AUDIO CLASSIFICATION

CSCI 6907: DEEP LEARNING AND NEURAL NETWORKS

PROJECT TEAM

Shambhavi Adhikari (G37903602)

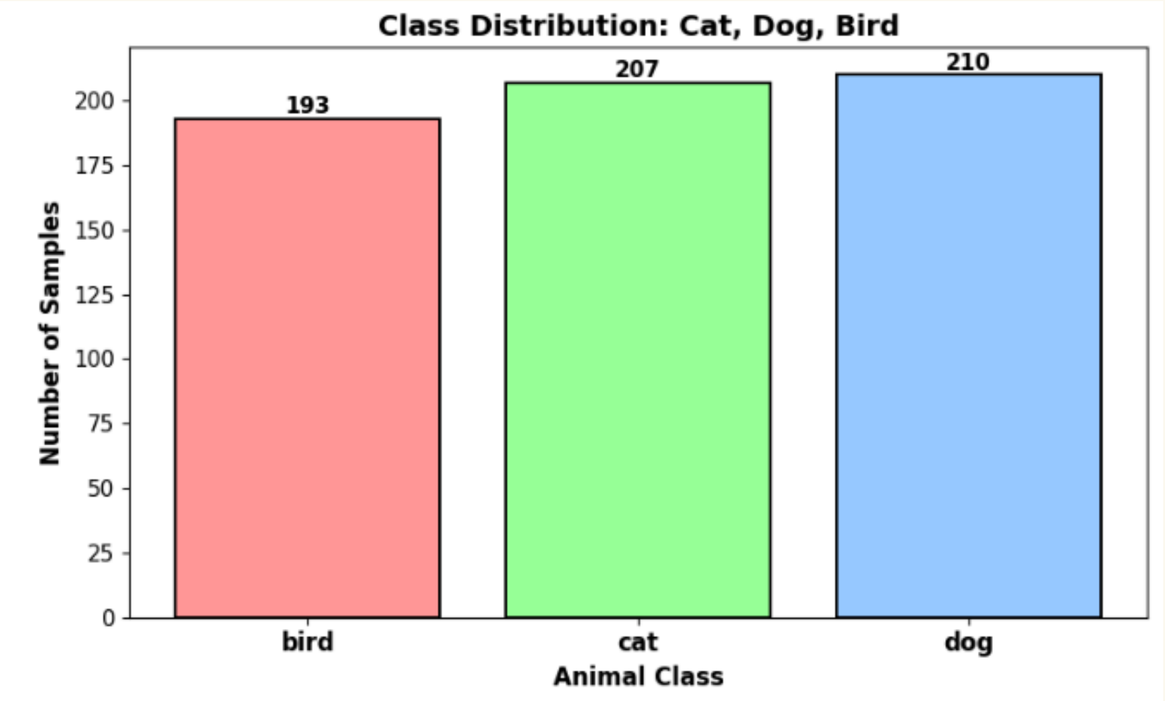
Rakshitha Mamilla (G23922354)

Abhiyan Sainju (G22510509)

Project Goal: The goal of this project is to classify cat, dog, and bird sounds using EDA-driven preprocessing, a fine-tuned 2D CNN baseline, and transfer learning with YAMNet.

