Fish Recognizer

Will and Joe Thompson

Project Objectives

- We wanted develop A project that would help with the identification of Fish
- For people who are unfamiliar with fish it is hard to identify what they are
- Some softwares exist but are not very accurate and are behind a paywall

Goal: Create an Fish image recognizer

Getting the Dataset

Problem: Unable to Find Dataset on Aquarium Fish

Solution: Custom Dataset using Web Scraping and Taking pictures

Data Sources:

- Web Crawlers scraping images from google and Bing (Using terms like *"guppy fish side view"*, *"single guppy swimming left"*)
- Original Photos taken at Petco

Initial Raw Image Problems:

- Duplicate Images
- Multiple Fish in Frame
- Non-Fish Images (shirts, memes, toys)
- Wrong type of fish

Cleaning Criteria:

- Exactly 1 Fish per Image
- Full Body (No cut offs)
- Decent Resolution for Feature recognition
- Fish must be Primary Focus
- No Distracting Text (text, watermarks, captions)

Good Quality Image

Poor Quality Image

Cleaning Process

Step 1

Manual Removal

- Manually removed obvious image issues (maybe example)

Step 2

Remove Duplicates

- Use code that opens images in binary mode and creates a md5 hash of the image (Image fingerprint)
- Compares the image “fingerprints”
- Automatically deletes duplicates “fingerprints”

Step 3

Flip Images

Add a 80/20 split for the way the fish is facing
-80% gives the model consistency, reliable that the “head will face left”
-20% adds some randomness so it doesn't only memorize that the fish will always face left
-if it was 50/50 the model would struggle to understand the shape of the fish

Step 4

Condense and Add padding

- finds the subject of the image crops the image
- adds padding to prevent image being warped
- must resize the image to standard for model 244px by 244px

Step 5

Apply Train, Test and Validation Split to Dataset

Result

A Custom Clean dataset of Aquarium fish
Containing:

- 283 total images
- 4 classes of fish (Bettas, Gourami, Guppies, and Killifish)

Simplified Class Labels (eg. Killifish includes many species)
Why these Fish?

- Simplified Data Collection
- Avoids dealing with the 1200+ species of Killifish
- Non-schooling behavior, typically capture alone with minimal background noise
- Easier for clean image collection and classification

Betta Fish

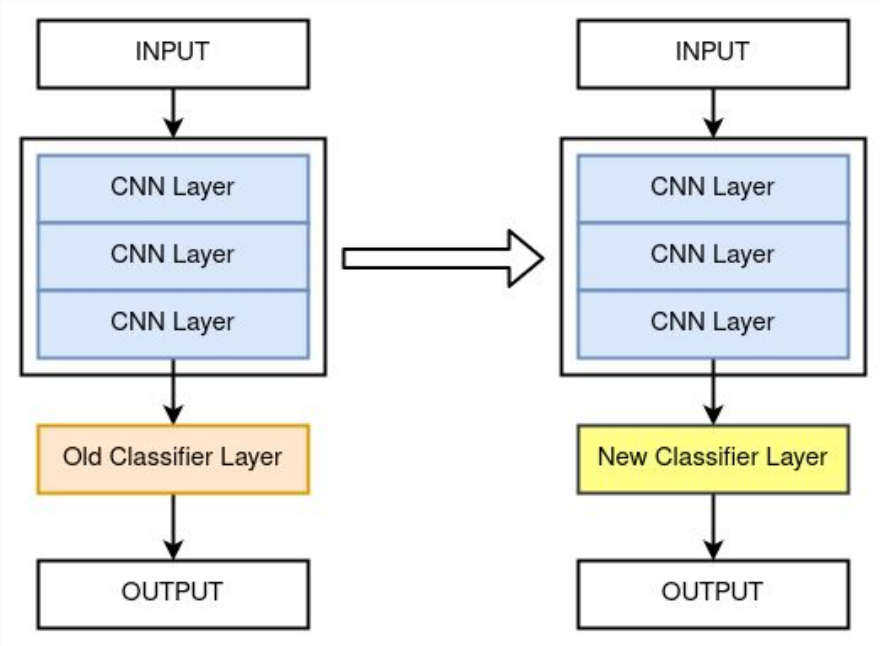
Gourami

Killifish

Guppy

Models Chosen

- Image Recognition Problem Used Deep Learning model
- Because Dataset is so small (283 clean images) total, we used pretrained models
  - MobileNetV2
  - MobileNetV3Small
  - EffiecientNetB0
  - ResNet50
- Applied Transfer learning to adapt the pretrained models
  1. Loaded pretrained models (trained originally on ImageNet)
  2. Removed the original classification head
  3. Froze the base layers to preserve the pretrained features
  4. Added custom head with 4 output classes (fish types)
  5. Trained the new head on the fish dataset



