan normalisasi adalah untuk mempercepat proses konvergensi selama pelatihan, menjaga stabilitas proses optimasi, serta meningkatkan performa model dalam mempelajari pola visual dari gambar wajah.

3. Data Augmentation. Untuk meningkatkan keragaman data dan mengurangi risiko *overfitting*, penelitian ini menerapkan teknik *data augmentation* pada data training. Beberapa transformasi yang digunakan meliputi rotasi gambar, pergeseran posisi horizontal dan vertikal, *horizontal flipping*, serta *zooming*. Data augmentation hanya diterapkan pada data training, sedangkan data validation dan data testing tidak mengalami augmentasi agar evaluasi performa model tetap objektif dan mencerminkan kondisi data nyata.

Secara keseluruhan, tahapan *pre-processing* data bertujuan untuk meningkatkan kualitas dan kesiapan dataset sebelum digunakan dalam pelatihan model. Dengan melakukan *resize*, normalisasi, dan data augmentation, model diharapkan mampu mempelajari representasi fitur yang lebih robust, memiliki kemampuan generalisasi yang lebih baik, serta lebih efektif dalam membedakan karakteristik visual antara gambar wajah asli dan gambar deephoax pada data yang belum pernah dilihat sebelumnya.

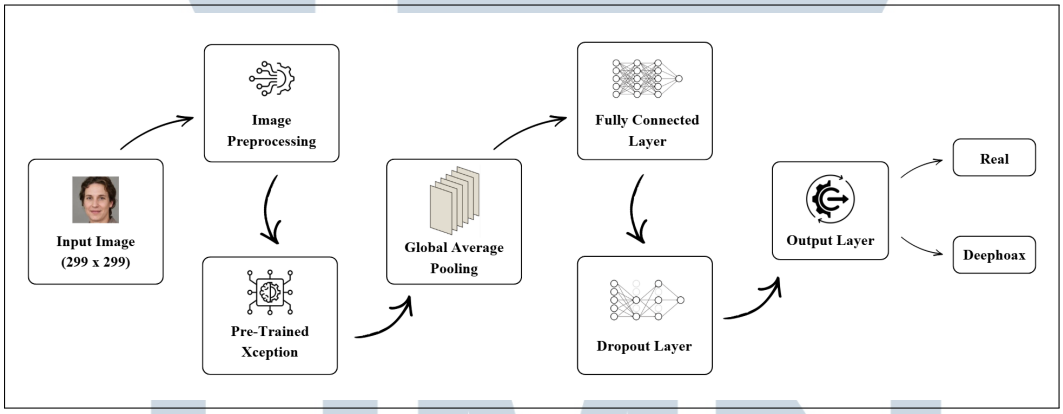
3.4 Penerapan Model

Arsitektur model yang digunakan dalam penelitian ini berbasis deep learning convolutional neural network (CNN) dengan memanfaatkan pendekatan transfer learning. Model utama yang digunakan adalah Xception (Extreme Inception) yang telah dilatih sebelumnya pada dataset ImageNet. Pemilihan arsitektur ini bertujuan untuk memperoleh representasi fitur visual yang kuat dan

efisien dalam membedakan karakteristik antara gambar wajah asli dan gambar deephoax.

3.4.1 Penerapan Transfer Learning

Pendekatan transfer learning yang diterapkan dalam penelitian ini adalah feature extraction dengan memanfaatkan model Xception pre-trained pada dataset ImageNet sebagai backbone. Pada pendekatan ini, seluruh lapisan konvolusi pada model Xception dibekukan (freeze) sehingga bobot hasil pelatihan sebelumnya tidak diperbarui selama proses pelatihan. Strategi ini bertujuan untuk mempertahankan kemampuan ekstraksi fitur visual tingkat tinggi yang telah dipelajari dari data skala besar serta mengurangi risiko overfitting pada dataset penelitian.



Gambar 3.2. Arsitektur Transfer Learning Menggunakan Xception dengan Pendekatan Feature Extraction

Berikut penjelasan Gambar 3.2 untuk proses pemanfaatan model pre-trained sebagai *feature extractor*.