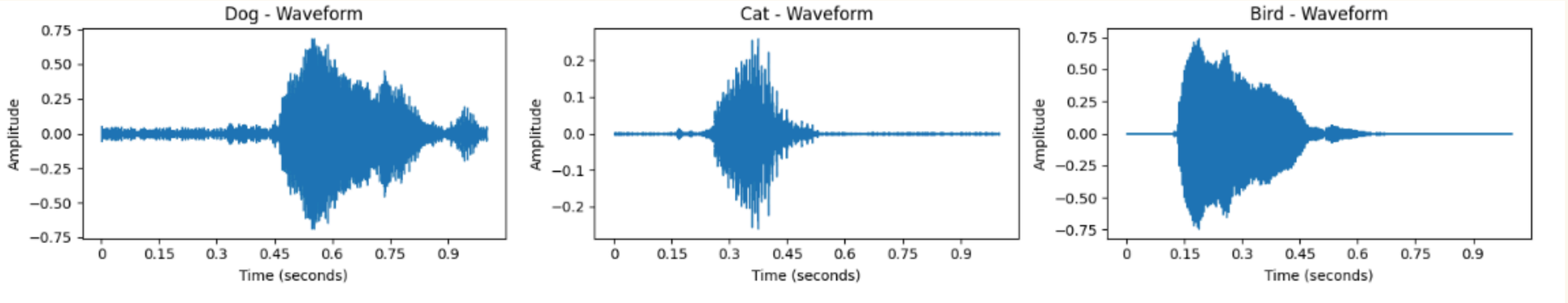
DATASET

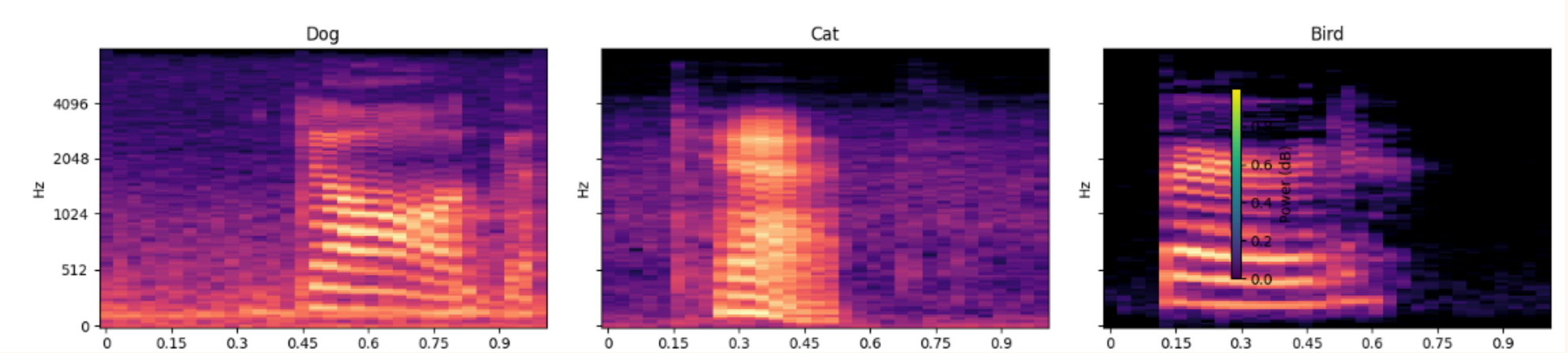
- High-quality WAV recordings of humans verbally saying the words “dog,” “cat,” and “bird”, provided with clean labels for supervised audio classification.



Link: Kaggle Dataset

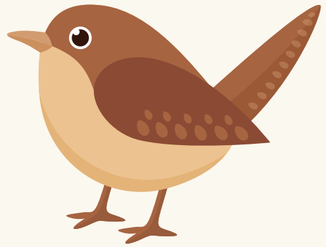
EDA



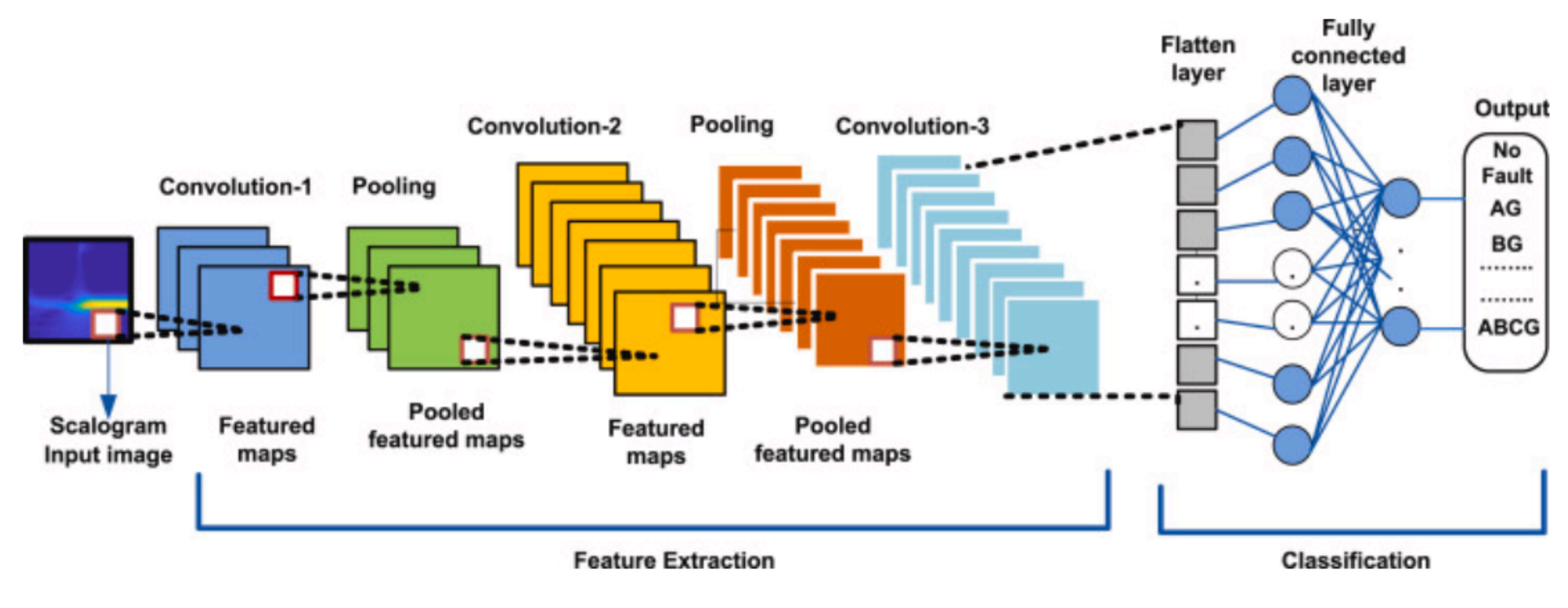


Seeing the waveforms and especially the spectrograms convinced us to treat this as an image problem: the time–frequency patterns are very visual, so we chose Mel-spectrograms plus a 2D CNN as our main model, which ultimately gave us the best performance.