Pipeline

Pretrained_Models = [MobileNetV2, MobileNetV3Small, EfficientNetB0, ResNet50]

for model in Pretrained_Models:

Step 1

Preprocessing

Adjust Dataset images to how the Pretrained Deep learning model is used to looking at them

Step 2

Applied Transfer Learning

Combined pretrained model with our model

Then we Compiles and Fit model

Step 3

Compare

Find Error metrics and build graphs to Compare and find Best performing model

Step 4

Grid Search

Apply Grid Search to best performing model to achieve best parameters

Model Results

Model	Test Accuracy	Average Precision	Average Recall	Average F1	Key Notes
MobileNetV2	85%	83%	82%	82%	
MobileNetV3Small	74%	72%	72%	72%	
EfficientNetB0	87%	88%	85%	86%	Best Performer
ResNet50	83%	86%	80%	80%	

Standard:

Best Parameters Of EffiecientNetB0

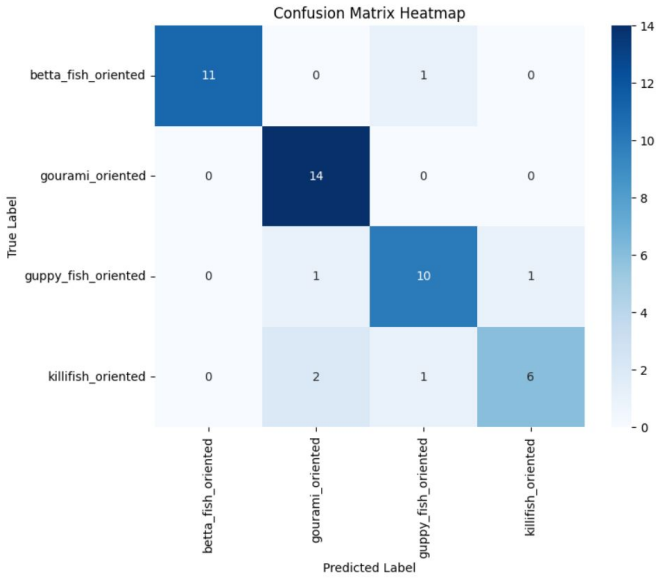
- Validation Accuracy peaked at Epoch 10
- After that Model began to overfit as Training accuracy dropped only a little, while Validation decreased sharply
- Therefore we selected the model weights at epoch 10 maximize generalization and prevent overfitting

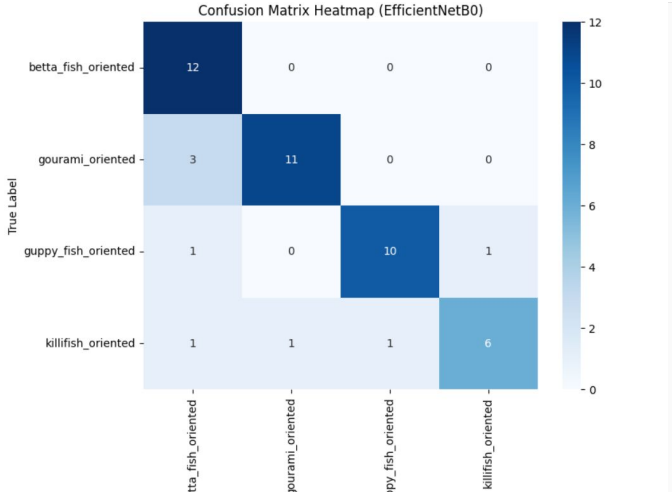
Best hyperparameters:
- Dense units: 64
- Dropout rate: 0.4
- Learning rate: 0.001

