Report Date: 07/15/2022
To: ematson@purdue.edu, ahsmith@purdue.edu, lhiday@purdue.edu, and lee3450@purdue.edu
From: What is today's lunch?

- Ilmun Ku(mun90505@hufs.ac.kr)
- Seungyeon Roh(shtmddus99@konkuk.ac.kr)
- Gyeongyeong Kim(kky57389@sunmoon.ac.kr)
- Charles Taylor(taylo869@purdue.edu)

Summary

Spectrogram Augmentation (MFCCs, Mel, Chroma, Tonnetz, Contrast) was conducted to utilize Deep Learning models to classify payload more accurately. Based on these data, Convolutional Neural Network(CNN) and Recurrent Neural Network(RNN) were conducted.

Data collection was held in a changed data collection method i.e., drone hovering in order that the quality of the audio data make higher.

What 'What is today's lunch?' completed this week:

- **Spectrogram Augment with new dataset**
  - Time Masking & Frequency masking (MFCCs, Mel, Chroma, Tonnetz, Contrast)

Figure 1. Time masking and Frequency masking

Spectrogram Augment was conducted. Two methods (Time masking, Frequency masking) will be utilized to classify payload with Deep Learning models.

```python
def freq_augment(spec: np.ndarray, num_mask=2,
                 freq_masking_max_percentage=0.15, time_masking_max_percentage=0.3):

    spec = spec.copy()
    for i in range(num_mask):
        all_frames_num, all_freqs_num = spec.shape
        freq_percentage = random.uniform(0.0, freq_masking_max_percentage)

        num_freqs_to_mask = int(freq_percentage * all_freqs_num)
        f0 = np.random.uniform(low=0.0, high=all_freqs_num - num_freqs_to_mask)
        f0 = int(f0)
        spec[:, f0:f0 + num_freqs_to_mask] = 0

    return spec
```

```python
def time_augment(spec: np.ndarray, num_mask=2,
                 freq_masking_max_percentage=0.15, time_masking_max_percentage=0.3):

    spec = spec.copy()
    for i in range(num_mask):
        all_frames_num, all_freqs_num = spec.shape
        time_percentage = random.uniform(0.0, time_masking_max_percentage)

        num_frames_to_mask = int(time_percentage * all_frames_num)
        t0 = np.random.uniform(low=0.0, high=all_frames_num - num_frames_to_mask)
        t0 = int(t0)
        spec[t0:t0 + num_frames_to_mask, :] = 0

    return spec
```

Figure 2. Time masking and Frequency masking code

- **Spectrogram Augment with new dataset Augmented data to json file**

```python
# loop through all labels
for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
    if dirpath is not dataset_path:
        dirpath_components = dirpath.split("/")
        semantic_label = dirpath_components[-1]
        dirpath_part = dirpath.split('/')
        label_idx = 0
        current_label = ""
        for i in labelList:
            if i in dirpath_part:
                label_idx = labelList.index(i)
                current_label = i
                if current_label not in data["mapping"]:
                    data["mapping"].append(current_label)
        print("\nProcessing{}, current_label:{}" .format(semantic_label,current_label ))

        print(f'dirpath: {dirpath}, dirnames:{dirnames}, label_idx:{label_idx}, current_label:{current_label}')
        for f in filenames:
            file_path = os.path.join(dirpath, f)
            signal, sr = librosa.load(file_path, sr = SAMPLE_RATE)

            for s in range(num_segments):
                start_sample = num_samples_per_segment * s
                finish_sample = start_sample + num_samples_per_segment

                chroma = librosa.feature.chroma_stft(signal, sr=sr, n_fft=2048)
                time_aug = time_augment(chroma)
                chroma = time_aug.T

                # store chroma for segment if it has the expected length
                if len(chroma) == expected_num_chroma_vectors_per_segment:
                    data['chroma_timeaugment'].append(chroma.tolist()) # numpy array -> list

                    data['labels'].append(label_idx)
                    print("{}, segment:{}, dirnames:{}, label_idx:{}, current_label:{}" .format(file_path, s+1, dirnames, label_idx, current_label))
# save as json
with open(json_path, "w") as fp: # file path
    json.dump(data, fp, indent = 4)
```

Figure 3. Feature Extraction and Data Augmentation code

After Raw data augmentation, Spectrogram augmentation, it is required to save as json file of augmented datasets.

- **CNN model for UAV payload detection has been built using new dataset.**
  - The CNN Model has recorded reasonable performance.

