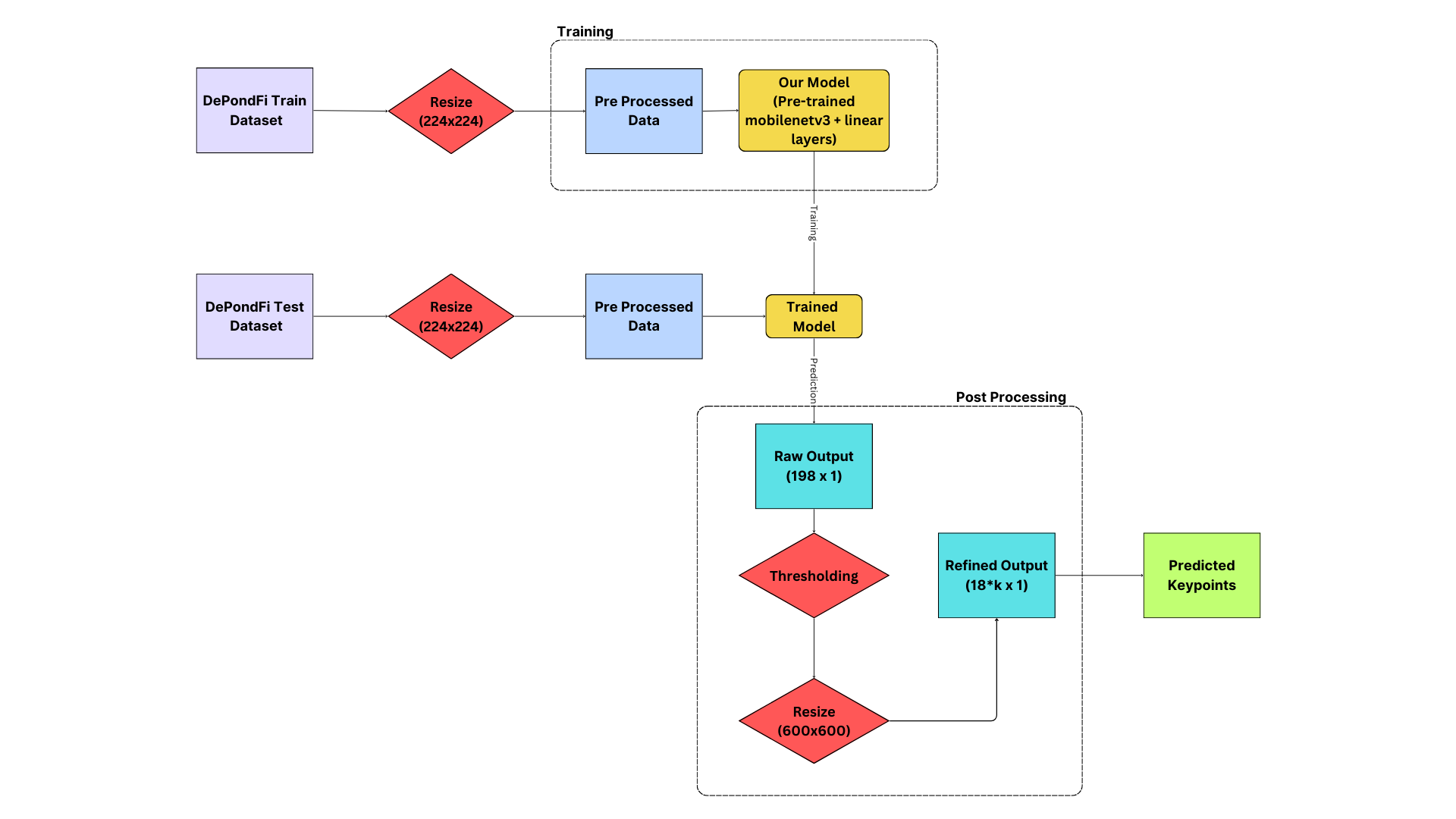
**NCVPRIPG2024**

**Detection of Pond Fish Challenge-2024 (DePondFi’24)**

| **Team Name:** | **Gladiators** |
| --- | --- |
| **Team member's names with affiliation:** | 1. **Tejas Gupta** 2. **Maulik Desai** |

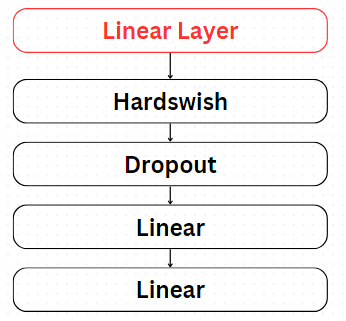
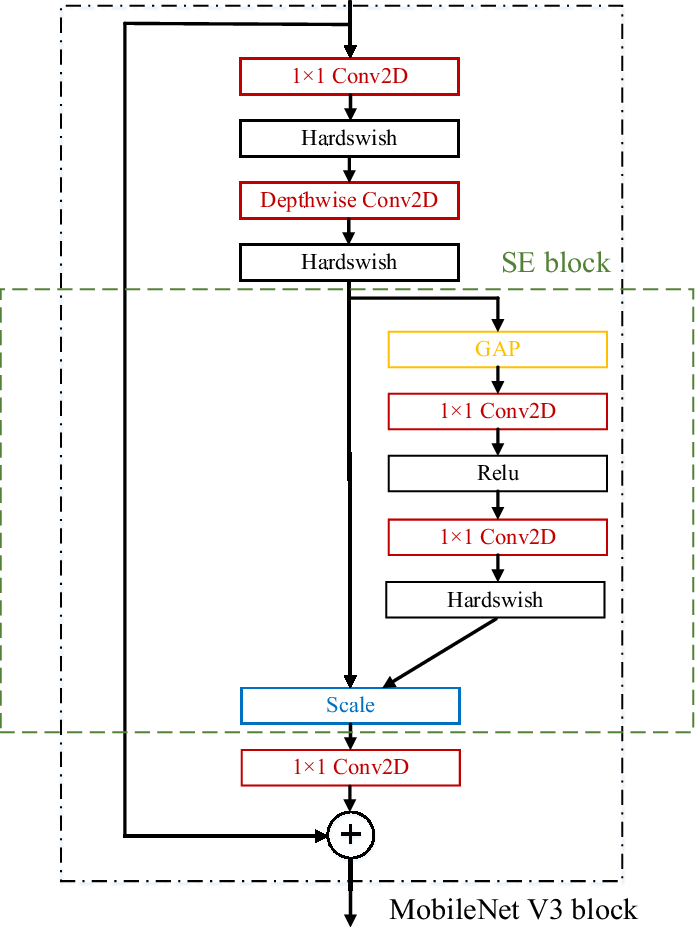
1. **General Block Diagram**

