COL334 Assignment 3.1

Saksham Kumar(2022CS51141), Himanshu Shekhar(2022CS11091)

November 2024

1 Convolutional Neural Network

- We have used a basic AlexNet [1] style implementation of CNN to classify the images of the birds species (`Identify-the-Birds` dataset) into 10 classes.
- The basic layers and the number of them used in our architecture are:

Layer	Convolutional	Batch Normalization	MaxPool	Fully Connected
Number of such layers	7	2D-7, 1D-2	7	3

Table 1: Distribution of different Layers

- We tried different model architectures that were inspired by different standard architectures like `ResNet101, AlexNet, GoogleNet, Stochastic Depth ResNet` but the model derived from `AlexNet` seemed to work best for us.

2 Model Architecture

- Our CNN model architecture derives its inspiration from AlexNet in terms of depth and width.
- The model is quite deep and wide with large number of parameters (`11044874`) which enables it to achieve a high accuracy on validation images right from 1st training epoch ($\approx 71\%$).
- We decided to not use any dropout layers in the architecture since the model was performing quite good in terms of not overfitting the data ($\approx 90\%$validation accuracy at $\approx 93\%$training accuracy).
- Moreover, the research papers promoting dropouts showed a positive effect in the model only when the number of training epochs were quite large ($\approx 20,000$) whereas our model runs for just 30 epochs.

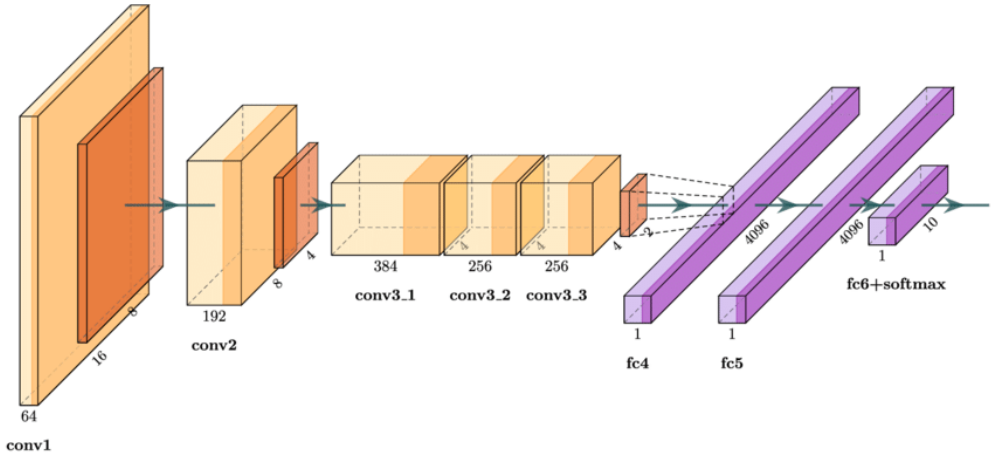

Figure 1: A basic structure of AlexNet type CNN

- The model architecture is:

Layer	Layer Specifications
Convolution2D	inChannels = 3, outChannels = 32, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 32
Activation	SiLU
MaxPool	kernel=2, stride=2, padding=0
Convolution2D	inChannels = 32, outChannels = 64, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 64
Activation	SiLU
MaxPool	kernel=2, stride=2
Convolution2D	inChannels = 64, outChannels = 128, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 128
Activation	SiLU
MaxPool	kernel=2, stride=1, padding=0
Convolution2D	inChannels = 128, outChannels = 256, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 256
Activation	SiLU
MaxPool	kernel=2, stride=1, padding=0
Convolution2D	inChannels = 256, outChannels = 512, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 512
Activation	SiLU
MaxPool	kernel=2, stride=1, padding=0
Convolution2D	inChannels = 512, outChannels = 256, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 256
Activation	SiLU
MaxPool	kernel=2, stride=1, padding=0
Convolution2D	inChannels = 256, outChannels = 128, kernel = 3, stride = 1, padding = 1
BatchNorm2D	size = 128
Activation	SiLU
MaxPool	kernel=2, stride=1, padding=0
Fully-Connected Linear	inputNeurons = 15488, outputNeurons = 512
BatchNorm1D	size = 512
Activation	SiLU
Fully-Connected Linear	inputNeurons = 512, outputNeurons = 128
BatchNorm1D	size = 128
Activation	SiLU
Fully-Connected Linear	inputNeurons = 128, outputNeurons = 10

