Report - GradCAM

Vidit Kumar

June 2025

1 Data Augmentation

Two models are to be fine-tuned on the provided dataset, namely `Resnt50` and `InceptionV3`. The data augmentations for both models is discussed below:

1.1 ResNet50

The images have been resized to 256×256, then they have been converted to RGB, after which they have been normalized and their mean and standard deviations are tuned to the one provided in the original paper.

1.2 InceptionV3

The images have been resized to 256×256, then cropped to 224×224, after that they have been gray-scaled and merged into 1 channel, after which they have been normalized and their mean and standard deviations are tuned to the one in `ResNet50`.

2 ResNet Model Overview

The default pre-trained weights from `PyTorch` have been imported and the models is fine-tuned on the given dataset.

2.1 ResNetModule

Each bottleneck block contains:

- 1×1 conv (reduce dims): $C_{in} \to C_{out}$
- 3×3 conv (stride as needed): $C_{out} \to C_{out}$
- 1×1 conv (expand dims): $C_{out} \to 4 \cdot C_{out}$
- BatchNorm after each conv
- Residual addition (with optional downsampling)
- Final ReLU

2.2 Network Architecture

The complete model, starting from an input of shape $(3, H, W)$, follows this sequence:

Layer	**Output Shape**	**Details**
Conv1 + BN + ReLU	$64 \times \frac{H}{2} \times \frac{W}{2}$	7×7 Conv, stride 2, padding 3
MaxPool	$64 \times \frac{H}{4} \times \frac{W}{4}$	3×3 MaxPool, stride 2, padding 1
Layer1	$256 \times \frac{H}{4} \times \frac{W}{4}$	3 Bottleneck blocks, 64 → 256
Layer2	$512 \times \frac{H}{8} \times \frac{W}{8}$	4 Bottleneck blocks, 128 → 512, stride 2
Layer3	$1024 \times \frac{H}{16} \times \frac{W}{16}$	6 Bottleneck blocks, 256 → 1024, stride 2
Layer4	$2048 \times \frac{H}{32} \times \frac{W}{32}$	3 Bottleneck blocks, 512 → 2048, stride 2
AvgPool	$2048 \times 1 \times 1$	Global average pooling
Flatten	2048	Flatten
FC	512	Fully connected layer

Table 1: Architecture of `myResNet50` model

3 InceptionV3 Model Overview

The default pre-trained weights from `PyTorch` have been imported and the models is fine-tuned on the given dataset.

3.1 InceptionModule

Each Inception block performs parallel operations on the same input:

- **Branch 1:** 1×1 Conv, output channels $= b_1$
- **Branch 2:** 1×1 Conv $\rightarrow 3 \times 3$ Conv, output channels $= b_{2b}$
- **Branch 3:** 1×1 Conv $\rightarrow 5 \times 5$ Conv, output channels $= b_{3b}$
- **Branch 4:** 3×3 MaxPool $\rightarrow 1 \times 1$ Conv, output channels = pool_proj

All branches use ReLU after each convolution. Outputs are concatenated along the channel dimension.

3.2 Network Architecture

The complete model, starting from an input of shape $(3, H, W)$, follows this sequence:

