Dokumentasi Aplikasi Web Deteksi Dini Penyakit Alzheimer

Skripsi: Pengembangan dan Evaluasi Model CNN untuk Diagnosis Dini Penyakit Alzheimer Menggunakan Citra MRI OASIS-1

Implementasi Antarmuka Web Berbasis Streamlit

---

1. Gambaran Umum

Aplikasi ini merupakan implementasi antarmuka web dari model *Convolutional Neural Network* (CNN) yang telah dilatih untuk mengklasifikasikan citra MRI otak ke dalam dua kelas, yaitu **CN (*Cognitively Normal*)** dan **AD (*Alzheimer's Disease*)**.

Aplikasi dibangun menggunakan *framework* **Streamlit**, yaitu pustaka Python yang memungkinkan pembuatan aplikasi web interaktif tanpa memerlukan pemrograman HTML, CSS, maupun JavaScript. Tujuan aplikasi adalah menyediakan sarana pengujian model secara praktis sehingga hasil penelitian dapat didemonstrasikan secara langsung kepada pengguna.

Aplikasi terdiri atas satu modul utama, yaitu **Prediksi Diagnosis**, di mana pengguna mengunggah citra MRI dan sistem mengembalikan hasil klasifikasi beserta tingkat keyakinannya.

2. Struktur Berkas

Seluruh berkas aplikasi berada di dalam direktori `streamlit_web/` dengan rincian sebagai berikut.

Berkas	Keterangan
`app.py`	Berkas utama yang berisi seluruh kode aplikasi Streamlit.
`model.onnx`	Model CNN dalam format ONNX; berkas inilah yang digunakan untuk proses inferensi.
`requirements.txt`	Daftar pustaka Python yang dibutuhkan aplikasi.
`best_model_alzheimer.h5`	Model hasil pelatihan dalam format Keras (arsip, tidak digunakan aplikasi).
`model.tflite`, `saved_model/`	Format ekspor model lainnya (arsip, tidak digunakan aplikasi).
`Normal.jpg`, `Dementia.jpg`	Contoh citra MRI untuk keperluan pengujian aplikasi.
`cm.png`, `roc curve.png`, `kurva.png`, `ringkasan.png`	Gambar hasil evaluasi model (dokumentasi penelitian).

2.1 Kebutuhan Pustaka

```
streamlit
onnxruntime
numpy
pandas
Pillow
```

> **Catatan mengenai pemilihan ONNX Runtime.** Model tidak dimuat menggunakan TensorFlow, melainkan dikonversi terlebih dahulu ke format ONNX dan dijalankan dengan `onnxruntime`. Pertimbangannya adalah ukuran dependensi: TensorFlow berukuran sekitar 500 MB, sedangkan `onnxruntime` hanya sekitar 15 MB. Hal ini penting apabila aplikasi di-*deploy* ke layanan seperti Streamlit Community Cloud yang memiliki batasan ukuran dan memori.

3. Alur Kerja Sistem

Proses yang terjadi sejak pengguna mengunggah citra hingga hasil ditampilkan dijelaskan pada tahapan berikut.

1. **Pengunggahan citra.** Pengguna memilih berkas citra MRI berformat JPG, JPEG, atau PNG melalui komponen *file uploader*.
2. **Pemuatan model.** Berkas `model.onnx` dimuat ke dalam sesi inferensi ONNX Runtime.
3. **Prapemrosesan citra.** Citra dikonversi ke skala keabuan, diubah ukurannya menjadi 224×224 piksel, dinormalisasi, dan disesuaikan dimensinya dengan masukan model.
4. **Inferensi.** Citra hasil prapemrosesan diteruskan ke model, yang menghasilkan nilai probabilitas kelas AD.
5. **Penentuan kelas.** Probabilitas dibandingkan dengan ambang batas 0,5 untuk menentukan kelas keluaran.
6. **Penyajian hasil.** Hasil klasifikasi, *confidence score*, dan distribusi probabilitas ditampilkan kepada pengguna.

3.1 Pemuatan Model

```
@st.cache_resource
def load_model():
    return ort.InferenceSession(MODEL_PATH)
```

Dekorator `@st.cache_resource` berfungsi menyimpan objek model di dalam *cache*. Hal ini diperlukan karena Streamlit menjalankan ulang keseluruhan skrip setiap kali terjadi interaksi pengguna. Tanpa dekorator tersebut, model akan dimuat berulang kali sehingga aplikasi menjadi lambat. Dengan *caching*, model cukup dimuat satu kali selama sesi berlangsung.

3.2 Prapemrosesan Citra

```
def preprocess_image(uploaded_file):
    img = Image.open(uploaded_file).convert("L")
    img = img.resize(TARGET_SIZE, Image.BILINEAR)
    arr = np.array(img, dtype=np.float32) / 255.0
    return arr[np.newaxis, :, :, np.newaxis]   # (1, 224, 224, 1)
```

Tahapan prapemrosesan dijelaskan sebagai berikut.

- **Konversi skala keabuan** — `convert("L")` mengubah citra menjadi satu kanal warna, sesuai dengan karakteristik citra MRI dan konfigurasi masukan model.

