

Project-4-Fish Image Classification

Documentation

- Importing necessary libraries.(tensorflow,keras,matplotlib(visualization),scikit-learn metrics(classification report)
- Used Visual Studio Code IDE and Google Colab environment for coding
- CNN Model coding was done in VS Code.
- Connecting the Google drive to the Colab environment for pre-trained models as Colab offers limited time GPU.
- The Dataset is in zip file so extracting the zipfile into individual folders.
- Defined the path for (train,test and val folders).

Approach – 1

Data Pre-processing and Augmentation.

- Using tensorflow.keras we have imported ImageDataGenerator.
- With the help of ImageDataGenerator we have made changes to training data,
 - i. **Rescale-1./255**-Scales the pixel values from the range 00,255] to [0,1].
 - ii. **Rotation (0.2)** – Rotates the image by 20 Degrees (+or-).
 - iii. **Width/Height (0.2)** – Shifts images horizontally and vertically up to 20%.
 - iv. **Shear-range (0.2)** – It distorts the image up to 20%.
 - v. **Zoom range (0.2)** – Zooms in and out by 20%.
 - vi. **Horizontal flip (True)** – It flips the image horizontally.

- vii. **Fill mode (nearest)** – when rotation, shifting, zooming empty areas are created, fill mode fills those gaps using nearest pixel values.
 - viii. **Validation split (0.2)** – 20% for Validation
 - ix. For testing and validation data only rescaling is done.
- **Data Loading**
 - i. **Target size (224,224)** – Resizes all the images into 224*224 pixels.
 - ii. **Batch size (32)** – Loads 32 images per batch.
 - iii. **Class mode(categorical)** –Set the labels into One Hot Encoding(categorical) format for Multiclass.

Approach – 2

Training CNN model from Scratch.

- i. We are using **Sequential** model where it creates layers of linear stacks in which each layer's output becomes input of the next layer.
- ii. We have used 3 **Convolutional Blocks**, where each block contains filters (32,64,128), Kernel size (3,3), activation='relu' and **Maxpooling** of (2,2).
 - Each filter detects different patterns like edges, textures etc and captures more complex features.
 - ReLU introduces non-linearity to help the model learn complex patterns.
 - Maxpooling reduces the spatial dimensions by taking the maximum value in each 2*2 window.
- iii. **Flatten()** is used to convert 3D features into 1D vector for input to the dense layers.
- iv. **Fully Connected Dense Layers** – Adds 128 neurons for learning high level patterns.
- v. **Dropout** –Drops 50% of neurons randomly during training, reducing overfitting.

- vi. **Output Dense Layer** – It has 11 neurons (one for each class), activation is '**softmax**' which outputs a probability distribution, ensuring total probabilities sum to 1.

Compiling the Model:

- i. **Optimizer = Adam**, Adaptive Moment Estimation (Adam) is used because it adjusts the learning rate dynamically during training. And also improves speed and stability. Handles large datasets.
- ii. **categorical_crossentropy** calculates the difference between the predicted probability distribution and the true distribution.
- iii. **Metrics = accuracy**, used to track the percentage of metrics accurately for classified images.

Model Summary:

The output of the model summary is

Total params: 11,170,379

Trainable params: 11,170,379

Non-trainable params: 0

Fitting the Model:

- The model is fitted to train the data with epochs (One Complete pass of the entire training dataset through the model).
- Used Early Stopping to ensure efficiency, the model training can stop early if the model starts overfitting based on the parameters passed.
- **Monitor = val_loss**, it monitors validation loss during training
- **Patience = 3**, defines how many consecutive epochs with no improvement should pass before stopping training.

- **Restore_best_weights**, reverts best weights from the epoch with the lowest validation loss. Without this, the model would keep the weights of the previous trained epoch, which may not be optimal.
- **Verbose**, used to control the training log's display during model fitting. **Verbose = 2**, indicates **Summary log** for each epoch which gives cleaner output for larger model.

