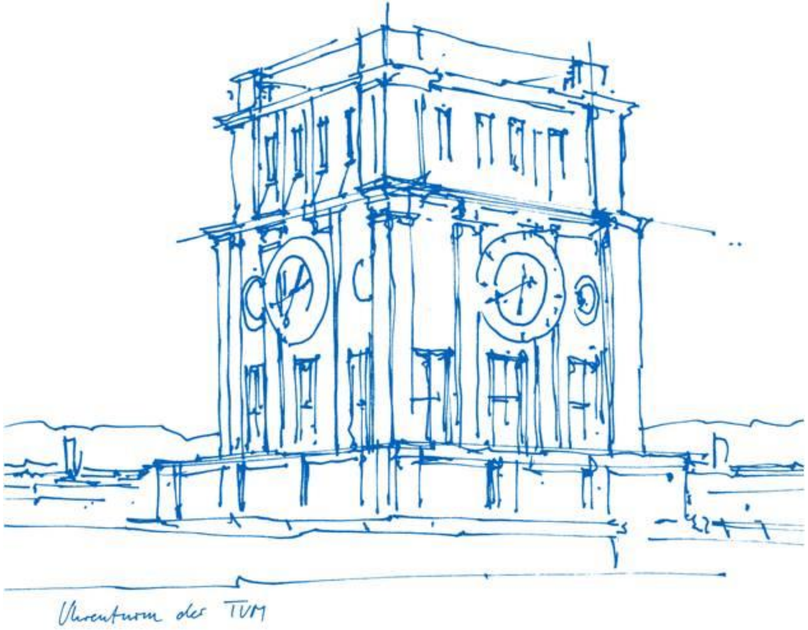
Sprint 1 Presentation

Ahmed Kaddah, Shao Jie Hu Chen, Marlon Müller

Edge Computing and the Internet of Things

Technische Universität München

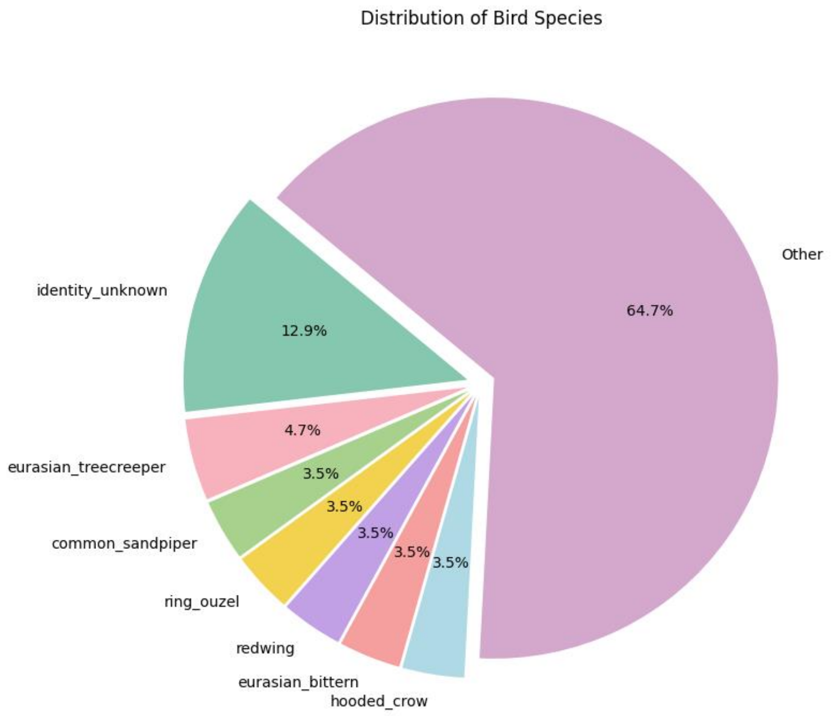
München, 24.11.2023



Dataset

```yaml
query:
  area: europe
  grp: "1"  # birds
  cnt: germany
  # loc: bavaria
  # box: LAT_MIN,LON_MIN,LAT_MAX,LON_MAX
  # lic: ""       # license
  # q: ">C"      # quality
  len: "5"       # length (s)
  # smp: ""      # sampling rate
  # since: ""    # upload date
```

```
file_name,class_id,class,xeno_id,slice,quality,length,sampling_rate
x-808740-1-0.mp3,1,eurasian_treecreeper,808740,0,B,0:05,48000
x-643995-0-0.mp3,0,common_sandpiper,643995,0,B,0:05,44100
x-829354-0-0.wav,0,common_sandpiper,829354,0,B,0:05,22050
x-583385-0-0.mp3,0,common_sandpiper,583385,0,A,0:05,44100
x-643994-2-0.mp3,2,ring_ouzel,643994,0,B,0:05,44100
x-295611-1-0.mp3,1,eurasian_treecreeper,295611,0,D,0:05,48000
x-695404-2-0.mp3,2,ring_ouzel,695404,0,C,0:05,44100
```



