Echoes of the Forest

Advancing Capuchin Bird Conservation through Deep Learning Audio Classification

Convolutional Neural Network for Capuchin Bird Call Recognition

Our Team

Sunidhi Goyal

(myp8ma)

Michael Hammer

(fhu9hn)

Karolina Straznikiewicz

(ssj9sw)

Motivation

Capuchin birds face threats such as habitat loss, climate change, and human activities. Accurately identifying and monitoring Capuchin bird populations through their vocalizations can aid conservation efforts by identifying population trends, habitat preferences, and areas requiring protection.

Training Data

- 217 parsed audio clips (.wav), each between 2-5 seconds long, that include specific bird calls from Capuchin birds.
- 593 parsed audio clips (.wav), each around 3 seconds long, that include sounds of other animals and birds.

Testing Data

- 100 audio recordings in MP3 format, each approximately 3 minutes in length. Recordings feature a mixture of Capuchin bird vocalizations alongside various background sounds, including other bird species, and environmental noises.

Strategy

TRAINING DATA

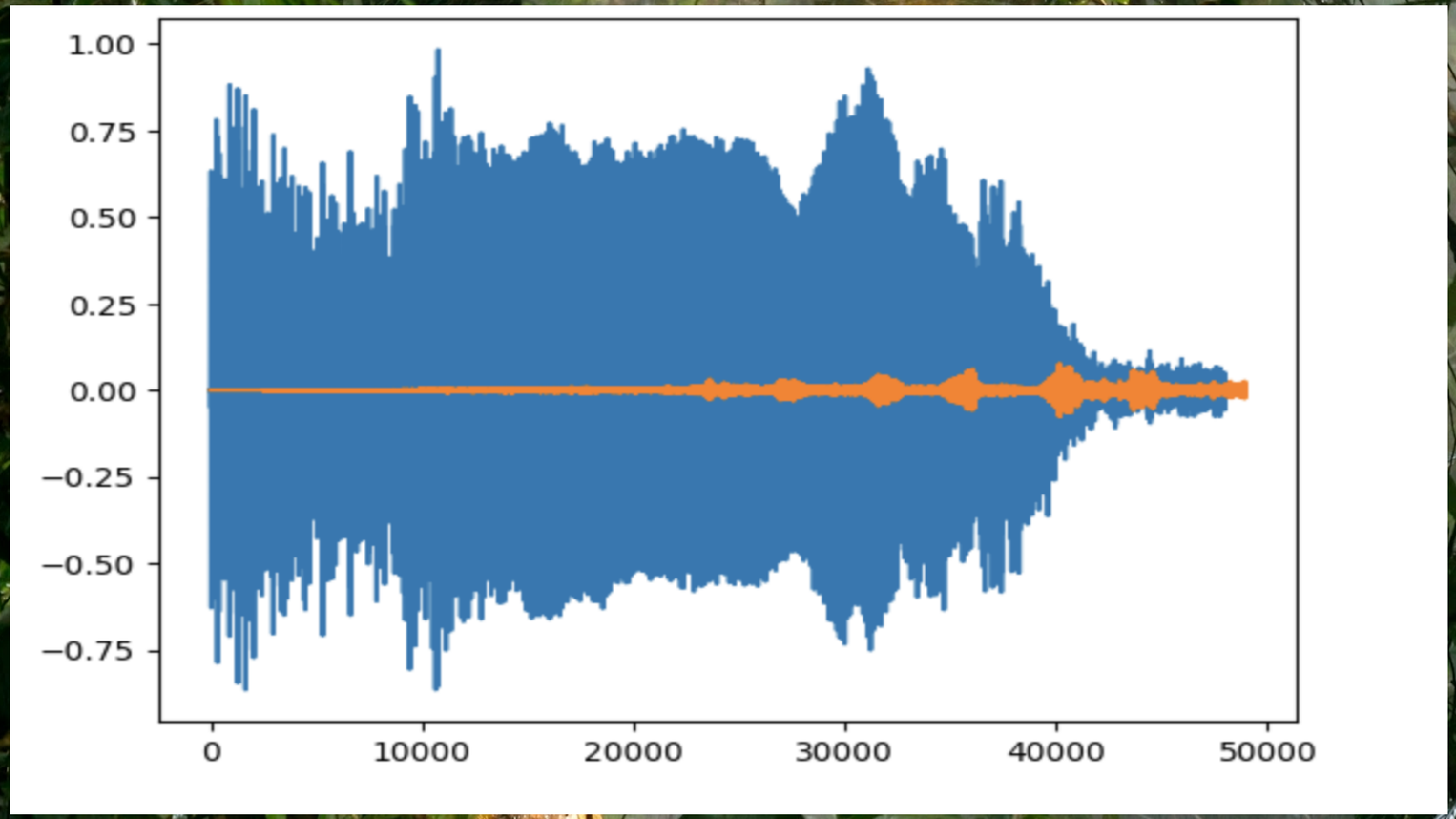
- Convert wav files into 16k mono
- Convert wav files into Short Term Fourier Transformations -> Spectrograms
- Feeding spectrogram images in the model and handling the parameters by making them 48000 Hz audio files and padding the rest with 0s, in order to tackle time and resources during training
- Training model on capuchin call positive and negative spectrograms
- Assessing the evaluation metrics