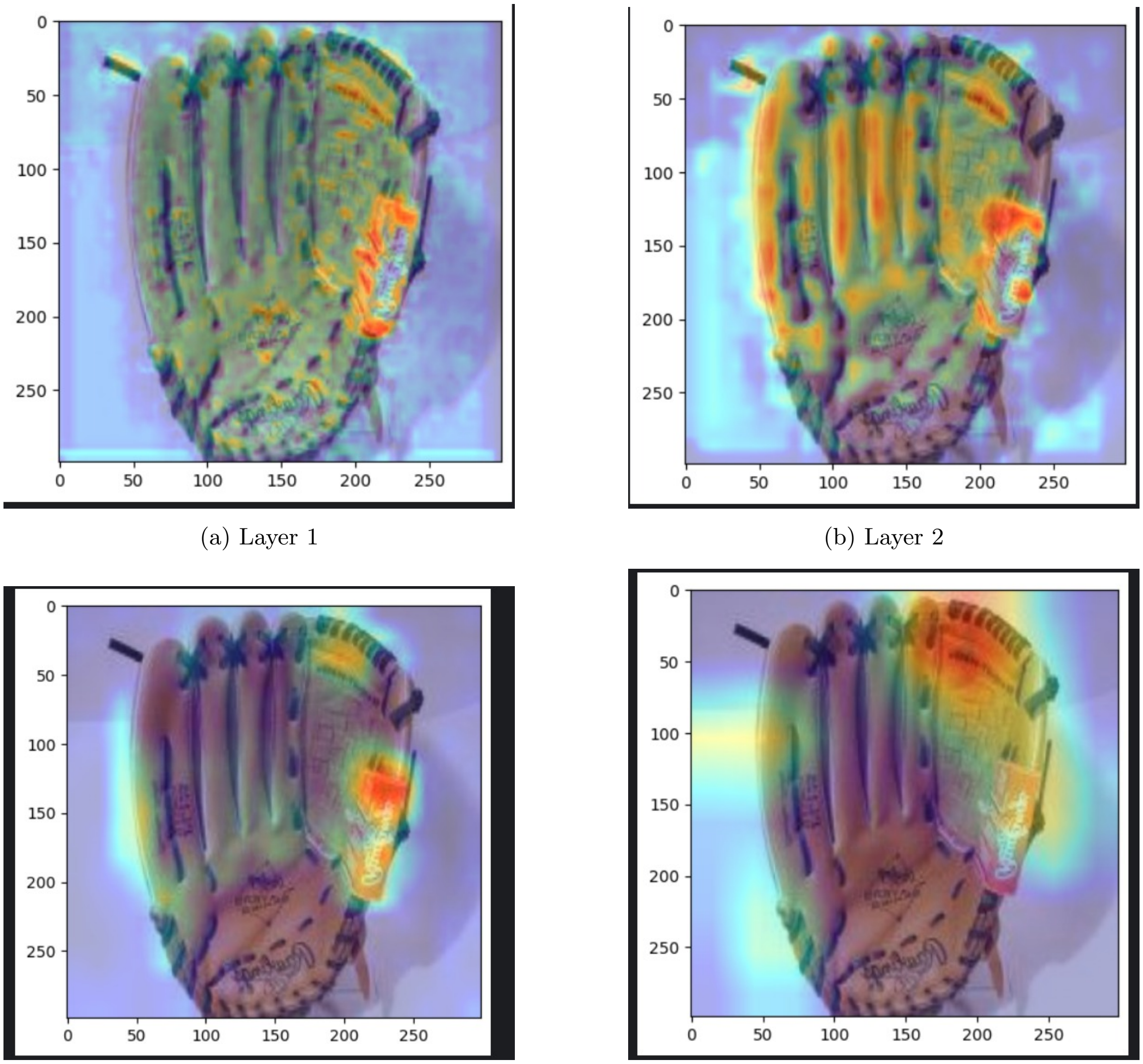
Layer	**Output Shape**	**Details**
Conv1 + ReLU	$32 \times \frac{H}{2} \times \frac{W}{2}$	3×3 Conv, stride 2
Conv2 + ReLU	$32 \times \frac{H}{2} \times \frac{W}{2}$	3×3 Conv
Conv3 + ReLU	$64 \times \frac{H}{2} \times \frac{W}{2}$	3×3 Conv, padding 1
MaxPool	$64 \times \frac{H}{4} \times \frac{W}{4}$	3×3 MaxPool, stride 2
Inception3a	$256 \times \frac{H}{4} \times \frac{W}{4}$	$(64, 48, 64, 64, 96, 32)$
Inception3b	$288 \times \frac{H}{4} \times \frac{W}{4}$	$(64, 48, 64, 64, 96, 64)$
MaxPool	$288 \times \frac{H}{8} \times \frac{W}{8}$	3×3 MaxPool, stride 2
Inception4a	$288 \times \frac{H}{8} \times \frac{W}{8}$	$(64, 48, 64, 64, 96, 64)$
Inception4b	$288 \times \frac{H}{8} \times \frac{W}{8}$	$(64, 48, 64, 64, 96, 64)$
AvgPool	$288 \times 1 \times 1$	Global average pooling
Flatten	288	Flatten
Dropout	288	$p = 0.4$
FC	257	Fully connected layer

Table 2: Architecture of `myInceptionV3` model

4 Results

For `myResNet50` the top-5 accuracy for validation set is 91.46% and for test set it is 99.96% (This one seems fishy). For `myInceptionV3`, the top-5 accuracy was approximately 45%.

The results have been verified by GradCAM:



(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Layer 4

Figure 1: Visualizations of ResNet Layers

5 Sources

- PyTorch Documentation and GitHub repositories
- Intuition for ensembler: `https://arxiv.org/pdf/1409.1556`
- GradCAM reference (skimmed): `https://arxiv.org/pdf/1610.02391`
- Some help from ChatGPT-4o for minor debugging