

---

# MINI RESEARCH PROJECT

Grad CAM for CNN Interpretability

Carlos Mendez

MSCS 2201

---



---

# TL;DR: MOTIVATION, MAIN IDEA, AND RESULTS

Deep CNNs are highly accurate but operate as black boxes (we cannot know how they are determining their choice).

Grad-CAM generates heatmaps showing which image regions influence the model's decision.

Results: The model focuses correctly on meaningful features (dog face, car doors), and exposes failure cases where it attends to irrelevant areas.

---

---

# TABLE OF CONTENTS

01	02	03	04	05	06	07
Motivation	Literature Review	Method / Approach	Demonstration (YouTube)	Results (Dog, Personal Dog, Car, Failure Case)	Discussion & Limitations	Conclusion

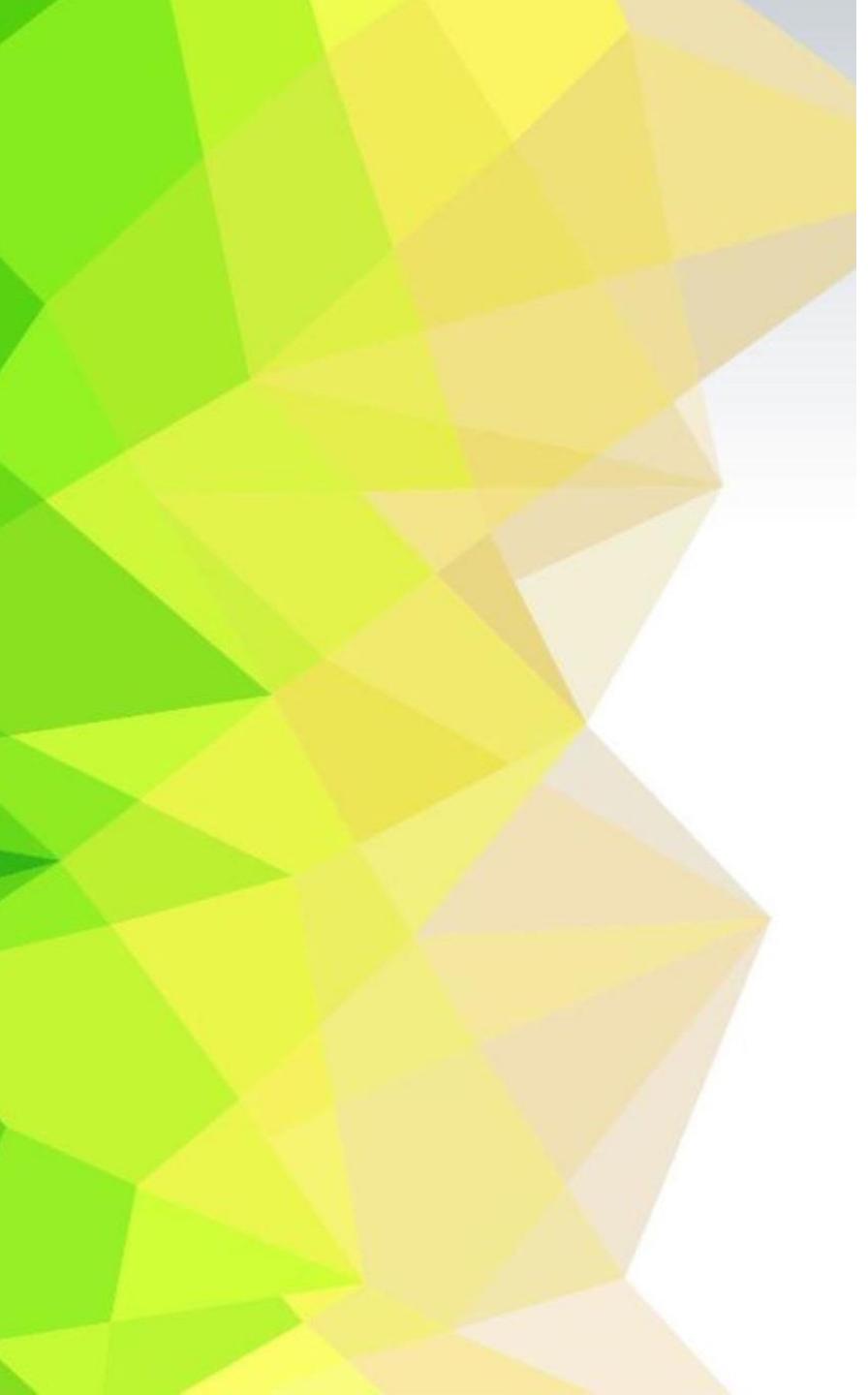
# MOTIVATION

Neural networks perform extremely well but lack transparency.

