Klasifikasi Audio Perintah: CNN - LSTM

Speech Recognition

Kelompok 17:
1. Ivan Setiawan - 2602109473
2. Yithro Paulus Tjendra - 2602112051
3. Muhammad Fathan - 2602196435

I. Latar Belakang

Di era teknologi yang sudah berkembang, tombol dan input fisik sudah biasa digunakan sebagai metode melakukan sebuah aksi, namun memiliki keterbatasan dalam hal kenyamanan dan aksesibilitas pengguna. Dengan melakukan klasifikasi audio perintah, kita dapat menggunakan suara sebagai metode input alternatif sehingga teknologi dapat bekerja menjadi lebih intuitif dan *user-friendly*. Suara memungkinkan user untuk melakukan interaksi dengan perangkat secara *hands-free* dan lebih natural, terutama dalam situasi di mana penggunaan tangan tidak memungkinkan ataupun tidak praktis.

Dalam pembuatan teknologi tersebut, muncullah tantangan baru dimana sebuah perintah harus bisa dideteksi secara akurat oleh sistem, karena kesalahan pada klasifikasi dapat menghasilkan aksi yang beda dan dapat berakibat fatal. Proses ekstraksi sebuah data audio juga relatif sulit untuk dilakukan. Namun, metode konvensional seringkali tidak mampu menangani keragaman dan noise pada data audio, sehingga muncullah teknologi deep learning seperti Convolution Neural Network (CNN) yang biasa digunakan untuk mengolah dan mengekstrak fitur pada data gambar. Selain itu, Long Short-Term Memory (LSTM) biasa digunakan untuk data yang memiliki urutan dan juga memiliki performa yang baik sehingga sering digunakan untuk data teks. Oleh karena itu, kami melakukan eksperimen ini untuk melihat performa gabungan dari kedua teknik ini pada data audio.

II. Rumusan Masalah

1. Bagaimana cara mengolah dan melakukan klasifikasi data audio?
2. Bagaimana performa model CNN dalam klasifikasi audio?
3. Bagaimana performa model CNN-LSTM dalam klasifikasi audio?

III. Tujuan Project

1. Mengolah data audio perintah.
2. Membangun model CNN dan CNN-LSTM untuk klasifikasi audio.
3. Membandingkan performa kedua model.
4. Menentukan seberapa layak model melakukan klasifikasi audio perintah.

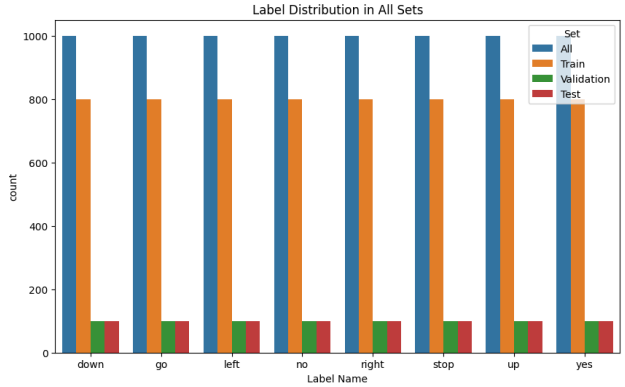
IV. Ruang Lingkup Project

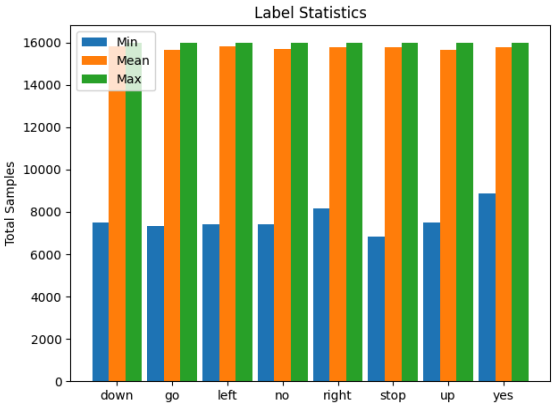
Proyek ini dibatasi pada klasifikasi audio dari dataset yang sudah tersedia, dengan fokus pada pengembangan dan evaluasi model CNN dan CNN-LSTM. Pengujian dilakukan menggunakan dataset yang terdiri delapan kelas dengan jumlah yang seimbang dan tidak mencakup data real-time.