Table 2: Model Architecture

3 Training Optimizations

3.1 Image Augmentations [3]

- Accuracy before Augmentations: `73.13%`
  Accuracy after Augmentations: `94.68%`

- The transformations done on the images are shown below.

```
TARGET_IMAGE_SIZE = (64,64)
mean = [0.4850, 0.4949, 0.4120]
std = [0.2442, 0.2419, 0.2658]
```

Transformation	Details
Resize	Resize to `TARGET_IMAGE_SIZE`
Random Crop	Random crop of size `TARGET_IMAGE_SIZE` with padding 4
Random Horizontal Flip	Flip image with probability 0.5
To Tensor	Convert image to a PyTorch tensor
Normalize	Normalize with `mean` and `std`

Table 3: Training Transformations (Train Transform 1)

Transformation	Details
Resize	Resize to `TARGET_IMAGE_SIZE`
Random Rotation	Rotate image randomly by 15 degrees
Color Jitter	Adjust brightness, contrast, and saturation
Random Horizontal Flip	Flip image with probability 0.5
To Tensor	Convert image to a PyTorch tensor
Normalize	Normalize with `mean` and `std`

Table 4: Training Transformations (Train Transform 2)

3.2 Class Imbalance

- Accuracy before CI: `94.68%`
  Accuracy after CI: `92.77%`
- Class Imbalance seemed to not work well with our augmentation technique, so we did not implement it.

3.3 Preventing Overfiiting by Regularization/ Dropouts

- Accuracy before Dropouts: `94.68%`
  Accuracy after Dropouts: `89.31%`
- Accuracy before Dropouts: `94.68%`
  Accuracy after Dropouts: `93.89%`
- Both of them did not make any improvement, so we did not apply these techniques too.

4 Results

- To evaluate the model, we used the standard test-train split of 20-80.

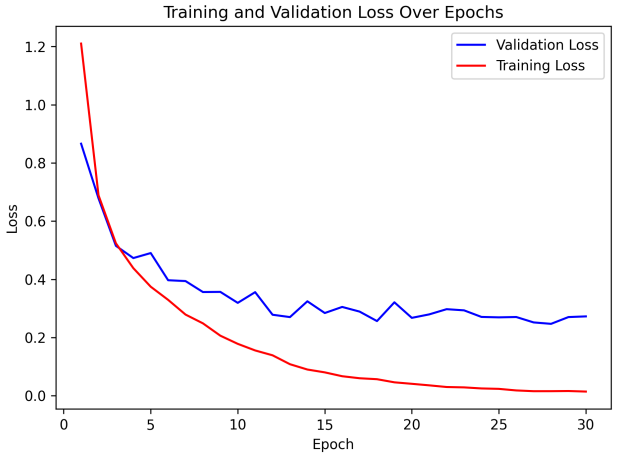

Figure 2: Train and Eval Loss over Epochs

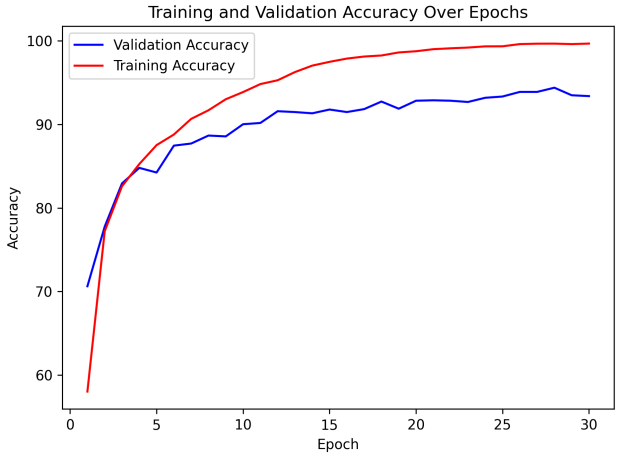

Figure 3: Train and Eval Accuracy over Epochs

- The best accuracy our model achieved when trained with 80% of the training data was **94.68%**. The overall sklearn classification report is shown below and the f1 scores are:

```
Micro-average F1 Score:  0.9468
Macro-average F1 Score:  0.9438
Weighted-average F1 Score:  0.9466
```