Interpretability increases trust, safety, and debuggability.

For vision tasks, visual explanations like heatmaps are the most intuitive.

Grad-CAM allows us to “look inside” a CNN without changing its architecture.



---

# LITERATURE REVIEW (PART 1)

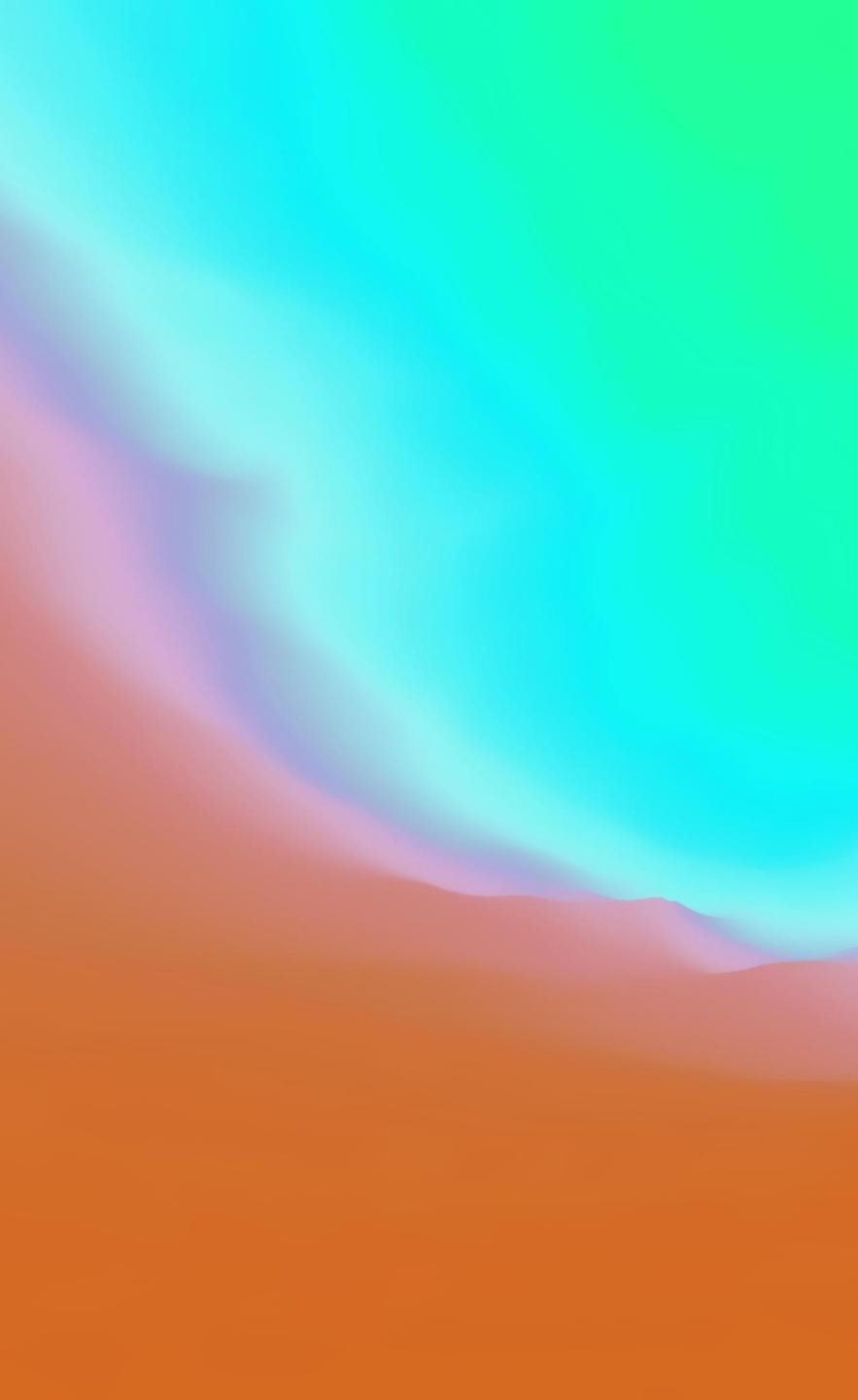
---

- Key interpretability methods:
  - LIME (2016): Creates local surrogate models to explain individual predictions.
  - SHAP (2017): Uses Shapley values to measure feature importance.
  - Grad-CAM (2017): Uses gradients + activations to produce class-specific heatmaps for CNNs.

