Laporan klasifikasi Lion,Tiger

Laporan Klasifikasi Gambar Hewan (Harimau vs Singa) menggunakan ResNet18

1. Pendahuluan

Proyek ini bertujuan untuk mengembangkan dan mengevaluasi model klasifikasi gambar yang mampu membedakan antara gambar harimau dan singa. Untuk mencapai tujuan ini, kami memanfaatkan arsitektur Convolutional Neural Network (CNN) yang sudah terbukti efektif, yaitu ResNet18, yang telah dilatih sebelumnya (\_pre-trained\_) pada dataset ImageNet. Pendekatan ini memungkinkan kami untuk memanfaatkan pengetahuan yang telah diperoleh model dari dataset yang besar dan beragam, sehingga mempercepat proses pelatihan dan meningkatkan akurasi pada dataset target yang lebih kecil. Laporan ini akan merinci seluruh tahapan proyek, mulai dari persiapan dataset, desain dan modifikasi model, proses pelatihan, hingga evaluasi hasil dan demonstrasi prediksi.

2. Dataset

Dataset yang digunakan dalam proyek ini adalah kumpulan gambar hewan yang secara spesifik terdiri dari gambar harimau dan singa. Dataset ini diorganisir dalam struktur direktori sebagai berikut:

datasets/

├── train/

│ ├── tiger/

│ │ ├── image1.jpg

│ │ ├── image2.jpg

│ │ └── ...

│ └── lion/

│ ├── image1.jpg

│ ├── image2.jpg

│ └── ...

└── test/

├── 1.webp

└── ... (gambar-gambar untuk prediksi)

Gambar-gambar dimuat menggunakan `ImageFolder` dari pustaka `torchvision`, yang secara otomatis mengidentifikasi kelas berdasarkan nama subfolder.

2.1 Pra-pemrosesan Data

Sebelum dimasukkan ke dalam model, setiap gambar melewati serangkaian transformasi standar untuk memastikan konsistensi dan kesesuaian format input model:

1. Pengubahan Ukuran: Gambar diubah ukurannya menjadi $224 \times 224$ piksel, ukuran input standar untuk model ResNet.

2. Konversi ke Tensor: Gambar dikonversi menjadi Tensor PyTorch.

3. Normalisasi: Gambar dinormalisasi menggunakan nilai rata-rata (`mean`) `[0.485, 0.456, 0.406]` dan standar deviasi (`std`) `[0.229, 0.224, 0.225]`. Nilai-nilai ini adalah statistik rata-rata channel RGB dari dataset ImageNet, yang sesuai dengan bobot pre-trained ResNet.

Dataset kemudian dibagi menjadi data latih (80%) dan data validasi (20%) secara acak. `DataLoader` digunakan untuk memuat data dalam batch berukuran 32, yang memungkinkan pelatihan yang efisien.

3. Arsitektur Model

Model yang dipilih untuk tugas klasifikasi ini adalah ResNet18. ResNet (Residual Network) adalah jenis CNN yang memperkenalkan koneksi \_skip\_ atau \_residual\_ untuk mengatasi masalah \_vanishing/exploding gradients\_ pada jaringan yang sangat dalam. ResNet18 adalah versi yang lebih ringan dengan 18 lapisan.

Kami menggunakan model ResNet18 yang telah dilatih sebelumnya pada dataset ImageNet (`ResNet18\_Weights.DEFAULT`). Ini berarti model telah mempelajari fitur-fitur umum dari berbagai objek, yang dapat ditransfer ke tugas klasifikasi hewan ini.

3.1 Modifikasi Lapisan Output

Karena tugas kita hanya memiliki dua kelas (harimau dan singa), lapisan \_fully connected\_ (fc) terakhir dari ResNet18 dimodifikasi. Lapisan `fc` asli, yang biasanya memiliki 1000 output (sesuai jumlah kelas ImageNet), diganti dengan lapisan `nn.Linear` baru yang memiliki 2 output. Ini memastikan bahwa output model sesuai dengan jumlah kelas yang ingin kita prediksi.

- Total Parameter Model: Sekitar 11.18 Juta parameter.

- Parameter yang Dapat Dilatih: Sekitar 11.18 Juta parameter (semua parameter dilatih, bukan hanya lapisan terakhir, meskipun praktik \_fine-tuning\_ seringkali hanya melatih lapisan terakhir atau lapisan yang lebih tinggi).

4. Pelatihan Model

Proses pelatihan model dilakukan dengan konfigurasi sebagai berikut:

- Device: Model dilatih menggunakan CUDA (GPU) jika tersedia, untuk mempercepat komputasi. Jika tidak, pelatihan akan beralih ke CPU.

- Fungsi Kerugian (Loss Function): `nn.CrossEntropyLoss` digunakan. Fungsi ini cocok untuk tugas klasifikasi multi-kelas dan secara implisit menggabungkan `softmax` dan \_negative log likelihood loss\_.