Output of model Fitting:

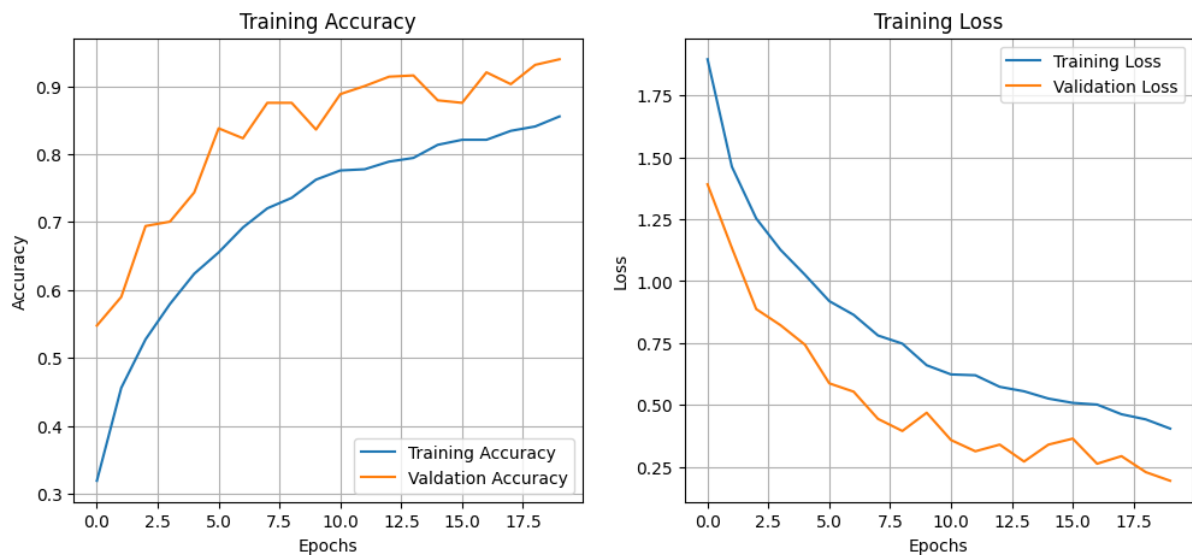
Key Observations:

- **Training Accuracy** increases from **31.92%** to **85.54%**
- **Validation Accuracy** increases from **54.76%** to **93.96%**
- **Training Loss** decreased gradually
- **Validation Loss** decreased consistently until Epoch 9, then fluctuates.
- Due to **Early Stopping**, the model stopped at Epoch 12 to prevent unnecessary training.

Evaluating the Model:

- The output of the Test Accuracy: **93.44%**. (**Not an over-fitting Model**).
- A **high validation and test accuracy (above 90%)** suggests that the model can effectively classify fish species in real-world applications.

Visualization of CNN Model:



- Accuracy is gradually increasing, no overfitting since both the curves are close together.
- Loss decreases gradually, indicates stable performances.