AUDIO CLASSIFICATION



WHAT IS 2D CNN

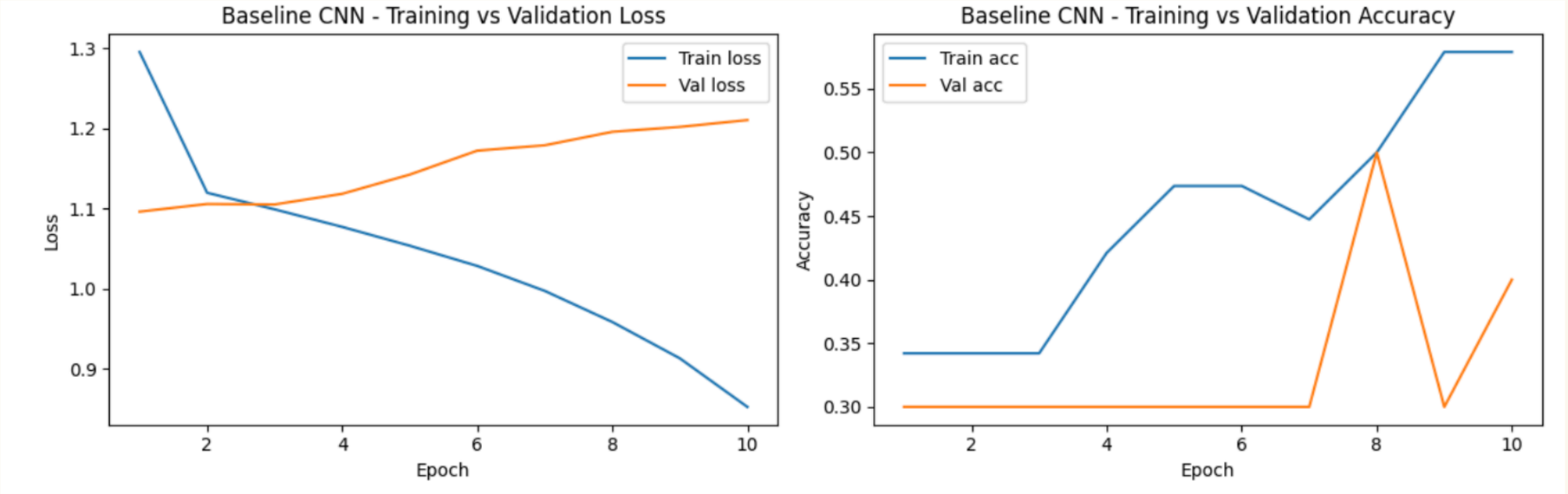


OUR BASELINE CNN MODEL

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 64)	4,194,368
dense_1 (Dense)	(None, 3)	195

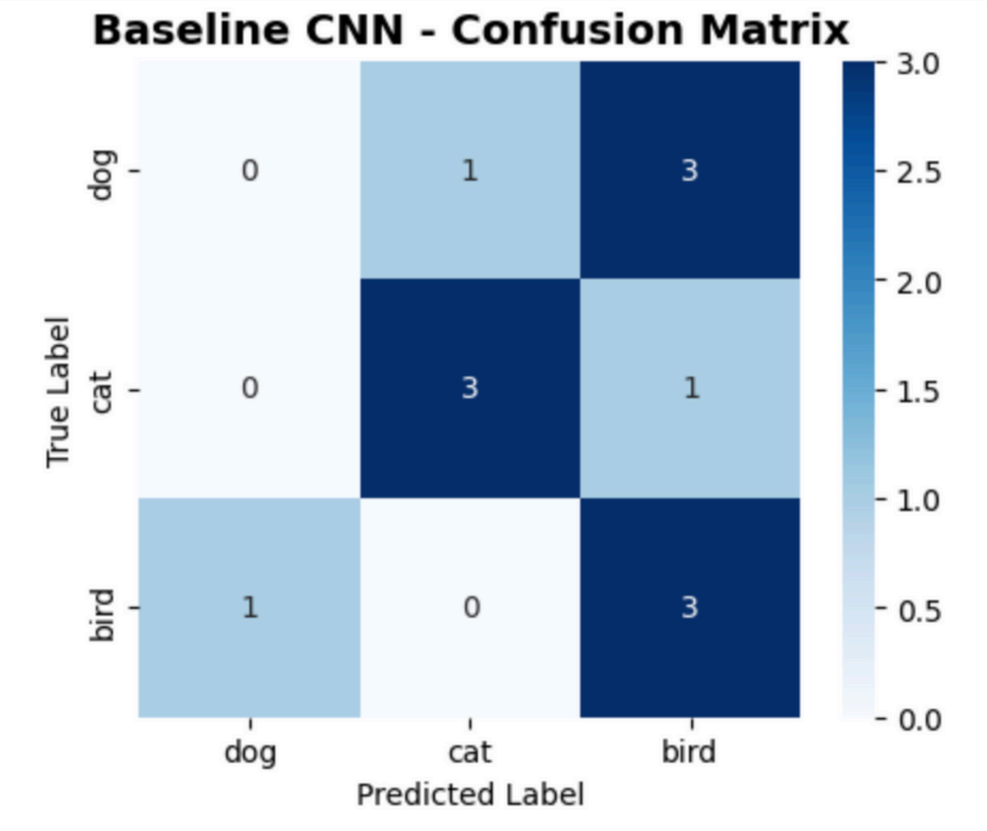
We converted audio into 128×128 Mel-spectrograms and trained a simple CNN to classify dog, cat, and bird sounds using train, validation, and test splits. The test accuracy was 0.33, as we only used 60 audio files.

OUR BASELINE CNN MODEL



Our baseline CNN could memorize the training set, but the validation curve was almost a flat line at chance level. That told us this architecture and data setup was not good enough yet.