Strategy

TESTING DATA

- Convert longer mp3 files into wav format and then STFT spectrograms
- Slicing the longer wav files into shorter windows, and then running the model on those windows
- This is done to find the number of times a Capuchin bird call is heard in the entire long audio file
- Assessing the results

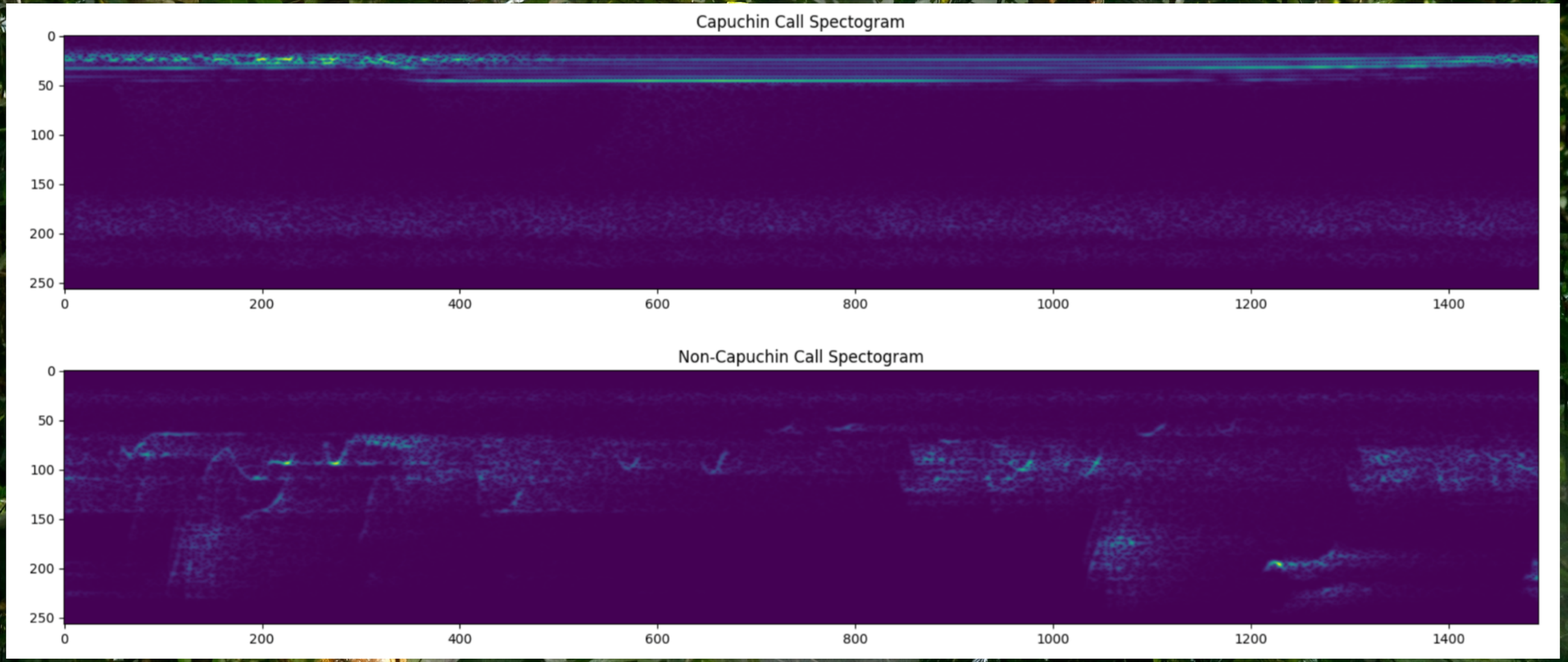
What do Wav Files Look Like?