1. Input Image. Gambar wajah berukuran 299 × 299 piksel digunakan sebagai masukan model, sesuai dengan ukuran input standar pada arsitektur Xception.
2. Image Preprocessing. Gambar input terlebih dahulu melalui tahap image preprocessing yang meliputi normalisasi nilai piksel dan penyesuaian format data agar sesuai dengan kebutuhan model.
3. Xception (Feature Extraction). Gambar yang telah dipreproses kemudian diproses oleh model Xception pre-trained yang berperan sebagai feature extractor untuk mengekstraksi fitur visual representatif dari wajah.

4. Global Average Pooling (GAP). Fitur hasil ekstraksi dari Xception diteruskan ke lapisan Global Average Pooling, yang berfungsi untuk mereduksi dimensi fitur spasial menjadi vektor satu dimensi, sehingga jumlah parameter dapat dikurangi dan kemampuan generalisasi model meningkat.

5. Fully Connected Layer (Dense). Vektor fitur kemudian diproses oleh Fully Connected Layer dengan fungsi aktivasi ReLU untuk mempelajari hubungan non-linear antar fitur.

6. Dropout Layer. Untuk mengurangi risiko overfitting, ditambahkan Dropout Layer yang secara acak menonaktifkan sebagian neuron selama proses pelatihan.

7. Output Layer. Tahap akhir model adalah Output Layer dengan fungsi aktivasi softmax yang menghasilkan probabilitas klasifikasi ke dalam dua kelas, yaitu Real dan Deephoax.

Berdasarkan tahapan tersebut, penerapan transfer learning dengan pendekatan feature extraction menggunakan arsitektur Xception memungkinkan model untuk memanfaatkan fitur visual tingkat tinggi yang telah dipelajari sebelumnya, sekaligus menyesuaikannya secara efektif untuk tugas klasifikasi gambar wajah real dan deephoax. Pendekatan ini diharapkan mampu menghasilkan performa klasifikasi yang optimal dengan efisiensi pelatihan yang lebih baik serta risiko overfitting yang lebih rendah pada dataset penelitian.

3.4.2 Model Pre-trained Xception

Xception merupakan arsitektur *Convolutional Neural Network* (CNN) yang dirancang untuk meningkatkan performa ekstraksi fitur dengan memanfaatkan *depthwise separable convolution*. Pendekatan ini memungkinkan pengurangan jumlah parameter dan kompleksitas komputasi tanpa mengorbankan akurasi model. Keunggulan tersebut menjadikan Xception sesuai untuk tugas klasifikasi gambar yang membutuhkan kemampuan mendeteksi perbedaan visual yang halus, seperti pada deteksi gambar deephoax.

Dalam penelitian ini, Xception digunakan sebagai model *pre-trained* dengan bobot hasil pelatihan pada dataset ImageNet. Penggunaan bobot ImageNet memungkinkan model memanfaatkan pengetahuan awal berupa fitur-fitur visual umum, seperti tepi, tekstur, dan struktur wajah, yang telah dipelajari dari dataset

berskala besar. Hal ini membantu mempercepat proses pelatihan dan meningkatkan stabilitas model meskipun dataset penelitian memiliki karakteristik yang berbeda.

```
1
2  def build_xception_model():
3      base_model = Xception(
4          weights="imagenet",
5          include_top=False,
6          input_shape=(299, 299, 3)
7      )
8
9      # Freeze backbone
10     base_model.trainable = False
11
12     x = base_model.output
13     x = GlobalAveragePooling2D()(x)
14     x = Dense(256, activation="relu")(x)
15     x = Dropout(0.3)(x)
16     output = Dense(NUM_CLASSES, activation="softmax")(x)
17
18     model = Model(inputs=base_model.input, outputs=output)
19
20     model.compile(
21         optimizer=Adam(learning_rate=LEARNING_RATE),
22         loss="categorical_crossentropy",
23         metrics=["accuracy"]
24     )
25
26     return model
```

Kode 3.1: Implementasi Arsitektur Model Xception dengan Pendekatan Feature Extraction

Implementasi arsitektur model Xception dengan pendekatan *feature extraction* ditunjukkan pada Kode 3.1. Pada implementasi tersebut, bagian *fully connected layer* bawaan Xception dihilangkan (`include_top=False`), sehingga Xception berperan sebagai *feature extractor* yang menghasilkan representasi fitur tingkat tinggi dari inpur gambar berukuran 299×299 piksel. Seluruh *convolutional layers* pada backbone Xception dibekukan, sementara lapisan klasifikasi tambahan dilatih untuk menyesuaikan dengan tugas klasifikasi gambar deephoax.