Class	Precision	Recall	F1-Score	Support
0	0.95	0.98	0.96	216
1	0.97	0.99	0.98	192
2	0.96	0.92	0.94	144
3	0.87	0.89	0.88	168
4	0.94	0.98	0.96	240
5	0.98	0.99	0.98	240
6	0.93	0.94	0.93	240
7	0.91	0.89	0.90	192
8	1.00	0.98	0.99	192
9	0.93	0.88	0.90	168
Accuracy	0.95			1992
Macro Avg	0.95	0.94	0.94	1992
Weighted Avg	0.95	0.95	0.95	1992

Table 5: Classification Report

5 Grad CAM Visualization

- The target layer for gradCAM [2] visualization is the 7th convolution layer, the last convolution layer in the model. Note that a transition from a blue region to a red region in the following images below implies increase in gradient or importance (a transition from cold to hot regions).
- GradCAM helps to visualise the model performance by highlighting the areas where the model lays its focus. A good model would generally focus on the subject of classification rather than the background.

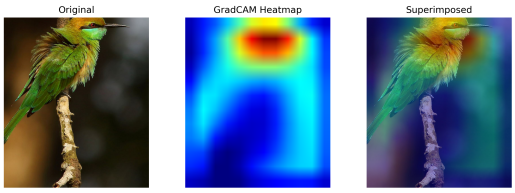

Figure 4: Class 0

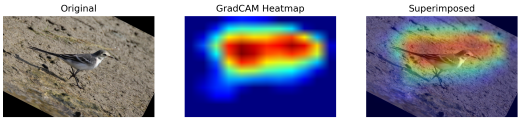

Figure 5: Class 1

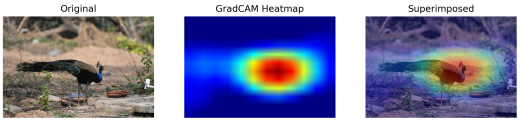

Figure 6: Class 2

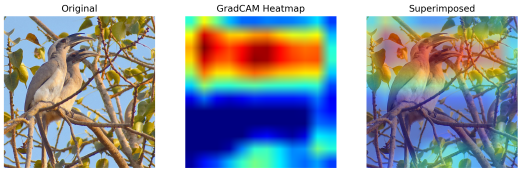

Figure 7: Class 3

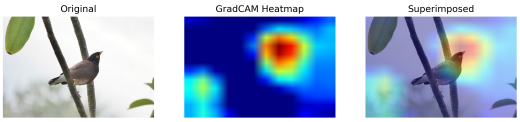

Figure 8: Class 4

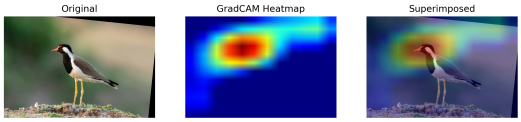

Figure 9: Class 5

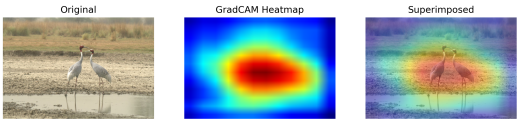

Figure 10: Class 6

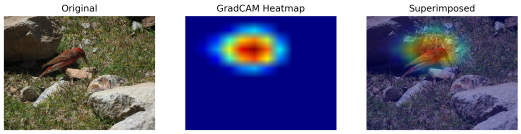

Figure 11: Class 7

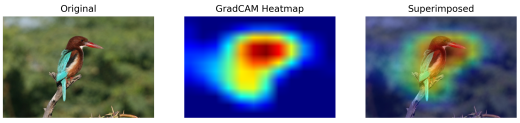

Figure 12: Class 8

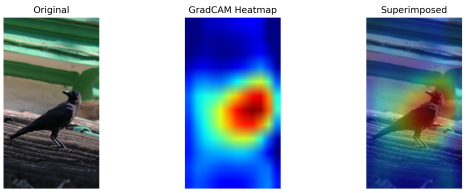

Figure 13: Class 9

5.1 Interpretation

- From the CAM images above, it can be observed that the most important regions of the image from which the model identifies the class are: `beak`, `head`. It is not paying much attention to the `feathers` or `legs`.
- Moreover, the model is also able to detect multiple subjects in the image as observed from the CAM image of class 3 and 6. The region of interest is far wider than the other images with a single subject.
- When the CAM images are generated for all the convolution layers, it is observed that the red regions start to concentrate and decrease in area as we move to the deeper layers. This also makes sense as the top layers would try to identify all the gradients in the image, therefore identifying all the objects but on going deeper, the area of interest must tone down to just the bird.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[2] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[3] Mingle Xu, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, 137:109347, 2023.