Report

We started to develop a convolutional neural network (CNN) model for classifying bird images, using an iterative approach to enhance accuracy and reduce overfitting. We began with a simpler architecture and progressively added layers, regularization techniques, and other optimization strategies, ultimately achieving an effective balance between model complexity and generalization performance.

Model Development Process

Initial Architecture: **3 Convolutional Layers + 2 Fully Connected Layers**

With this architecture, we obtained **a training accuracy of 97%** and a **validation accuracy of 63%**.
The high training accuracy, combined with relatively low validation accuracy, indicated significant overfitting. This outcome suggested that the model was memorizing training data patterns but struggled to generalize to unseen validation data.

Second Architecture: 4 Convolutional Layers + 2 Fully Connected Layers with **Dropout**
To address overfitting, we expanded the architecture by adding another convolutional layer. Additionally, we introduced **dropout regularization with a rate of 0.8** in the fully connected layers.

This modification yielded notable improvements:
- Training accuracy: 89%
- Validation accuracy: 83%

The added convolutional layer allowed the model to capture more complex patterns, while dropout regularization improved generalization by preventing the network from relying too heavily on specific neurons. By deactivating random neurons during training, dropout effectively reduced overfitting.

However, this architecture—with an aggressive dropout rate of 0.8—may risk underfitting, as it could restrict the model's ability to learn meaningful patterns from data. Recognizing this trade-off, we proceeded to explore alternative architectures with more balanced regularization..

Final Architecture: **5 Convolutional Layers + 2 Fully Connected Layers**

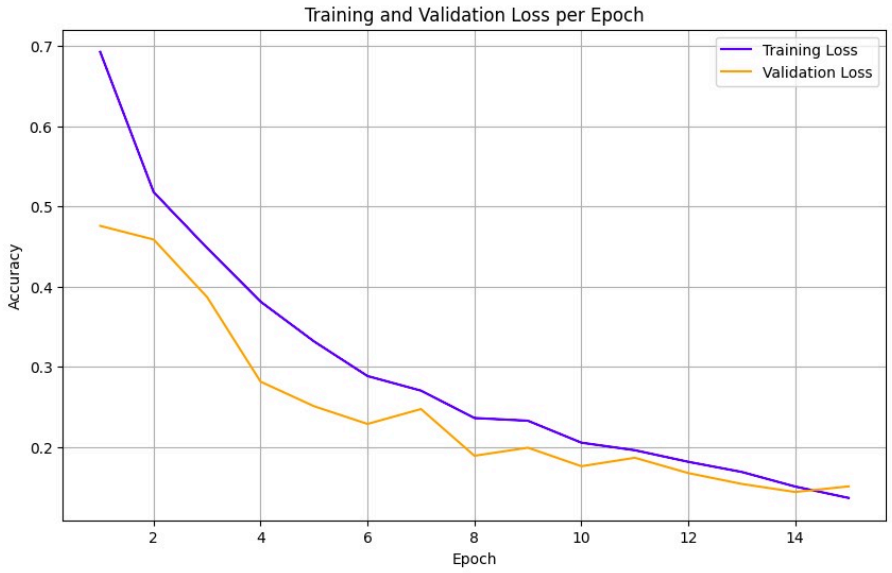
Encouraged by the results of the previous configuration, we further expanded the model to include 5 convolutional layers + 2 fully connected layers . We configured the dropout regularization with a rate of 0.5 and also used L2 regularization (0.5). This architecture yielded the best performance:

- Training accuracy: 87%

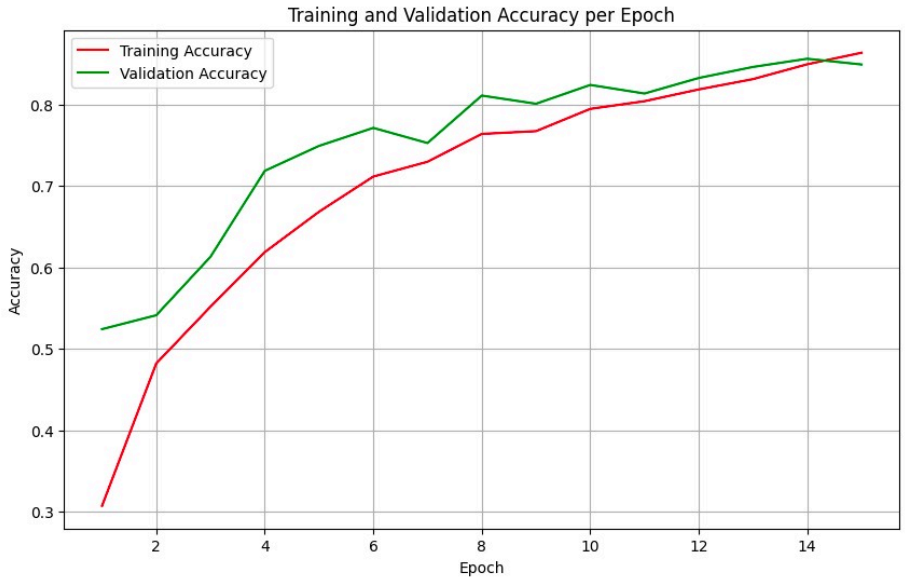
- Validation accuracy: 84%

1.