V. Deskripsi Dataset

Dataset yang digunakan dalam proyek ini adalah dataset *mini_speech_commands*, yang berisi rekaman audio dari beberapa kelas berbeda. Dataset ini dipilih karena kelengkapannya dan penggunaannya yang luas dalam penelitian pengenalan suara. Terdapat 8000 data audio dengan *sample rate* 16000 Hz dengan panjang durasi terkecil sekitar 500 milisekon, durasi terbesar di 1000 milisekon, dan rata-rata di sekitar 1000 milisekon. Dataset terdiri dari delapan kelas label perintah, yaitu *'down', 'go', 'left', 'no', 'right', 'stop', 'up',* dan *'yes'*.

Berdasarkan tabel Explanatory Data Analysis yang ada di bawah, kita dapat melihat bahwa jumlah distribusi label seimbang dengan total 1000 data per label dan dipecah menjadi set pelatihan, validasi, maupun uji dengan metode stratifikasi untuk memastikan tidak ada bias dalam pelatihan model. Dengan ini data-data diharapkan mampu mendukung performa dan generalisasi model menjadi lebih baik.





VI. Tahap-tahap Eksperimen

1. Setup Environment

a. Dalam tahap *setup environment*, kami melakukan instalasi berbagai *library* yang diperlukan untuk proses training model, seperti Tensorflow, Tensorflow.io, Numpy, Matplotlib, Seaborn, Scikit-learn, dan Pandas. Instalasi ini memastikan bahwa semua tool yang diperlukan tersedia dan dapat diimpor ke dalam project. Selain itu, kami memastikan Tensorflow menggunakan GPU untuk mempercepat proses komputasi. Kami juga menentukan nilai seed untuk eksperimen agar hasil yang diperoleh dapat direproduksi lebih konsisten. Meski demikian, hasil eksperimen dapat memiliki variasi dalam performa dikarenakan arsitektur unit pemrosesan grafis dalam GPU tidak sepenuhnya presisi.

2. Menyiapkan Dataset

a. Tahap menyiapkan dataset melibatkan instalasi dan *loading* dataset yang akan digunakan. Data yang telah di-load kemudian dibagi (*splitting*) menggunakan stratifikasi untuk memastikan distribusi label yang seimbang antara *training set, validation set,* dan *test set*. Ini penting untuk menjaga representasi yang konsisten dari masing-masing kelas dalam data, sehingga model dapat belajar dengan baik dari seluruh variasi yang ada.

3. Membuat EDA (Explanatory Data Analysis)

a. Selama tahap EDA, kami melakukan analisis distribusi label dalam dataset untuk memahami proporsi masing-masing kelas. Selain itu, kami melihat statistik deskriptif dari label, seperti rata-rata, median, dan distribusi frekuensi. Analisis ini membantu dalam memahami karakteristik data dan mengidentifikasi potensi masalah seperti ketidakseimbangan kelas yang perlu diperhatikan sebelum melanjutkan ke tahap selanjutnya.

4. Preprocessing

a. Dalam tahap preprocessing, kami melakukan *load* dataset dan menerapkan berbagai teknik preprocessing. Langkah ini meliputi padding dan penentuan bentuk file audio untuk memastikan data dalam format yang konsisten. Kami juga melakukan setting ukuran batch untuk efisiensi pemrosesan, membentuk spesifikasi dataset, dan mengubah waveform menjadi spectrogram dengan tambahan satu dimensi. Selain itu, kami menyimpan *cache* dan melakukan operasi *prefetch* untuk mengurangi *read latency* saat training model.