3.5 Training Model

Proses pelatihan model dilakukan dengan memanfaatkan data training dan data validation yang telah disiapkan sebelumnya. Model dirancang untuk melakukan klasifikasi dua kelas, yaitu gambar deephoax dan gambar asli (real). Oleh karena itu, fungsi loss categorical cross-entropy digunakan karena sesuai dengan penggunaan softmax activation function pada lapisan output serta format label kategorikal yang dihasilkan oleh data loader.

Tabel 3.3. Konfigurasi Hyperparameter Pelatihan Model

Parameter	**Nilai**
Model	Xception (Pre-trained ImageNet)
Ukuran Input	299×299
Jumlah Kelas	2 (Deephoax, Real)
Fungsi Aktivasi Output	Softmax
Fungsi Loss	Categorical Cross-Entropy
Optimizer	Adam
Learning Rate	0.0001
Batch Size	32
Epoch	10

Konfigurasi hyperparameter yang digunakan dalam proses pelatihan model ditunjukkan pada Tabel 3.3. Model dilatih menggunakan optimizer Adam dengan learning rate sebesar 0.0001. Pemilihan optimizer Adam didasarkan pada kemampuannya dalam menyesuaikan laju pembelajaran secara adaptif, sehingga dapat mempercepat proses konvergensi dan menjaga stabilitas pelatihan. Seluruh gambar input diubah ukurannya menjadi 299 × 299 piksel agar sesuai dengan spesifikasi input model Xception. Proses pelatihan dilakukan dengan batch size sebesar 32 dan jumlah epoch sebanyak 10. Alasan jumlah epoch tersebut dipilih adalah untuk mengamati perilaku pembelajaran model secara efektif sekaligus mempertimbangkan keterbatasan sumber daya komputasi yang tersedia.

```
callbacks = [
        EarlyStopping(
            monitor="val_loss",
            patience=2,
            restore_best_weights=True
        ),
```

```
7
8           ModelCheckpoint(
9                   filepath=checkpoint_path,
10                  monitor="val_loss",
11                  save_best_only=True,
12                  save_weights_only=False,
13                  verbose=1
```

Kode 3.2: Implementasi Callback Early Stopping dan Model Checkpoint pada Proses Training

Untuk meningkatkan stabilitas pelatihan dan mencegah terjadinya overfitting, diterapkan beberapa mekanisme callback seperti yang di tunjukkan pada Kode 3.2. Early Stopping digunakan dengan memantau nilai validation loss dan patience sebanyak dua epoch, sehingga proses pelatihan akan dihentikan secara otomatis apabila tidak terjadi peningkatan performa. Selain itu, Model Checkpoint diterapkan untuk menyimpan model dengan performa terbaik berdasarkan nilai validation loss. Dengan mekanisme ini, model yang digunakan pada tahap evaluasi merupakan model dengan performa optimal selama proses pelatihan. Seluruh eksperimen dilakukan menggunakan konfigurasi pelatihan yang sama, dan seluruh hyperparameter dijaga tetap konstan untuk memastikan evaluasi model yang adil, konsisten, dan dapat direproduksi. Fokus penelitian ini diarahkan pada evaluasi performa model transfer learning berbasis Xception dalam mendeteksi gambar deephoax, bukan pada optimasi hyperparameter secara ekstensif.

3.6 Evaluasi Model

Confusion Matrix merupakan metode evaluasi yang umum digunakan dalam pembelajaran mesin untuk menilai performa model klasifikasi, khususnya pada tugas klasifikasi biner dan multi-kelas. Confusion Matrix disajikan dalam bentuk tabel persegi yang menggambarkan perbandingan antara kelas sebenarnya (ground truth) dan kelas hasil prediksi model. Baris pada matriks merepresentasikan kelas aktual, sedangkan kolom menunjukkan kelas yang diprediksi oleh model.