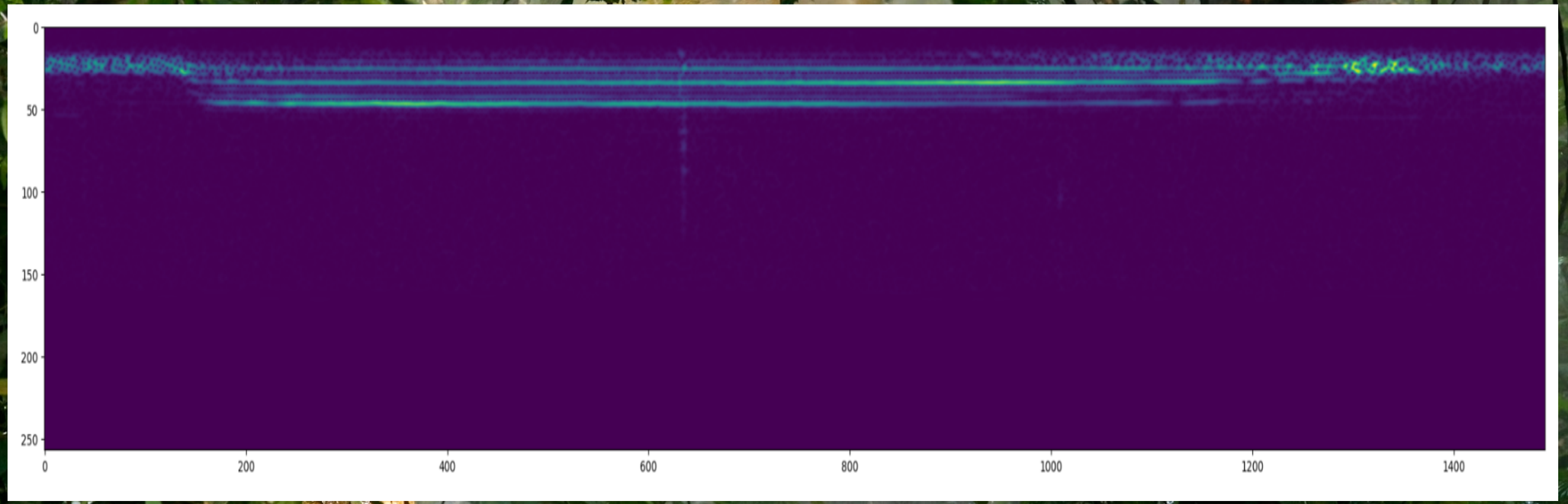
Capuchin Call

Non-Capuchin Call

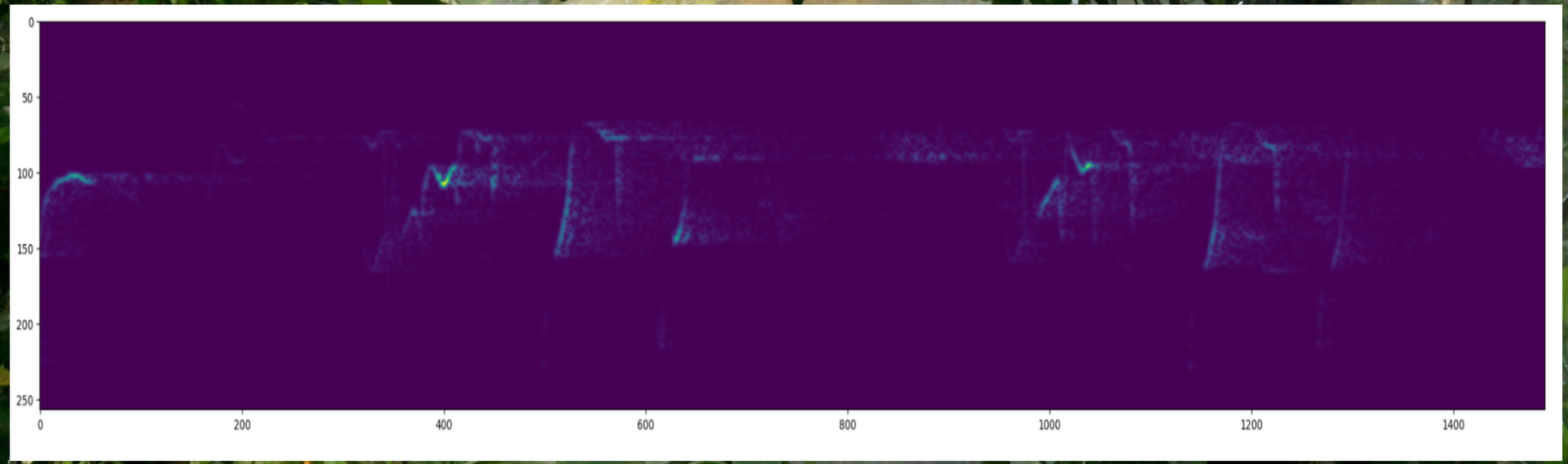
Spectrograms



Capuchin Positive Spectrogram Example

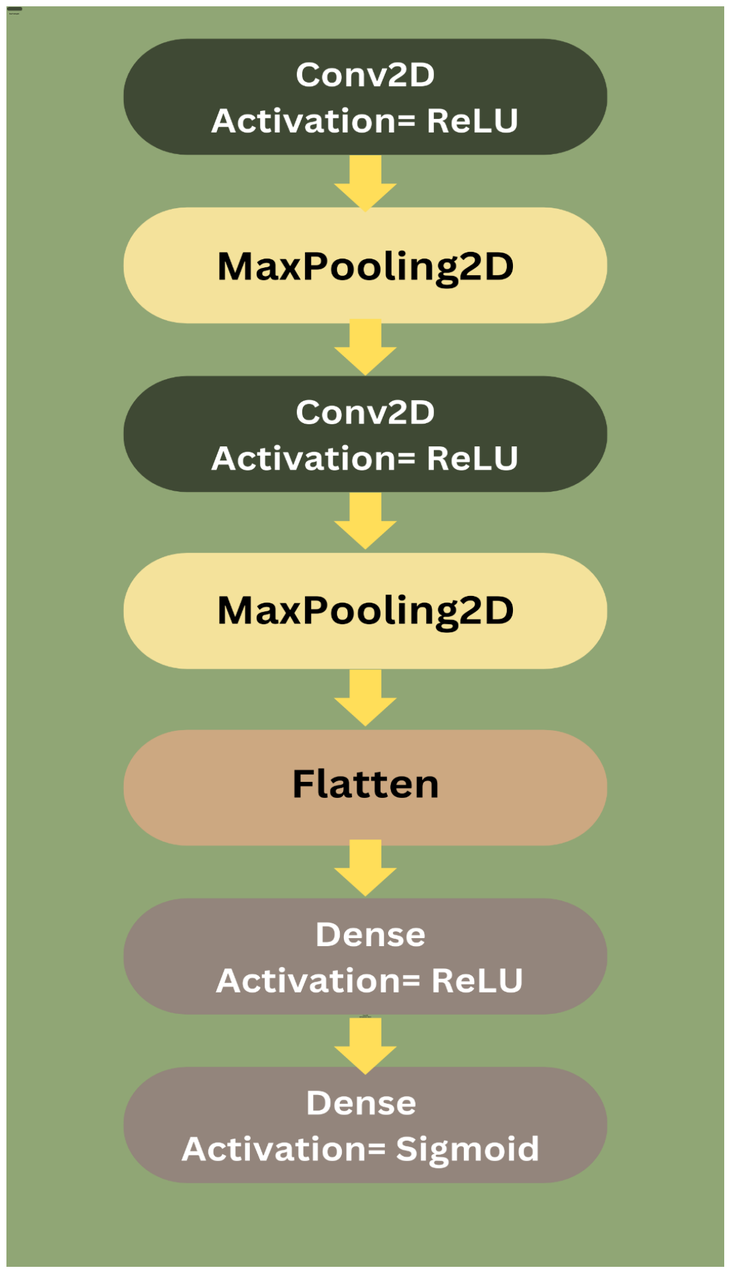


Capuchin Negative Spectrogram Example



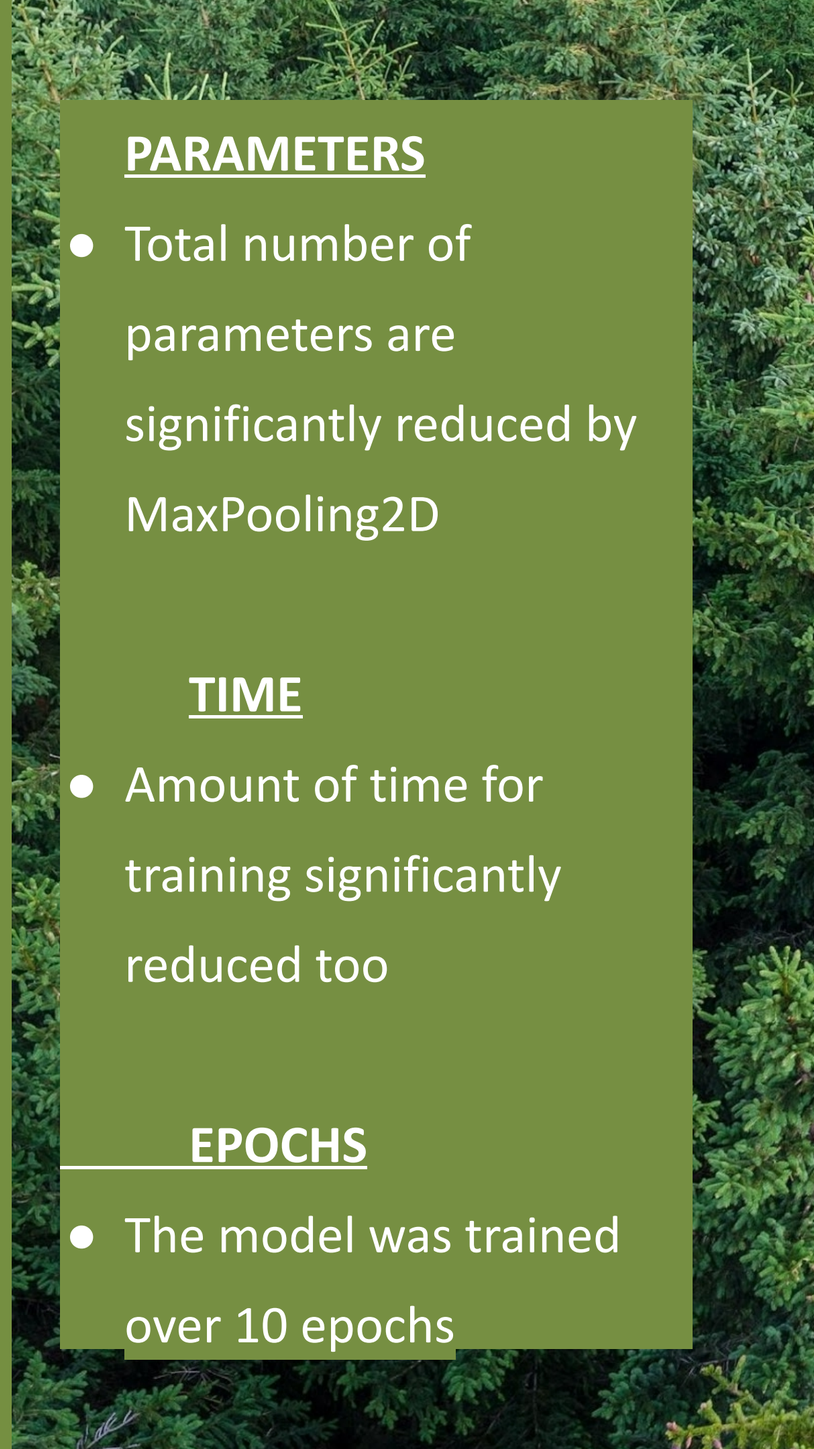
Model

MODEL ARCHITECTURE



DESCRIPTION

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 1489, 255, 16)     160

 max_pooling2d_7 (MaxPoolin  (None, 744, 127, 16)      0
 g2D)

 conv2d_16 (Conv2D)          (None, 742, 125, 16)      2320

 max_pooling2d_8 (MaxPoolin  (None, 371, 62, 16)       0
 g2D)

 flatten_9 (Flatten)         (None, 368032)            0

 dense_17 (Dense)            (None, 64)                23554112

 dense_18 (Dense)            (None, 1)                 65

=================================================================
Total params: 23556657 (89.86 MB)
Trainable params: 23556657 (89.86 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