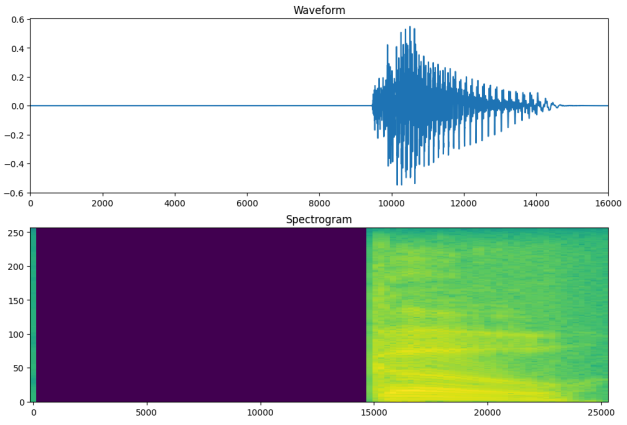
5. **Modelling**
    a. Tahap modelling dimulai dengan menentukan *shape input* dan *output* yang diperlukan oleh model. kami membangun model CNN dan menggabungkannya dengan LSTM untuk memanfaatkan kekuatan kedua model dalam menangani data urutan dan spasial. Model kemudian dikompilasi dengan menentukan *optimizer*, *loss function*, dan *evaluation metrics*. Setelah itu, kami melatih model dengan data training yang telah di-*preprocess* diikuti dengan validasi menggunakan data validasi. Lalu, melakukan visualisasi *loss* serta performa model selama proses pelatihan. Model yang telah dilatih dan *history* dari pelatihannya kemudian disimpan untuk evaluasi lebih lanjut.

6. **Evaluation**
    a. Pada tahap evaluasi, kami melakukan *load* model yang telah disimpan dan mengujinya menggunakan *test set*. Evaluasi ini mencakup pembuatan classification report untuk menilai performa model berdasarkan metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score*. Selain itu, kami membuat *confusion matrix* untuk mendapatkan gambaran visual tentang prediksi model terhadap label sebenarnya, yang membantu dalam mengidentifikasi area yang memerlukan perbaikan lebih lanjut.

VII. Preprocessing

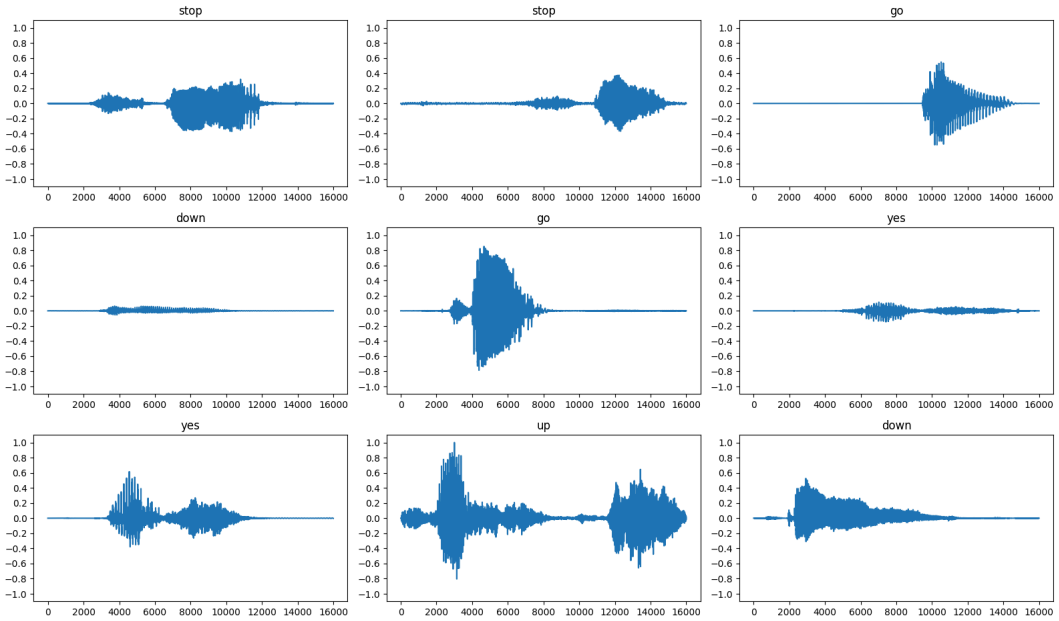
Dalam tahap preprocessing, kami melakukan load dataset dan menerapkan berbagai teknik preprocessing. Langkah ini meliputi padding dan penentuan bentuk file audio untuk memastikan data dalam format yang konsisten. kami juga melakukan setting ukuran batch 64 untuk efisiensi pemrosesan, membentuk spesifikasi dataset, dan mengubah waveform menjadi spectrogram dengan tambahan satu dimensi. Penambahan satu dimensi dilakukan agar spectrogram memiliki dimensi *channel* yang biasa dimiliki oleh data gambar sehingga data dapat dimasukkan ke dalam model CNN. Selain itu, kami menyimpan *cache* dan melakukan operasi *prefetch* untuk mengurang*i read latency* saat training model.

