## Exercises

**1. Replace freefield1010 with warblrb to train the CNN. Is the accuracy improved or worsened?**

**freefield1010**

```
* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 16, 188, 1), dtype=tf.float32, name='keras_tensor_24')
Output Type:
  TensorSpec(shape=(None, 2), dtype=tf.float32, name=None)
Captures:
  136917097541520: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097545360: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097531344: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097542288: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097544784: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097545552: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097543632: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097544976: TensorSpec(shape=(), dtype=tf.resource, name=None)
TFLite model saved as "bird_activity_simplified_with_weights.tflite"
Final Validation Accuracy: 0.7789
```

**warblrb**

```
* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 16, 188, 1), dtype=tf.float32, name='keras_tensor')
Output Type:
  TensorSpec(shape=(None, 2), dtype=tf.float32, name=None)
Captures:
  135341194566672: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341198800080: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049154064: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049162512: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049165200: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049152720: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049155024: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341049164432: TensorSpec(shape=(), dtype=tf.resource, name=None)
TFLite model saved as "bird_activity_warblrb10k_with_weights.tflite"
Final Validation Accuracy: 0.7556
```

When we train the CNN on Freefield1010, the accuracy is a bit higher (around 77.9%) compared to when we use Warblrb10k (around 75.6%). The main reason is that the two datasets are very different in quality and difficulty. Freefield1010 mostly contains outdoor environmental recordings where the difference between bird and non-bird sounds is clearer. In contrast, Warblrb10k was collected from many different people, and it often includes background noise like cars, people talking, or dogs barking, which makes it harder for the model to separate bird sounds from everything else.

Another factor is the labels. Freefield1010 labels are more carefully checked, so they are usually correct. Warblrb10k uses crowd-sourced labels, which means sometimes the “bird” tag is used even when the bird sound is very faint or unclear. This confuses the model and lowers its accuracy. The datasets are also different in balance: Freefield1010 has more non-bird clips, while Warblrb10k has more bird clips. Even if we try to adjust for this, the imbalance can still affect performance.

Finally, the CNN we are using is very simple. It only has a couple of small convolutional layers, which is enough for a cleaner dataset like Freefield1010 but not powerful enough to

deal with the messier Warblrb10k recordings. In short, Freefield1010 is like an easy test with clear answers, while Warblrb10k is a harder test with more distractions. That's why the accuracy drops when we switch datasets.

**2. Use the TinyChirp dataset to train the CNN. This dataset has fixed train/validate/test splits. What kind of modifications are needed to properly train your CNN?**