# LITERATURE REVIEW (PART 2)

---

- Interpretability vs Explainability (Lecture 9):
  - Interpretability: Understanding input-output behavior.
  - Explainability: Producing human-like reasoning after the fact.
  - Grad-CAM is a post-hoc explainability technique; it explains the model's behavior without modifying the model.



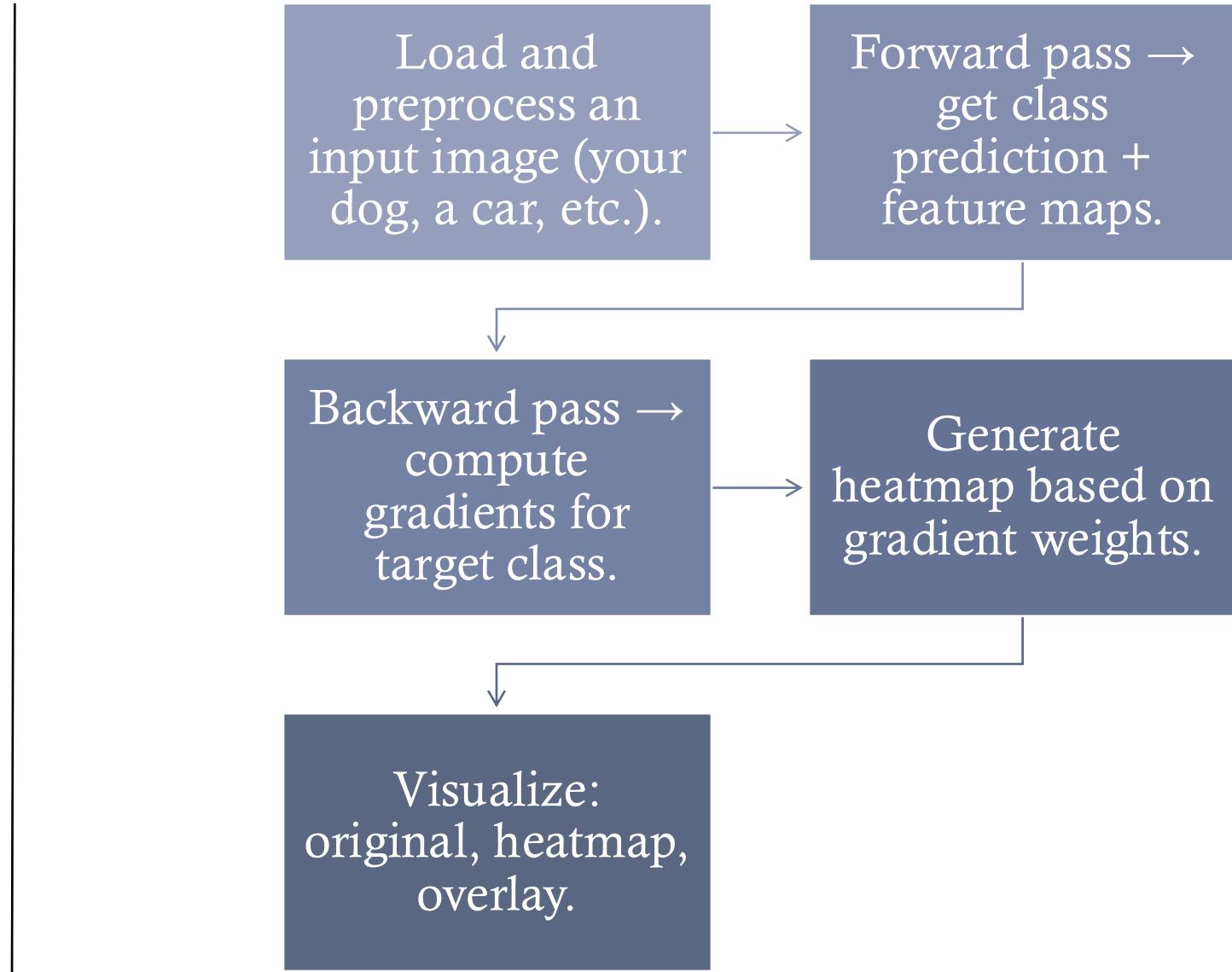
---

# APPROACH / METHOD USED

---

- Pretrained ResNet50 (ImageNet) used as the model.
- Capture activations and gradients from the final convolutional layer (layer4).
- Combine gradients with activations to compute a weighted heatmap.
- Resize and overlay heatmap on top of original image.
- Implemented end-to-end in Google Colab using PyTorch.