PARAMETERS

- Total number of parameters are significantly reduced by MaxPooling2D

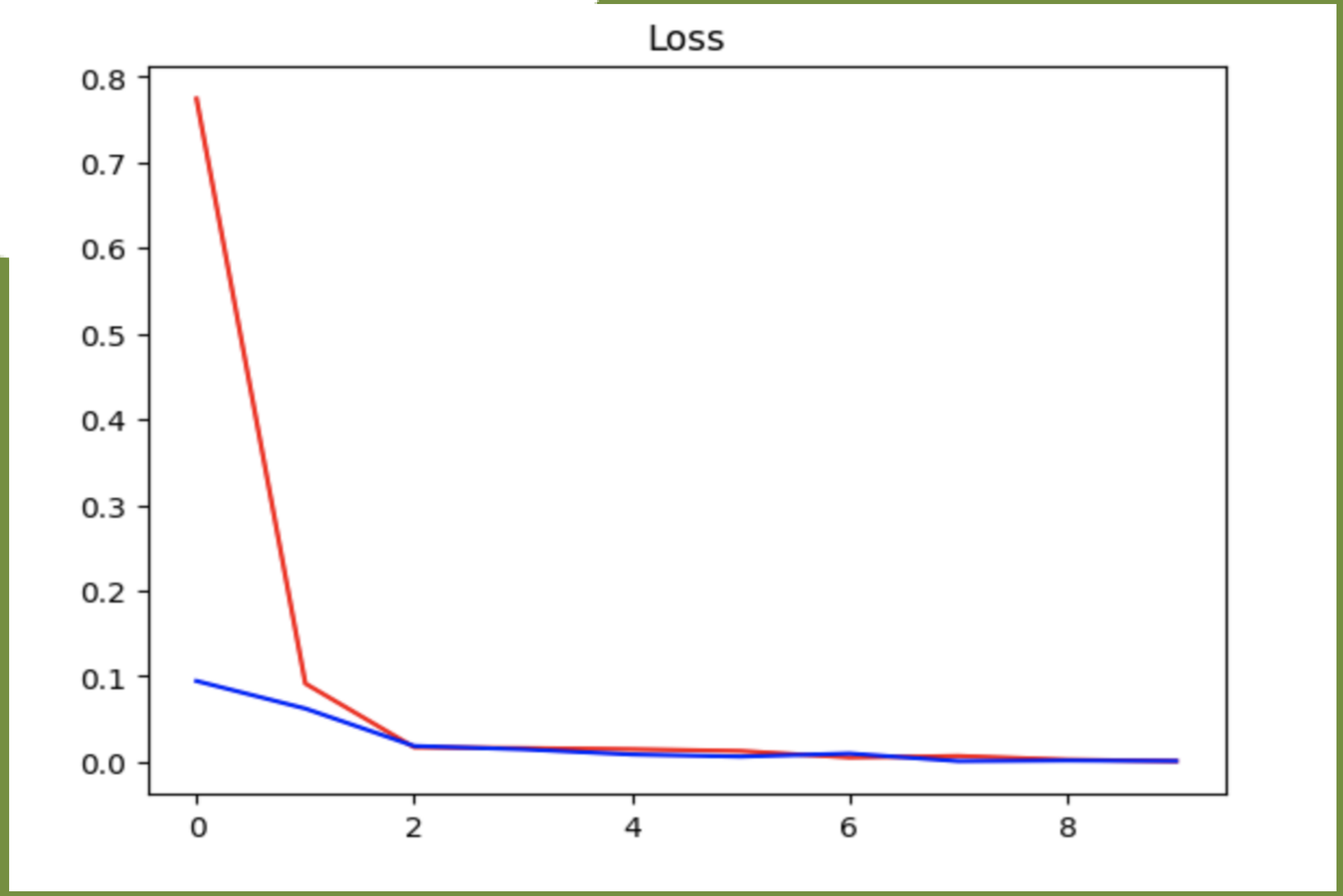
TIME

- Amount of time for training significantly reduced too

EPOCHS