Preprocessing

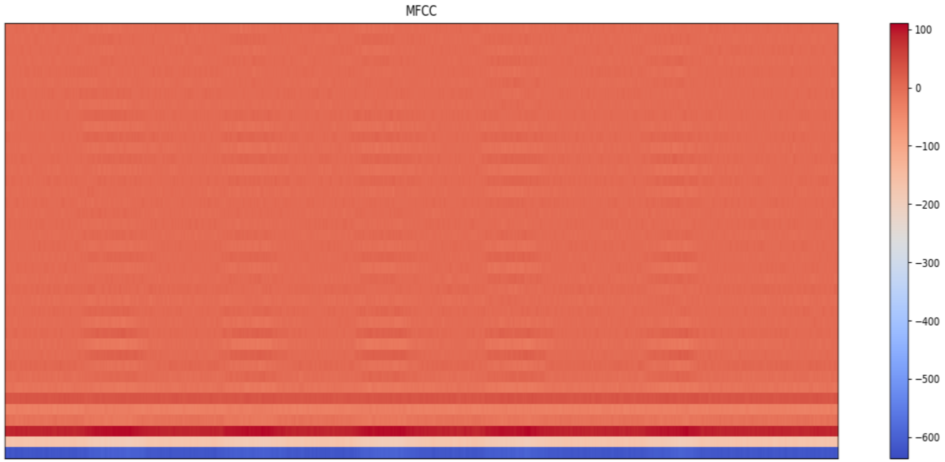
```
// Framing computation

for (size_t i = 0; i<num_frames; ++i) {
    size_t ref_frame = i * HOP_LENGTH_SAMPLES;

    for (size_t j=0; j < WIN_LENGTH_SAMPLES; ++j) {
        fft_operand[2*j] = wav_values[ref_frame + j] * window[j]; // Re
        fft_operand[2*j + 1] = 0; // Im
    }

    ret = dsps_fft2r_fc32(fft_operand, WIN_LENGTH_SAMPLES);
    ret = dsps_bit_rev_fc32(fft_operand, WIN_LENGTH_SAMPLES);

    if (ret != ESP_OK) {
        ESP_LOGE(PREPROCESS_TAG, "Error FFT computation");
        return ret;
    }

    // Power spectrum computation

    // Notice that we only need the first half of the FFT result,
    // since the second half has the same values but in reverse order.

    for (size_t j=0; j < NUM_FFT/2 + 1; ++j) {
        float re_part = fft_operand[2*j]; // Re
        float im_part = fft_operand[2*j + 1]; // Im

        float power = re_part * re_part + im_part * im_part;

        power_spectrum[j] = power;
    }

    // Mel filter bank computation
```

Espressif DSP Library

ESP-DSP is the official DSP library for ESP32 and ESP32-S3 chips.

Overview

ESP-DSP is intended to be used as an ESP-IDF component. For the introduction to ESP-IDF, refer to the ESP-IDF Programming Guide.



Model

```python
class CustomModel(nn.Module):

    def __init__(self, input_size, num_classes):
        super(CustomModel, self).__init__()

        print(f"Size of input: {input_size}")
        channels, height, width = input_size


        # Features (note: no support for nn.Sequential)
        self.conv1 = nn.Conv2d(channels, 16, kernel_size=5, stride=2, padding=3)
        self.relu1 = nn.ReLU(inplace=False)
        self.pool1 = nn.MaxPool2d((2, 1), stride=(2, 1))
        self.conv2 = nn.Conv2d(16, 16, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU(inplace=False)
        self.pool2 = nn.MaxPool2d((3, 2), stride=(3, 2))
        self.conv3 = nn.Conv2d(16, 8, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU(inplace=False)
        self.pool3 = nn.MaxPool2d((3, 1), stride=(3, 1))

        # Classifier
        self.fc1 = nn.Linear(self._calculate_feature_size(channels, height, width), 128)
        self.relu4 = nn.ReLU(inplace=False)
        self.fc2 = nn.Linear(128, num_classes)
```

PyTorch

ONNX

ESPRESSIF

```c
const static __attribute__((aligned(16))) int8_t _conv2_conv_filter_element[] = {
        70,    64,    66,    81,    84,    69,    78,    78,    74,    68,    77,    77,    67,    65,    71,
       -79,    76,    71,    69,    83,   -70,    69,   -83,    64,   -64,    75,    65,   -71,    77,   -68,
        74,    68,   -83,    81,   -75,   -73,    77,    82,    64,   -77,   -83,   -67,   -73,    77,    76,
       -63,    71,   -64,    68,   -69,   -75,   -73,    81,   -66,   -77,   -64,   -74,   -83,   -79,   -70,
       -63,   -80,    69,    72,   -68,   -69,   -80,   -64,   -71,   -77,    78,   -69,   -81,   -68,    66,
```

Objectives for Sprint 2

In general, assemble a **functional prototype**

- Integrate auxiliary data (e.g., ESC-50), resample (and split) audio files
- Build, train, and evaluate a first complete dataset
- Interconnect devices and enable sensor functionality
- Resolve MFCC discrepancies and execute the trained model on the ESP32
- (Optional) Start the development of a website/dashboard for visualization