Classification Report(CNN Model):

```
100/100 [=====] - 35s 344ms/step
              precision    recall  f1-score   support

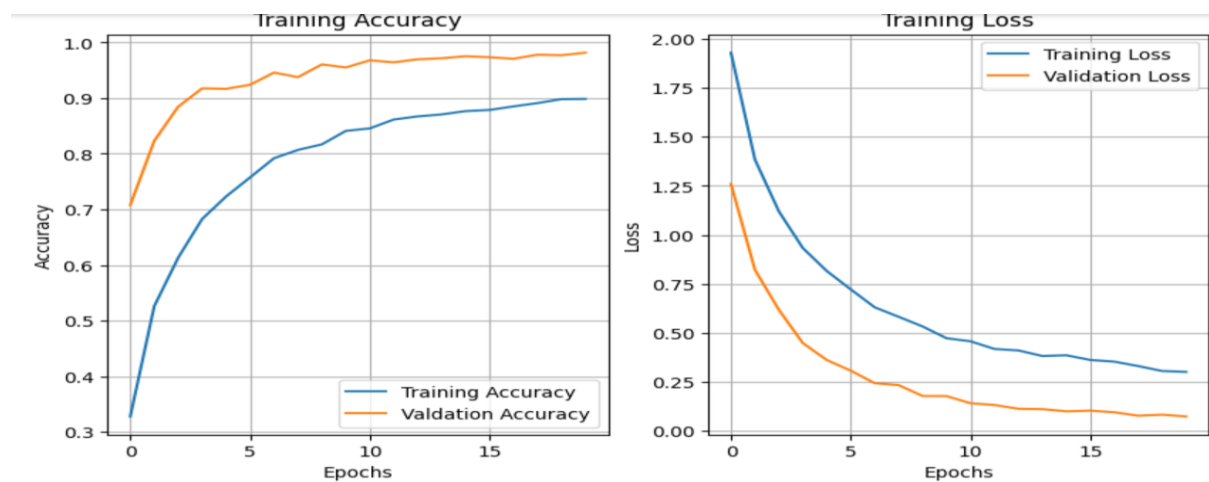
    animal fish              0.15      0.16      0.15         520
    animal fish bass          0.00      0.00      0.00          13
 fish sea_food black_sea_sprat 0.10      0.11      0.10         298
 fish sea_food gilt_head_bream 0.11      0.08      0.09         305
 fish sea_food hourse_mackerel 0.11      0.10      0.11         286
    fish sea_food red_mullet    0.06      0.06      0.06         291
 fish sea_food red_sea_bream    0.09      0.11      0.10         273
    fish sea_food sea_bass      0.11      0.10      0.11         327
    fish sea_food shrimp        0.06      0.06      0.06         289
 fish sea_food striped_red_mullet 0.11      0.11      0.11         293
    fish sea_food trout         0.09      0.09      0.09         292

      accuracy                   0.10         3187
    macro avg              0.09      0.09      0.09         3187
    weighted avg            0.10      0.10      0.10         3187
```

1)Pre-trained Model – VGG16:

- Loaded the model without the top layer (fully connected layer).
- Freezing the initial layers to retain the pre-trained features.
- Added custom layers on the top.
- Global Average Pooling reduces the number of parameters, making the model prone to overfitting.
- Created, Compiled and trained the model with 20 Epochs.
- Testing Accuracy: **98.53%** achieved.

Visualization-VGG16:



- Training Accuracy and validation accuracy increases rapidly.

Classification Report-VGG16:

100/100	20s	196ms/step			
	precision	recall	f1-score	support	
animal fish	0.98	1.00	0.99	520	
animal fish bass	0.00	0.00	0.00	13	
fish sea_food black_sea_sprat	0.99	0.98	0.99	298	
fish sea_food gilt_head_bream	0.99	1.00	1.00	305	
fish sea_food hourse_mackerel	0.98	0.98	0.98	286	
fish sea_food red_mullet	0.97	0.98	0.98	291	
fish sea_food red_sea_bream	1.00	0.99	0.99	273	
fish sea_food sea_bass	1.00	0.98	0.99	327	
fish sea_food shrimp	0.99	1.00	1.00	289	
fish sea_food striped_red_mullet	0.96	0.98	0.97	293	
fish sea_food trout	1.00	0.99	0.99	292	
accuracy			0.99	3187	
macro avg	0.90	0.90	0.90	3187	
weighted avg	0.98	0.99	0.98	3187	

