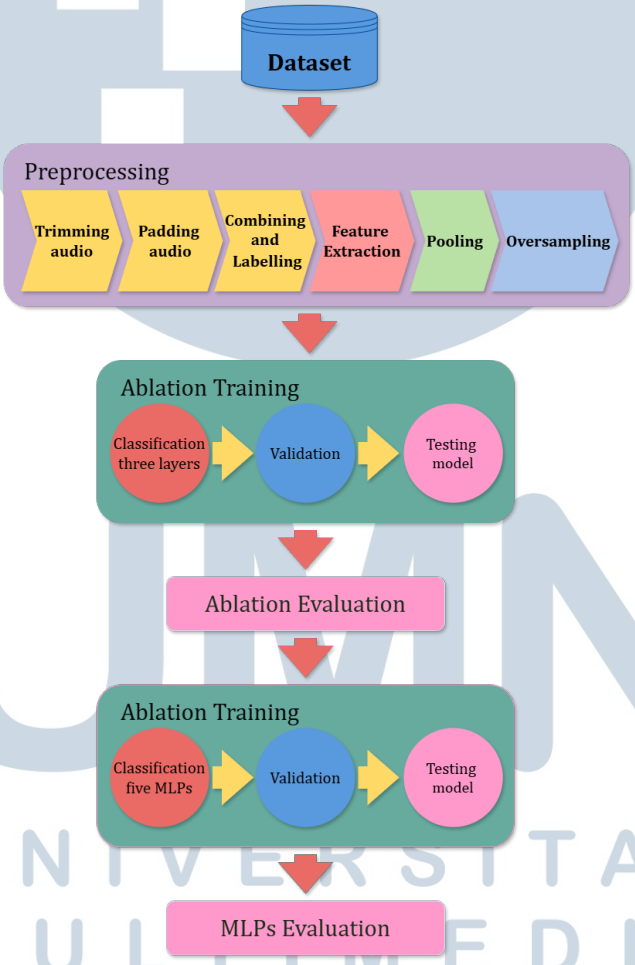
BAB 3
METODE PENELITIAN

Penelitian ini menggunakan *frozen features* yang diekstraksi dari waveform audio menggunakan *pre-trained model* wav2vec 2.0. Fitur-fitur tersebut kemudian digunakan untuk melatih dan menguji empat konfigurasi Multilayer Perceptron (MLP) yang berbeda. Untuk mengatasi ketidakseimbangan kelas pada tahap akhir praproses, dilakukan *oversampling* menggunakan SMOTE. Alur proses penelitian ditunjukkan pada Gambar 3.1.



Gambar 3.1. Metodologi Penelitian Deteksi DeepHoax Audio

3.1 Deskripsi Dataset

Pada ini, dataset yang digunakan adalah dataset Fake-or-Real (FoR) [33]. Dataset ini berisi lebih dari 195.000 ujaran yang mencakup ucapan asli oleh manusia dan ucapan sintetis yang dihasilkan oleh sistem *text-to-speech* yang ahli.

Terdapat empat versi dari dataset ini, yaitu for-original, for-norm, for-2-sec, dan for-rerec. Versi for-original berisi file audio yang dikumpulkan tanpa modifikasi apa pun. For-norm merupakan versi dari for-original yang telah diseimbangkan dalam hal gender dan kelas, serta dinormalisasi pada sample rate, volume, dan jumlah *channel*. For-2-sec adalah versi for-norm yang durasinya telah diseragamkan menjadi 2 detik. Versi terakhir, for-rerec, merupakan versi for-2-sec yang direkam ulang untuk mensimulasikan skenario di mana penyerang mengirimkan pesan melalui sarana suara.

Penelitian ini menggunakan versi for-norm. Versi ini telah diseimbangkan dan dinormalisasi sehingga tidak memerlukan proses pembersihan tambahan. Meskipun versi dua detik dari dataset dapat memberikan kepadatan konten yang lebih baik dan mengurangi beban komputasi, versi tersebut berpotensi kehilangan konteks tambahan yang berguna dalam menentukan apakah suatu sampel merupakan hoaks atau fakta.

Sebelum diproses lebih lanjut, seluruh audio dipotong menjadi durasi lima detik atau setara dengan 80.000 *data points*. Audio yang melebihi ambang *high threshold* lima detik akan dipotong menjadi beberapa potongan (*chunks*). Untuk audio atau potongan dengan durasi di antara *high threshold* dan *low threshold* tiga detik, dilakukan *padding* dengan nilai nol. Sementara itu, jika durasinya berada di bawah *low threshold*, seluruh audio atau potongan tersebut akan dilewati sepenuhnya. Hal ini dilakukan karena *classifier* MLP mengharuskan semua sampel *input* memiliki jumlah fitur yang sama, yang berarti memiliki dimensi yang sama.

3.2 *Feature Extraction* dengan wav2vec 2.0

Ekstraksi fitur dilakukan menggunakan *pre-trained model* wav2vec 2.0 dari *library* torchaudio. Penelitian ini menggunakan model *base*, yang menghasilkan embedding berdimensi 768.