****

1. **Algorithm Or Pseudo Code Used**

* We took the training images and resized them to 224x224
* We put that into a neural network with pre-trained weights of Google’s MobileNetv3 with the following additions:-
  + Phase 1 → one linear layer (output pretrained, 1024) another layer (1024, 198)
  + Phase 2 → one linear layer (output pretrained, 1024) then (1024, 512) and another linear layer (512, 198)
* We trained the neural network with relevant regularization techniques.
* For post processing, we took the output vector (198x1) and checked every 18th value (indexed at 0), if it was less than a certain threshold (0.2) then we assign 0 to all values after it
* The remaining output vector after resizing is the predicted coordinates of our model with x coordinates at even indices and y coordinates at their successive index.

1. **Architecture Diagram**

****

1. **Implementation details (include hardware and software used)**

* We implemented the entire code on Google Colab with T4 GPU runtime, no external hardware was used.

1. **Summary of the approach and experimental result – Phase-I**

**(Detail Explanation)**

* We took the training images and resized them to 224x224
* We put that into a neural network with pre-trained weights of Google’s MobileNetv3 with two linear layers (output, 1024) and (1024, 198)
* We trained the neural network with relevant regularization techniques.
* We used Adaptive Wing Loss as the loss function for this approach
* For post processing, we took the output vector (198x1) and checked every 18th value (indexed at 0), if it was less than a certain threshold (0.2) then we assign 0 to all values after it
* The remaining output vector after resizing to 600x600 is the predicted coordinates of our model with x coordinates at even indices and y coordinates at their successive index.

1. **Summary of the approach and experimental result – Phase-II**

**(Detail Explanation)**

* We took the training images and resized them to 224x224
* We put that into a neural network with pre-trained weights of Google’s MobileNetv3 with three linear layers (output, 1034) then (1024, 512) and another linear layer (512, 198)
* We trained the neural network with relevant regularization techniques.
* For post processing, we took the output vector (198x1) and checked every 18th value (indexed at 0), if it was less than a certain threshold (0.2) then we assign 0 to all values after it
* The remaining output vector after resizing to 600x600 is the predicted coordinates of our model with x coordinates at even indices and y coordinates at their successive index.

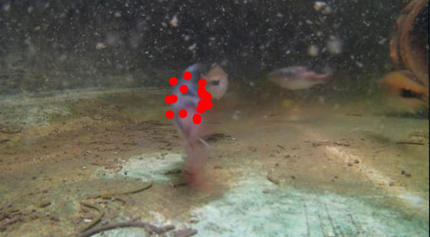
**Tabulation**

| **S.No.** | **Parameters** | **Details – Phase I** | **Details- Phase II** |
| --- | --- | --- | --- |
| 1 | Train Count | 80% | 80% |
| 2 | Validation Count | 20% | 20% |
| 3 | Augmentation | None | None |
| 4 | Loss Function Name | Adaptive Wing Loss | Mean Absolute Error |
| 5 | Ensemble Learning (Yes/No) | No | No |
| 6 | Experimental Setup Details |  |  |
| * TensorFlow/Pytorch | PyTorch | PyTorch |
| * Learning Rate | 1e-2 | 1e-2 |
| * Optimizer | Adam | Adam |
| * Others (if any) | Hardswish  Dropout = 0.2  Weight decay = 1e-4 | Hardswish  Dropout = 0.2  Weight decay = 1e-4 |
| 7 | Model Details with number of parameters | 1,720,806 | 2,144,230 |
| 8 | Details about Pre-Processing | Resize to 224x224 | Resize to 224x224 |
| 9 | Details about Post-Processing | Thresholding  Resizing | Thresholding  Resizing |
| 10 | Others (if any specific) | Learning Rate Scheduler | Learning Rate Scheduler |

1. **Visualization of results showing detected key points - Phase I and Phase II**



**Phase-1**



**Phase-2**

1. **Conclusion**

Our team, Gladiators, utilized Google’s MobileNetv3 architecture in two phases to tackle the Detection of Pond Fish Challenge 2024 (DePondFi’24). Phase I focused on refining predictions with Adaptive Wing Loss, while Phase II enhanced the model with an additional layer, leading to incremental improvements in accuracy and robustness.

Looking ahead, integrating last year's fish detection model would have facilitated ROI localization and training keypoint extraction, potentially enhancing overall performance and scalability.

1. **References**

Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. *ArXiv*. /abs/1905.02244

[elliottzheng/AdaptiveWingLoss: PyTorch Implementation of Adaptive Wing Loss and Wing Loss (github.com)](https://github.com/elliottzheng/AdaptiveWingLoss)