OUR BASELINE CNN MODEL



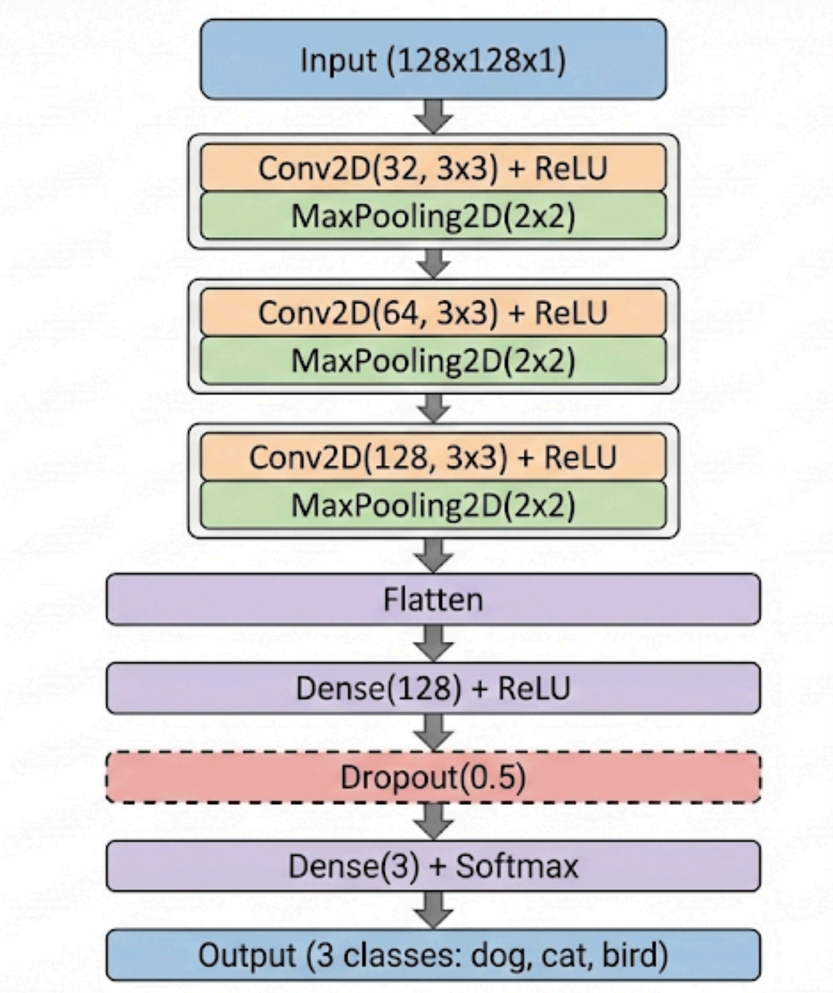
- The confusion matrix told us that the model was basically a ‘always say cat’ classifier. It completely ignored dogs and struggled to separate cats from birds.

- So this confusion matrix shows that our baseline CNN was systematically biased toward one class. That’s what motivated the improved CNN + Dropout model that finally reached ~88% test accuracy

FULL DATASET TRAINING CNN MODEL

Why this architecture?

- Treat each audio clip as a 128×128 Mel-spectrogram image.
- 3 Conv + MaxPool blocks (32→64→128): learn low → mid → high-level time–frequency patterns for dog / cat / bird.
- Flatten + Dense(128, ReLU): combine all features into a compact representation.
- Dropout(0.5): strong regularization to fight overfitting on our small subset.
- Dense(3, Softmax): outputs class probabilities for dog, cat, bird.

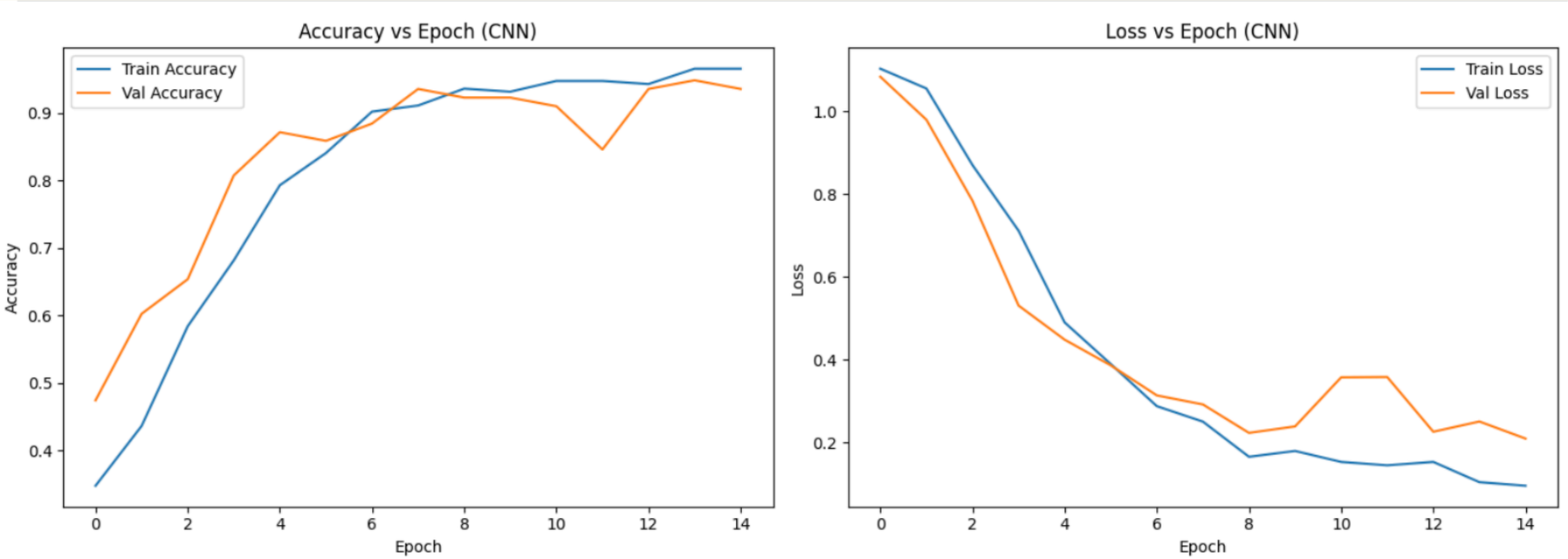


FULL DATASET TRAINING CNN MODEL

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 126, 126, 32)	320
max_pooling2d_4 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_5 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_6 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_6 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten_2 (Flatten)	(None, 25088)	0
dense_4 (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 3)	387

We converted audio into 126×126 Mel-spectrograms and trained a simple CNN to classify dog, cat, and bird sounds using train, validation, and test splits. Here, we used the full dataset and got a test accuracy of 0.93.