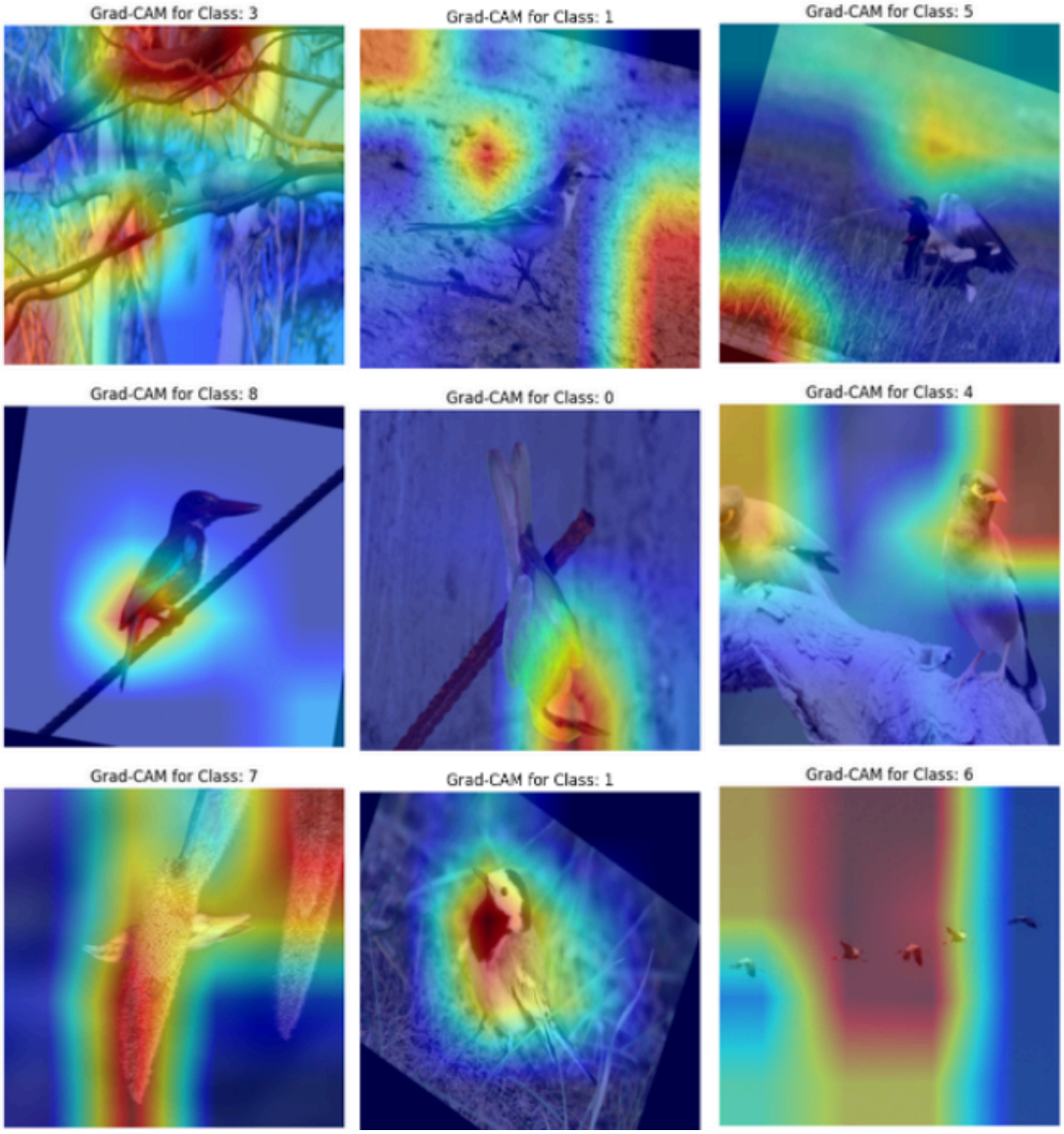
2. **Train and Validation Loss vs. Epochs:** Plot the training and validation loss over the epochs to demonstrate how the model's learning progresses over time.