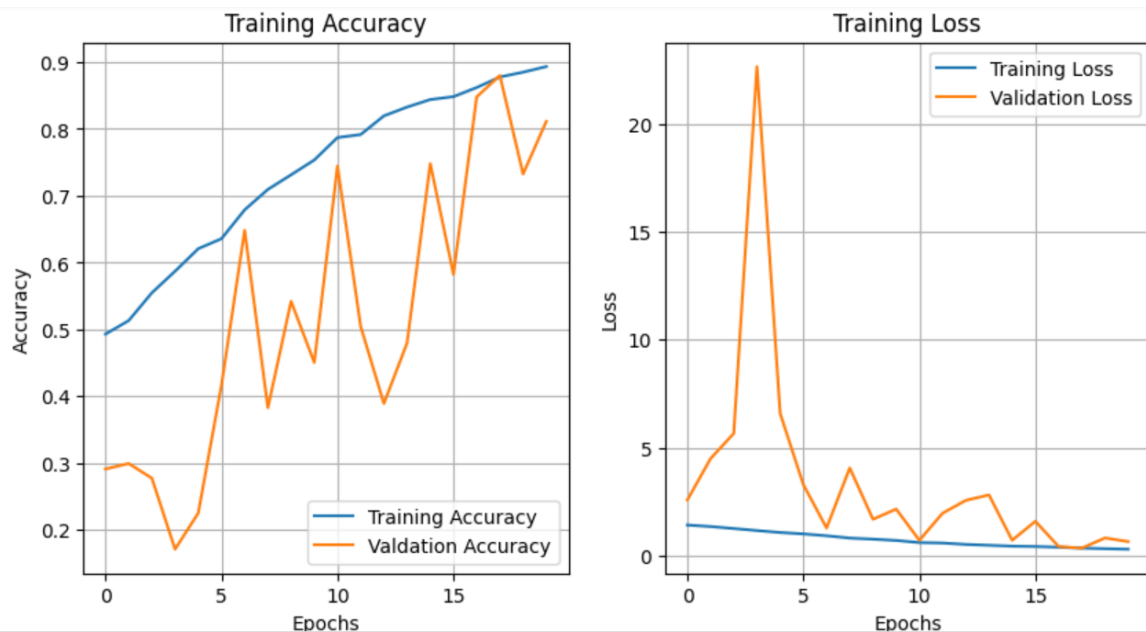
2) Pre-Trained Model ResNet50:

- Loaded the model without the top layer (fully connected layer).
- Freezing the initial layers to retain the pre-trained features.
- Added custom layers on the top.
- Created, Compiled and trained the model with 20 Epochs.
- **Test Accuracy: 32.60%**
- **Validation Accuracy: 31.96%**
- Since test accuracy is slightly higher than the validation accuracy there is no severe overfitting and ResNet50 is deep model so the accuracy will start low only. Next is fine tuning.
- Fine-Tuned by unfreezing last 50 layers for training and also changed Learning Rate to 0.0001(1e-4) and started training the model.
- **Final Training Accuracy: 89.33%**
Final Validation Accuracy: 81.14%
Test Accuracy: 80.82%.

Classification Report of ResNet50:

100/100 ————— 16s 128ms/step				
	precision	recall	f1-score	support
animal fish	0.91	0.96	0.93	520
animal fish bass	0.00	0.00	0.00	13
fish sea_food black_sea_sprat	1.00	0.47	0.64	298
fish sea_food gilt_head_bream	0.99	0.49	0.65	305
fish sea_food hourse_mackerel	0.89	0.92	0.91	286
fish sea_food red_mullet	0.90	0.87	0.89	291
fish sea_food red_sea_bream	0.93	0.82	0.87	273
fish sea_food sea_bass	0.79	0.92	0.85	327
fish sea_food shrimp	0.87	0.93	0.90	289
fish sea_food striped_red_mullet	0.75	0.80	0.77	293
fish sea_food trout	0.56	1.00	0.71	292
accuracy			0.82	3187
macro avg	0.78	0.74	0.74	3187
weighted avg	0.86	0.82	0.82	3187

Visualization ResNet50:



- The Validation Accuracy in the left plot fluctuates more indicating overfitting.
- The Validation Loss in right plot fluctuates more indicating poor generalization.
- These both occur may be due to class imbalance. We have fine tuned and achieved Testing accuracy from 31% to 81%

3) Pre-Trained Model MobileNet:

- Fine-Tuned **ImageDataGenerator** as MobileNet is light weighted.
- Loaded the model without the top layer (fully connected layer).
- Freezing the initial layers to retain the pre-trained features.
- Added custom layers on the top. Reduced Dropout to 0.3 because MobileNet has Batch Normalization which is used for generalizations.
- Created, Compiled and trained the model with 10 Epochs.
- At 3rd Epoch the training accuracy achieves **99.66%** and stopped executing, tried again but it stops executing at high training accuracy so manually interrupted/stopped the kernel.
- The **Testing Accuracy: 99.62%**, this is the best model among other pre-trained models.
- Saved the Model in .pkl format.