Untuk keperluan *ablation study*, fitur diambil dari tiga layer yang berbeda pada model wav2vec 2.0 base, yaitu layer 4, 8, dan 12, yang masing-masing merepresentasikan layer bawah, tengah, dan atas. Hal ini dilakukan karena layer

teratas adalah layer terdekat dengan output akhir sementara layer bawah lebih dekat kepada *raw inputs* sehingga dengan mengambil tiga sampel di masing masing bagian layer, kontras hasil yang didapatkan bisa signifikan [34]. Fitur dari setiap layer dibekukan (frozen) dan digunakan sebagai input untuk klasifier MLP.

Setelah proses ekstraksi, setiap sampel direpresentasikan dalam bentuk tensor 3D dengan ukuran [1, 249, 768].

Adapun implementasi wav2vec 2.0 ditunjukkan dalam kode 3.1 Dalam kode tersebut, hidden_states adalah list berisi tensors. Setiap tensor ialah output dari satu *transformer layer*. parameter terakhir dari fungsi python yang tertera mengambil argumen layer_index. Jika layer_index ditetapkan sebagai None, maka *feature extraction* dilakukan dengan seluruh layer dalam list (12 layer). Sedangkan jika layer_index berisikan index lapisan wav2vec 2.0 (0-11), maka hanya *feature extraction* hanya dilakukan hingga layer dengan index tersebut.

```
1
2  def feature_extractor(sample_sequence, model_size="base",
       layer_index=None):
3      # Select model
4      if model_size == "base":
5          bundle = torchaudio.pipelines.WAV2VEC2_BASE
6      else:
7          bundle = torchaudio.pipelines.WAV2VEC2_LARGE
8
9      extractor_model = bundle.get_model()
10     extractor_model.eval()
11
12     print(f"Using model: WAV2VEC2_{model_size.upper()}")
13     print("Expected sample rate:", bundle.sample_rate)
14     print(f"Starting extraction for {len(sample_sequence)} samples
       ...\n")
15
16     features = []
17     total = len(sample_sequence)
18
19     for idx, sample in enumerate(sample_sequence, start=1):
20         waveform = flatten_waveform(sample)
21
22         # Convert list      tensor if needed
23         if isinstance(waveform, list):
24             waveform = torch.tensor(waveform)
25
```

```
        # Ensure shape: [1, time]
        if waveform.ndim == 1:
            waveform = waveform.unsqueeze(0)

        with torch.inference_mode():
            # Extract all hidden layers
            # This returns: (list_of_hidden_states, output_dict)
            hidden_states, _ = extractor_model.extract_features(
    waveform)

        # Decide what to save
        if layer_index is None:
            # Save ALL layers (list of tensors)
            feature = hidden_states
        else:
            # Save ONE layer (tensor)
            feature = hidden_states[layer_index]

        features.append(feature)

        print(f"[{idx}/{total}] OK      waveform shape {tuple(
    waveform.shape)}")

    print("\nFeature extraction completed.")
    return features
```

Kode 3.1: Contoh potongan kode dalam bahasa pemorgraman Python dengan judul yang panjang melampaui satu baris

3.3 *Mean Pooling*

Dalam penelitian ini, bentuk akhir fitur hasil ekstraksi sudah seragam yaitu [1, 249, 768], yang masing-masing merepresentasikan ukuran batch, jumlah frame, dan dimensi fitur dari wav2vec 2.0. Frame merupakan potongan kecil audio.Karena audio telah dipotong menjadi lima detik, jumlah frame untuk seluruh sampel sama, yaitu 249 frame.

Dengan demikian, total nilai fitur sebelum pooling mencapai 191.232 fitur. Jika dikalikan dengan jumlah sampel, hal ini berisiko menyebabkan overfitting. untuk mencegah hal tersebut, diterapkanlah *mean pooling*. Proses ini mengubah representasi fitur dari tensor 3D menjadi vektor 1D dengan ukuran (768,) untuk setiap sampel. Hasil *pooling* ini merupakan representasi akhir fitur yang digunakan

sebagai input ke klasifier.

3.4 Oversampling SMOTE

Selama tahap *preprocessing*, beberapa sampel dihapus karena tidak memenuhi *threshold* durasi minimum, yang berpotensi menyebabkan ketidakseimbangan kelas. Untuk mengatasi hal ini, digunakan metode *oversampling* SMOTE pada data latih hingga distribusi kelas menjadi seimbang.