- The model was trained over 10 epochs

LOSS WHILE TRAINING

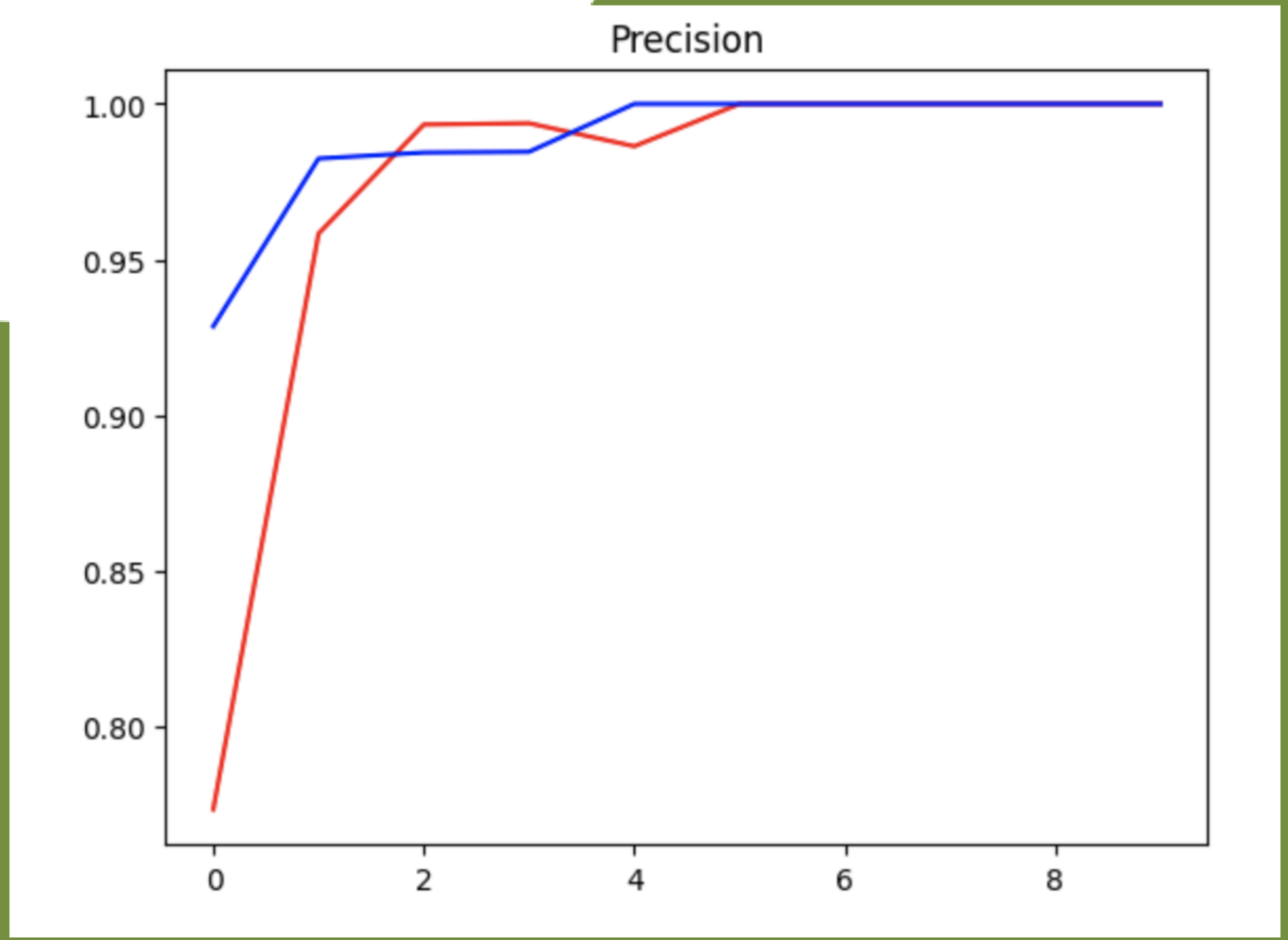


Loss could be observed converging towards 0 after a few epochs which indicated good training