Sebagai visualisasi, di samping ini kami lampirkan perbandingan gambar dari waveform dan spectrogram dari kata “go”.

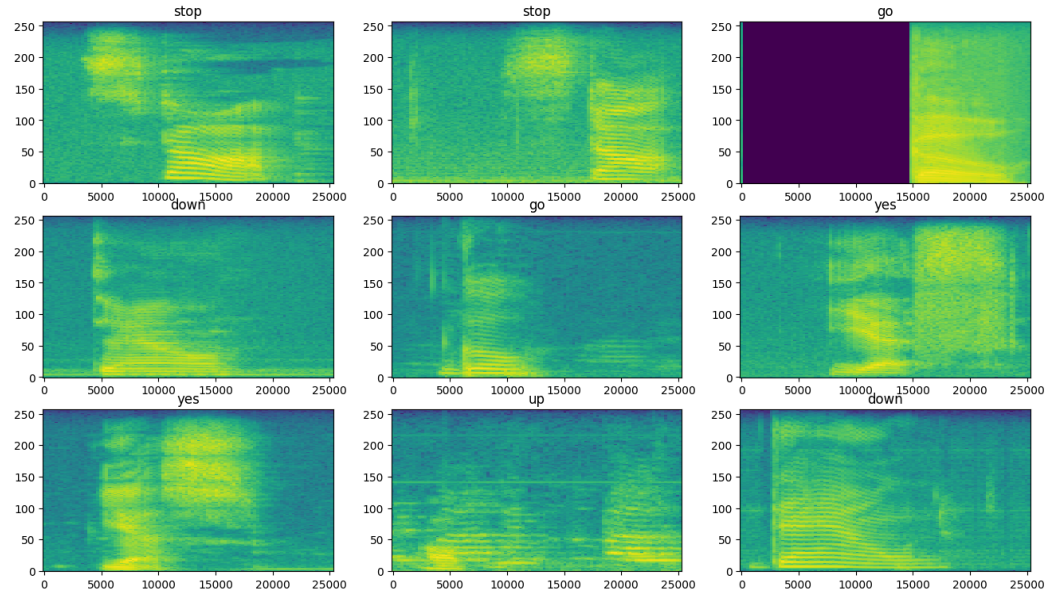


Selain itu, kami juga lampirkan beberapa grafik dari setiap kata dalam bentuk visualisasi waveform dan spectrogram.

Waveforms



Spectrograms



Di dalam visualisasi spectrogram terlihat bagian grafik yang tidak terisi (yang berwarna ungu gelap). Hal ini dikarenakan audio tersebut tidak memiliki suara atau diam.

VIII. Rancangan Model

Terdapat tiga arsitektur model yang kami gunakan untuk perbandingan dua model, yaitu CNN, Dense, dan LSTM. CNN berperan sebagai *feature extraction* dimana model dapat mempelajari fitur melalui visualisasi yang ada di dalam spectrogram. Dense dan LSTM berperan sebagai klasifikator yang akan melakukan klasifikasi visualisasi spectrogram tersebut menjadi label perintah. Data spectrogram akan masuk melalui *input layer* dengan *input shape* yang sesuai. Kemudian akan dilakukan proses *resizing* dan *normalization.* Pada tahap *resizing*, dimensi spectrogram akan diturunkan dari 98 x 257 x 1 menjadi 64 x 64 x 1 untuk mempercepat proses pelatihan. Kemudian, normalisasi dilakukan agar skala pada spectrogram memiliki rentang nilai yang sama.

Arsitektur CNN:

- **Convolutional Layer**: Dua lapisan konvolusi dengan filter berukuran 64 dan 128 menggunakan *kernel size* 3 beserta fungsi aktivasi ReLU.
- **Max Pooling Layer**: Lapisan pooling untuk mengurangi dimensi data dengan nilai maksimum setiap 2 x 2 *pool size*.
- **Dropout Layer**: Lapisan untuk mengurangi overfitting dengan mengabaikan 25% dari neuron selama pelatihan.

Arsitektur Dense:

- **Flatten Layer**: Mengubah peta fitur multidimensi yang dihasilkan oleh lapisan konvolusi menjadi monodimensi sehingga dapat diproses di layer berikutnya.
- **Dense Layer**: Lapisan dense dengan 128 neuron, fungsi aktivasi ReLU, dan regulasi L2 untuk mencegah overfitting.
- **Dropout Layer**: Lapisan untuk mengurangi overfitting dengan mengabaikan 50% dari neuron selama pelatihan.