- Optimizer: `optim.Adam` digunakan sebagai optimizer. Adam adalah algoritma optimasi yang populer dan adaptif yang efisien dalam mengelola laju pembelajaran.

- Laju Pembelajaran (Learning Rate): Ditetapkan sebesar `1e-3`.

- Epoch: Model dilatih selama 10 epoch.

Selama setiap epoch, model dilatih pada data latih dan dievaluasi pada data validasi. Metrik yang dicatat meliputi kerugian (loss) pada data latih dan validasi, serta akurasi pada data validasi. Model beserta bobot optimizer dan riwayat loss disimpan dalam file `animal\_classifier\_checkpoint.pth` pada akhir pelatihan, memungkinkan pemulihan atau penggunaan kembali model.

5. Hasil

Setelah 10 epoch pelatihan, model menunjukkan performa yang menjanjikan pada data validasi. Berikut adalah ringkasan hasil yang diamati berdasarkan grafik loss dan confusion matrix pada epoch terakhir:

- Epoch 10:

- Train Loss: Sekitar 0.0250 (diestimasi dari grafik)

- Val Loss: Sekitar 0.0600 (diestimasi dari grafik)

- Val Accuracy: 94.25%

Grafik kerugian pada data latih dan validasi menunjukkan tren penurunan yang konsisten, mengindikasikan bahwa model belajar dengan efektif dari data. Perbedaan kecil antara \_train loss\_ dan \_val loss\_ menunjukkan bahwa model tidak mengalami \_overfitting\_ yang signifikan pada data latih. Akurasi validasi sebesar 94.25% menunjukkan kemampuan model yang kuat dalam membedakan antara harimau dan singa.

6. Prediksi

Proyek ini juga menyediakan fungsionalitas untuk melakukan prediksi pada gambar baru menggunakan model yang telah dilatih.

- Fungsi `load\_model` digunakan untuk memuat checkpoint model yang disimpan (`animal\_classifier\_checkpoint.pth`).

- Fungsi `predict\_image` menerima path ke gambar baru, menerapkan transformasi pra-pemrosesan yang sama seperti saat pelatihan, dan kemudian menggunakan model untuk menghasilkan prediksi kelas (`harimau` atau `singa`).

Contoh prediksi dilakukan pada gambar `datasets/test/1.webp`, dengan hasil prediksi dan visualisasi gambar tersebut.

7. Visualisasi

Untuk memantau dan menganalisis performa model, beberapa visualisasi penting disertakan:

- Loss Plot: Menampilkan kurva kerugian (loss) data latih dan data validasi per epoch. Ini membantu dalam memantau konvergensi model dan mendeteksi \_overfitting\_.

- Accuracy Plot: Menampilkan kurva akurasi validasi per epoch.

- Confusion Matrix: Matriks kebingungan dihasilkan untuk mengevaluasi performa model secara lebih detail pada data validasi, menunjukkan jumlah \_True Positives\_, \_True Negatives\_, \_False Positives\_, dan \_False Negatives\_.

- Prediction Display: Gambar yang diprediksi ditampilkan beserta label prediksinya, memberikan demonstrasi visual dari kapabilitas model.

8. Ketergantungan

Proyek ini dikembangkan menggunakan pustaka Python berikut:

- `torch`

- `torchvision`

- `matplotlib`

- `numpy`

- `scikit-learn`

- `seaborn`

- `Pillow`

9. Cara Menjalankan

Untuk menjalankan proyek ini:

1. Persiapan Dataset: Pastikan dataset gambar hewan (`harimau` dan `singa`) tersedia pada path `datasets/train` dengan struktur folder yang sesuai (`tiger`, `lion`). Siapkan juga folder `datasets/test` untuk gambar uji jika ingin menggunakan path yang sama untuk prediksi.

2. Instalasi Ketergantungan: Instal semua pustaka Python yang tercantum di bagian "Ketergantungan" menggunakan `pip`, misalnya:

```bash

pip install torch torchvision matplotlib numpy scikit-learn seaborn Pillow

```

3. Menjalankan Notebook:

Untuk melatih model dari awal: Buka file `klasifikasi hewan tiger,lion.ipynb` dan jalankan semua sel secara berurutan. Ini akan melatih model, mengevaluasinya, dan menyimpan checkpoint.

Untuk melakukan prediksi menggunakan model yang telah dilatih\*\*: Pastikan file `animal\_classifier\_checkpoint.pth` ada di direktori yang sama dengan notebook (atau sesuaikan path-nya). Jalankan sel-sel yang berisi import library, definisi dataset (untuk `class\_names`), definisi fungsi `load\_model` dan `predict\_image`. Kemudian, modifikasi variabel `img\_path` pada bagian prediksi untuk menunjuk ke gambar yang ingin Anda prediksi dan jalankan sel tersebut untuk melihat hasil dan visualisasi.