Training

Validation

PRECISION

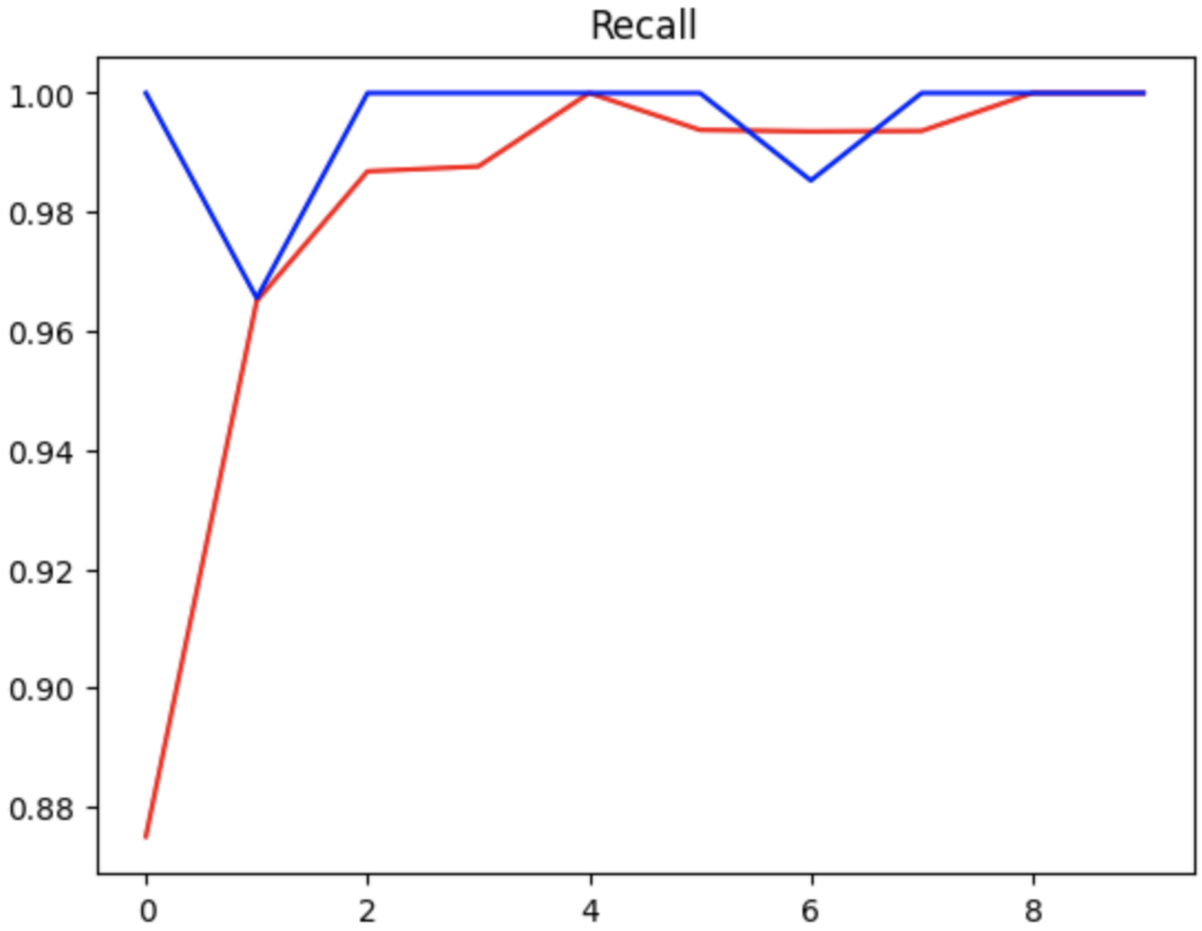


High Precision, converging towards 1 over the span of training was observed. Indicator of a good model!

Training

Validation

RECALL



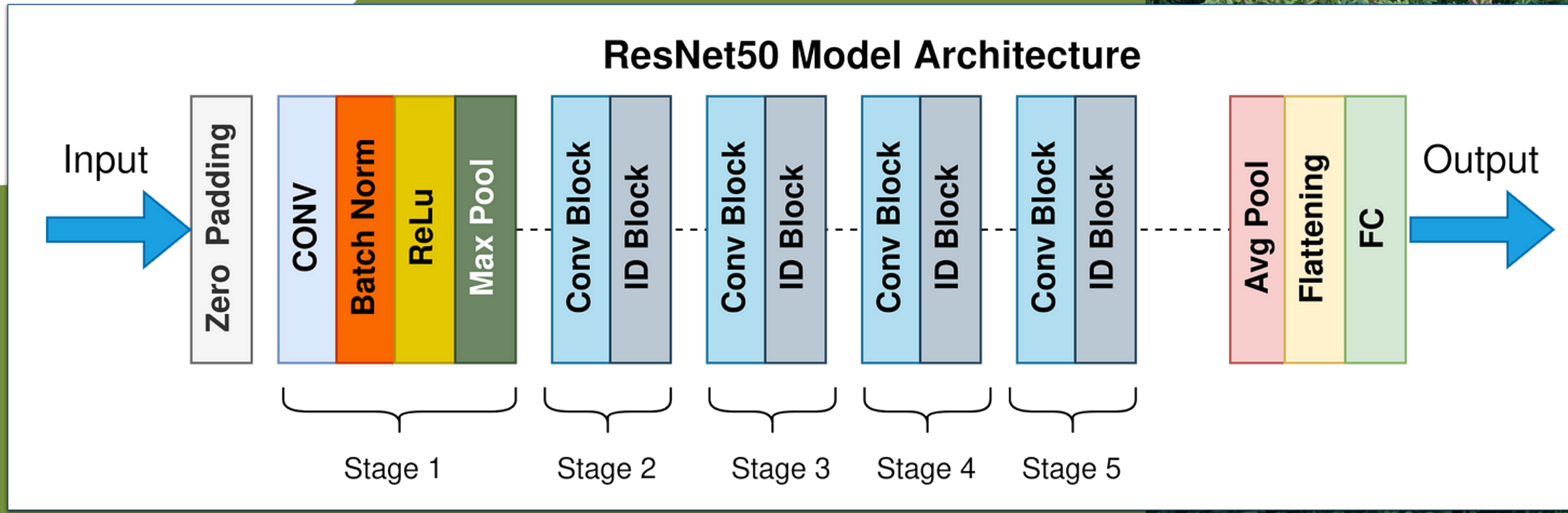
High Recall, converging towards 1 over the span of training was observed. Also an indicator of a good model!