Arsitektur LSTM:

- **Permute Layer**: Lapisan yang menukar dimensi spasial agar dimensi lebar menjadi dimensi waktu sehingga LSTM dapat dilakukan di sepanjang dimensi lebar spectrogram karena sumbu x atau dimensi lebar pada spectrogram yang menandakan waktu.
- **LSTM Layer**: Menerapkan LSTM pada setiap frame waktu secara independen, dengan 128 unit di setiap LSTM untuk menangkap informasi temporal.
- **Dense Architecture**: Menerapkan arsitektur dense untuk tiga layer terakhir.

Setelah itu, fitur akan menuju output layer yang berupa dense layer terakhir berjumlah dari banyaknya label perintah untuk klasifikasi menggunakan fungsi aktivasi softmax.

Dari arsitektur tersebut, kami membuat dua macam model untuk klasifikasi audio perintah yaitu model CNN-Dense dan CNN-LSTM yang rinciannya terdapat di bawah ini:

1. **CNN - Dense (Biasa disebut CNN saja):**

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_2 (Resizing)       (None, 64, 64, 1)         0
 normalization_2 (Normalizat (None, 64, 64, 1)         3
 ion)
 conv2d_4 (Conv2D)           (None, 62, 62, 64)        640
 conv2d_5 (Conv2D)           (None, 60, 60, 128)       73856
 max_pooling2d_2 (MaxPooling (None, 30, 30, 128)       0
 2D)
 dropout_4 (Dropout)         (None, 30, 30, 128)       0
 flatten_2 (Flatten)         (None, 115200)            0
 dense_4 (Dense)             (None, 128)               14745728
 dropout_5 (Dropout)         (None, 128)               0