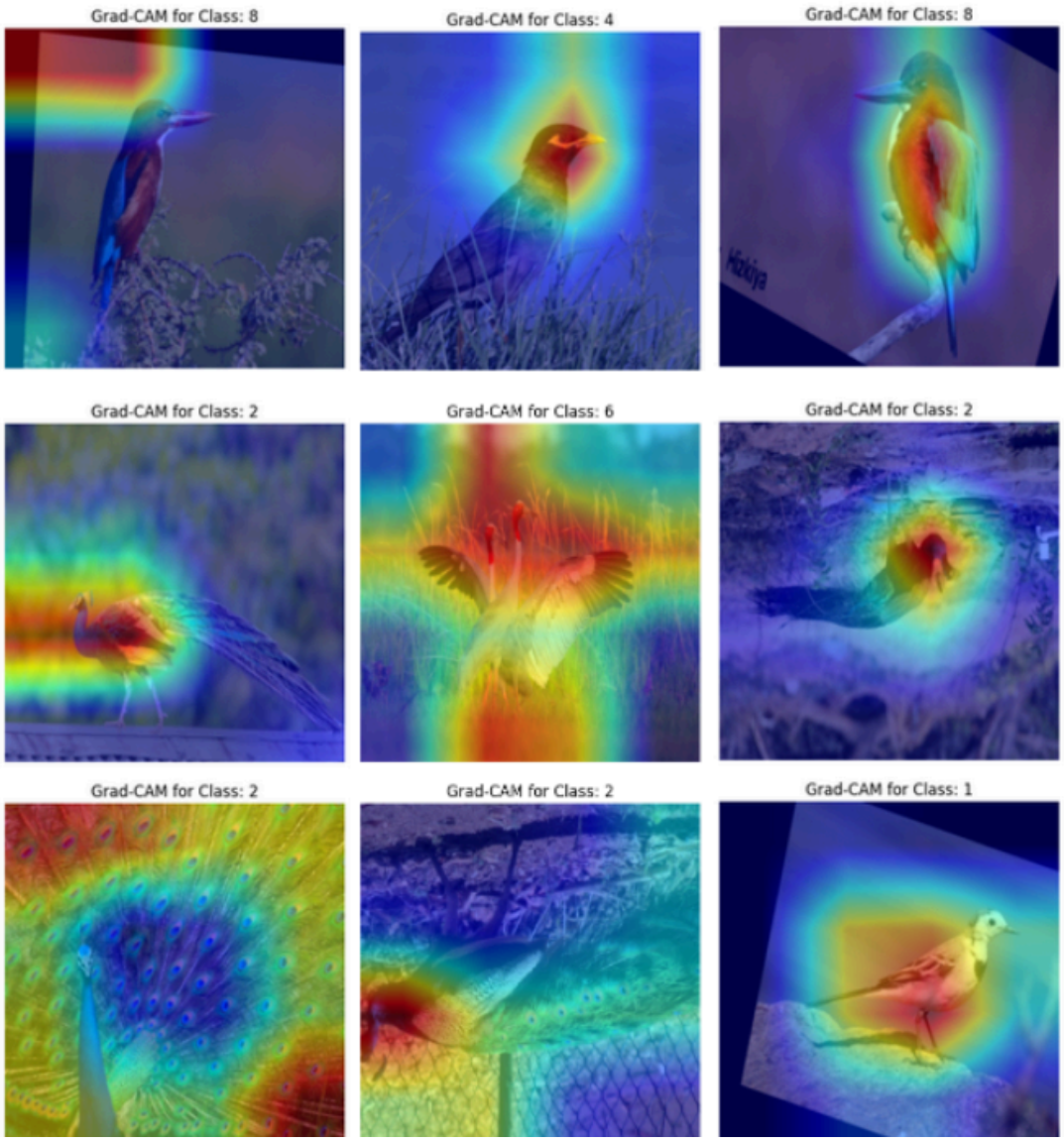


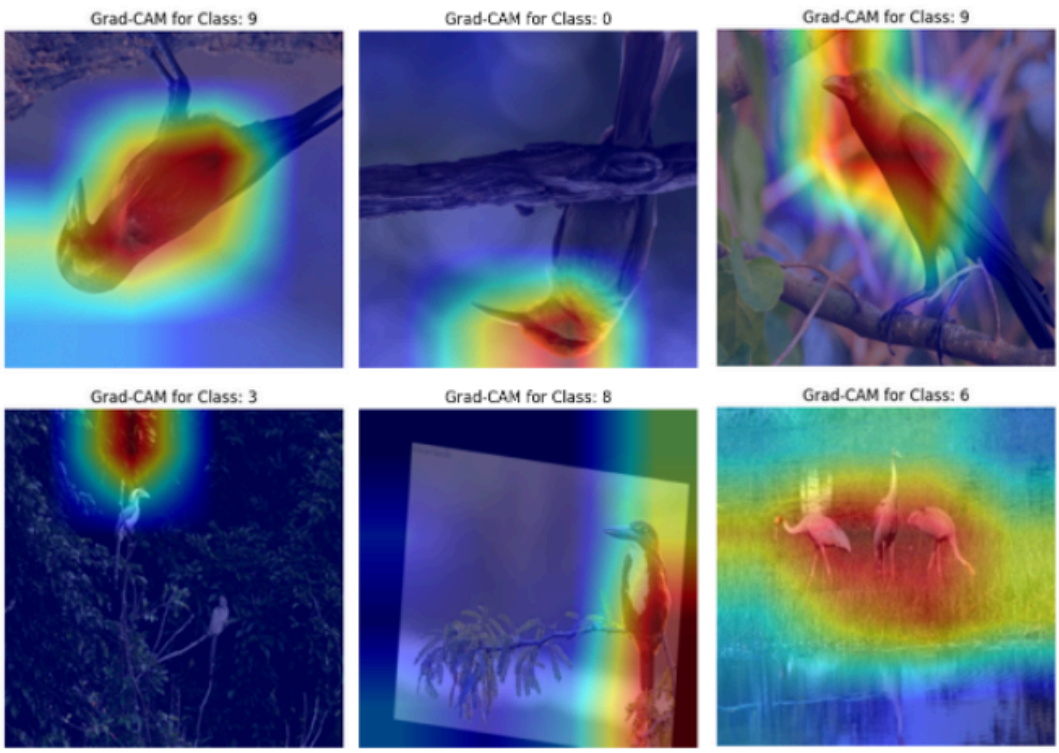
3. **Train and Validation Accuracy vs. Epochs**: Plot the training and validation accuracy across epochs to show how well the model is performing on both the training and validation sets.



4. **Class Activation Maps (CAM**): For each class, provide visualizations of the Class Activation Maps and explain your observations. Discuss how the model interprets different regions of the image for each class and what insights you can draw from the highlighted areas.


Grad-CAM for Class: 3
Grad-CAM for Class: 1
Grad-CAM for Class: 5
Grad-CAM for Class: 8
Grad-CAM for Class: 0
Grad-CAM for Class: 4
Grad-CAM for Class: 7
Grad-CAM for Class: 1
Grad-CAM for Class: 6


Grad-CAM for Class: 8
Grad-CAM for Class: 4
Grad-CAM for Class: 8
Grad-CAM for Class: 2
Grad-CAM for Class: 6
Grad-CAM for Class: 2
Grad-CAM for Class: 2
Grad-CAM for Class: 2
Grad-CAM for Class: 1



5. **Model Architecture Summary: 5 Conv2D layer + 2 Fully Connected Layers**

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 8, 256, 256]             224
       BatchNorm2d-2          [-1, 8, 256, 256]              16
              ReLU-3          [-1, 8, 256, 256]               0
         MaxPool2d-4          [-1, 8, 128, 128]               0
            Conv2d-5         [-1, 16, 128, 128]           1,168
       BatchNorm2d-6         [-1, 16, 128, 128]              32
              ReLU-7         [-1, 16, 128, 128]               0
         MaxPool2d-8           [-1, 16, 64, 64]               0