Training

Validation

Model #2

MODEL ARCHITECTURE



MODEL SELECTION

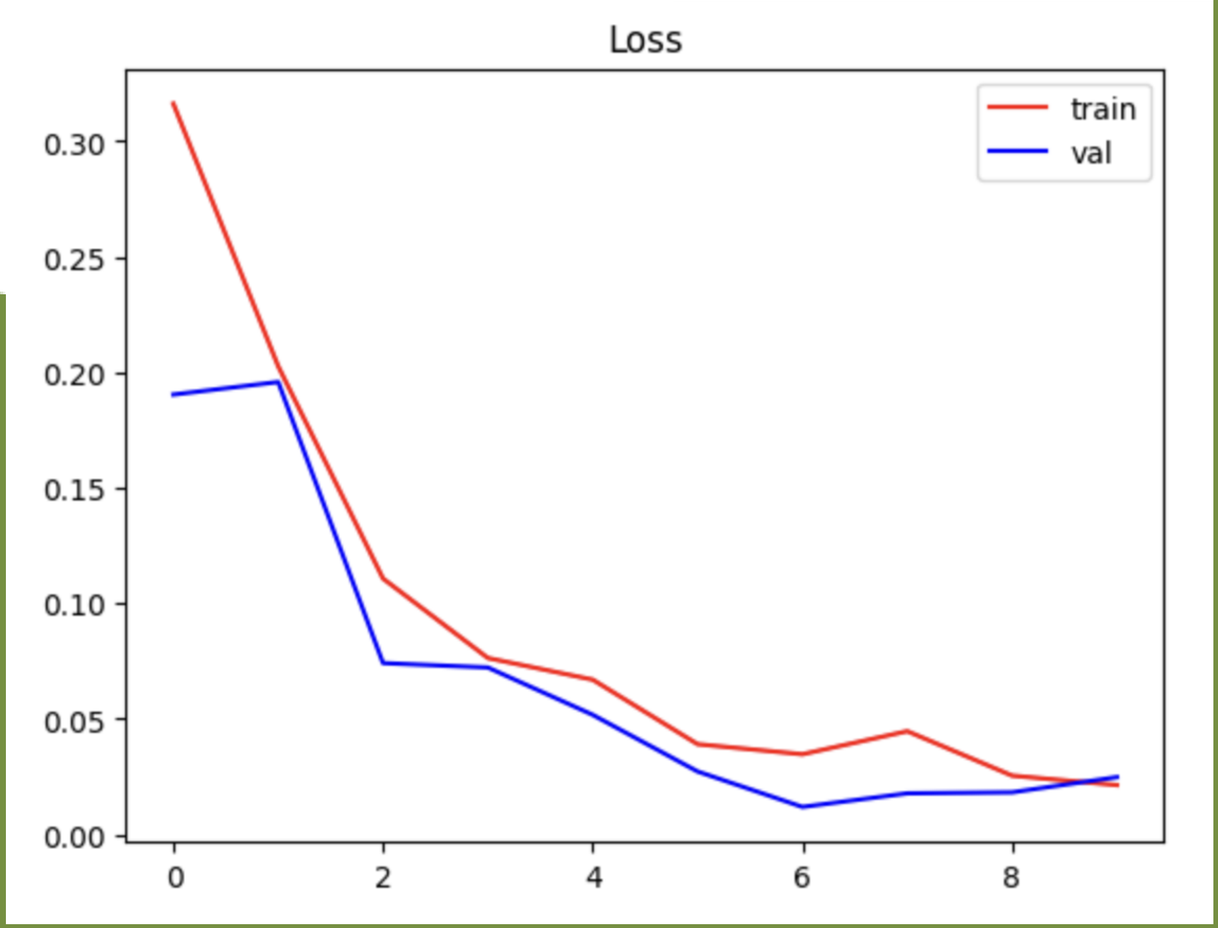
- Problem: most pre-trained models are trained on RGB input data

- Two possible solutions:
  - modify existing networks or pre-process input data
  - Audio classification-specific networks trained on spectrograms

DESCRIPTION

```
Model: "ResNet50_modified"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 1489, 255, 3)      30

 max_pooling2d_2 (MaxPoolin  (None, 744, 127, 3)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 742, 125, 3)       84

 max_pooling2d_3 (MaxPoolin  (None, 371, 62, 3)        0
 g2D)

 resnet50 (Functional)       (None, 47, 9, 2048)       23587712

 flatten_1 (Flatten)         (None, 49152)             0

 dense_2 (Dense)             (None, 32)                1572896

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 25160755 (95.98 MB)
Trainable params: 1573043 (6.00 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
```

LOSS WHILE TRAINING



Loss trends towards 0, relatively stable after ~ 5 epochs. Doesn’t perform quite as well as the first model

DALL-E struggled.

Results

Converting Logits to Classes

```
[ ]  yhat = [1 if prediction > 0.5 else 0 for prediction in yhat]
```

```
yhat
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1]
```

```
[ ]  tf.math.reduce_sum(yhat)
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=5>
```

5 capuchins were heard in that clip!!!

```
tf.math.reduce_sum(y_test)
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=5.0>
```

5 capuchins were heard in that clip too!!! -> Accurate prediction

Each forest recording clip is analyzed for how many Capuchin calls are recognized

Both models performed similarly in terms of identifying recordings with low / high Capuchin occurences

Conclusions

Tensor dimensions will be the death of me.

CNN models can have real practical uses in wildlife conservation of Capuchin birds by providing accurate species recognition.

Models can identify Capuchin birds with high accuracy

Pretrained image classifiers can be modified to work well with audio data

Thank You!

References

Renotte, N. (2022, April 16). *Build a deep audio classifier with python and tensorflow*. YouTube. https://www.youtube.com/watch?v=ZLIPkmmDJAc