Dalam penelitian ini, Confusion Matrix digunakan untuk mengevaluasi performa model dalam mendeteksi gambar deephoax, yang merupakan permasalahan klasifikasi biner dengan dua kelas, yaitu gambar asli (real) dan gambar hasil manipulasi (deephoax). Oleh karena itu, Confusion Matrix yang digunakan berukuran 2×2 dan terdiri dari empat komponen utama, yaitu true positive (TP), false positive (FP), true negative (TN), dan false negative (FN).

$$ConfusionMatrix = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{3.2}$$

Keempat komponen ini menjadi dasar dalam pengukuran kinerja model secara lebih komprehensif. Komponen Confusion Matrix pada klasifikasi gambar deephoax dapat dijelaskan sebagai berikut:

1. True Positive (TP): Jumlah gambar deephoax yang berhasil diklasifikasikan dengan benar sebagai deephoax.
2. False Positive (FP): Jumlah gambar asli yang salah diklasifikasikan sebagai deephoax.
3. True Negative (TN): Jumlah gambar asli yang berhasil diklasifikasikan dengan benar sebagai gambar asli.
4. False Negative (FN): Jumlah gambar deephoax yang salah diklasifikasikan sebagai gambar asli.

Berdasarkan nilai-nilai pada confusion matrix, berbagai metrik evaluasi dapat dihitung untuk menilai performa model secara lebih komprehensif, meliputi accuracy, precision, recall, dan F1-score [19, 20].

Metrik accuracy mengukur tingkat ketepatan model secara keseluruhan. Accuracy dirumuskan sebagai berikut.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

Metrik precision mengukur ketepatan model dalam memprediksi kelas positif (deephoax). Precision dirumuskan sebagai berikut.

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

Metrik recall mengukur kemampuan model dalam mendeteksi kemampuan positif (deephoax) yang ada. Recall dirumuskan sebagai berikut.

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

Selanjutnya, F1-score merupakan rata-rata harmonik yang menyeimbangkan presisi dan recall. F1-score dirumuskan sebagai berikut.

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.6}$$

Penggunaan Confusion Matrix beserta metrik turunannya memberikan evaluasi yang lebih informatif dibandingkan penggunaan accuracy saja, khususnya pada kasus deteksi deephoax yang berpotensi memiliki distribusi data antar kelas yang tidak seimbang.

3.6.1 Testing Model

Tahap testing dilakukan menggunakan testing dataset untuk mengevaluasi kemampuan generalisasi model terhadap data yang belum pernah digunakan selama pelatihan. Model terbaik yang diperoleh melalui mekanisme Model Checkpoint digunakan untuk melakukan prediksi pada seluruh data uji. Setiap gambar diproses oleh model untuk menghasilkan probabilitas melalui fungsi aktivasi softmax pada lapisan output. Kelas prediksi ditentukan menggunakan argmax, yaitu memilih kelas dengan nilai probabilitas tertinggi sebagai hasil klasifikasi, baik deephoax maupun real.

```
    y_prob = best_model.predict(test_gen, verbose=0)

    # Fake jika P(fake) > P(real)
    y_pred = np.argmax(y_prob, axis=1)
    y_true = test_gen.classes

    class_names = ["Fake", "Real"]

    df_pred = pd.DataFrame({
        "filename": test_gen.filenames,
        "true_label": y_true,
        "true_class": [class_names[i] for i in y_true],
        "pred_label": y_pred,
        "pred_class": [class_names[i] for i in y_pred],
        "prob_fake": y_prob[:, 0],
        "prob_real": y_prob[:, 1]
    })
```

```
18
19      csv_path = os.path.join(
20          os.path.dirname(MODEL_PATH),
21          "test_predictions.csv"
22      )
23      df_pred.to_csv(csv_path, index=False)
24      print(f"\nTest predictions saved to: {csv_path}")
```

Kode 3.3: Proses Implementasi Proses Testing Model dan Penyimpanan Hasil Prediksi

Hasil prediksi yang diperoleh kemudian dibandingkan dengan true_label untuk menghitung metrik evaluasi performa model, meliputi akurasi, presisi, recall, dan F1-score. Selain itu, seluruh hasil prediksi disimpan dalam bentuk file CSV yang memuat informasi nama file gambar, label aktual, label prediksi, serta nilai probabilitas untuk masing-masing kelas. Penyimpanan hasil prediksi ini bertujuan untuk mendukung analisis lanjutan secara kuantitatif maupun kualitatif serta meningkatkan transparansi proses evaluasi. Tahap testing ini menjadi dasar dalam menilai performa akhir model transfer learning berbasis Xception dalam mendeteksi gambar deephoax serta mengukur kemampuan model dalam melakukan klasifikasi secara akurat pada data baru di luar proses pelatihan.