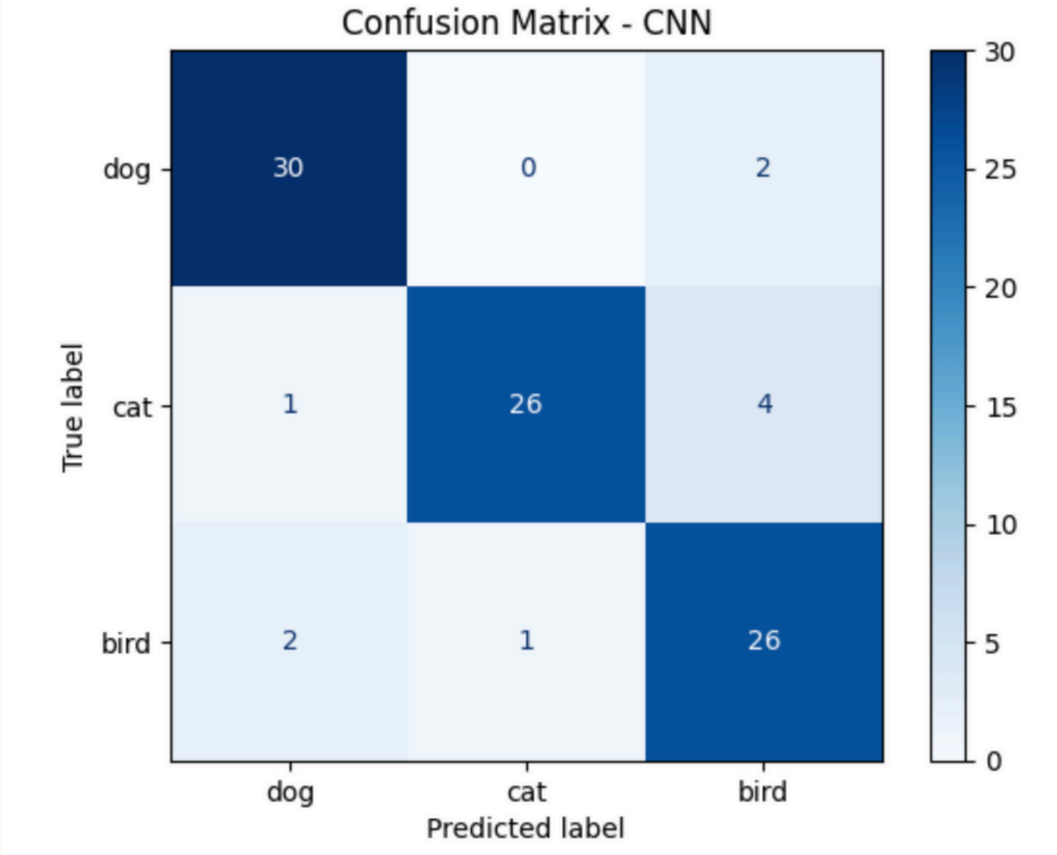
FULL DATASET TRAINING CNN MODEL




```
CNN - Test Loss: 0.2629, Test Accuracy: 0.9098
```

The training curves show steady learning with slightly higher training than validation performance after ~8 epochs, indicating mild overfitting but overall strong model learning.

FULL DATASET TRAINING CNN MODEL



- Correct predictions: 82/92
- Most confused classes: cat → bird (4 cases)

- The CNN accurately distinguishes all three classes, with perfect cat classification and only a few dog–bird misclassifications.
- The main confusion we saw is cat being misclassified as bird a few times.
- That suggests that for certain pronunciations, the spectrogram of ‘cat’ looks more similar to ‘bird’ than to ‘dog’.
- Overall though, the errors are relatively symmetric and there’s no class that the model completely struggles with

```
Random test sample index: 100
True label      : cat (class 1)
Predicted label: cat (class 1)
Prediction CORRECT
```

FULL DATASET TRAINING CNN MODEL

Accuracy

92%

Precision

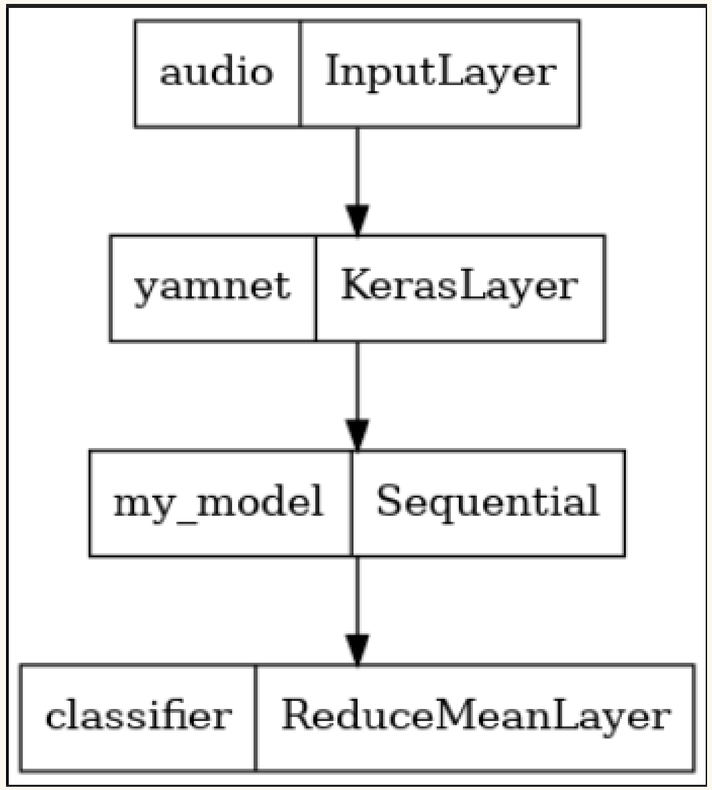
89%

Recall

89%

So compared to our early baseline, using the full dataset and adding dropout (0.5) gives us a big boost in both accuracy and generalization.