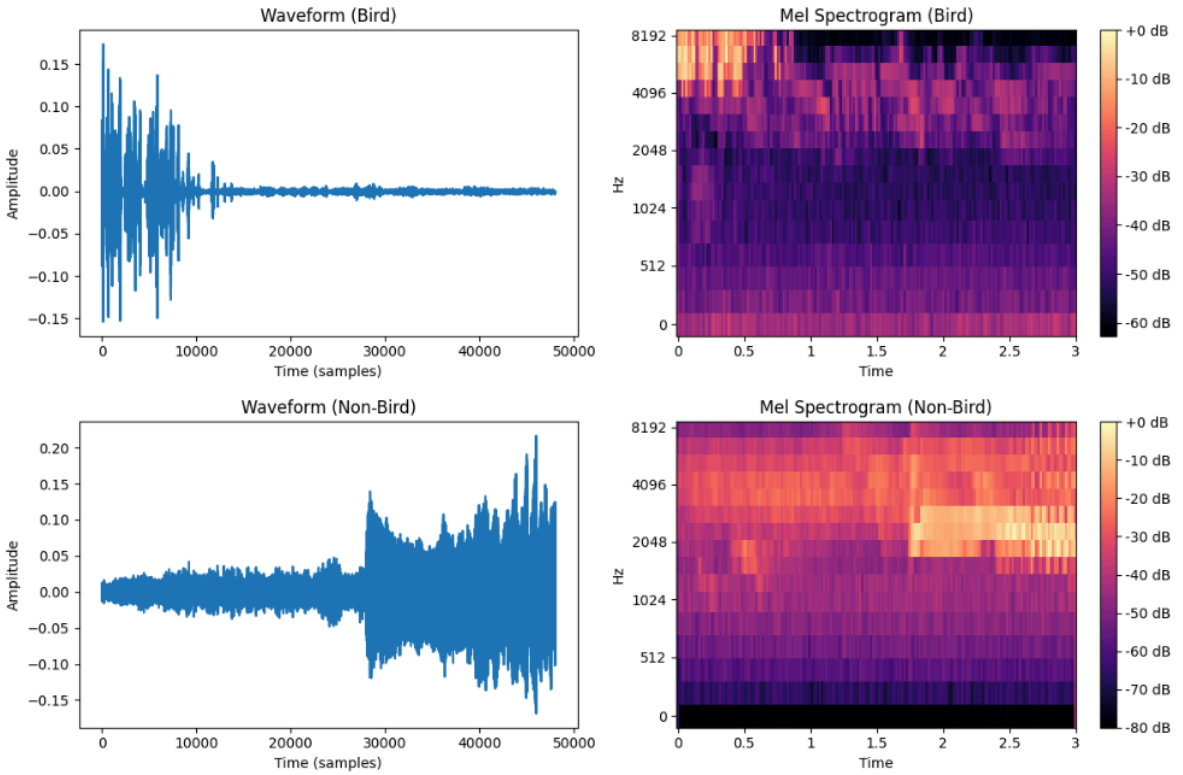
```
Loading split: validate from /content/tiny_chirp/validation

Loading split: test from /content/tiny_chirp/testing
Train: (11292, 16, 188, 1), Val: (1380, 16, 188, 1), Test: (1392, 16, 188, 1)
Training model for 10 epochs...
Epoch 1/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 16s 39ms/step - accuracy: 0.7584 - loss: 0.7102 - val_accuracy: 0.8891 - val_loss: 0.2974
Epoch 2/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 20s 39ms/step - accuracy: 0.8760 - loss: 0.2882 - val_accuracy: 0.9167 - val_loss: 0.2117
Epoch 3/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 20s 57ms/step - accuracy: 0.8982 - loss: 0.2422 - val_accuracy: 0.9500 - val_loss: 0.1622
Epoch 4/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 14s 39ms/step - accuracy: 0.9123 - loss: 0.2045 - val_accuracy: 0.9377 - val_loss: 0.1560
Epoch 5/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 20s 38ms/step - accuracy: 0.9351 - loss: 0.1653 - val_accuracy: 0.9529 - val_loss: 0.1391
Epoch 6/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 18s 52ms/step - accuracy: 0.9413 - loss: 0.1465 - val_accuracy: 0.9638 - val_loss: 0.1002
Epoch 7/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 18s 47ms/step - accuracy: 0.9518 - loss: 0.1235 - val_accuracy: 0.9623 - val_loss: 0.1168
Epoch 8/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 18s 38ms/step - accuracy: 0.9553 - loss: 0.1187 - val_accuracy: 0.9652 - val_loss: 0.1159
Epoch 9/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 19s 54ms/step - accuracy: 0.9587 - loss: 0.1033 - val_accuracy: 0.9659 - val_loss: 0.0994
Epoch 10/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 24s 64ms/step - accuracy: 0.9628 - loss: 0.0950 - val_accuracy: 0.9746 - val_loss: 0.0771
Final Test Accuracy: 0.9612
```

The TinyChirp dataset requires several modifications to properly train the CNN. First, the dataset already provides fixed splits into "training," "validation," and "testing," so we must load data directly from these folders instead of creating our own random splits. Second, the labels are organized into folders named "target" (bird) and "non_target" (non-bird), so the code must be adapted to map these folder names into numeric labels (e.g., 1 and 0). Third, unlike Freefield1010, no metadata file is needed, and class weights are not required because the dataset is smaller and more balanced. Apart from these changes, the preprocessing steps such as padding/trimming audio and converting to Mel spectrograms remain the same.

The TinyChirp dataset produces high accuracy (around 96%) mainly because of its structure and size. First, it is a curated dataset with clear and consistent recordings of "target" (bird) and "non_target" sounds, making classification easier compared to noisier, real-world datasets like Freefield1010. Second, the dataset is relatively small and balanced, so the CNN can learn patterns quickly without facing extreme class imbalance. Third, the audio clips are already trimmed and well-prepared, reducing background noise and irrelevant audio that could confuse the model. As a result, the CNN achieves high accuracy since the task is simpler and the dataset is clean.

**3. Load a few audio files and plot their waveforms and Mel spectrograms side by side. Describe what differences you observe between bird and non-bird clips.**




The comparison between bird and non-bird clips in both waveform and Mel spectrogram highlights clear differences. For the bird clip, the waveform shows short, sharp spikes at the beginning that quickly fade out, which correspond to distinct chirps or whistles produced by birds. Its Mel spectrogram displays structured horizontal frequency bands, concentrated at specific ranges, and these patterns appear more organised and repetitive, reflecting the regular nature of bird calls.

In contrast, the non-bird clip has a waveform that looks flatter and more continuous, with gradually increasing energy and no distinct spikes. This indicates background noise or other environmental sounds rather than short chirps. Its Mel spectrogram shows diffuse and spread-out energy across many frequencies, without clear, repetitive patterns. Instead of clean lines, the energy is more random and scattered, which is typical of noise, wind, or human-made sounds. Overall, the bird sounds are characterised by clean and structured frequency patterns, while the non-bird sounds are noisier, more irregular, and lack distinct structure.