Classification Report:

100/100	10s	83ms/step			
		precision	recall	f1-score	support
animal fish	0.99	1.00	1.00	520	
animal fish bass	1.00	0.77	0.87	13	
fish sea_food black_sea_sprat	1.00	1.00	1.00	298	
fish sea_food gilt_head_bream	1.00	0.99	1.00	305	
fish sea_food hourse_mackerel	0.99	1.00	0.99	286	
fish sea_food red_mullet	1.00	1.00	1.00	291	
fish sea_food red_sea_bream	1.00	1.00	1.00	273	
fish sea_food sea_bass	0.99	1.00	1.00	327	
fish sea_food shrimp	1.00	1.00	1.00	289	
fish sea_food striped_red_mullet	1.00	0.99	0.99	293	
fish sea_food trout	1.00	1.00	1.00	292	
accuracy			1.00	3187	
macro avg	1.00	0.98	0.99	3187	
weighted avg	1.00	1.00	1.00	3187	

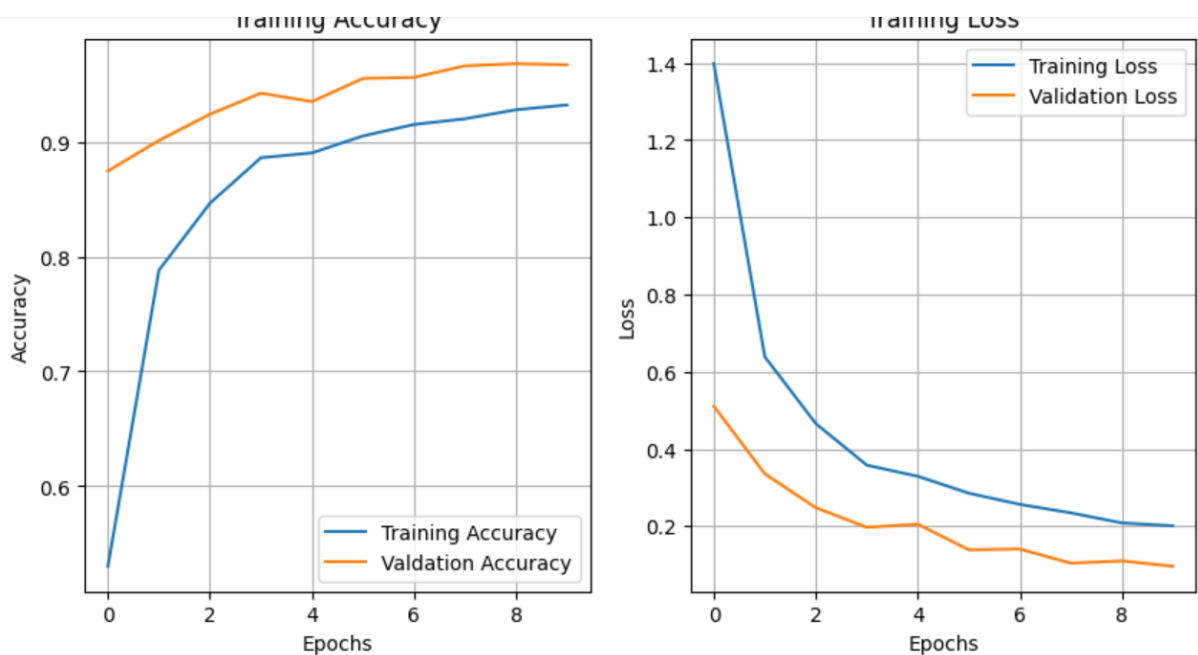
4)Pre-Trained Model(Inception V3):