- **Penyeragaman ukuran** — citra diubah menjadi 224×224 piksel dengan interpolasi bilinear agar seragam dengan dimensi masukan model.
- **Normalisasi** — nilai piksel dibagi 255 sehingga berada pada rentang 0 sampai 1, yang mempercepat konvergensi dan menjaga kestabilan numerik.
- **Penyesuaian dimensi** — larik diubah menjadi bentuk `(1, 224, 224, 1)` yang merepresentasikan (*batch*, tinggi, lebar, kanal).

> **Penting.** Tahapan prapemrosesan pada aplikasi harus identik dengan prapemrosesan yang digunakan pada saat pelatihan model. Perbedaan sekecil apa pun, misalnya pada metode normalisasi atau interpolasi, akan menyebabkan distribusi masukan bergeser sehingga hasil prediksi menjadi tidak valid.

3.3 Proses Inferensi

```
def predict(session, img_array):
    input_name = session.get_inputs()[0].name
    result = session.run(None, {input_name: img_array})
    prob_ad = float(result[0][0][0])
    return prob_ad
```

Model menggunakan satu neuron keluaran dengan fungsi aktivasi *sigmoid*, sehingga menghasilkan sebuah nilai probabilitas pada rentang 0 sampai 1 yang menyatakan kemungkinan citra termasuk kelas AD. Probabilitas kelas CN diperoleh sebagai komplemennya, yaitu 1 − `prob_ad`. Penentuan kelas dilakukan dengan ambang batas 0,5:

- `prob_ad ≥ 0,5` → diklasifikasikan sebagai **AD (*Alzheimer's Disease*)**
- `prob_ad < 0,5` → diklasifikasikan sebagai **CN (*Cognitively Normal*)**

4. Antarmuka Pengguna

4.1 Panel Samping (*Sidebar*)

Panel samping memuat judul aplikasi serta identitas penelitian, yaitu judul skripsi yang menjadi dasar pengembangan aplikasi. Panel ini bersifat informatif dan tampil pada seluruh tampilan aplikasi.

4.2 Halaman Prediksi Diagnosis

Halaman utama diawali dengan judul dan keterangan singkat mengenai fungsi aplikasi, diikuti komponen unggah berkas. Selama belum ada citra yang diunggah, aplikasi menampilkan pesan informasi berisi arahan bagi pengguna.

Setelah citra diunggah, tampilan terbagi menjadi dua kolom dengan pembagian sebagai berikut.

Bagian	Komponen yang Ditampilkan
Kolom kiri	Pratinjau citra MRI yang diunggah oleh pengguna, ditampilkan menyesuaikan lebar kolom.
Kolom kanan	Hasil klasifikasi yang mencakup: • Label hasil prediksi dengan penanda warna (merah untuk AD, hijau untuk CN); • *Confidence score* dalam satuan persen; • Diagram batang distribusi probabilitas kelas CN dan AD; • *Progress bar* sebagai representasi visual probabilitas AD.

Selama proses berlangsung, aplikasi menampilkan indikator pemuatan melalui komponen `st.spinner` sehingga pengguna memperoleh umpan balik bahwa citra sedang diproses.

4.3 Sanggahan (*Disclaimer*)

Pada bagian bawah hasil prediksi ditampilkan pernyataan sanggahan yang menegaskan bahwa hasil prediksi bersifat suportif dan bukan merupakan pengganti diagnosis klinis oleh dokter spesialis. Aplikasi hanya ditujukan sebagai alat bantu skrining awal. Pencantuman pernyataan ini merupakan bentuk pemenuhan aspek etika dalam penerapan kecerdasan buatan pada bidang kesehatan.

5. Petunjuk Menjalankan Aplikasi

Aplikasi dijalankan melalui terminal dengan tahapan berikut.

```
cd streamlit_web
pip install -r requirements.txt
streamlit run app.py
```

Setelah perintah dijalankan, Streamlit akan mengaktifkan peladen lokal dan membuka aplikasi pada peramban, umumnya melalui alamat `http://localhost:8501`.

> **Perhatian.** Lokasi berkas model ditulis sebagai lintasan relatif (`MODEL_PATH = "model.onnx"`). Oleh karena itu, aplikasi harus dijalankan dari dalam direktori `streamlit_web/`. Apabila dijalankan dari direktori lain, berkas model tidak akan ditemukan dan aplikasi akan mengalami galat.

6. Catatan Teknis

- **Ambang batas klasifikasi.** Nilai ambang yang digunakan adalah 0,5. Nilai ini dapat disesuaikan apabila diperlukan penekanan pada sensitivitas atau spesifisitas tertentu sesuai kebutuhan penerapan klinis.
- **Format masukan.** Aplikasi menerima berkas berformat JPG, JPEG, dan PNG. Citra MRI dalam format DICOM perlu dikonversi terlebih dahulu sebelum dapat digunakan.
- **Konsistensi prapemrosesan.** Setiap perubahan pada prosedur prapemrosesan harus diterapkan secara konsisten, baik pada tahap pelatihan maupun pada tahap inferensi di aplikasi.
- **Pemuatan model.** Model dimuat secara *lazy*, yakni hanya pada saat pengguna mengunggah citra, sehingga waktu muat awal aplikasi tetap ringan.

Dokumen ini menjelaskan implementasi antarmuka web pada berkas `streamlit_web/app.py`.