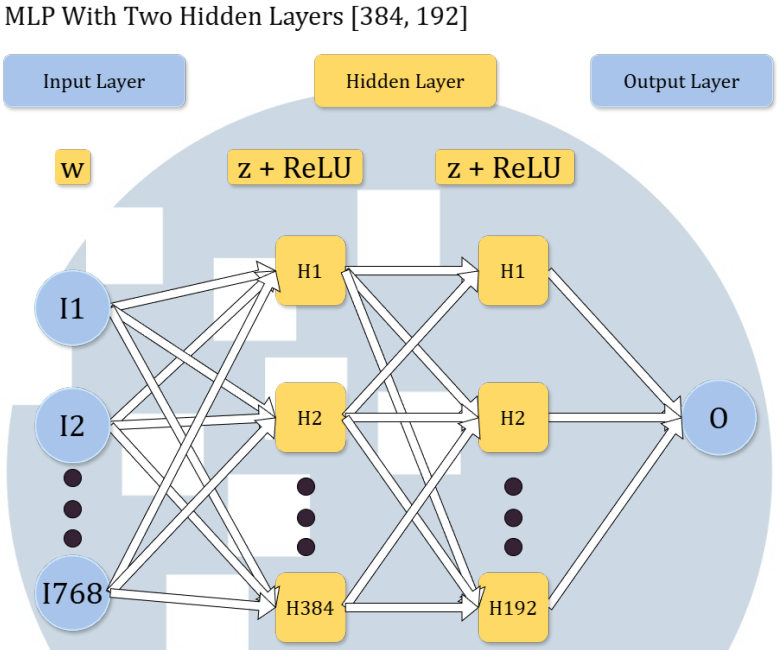
3.5 *Training* MLP *Classifier*

Klasifikasi dilakukan menggunakan Multilayer Perceptron (MLP) dengan fitur hasil ekstraksi wav2vec 2.0 sebagai input. Pada tahap studi ablasi, digunakan arsitektur MLP yang homogen dengan dua *hidden layer* berukuran (384, 192) dan *dropout* sebesar 0,3. Konfigurasi ini digunakan untuk seluruh representasi fitur dari layer bawah, tengah, dan atas wav2vec 2.0 guna memastikan perbandingan yang adil antar layer.

Pada tahap perbandingan model akhir, digunakan empat konfigurasi MLP dengan jumlah dan ukuran *hidden layer* yang berbeda, yaitu:

- 2 lapisan: (384, 192)
- 3 lapisan: (384, 192, 96)
- 4 lapisan: (512, 384, 256, 128)
- 5 lapisan: (640, 512, 384, 256, 128)

Seluruh model menggunakan *activation function* ReLU pada setiap lapisan, BCE sebagai *loss function*, dan *dropout* sebesar 0,3 untuk mengurangi risiko *overfitting*.



Gambar 3.2. Arsitektur MLP dengan dua *hidden layer*

Konfigurasi lapisan tersembunyi dirancang berdasarkan arsitektur heuristik sebagai berikut. Untuk konfigurasi dangkal (2 dan 3 lapisan), digunakan aturan *geometric scale* untuk efisiensi kompresi fitur [35]. Untuk konfigurasi lebih dalam (4 dan 5 lapisan), arsitektur menggunakan *linear reduction* dari ukuran awal yang lebih besar (640 unit) guna menjaga kapasitas informasi yang seimbang diantara lapisan [36]. Selain itu, seluruh ukuran lapisan dipilih sebagai kelipatan 128 untuk mengoptimalkan throughput komputasi.

MLP dalam penelitian ini digunakan pada dua skenario, yaitu *ablation study* dan perbandingan model akhir. *Ablation study* bertujuan untuk mengevaluasi pengaruh representasi fitur dari berbagai layer wav2vec 2.0, sedangkan perbandingan model akhir berfokus pada pengaruh kedalaman arsitektur MLP terhadap performa klasifikasi.

3.6 *Testing Model*

Setelah proses pelatihan selesai, model diuji menggunakan data uji (*test set*) yang tidak pernah dilibatkan selama proses pelatihan maupun validasi. Pengujian dilakukan untuk mengukur kemampuan generalisasi model dalam membedakan audio fakta dan deephoax.

Pada studi ablasi, setiap representasi fitur dari layer wav2vec 2.0 (layer

bawah, tengah, dan atas) diuji menggunakan arsitektur MLP yang sama dengan dua *hidden layer*. Hal ini memungkinkan evaluasi langsung terhadap kontribusi setiap layer terhadap performa klasifikasi.

Pada tahap perbandingan model akhir, seluruh konfigurasi MLP diuji menggunakan representasi fitur terbaik yang diperoleh dari *ablation study*. Setiap model menghasilkan prediksi kelas untuk seluruh sampel data uji, yang kemudian digunakan sebagai dasar perhitungan metrik evaluasi.

3.7 Evaluasi Model

Evaluasi performa model dilakukan menggunakan *confusion matrix* dan metrik evaluasi berbasis klasifikasi. Evaluasi *ablation* membandingkan hasil dari tiga lapisan wav2vec 2.0 yang diklasifikasikan dengan MLP dengan 2 *hidden layers*. Hasil *ablation* akan menghasilkan akurasi, presisi, *recall*, F1, dan *average runtime*. Sedangkan pada evaluasi akhir, 5 model MLP dengan konfigurasi *hidden layer* berbeda akan dibandingkan dan menghasilkan metrik keluaran yang sama dengan *ablation study*.