- Pre processed with ImageDataGenerator for Augmentation.
- Loaded the data and loaded the base layers.
- Freezing the base layers.
- Adding custom layers, used two dropouts as Inception V3 is a complex model.
- Defined, compiled and trained the model using 10 Epochs.
- **Training Accuracy: 93.20%**
- **Validation Accuracy: 96.70%**
- **Testing Accuracy: 97.62%**

Classification Report:

100/100		19s	127ms/step			
		precision	recall	f1-score	support	
animal fish	0.97	1.00	0.98	520		
animal fish bass	0.00	0.00	0.00	13		
fish sea_food black_sea_sprat	1.00	0.96	0.98	298		
fish sea_food gilt_head_bream	0.99	0.98	0.99	305		
fish sea_food hourse_mackerel	0.97	0.99	0.98	286		
fish sea_food red_mullet	0.96	0.99	0.97	291		
fish sea_food red_sea_bream	0.99	0.99	0.99	273		
fish sea_food sea_bass	0.95	0.97	0.96	327		
fish sea_food shrimp	1.00	1.00	1.00	289		
fish sea_food striped_red_mullet	0.96	0.90	0.93	293		
fish sea_food trout	0.98	1.00	0.99	292		
accuracy			0.98	3187		
macro avg	0.89	0.89	0.89	3187		
weighted avg	0.97	0.98	0.97	3187		

Visualization:



- Training accuracy and validation accuracy increases rapidly.
- Validation accuracy starts high and stabilizes.
- Training loss and validation loss decreases slowly.

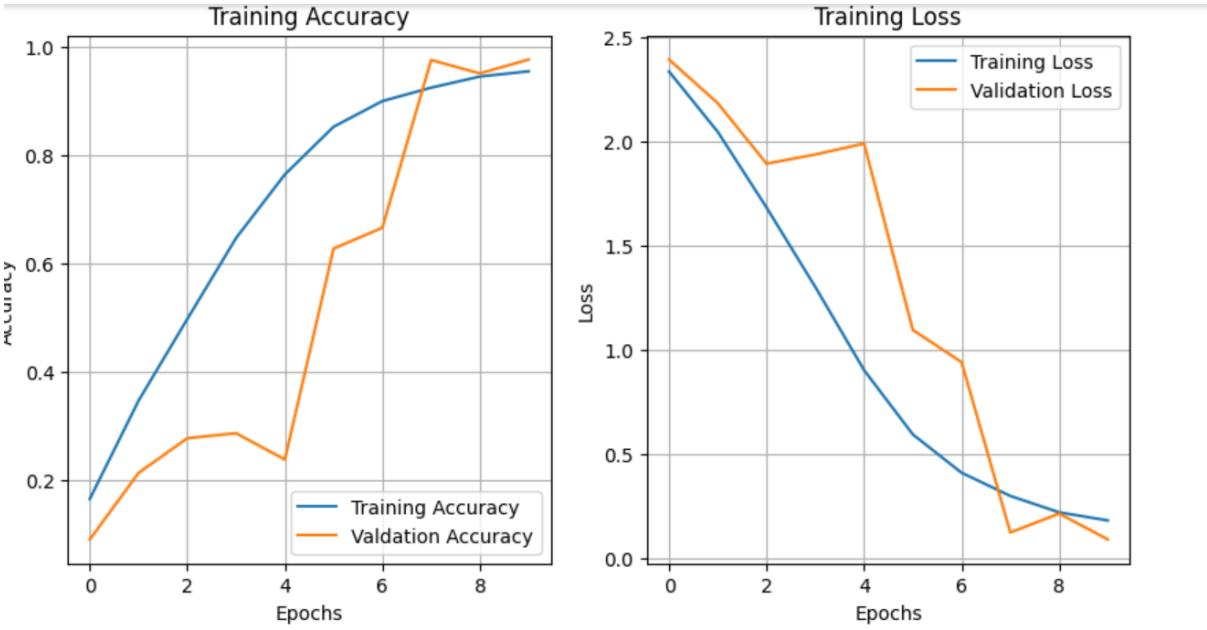
5) Pre-Trained Model(EfficientNetB0):