 dense_5 (Dense)             (None, 8)                 1032
=================================================================
Total params: 14,821,259
Trainable params: 14,821,256
Non-trainable params: 3
```

2. CNN-LSTM:

Model: "sequential_3"

Layer (type)	Output Shape	Param #
resizing_3 (Resizing)	(None, 64, 64, 1)	0
normalization_3 (Normalization)	(None, 64, 64, 1)	3
conv2d_6 (Conv2D)	(None, 62, 62, 64)	640
conv2d_7 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 30, 30, 128)	0
dropout_6 (Dropout)	(None, 30, 30, 128)	0
permute_1 (Permute)	(None, 30, 30, 128)	0
time_distributed_1 (TimeDistributed)	(None, 30, 128)	131584
flatten_3 (Flatten)	(None, 3840)	0
dense_6 (Dense)	(None, 128)	491648
dropout_7 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 8)	1032

Total params: 698,763

Trainable params: 698,760

Non-trainable params: 3

IX. Proses Pelatihan Model

Untuk proses pelatihan, model dikompilasi menggunakan *Adam optimizer, sparse categorical cross-entropy loss function*, dan *accuracy evaluation metrics*. Setelah itu, kami melatih model dengan *data training* disertai dengan validasi menggunakan data validasi. Setiap model akan dilatih sebanyak 100 *epoch* dengan *early stopping* dan tingkat *patience* 5 *epoch*. Hal ini berarti di setiap *epoch*, seluruh dataset akan dilatih ke dalam model, jika tidak ada perkembangan selama 5 *epoch*, proses pelatihan akan berhenti.

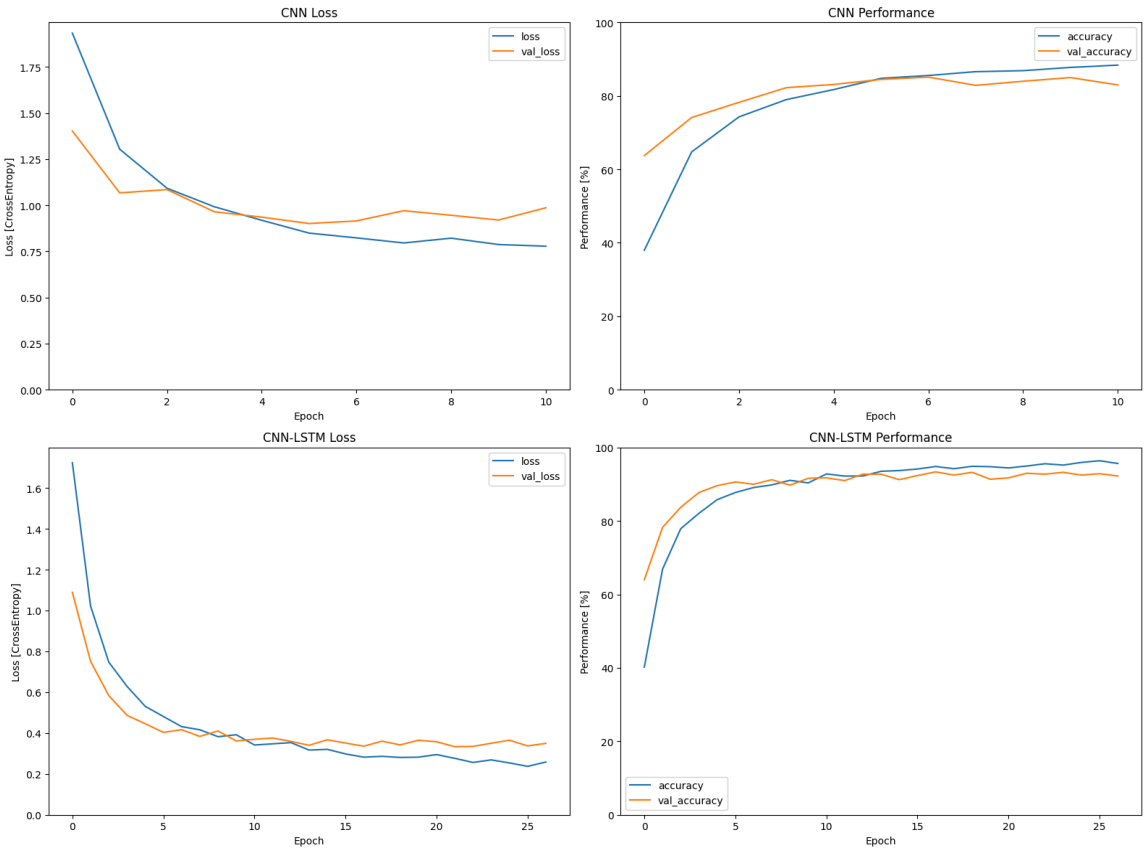
CNN-Dense