Accuracy: 83%
Precision: 85%
Recall: 82%
F1: 82%

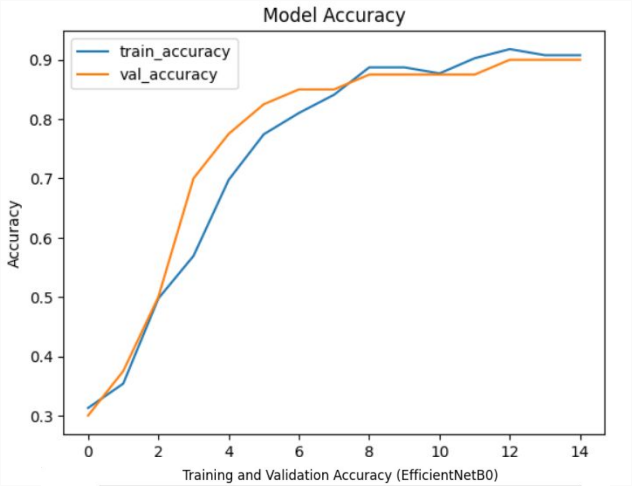
After Hyperparameter tuning :

After hyperparameter tuning, validation instability was observed. Therefore, the final model was trained using the standard EfficientNetB0 configuration to prioritize stable generalization performance."