            Conv2d-9           [-1, 32, 64, 64]           4,640
      BatchNorm2d-10           [-1, 32, 64, 64]              64
             ReLU-11           [-1, 32, 64, 64]               0
        MaxPool2d-12           [-1, 32, 32, 32]               0
           Conv2d-13           [-1, 64, 32, 32]          18,496
      BatchNorm2d-14           [-1, 64, 32, 32]             128
             ReLU-15           [-1, 64, 32, 32]               0
        MaxPool2d-16           [-1, 64, 16, 16]               0
           Conv2d-17          [-1, 128, 16, 16]          73,856
      BatchNorm2d-18          [-1, 128, 16, 16]             256
             ReLU-19          [-1, 128, 16, 16]               0
        MaxPool2d-20            [-1, 128, 8, 8]               0
           Conv2d-21            [-1, 256, 8, 8]         295,168
      BatchNorm2d-22            [-1, 256, 8, 8]             512
             ReLU-23            [-1, 256, 8, 8]               0
        MaxPool2d-24            [-1, 256, 4, 4]               0
           Linear-25                  [-1, 256]       1,048,832
             ReLU-26                  [-1, 256]               0
          Dropout-27                  [-1, 256]               0
           Linear-28                   [-1, 10]           2,570
================================================================
Total params: 1,445,962
Trainable params: 1,445,962
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 25.60
Params size (MB): 5.52
Estimated Total Size (MB): 31.87
----------------------------------------------------------------
```