```
Epoch 11/100
100/100 [==============================] - 2s 25ms/step - loss: 0.7783 - accuracy: 0.8839 - val_loss: 0.9867 - val_accuracy: 0.8300
Epoch 11: early stopping
```

CNN-LSTM

```
Epoch 27/100 100/100 [==============================] - 3s 32ms/step - loss: 0.2584 - accuracy: 0.9566 - val_loss: 0.3495 - val_accuracy: 0.9225
Epoch 27: early stopping
```

Berikut adalah visualisasi dari Loss dan Performance



X. Hasil Evaluasi dan Analisis

Setelah melakukan evaluasi, kami mendapatkan Classification Report dan Confusion Matrix yang akan dilampirkan di bawah ini

- **Classification Report**

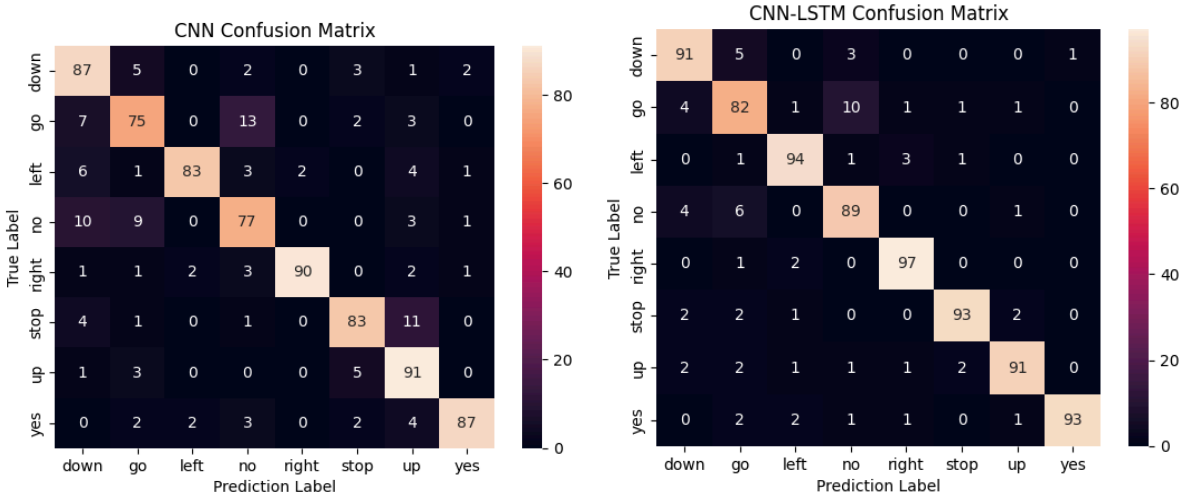
Classification Report CNN

	precision	recall	f1-score	support
down	0.75	0.87	0.81	100
go	0.77	0.75	0.76	100
left	0.95	0.83	0.89	100
no	0.75	0.77	0.76	100
right	0.98	0.90	0.94	100
stop	0.87	0.83	0.85	100
up	0.76	0.91	0.83	100
yes	0.95	0.87	0.91	100
accuracy			0.84	800
macro avg	0.85	0.84	0.84	800
weighted avg	0.85	0.84	0.84	800

Classification Report CNN-LSTM

	precision	recall	f1-score	support
down	0.88	0.91	0.90	100
go	0.81	0.82	0.82	100
left	0.93	0.94	0.94	100
no	0.85	0.89	0.87	100
right	0.94	0.97	0.96	100
stop	0.96	0.93	0.94	100
up	0.95	0.91	0.93	100
yes	0.99	0.93	0.96	100
accuracy			0.91	800
macro avg	0.91	0.91	0.91	800
weighted avg	0.91	0.91	0.91	800

- **Confusion Matrix**



Berdasarkan confusion matrix dan classification report, model CNN-LSTM menunjukkan performa yang lebih baik dibandingkan model CNN. Hal ini terlihat dari peningkatan nilai precision, recall, dan f1-score pada hampir semua kelas. Model CNN-LSTM memiliki akurasi keseluruhan sebesar 91%, sedangkan model CNN hanya 84%. Kelas *"left"*, *"right"*, dan *"yes"* menunjukkan peningkatan signifikan pada model CNN-LSTM, dengan precision dan recall yang lebih tinggi dibandingkan model CNN. Sebagai contoh, kelas "right" memiliki f1-score 0.96 pada model CNN-LSTM dibandingkan 0.94 pada model CNN, menunjukkan bahwa model CNN-LSTM lebih efektif dalam mengenali pola-pola dalam data. Diantara delapan kelas, yang memiliki kemungkinan kegagalan klasifikasi tertinggi adalah kelas *"go"* yang sering diklasifikasikan menjadi kelas *"no"* di kedua model. Hal ini dapat disebabkan oleh struktur suara yang mirip antara kedua kelas tersebut.

XI. Kesimpulan dan Saran

Model CNN-LSTM memberikan performa yang lebih baik daripada model CNN dalam klasifikasi dataset ini, dengan peningkatan akurasi dan metrik evaluasi lainnya. Hal ini dikarenakan sifat LSTM yang dapat menangkap informasi temporal. Informasi tersebut sangat penting dalam sebuah data audio sebab pada nyatanya, phoneme yang dikeluarkan dari suara memiliki relasi terhadap phoneme sebelum dan berikutnya yang biasa disebut dengan *context dependent*.

Saran untuk perbaikan lebih lanjut termasuk mengoptimalkan hyperparameter dan mencoba teknik ekstraksi data untuk meningkatkan performa model lebih jauh. Selain itu, mengeksplorasi arsitektur model yang lebih kompleks seperti *transformer* yang mungkin memberikan hasil yang lebih baik dengan harapan dapat meningkatkan kemampuan model dalam mengklasifikasikan data secara lebih akurat.

XII. Lampiran

Berikut adalah notebook code yang sudah diberi comment dan dieksekusi, serta riset lebih lanjut yang serupa:

- ***Notebook Code*:** https://github.com/vanssnn/command-audio-classification-cnn-lstm/blob/main/main.ipynb
- **Riset Serupa:** https://paperswithcode.com/dataset/speech-commands