WHAT IS YAMNET?



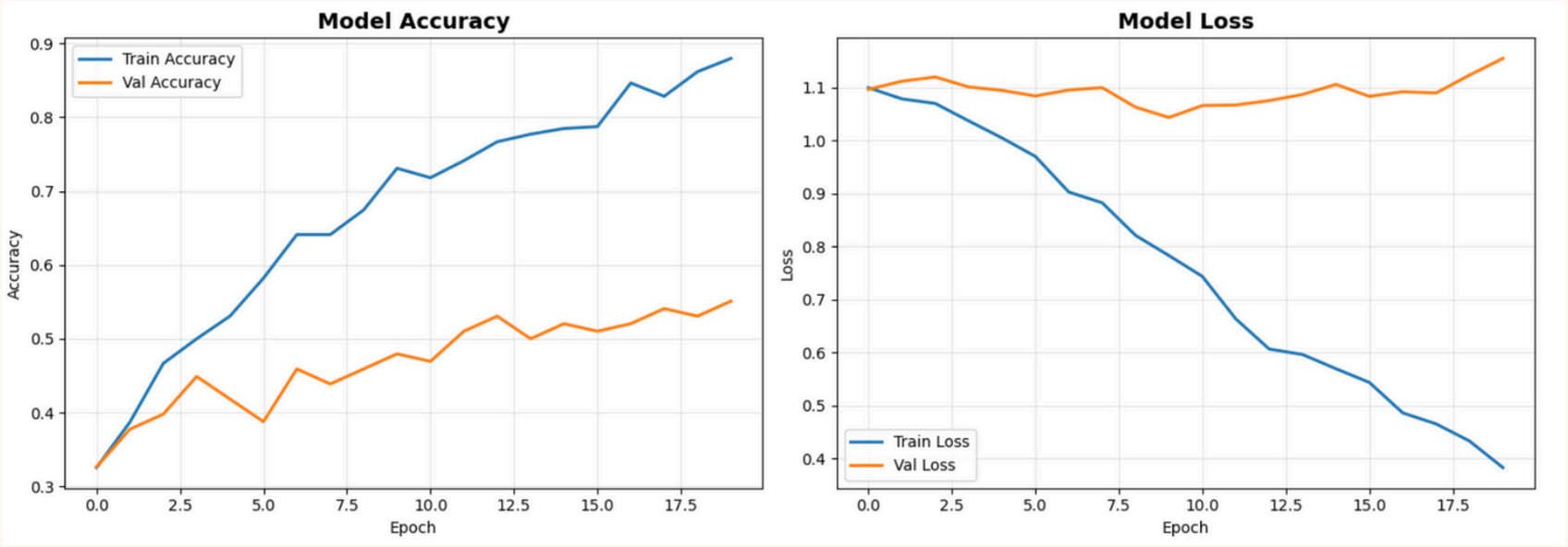
- We feed raw audio into the pretrained YAMNet model to extract embeddings, then pass those embeddings through our custom classifier to predict dog, cat, or bird.

YAMNET

Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 102400)	0
dense_6 (Dense)	(None, 16)	1,638,416
dropout_1 (Dropout)	(None, 16)	0
dense_7 (Dense)	(None, 16)	272
dropout_2 (Dropout)	(None, 16)	0
dense_8 (Dense)	(None, 16)	272
dense_9 (Dense)	(None, 3)	51

This model takes the YAMNet embeddings, flattens them, and passes them through a few small Dense layers to learn the final 3-class prediction.