**4. Add another convolutional layer to the CNN. What happens the accuracy? Can we keep on adding layers?**

**Baseline CNN (2 Convolutional Layers) – Validation Accuracy: 77.9%**

```
* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 16, 188, 1), dtype=tf.float32, name='keras_tensor_24')
Output Type:
  TensorSpec(shape=(None, 2), dtype=tf.float32, name=None)
Captures:
  136917097541520: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097545360: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097531344: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097542288: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097544784: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097545552: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097543632: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136917097544976: TensorSpec(shape=(), dtype=tf.resource, name=None)
TFLite model saved as "bird_activity_simplified_with_weights.tflite"
Final Validation Accuracy: 0.7789
```

**CNN with Added 3rd Convolutional Layer (No Regularization) – Validation Accuracy: 74.8%**

```
* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 16, 188, 1), dtype=tf.float32, name='keras_tensor_83')
Output Type:
  TensorSpec(shape=(None, 2), dtype=tf.float32, name=None)
Captures:
  135341035185808: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017012304: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341035189840: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341035181200: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341035188688: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135340851246288: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135340851246864: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135340851244752: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135340851248016: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135340851246480: TensorSpec(shape=(), dtype=tf.resource, name=None)
TFLite model saved as "bird_activity_simplified_with_weights.tflite"
Final Validation Accuracy: 0.7484
```

**Corrected 3-Layer CNN (Batch Normalization + Dropout) – Validation Accuracy: 82.9%**

```
* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 16, 188, 1), dtype=tf.float32, name='keras_tensor_58')
Output Type:
  TensorSpec(shape=(None, 2), dtype=tf.float32, name=None)
Captures:
  135341017014224: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017014800: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017007504: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017015184: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017009040: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017007888: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017011728: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017006352: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017007696: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017012112: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017007312: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017006736: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017006544: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017017872: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017009232: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017016528: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017009616: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017010768: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017013264: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017016144: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017008656: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135341017012496: TensorSpec(shape=(), dtype=tf.resource, name=None)
TFLite model saved as "bird_activity_simplified_with_weights.tflite"
Final Validation Accuracy: 0.8290
```

A baseline CNN with two convolutional layers was initially trained on the Freefield1010 dataset, achieving a validation accuracy of 77.9%. Each epoch completed relatively quickly due to the smaller number of parameters. When a third convolutional layer was added without Batch Normalization or Dropout, the training accuracy increased, but the validation accuracy dropped to 74.8%. This decline was caused by overfitting, as the deeper network increased the number of parameters, and unstable gradients occurred due to the lack of

regularization. The longer per-epoch training time also highlighted the added computational cost of the deeper network.

A corrected three-layer CNN was then constructed, with Batch Normalization applied after each convolutional layer and Dropout added after Flatten and Dense layers. The Dense layer size was slightly increased. This configuration stabilized training, mitigated overfitting, and allowed the network to extract more complex hierarchical features from the Mel spectrograms. The network achieved a validation accuracy of 82.9% after 10 epochs, with each epoch taking longer than the original two-layer model due to the increased computational workload.

Regarding further layer additions, each new convolutional layer increases network capacity and the potential to capture richer features, but without sufficient data or proper regularization, overfitting and unstable training are likely. When layers are added carefully with stabilization and regularization techniques, deeper networks can improve validation accuracy. Otherwise, indiscriminate stacking of layers usually reduces performance. In conclusion, the effect of adding layers depends on network design, dataset size, and regularization; with proper design, additional layers enable richer feature extraction and potential accuracy improvement, albeit at the cost of longer training time.

### 4.1 Reflection

- **Importance of stratified splitting for unbalanced datasets:**
  Stratified splitting ensures that both the training and validation sets maintain the same class distribution as the original dataset. This prevents scenarios where one class is underrepresented in either set, which could lead to biased training and misleading validation accuracy. By preserving the class proportions, the model is trained and evaluated more reliably.

- **Effect of class weights on CNN training with unbalanced data:**
  Class weights adjust the loss contribution of each class during training, giving more importance to minority classes. This prevents the network from being biased toward the majority class, improving learning for underrepresented classes and producing a more balanced performance across all classes. Without class weights, the CNN might achieve high overall accuracy by mainly predicting the majority class, masking poor performance on the minority class.

- **Why adding a channel dimension is necessary for CNN input:**
  Convolutional layers expect input with a channel dimension (e.g., (height, width, channels)), even for single-channel data. Simply reshaping the array without adding a channel dimension would result in a shape incompatible with convolution operations. By explicitly adding the channel dimension, the network can correctly apply filters across the spatial dimensions, enabling feature extraction from the input data.

Colab:
https://colab.research.google.com/drive/1elDAfwBT_wS2gn0wXZWxfOSAbeV60qjv?usp=sharing