- Data pre-processed and augmentation done.
- Loaded the data and loaded EfficientNet to the base layers.
- Initially freezes the base layer, added custom layers and trained the model with learning rate of **0.0001(1e-4)**.
- **Training Accuracy: 17.62%**
- **Validation Accuracy: 17.12%**
- **Testing Accuracy: 16.32%**
- The above is very low so now fine-tuned the model by changing the learning to **0.00001(1e-5)** and trained the model with 10 Epochs.
- **New Training Accuracy: 95.53%**
- **New Validation Accuracy: 97.71%**
- **New Testing Accuracy: 98.24%**

Classification Report:

100/100	20s 140ms/step			
	precision	recall	f1-score	support
animal fish	0.16	0.16	0.16	520
animal fish bass	0.00	0.00	0.00	13
fish sea_food black_sea_sprat	0.08	0.08	0.08	298
fish sea_food gilt_head_bream	0.11	0.10	0.10	305
fish sea_food hourse_mackerel	0.08	0.08	0.08	286
fish sea_food red_mullet	0.10	0.10	0.10	291
fish sea_food red_sea_bream	0.08	0.09	0.08	273
fish sea_food sea_bass	0.09	0.09	0.09	327
fish sea_food shrimp	0.08	0.08	0.08	289
fish sea_food striped_red_mullet	0.08	0.08	0.08	293
fish sea_food trout	0.08	0.08	0.08	292
accuracy			0.10	3187
macro avg	0.09	0.09	0.09	3187
weighted avg	0.10	0.10	0.10	3187

Visualization:



- Training accuracy steadily increases.
- Validation accuracy has some fluctuations but improves significantly after epoch 5.
- Training loss decreases smoothly, which is good.
- Validation loss initially fluctuates, but then it drops significantly.


Deployment:

- Built a Streamlit application, where the user will upload the fish image.
- The Model will predict its class and the confidence score.
- Entire Streamlit coding done in Google Colab environment.
- Installed localtunnel in the notebook, which gives the temporary URL and password.

Multiclass Fish Image Classification

Upload an image, get the predicted class and confidence score.

Choose an Image...

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files



0P2GBIIYQUY2.jpg 23.2KB



The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.



Uploaded Image

Classify

Predicted Class: animal fish

Confidence Score: 100.00

Difficulties faced:

- Since 11M parameters are used it takes heavy computational costs/time so used callbacks method for Early Stopping to ensure smooth transitions.
- The classification report of CNN model shows more Imbalance data due to fewer samples. So experimented with the pre-trained models.
- For VGG16, tried Coding in Visual Studio Code, since there is no GPU available in my local system training the model takes huge time even to run single epoch. So switched to Google Colab where free access GPU is available.
- For ResNet50, initial Testing Accuracy is **32.60%**, so fine-tuned by unfreezing last 50 layers and changing learning rate of Adam to 0.0001(1e-4) and trained the model, the Testing Accuracy after fine-tuning is **80.82%**, Training Accuracy is **89.33%** and Validation Accuracy is **81.14%**.

- For MobileNet Model, started to train the model having 10 Epochs but the execution stops when the training accuracy 99.60% around 3 Epoch itself, again tried but it stops around the 99.60%.
- In Streamlit UI building tried with CSS Styling and with more appealing but localtunnel doesn't support high styling pages as the page keeps on loading.

Practical Applications:

- This model can be beneficial for **marine biologists, fisheries, and seafood industries** to automate fish species identification.
- In **supply chain and market inspection**, it can help verify fish species for labeling accuracy and fraud prevention.

Thank You