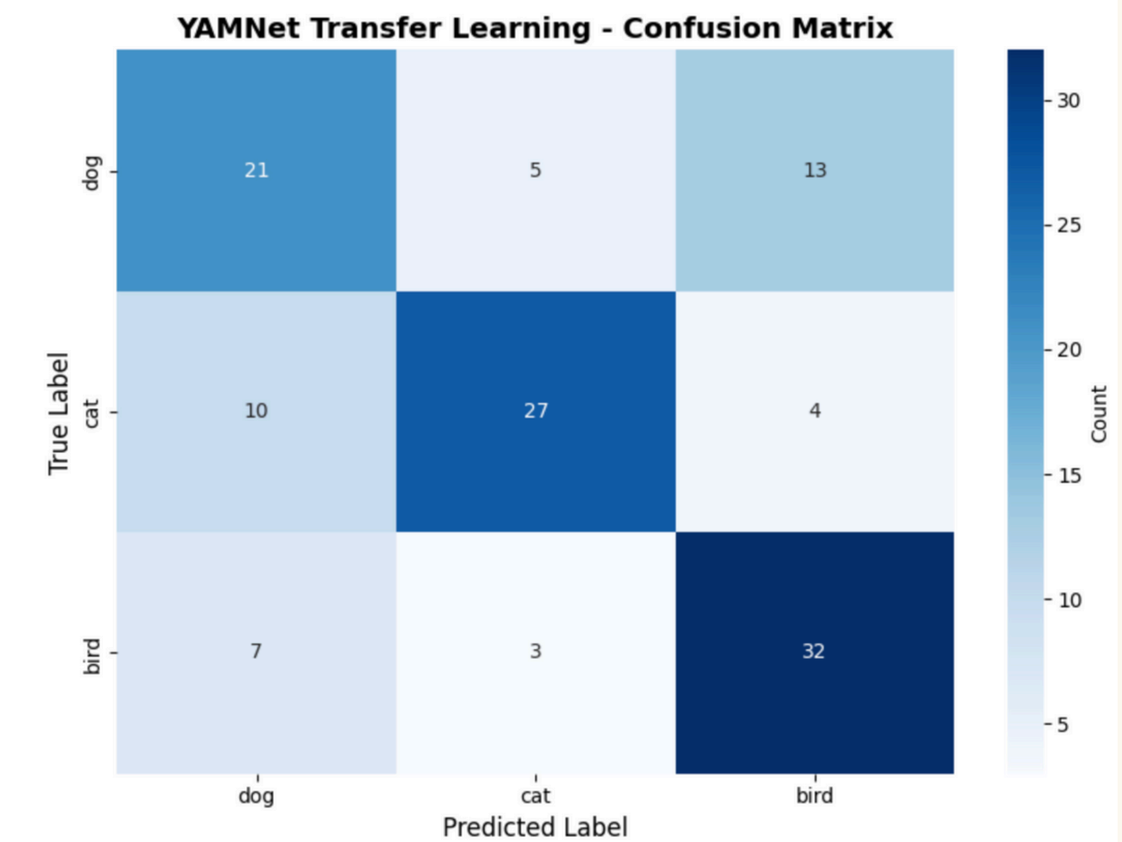
YAMNET



The model gets better on the training data, but not on the validation data, which means it is overfitting and not learning to generalize well.

YAMNET

- YAMNet correctly classifies many samples but often mixes up dog and bird sounds, showing weaker overall performance than the CNN model.



YAMNET

Accuracy

65.57%

Precision

65%

Recall

65%

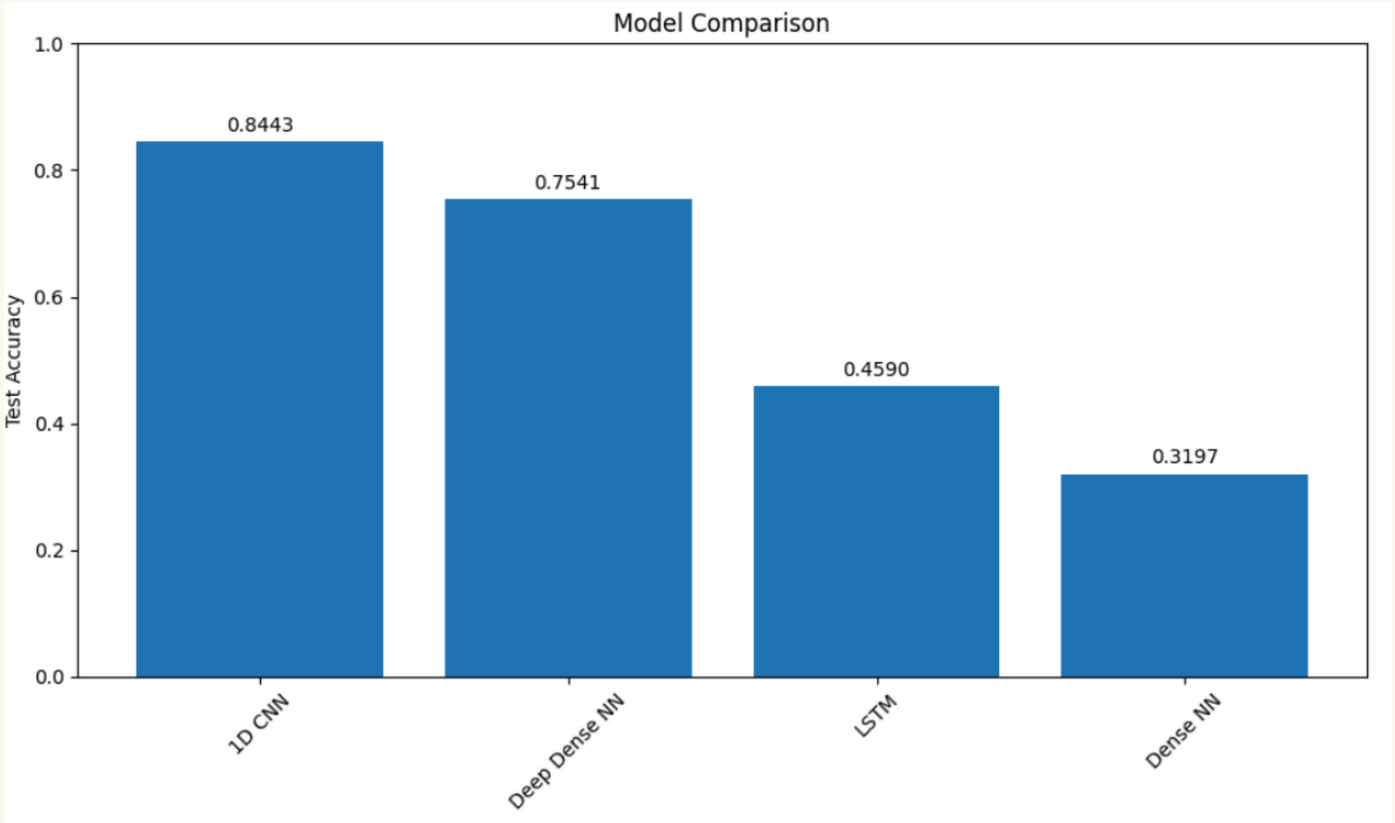
WHY OUR CNN BEAT YAMNET?