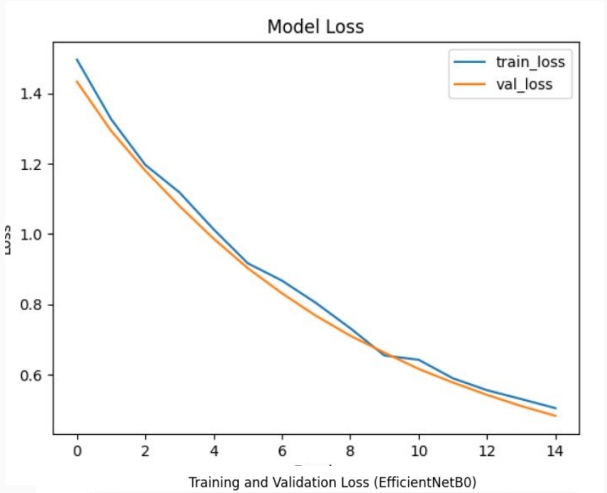


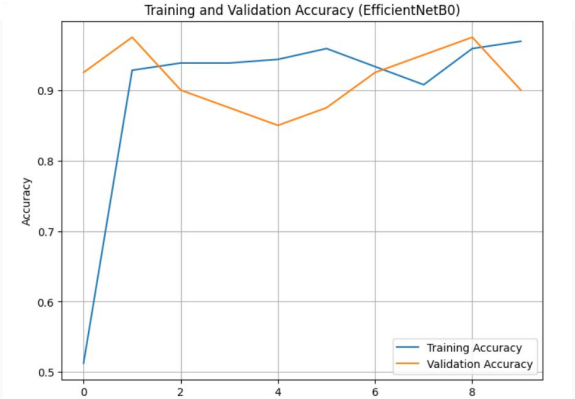


Results of EfficientNetB0

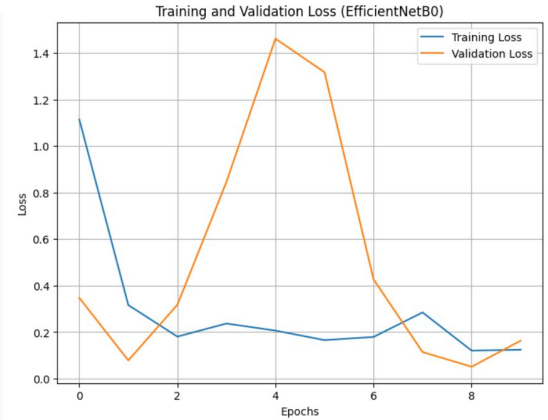


Standard



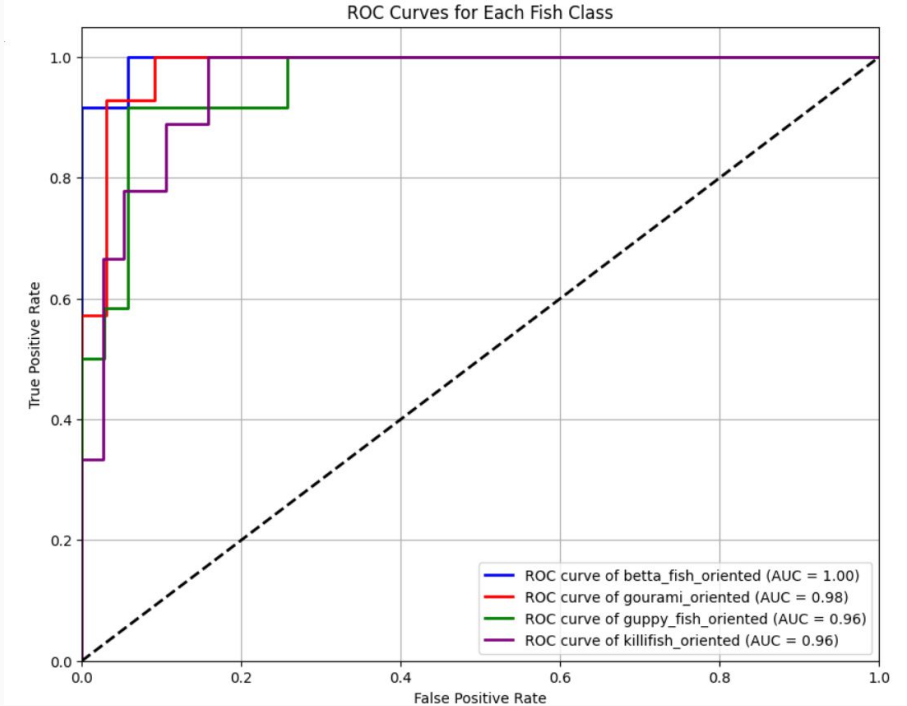


After Hyperparameter tuning

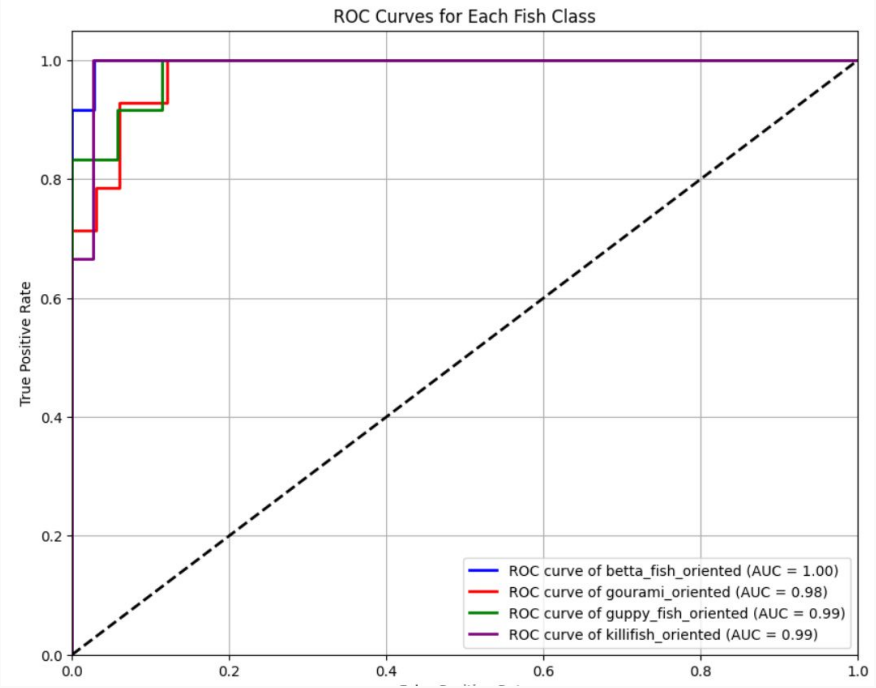


Results of EfficientNetB0

Standard



After Tuning



Challenges

- Initially models were performing poorly
- Fixed by giving each model its on preprocessing parameters

- The version of Tensorflow we were using wasn't allowing our models to load well after saving
- Fixed by Installing a different version of Tensorflow

CONCLUSIONS

What we Learned

Learned alot about creating, cleaning, and collecting for Datasets

Learned about Transfer Learning

Things to Add in Future

Create an app that you can use on your phone

Add more images to the Dataset Especially to the Killifish Class