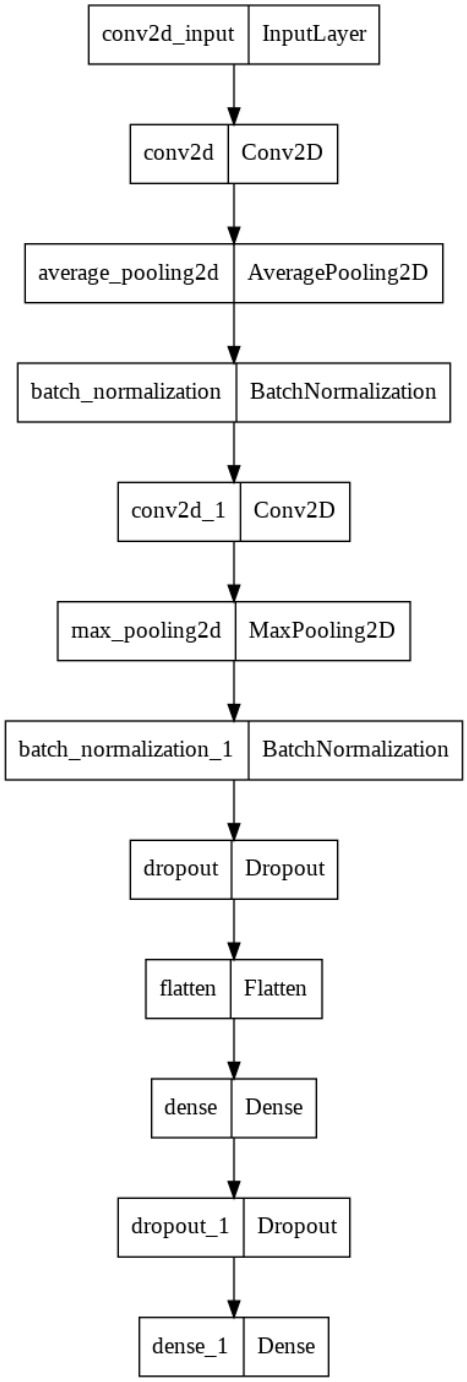

Figure 4. CNN Architecture for UAV Payload Detection

Table 1. Test Accuracy for CNN

Feature	Accuracy
MFCCs	0.74
Mel	0.69
Chroma_stft	0.62
Contrast	0.56
Tonnetz	0.61
Augmented MFCCs	0.78

- **RNN model for UAV payload detection has been built using new dataset.**
  - The RNN Model has recorded reasonable performance.

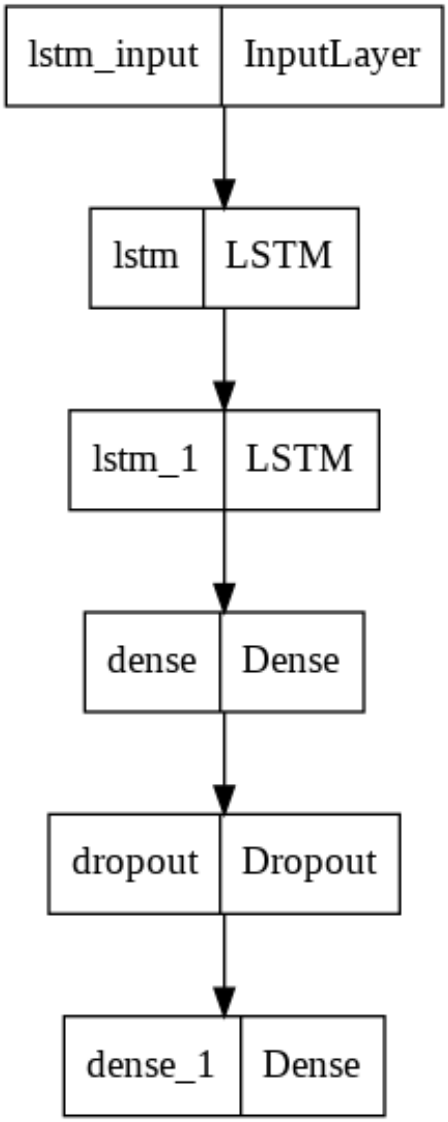

Figure 5. CNN Architecture for UAV Payload Detection

Feature	Accuracy
MFCCs	0.69
Mel	0.65
Chroma_stft	0.55
Contrast	0.62

Table 2. Test Accuracy for RNN

Things to do by next week

- Build CNN model using another augmented audio feature
- Going on a data collection trip with Mia.

Problems or challenges:

- Model performance of classifying 1 payload and 2 payloads is relatively low. Further data collection is required.

References