- Domain Mismatch – YAMNet is trained on real animal sounds, whereas our data consists of human speech.
- Better Features – Our CNN learns full Mel-spectrogram patterns; YAMNet compresses audio and loses key speech details.
- Enough Data – With 440 training samples, a custom CNN learns the task better than generic pre-trained embeddings.

```
======================================
CNN Model - Comprehensive Metrics
======================================

 Validation Set Metrics:
   Accuracy:  0.9359 (93.59%)
   Precision: 0.9362 (93.62%)
   Recall:    0.9353 (93.53%)
   F1-Score:  0.9355 (93.55%)

 Test Set Metrics:
   Accuracy:  0.8913 (89.13%)
   Precision: 0.8949 (89.49%)
   Recall:    0.8909 (89.09%)
   F1-Score:  0.8907 (89.07%)
======================================
```

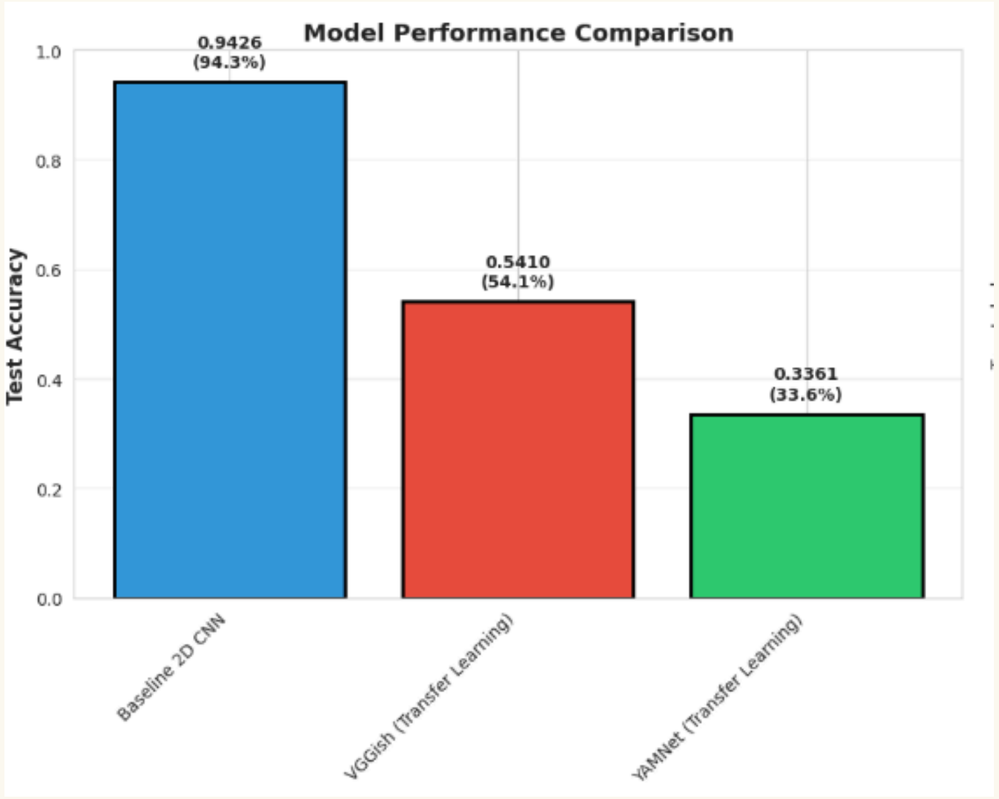
```
================================================
YAMNet Transfer Learning - Comprehensive Metrics
================================================

 Training Set Metrics (from validation_split=0.2):
   Accuracy:  0.8586 (85.86%)
   Precision: 0.8588 (85.88%)
   Recall:    0.8576 (85.76%)
   F1-Score:  0.8579 (85.79%)

 Test Set Metrics:
   Accuracy:  0.6557 (65.57%)
   Precision: 0.6590 (65.90%)
   Recall:    0.6530 (65.30%)
   F1-Score:  0.6531 (65.31%)
================================================
```

OTHER MODELS WE HAVE USED



- We also tested several other models, where the 1D CNN and Deep Dense Network performed reasonably well, while LSTM and simple Dense models struggled to classify the audio accurately.

OTHER MODELS WE HAVE USED



CNN:

- Our custom CNN achieved the highest accuracy, showing that training directly on Mel-spectrograms worked best for this dataset.

VGGish Transfer Learning:

- VGGish performed moderately well but still fell behind the CNN, likely due to feature mismatch with our specific audio classes.

YAMNET (Initial Result):

- This is the initial YAMNet accuracy, which was low at first, but we later improved its performance through better preprocessing and classifier tuning.

FINAL RESULTS AND CONCLUSIONS

Model	Feature Type	Test Accuracy	Test Loss
CNN + Dropout(0.5)	Mel-spectrogram (128×128)	**~92%**	~0.24
Baseline CNN (Full Data)	Mel-spectrogram (128×128)	~90%	~0.57
YAMNet Transfer Learning	Embeddings (100×1024)	~66%	~0.96

- Our custom CNN trained on Mel-spectrograms performed the best, achieving ~92% accuracy and consistently classifying dog, cat, and bird sounds.
- Regularization improves generalization: Adding Dropout(0.5) improved test loss from 0.57 to 0.24, indicating better model calibration and reduced overfitting.
- YAMNet and other transfer-learning models struggled to generalize, often confusing classes and showing signs of overfitting.
- Overall, we found that training a simple CNN directly on Mel-spectrograms works better for our dataset than using pretrained audio models like YAMNet.

FUTURE WORK

- Stronger Augmentation – Noise, pitch shift, time stretch, SpecAugment for robustness.
- Real-World Testing – Use real animal sounds + environmental variations; explore mobile deployment.
- New Architectures – Add attention, residual blocks, multi-scale CNNs, or ensemble models.

THANK YOU!
QUESTIONS?