# IMPLEMENTATION OVERVIEW



```
class GradCAM:
    def __init__(self, model, target_layer):
        """
        model: a CNN model (e.g., torchvision.models.resnet50)
        target_layer: name of the last conv layer (e.g., "layer4")
        """
        self.model = model
        self.model.eval()

        # Get the layer module by name
        self.target_layer = dict(self.model.named_modules())[target_layer]

        self.activations = None
        self.gradients = None

        # Forward hook: store activations
        def forward_hook(module, input, output):
            self.activations = output

        # Backward hook: store gradients
        def backward_hook(module, grad_in, grad_out):
            self.gradients = grad_out[0]

        self.target_layer.register_forward_hook(forward_hook)
        self.target_layer.register_backward_hook(backward_hook)

    def generate(self, input_tensor, class_idx=None):
        """
        input_tensor: shape (1, C, H, W)
        class_idx: target class index; if None, use predicted class.
        returns:
        - cam: Grad-CAM heatmap as numpy array in [0,1]
        - class_idx: index of the class used
        """
        # Forward pass
        output = self.model(input_tensor)

        # If no target class is provided, use the predicted one
        if class_idx is None:
            class_idx = output.argmax(dim=1).item()

        # Zero gradients
        self.model.zero_grad()

        # Backward w.r.t. target class score
        target = output[0, class_idx]
        target.backward()

        # activations: [B, C, H, W]
        # gradients: [B, C, H, W]
        gradients = self.gradients
        activations = self.activations

        # Global average pooling over H and W
```

---

# IMPLEMENTATION OVERVIEW

- Defines the Grad-CAM class, which captures activations and gradients from the last convolution layer.
- Combines these to create a heatmap showing what parts of the image the model used for its prediction.
- Loads a pretrained ResNet50 model and attaches Grad-CAM to its final conv block (`layer4`).
- This allows us to generate clear visual explanations of the model's decisions

# IMPLEMENTATION OVERVIEW

- Loads an image (dog, personal photo, car, or ambiguous image) and preprocesses it so ResNet50 can read it.
- Converts the image to a tensor, normalizes it, and sends it to the model's device.
- Runs the image through the model and uses Grad-CAM to compute a class-specific heatmap.
- Identifies the predicted class and returns the raw Grad-CAM map for visualization in the next step.

```
# Load pretrained ResNet50
model = models.resnet50(weights=models.ResNet50_Weights.IMGNET1K_V2)

# Send to device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

print("Using device:", device)

# Initialize Grad-CAM with the last convolution block "layer4"
gradcam = GradCAM(model, target_layer="layer4")

Downloading: "https://download.pytorch.org/models/resnet50-11ad3fa6.pth" to /root/.cache/torch/hub/checkpoints/resnet50-11ad3fa6.pth
100%|██████████| 97.8M/97.8M [00:00<00:00, 127MB/s]
Using device: cpu
```

```
# Change this filename to your uploaded image
image_path = "/Abstract.jpg" # for example "dog.jpg" or "car.jpg"

# Load and preprocess
img = Image.open(image_path).convert("RGB")
input_tensor = preprocess(img).unsqueeze(0).to(device)

plt.imshow(img)
plt.title("Input Image")
plt.axis("off")
plt.show()
```



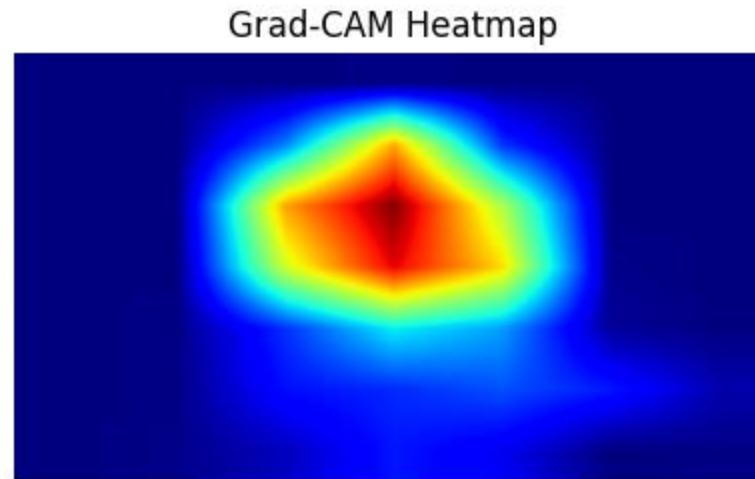
# DEMO (YOUTUBE LINK)

---

- <https://youtu.be/bLNHfx-hbgU>

# RESULTS: GOOGLE DOG IMAGE

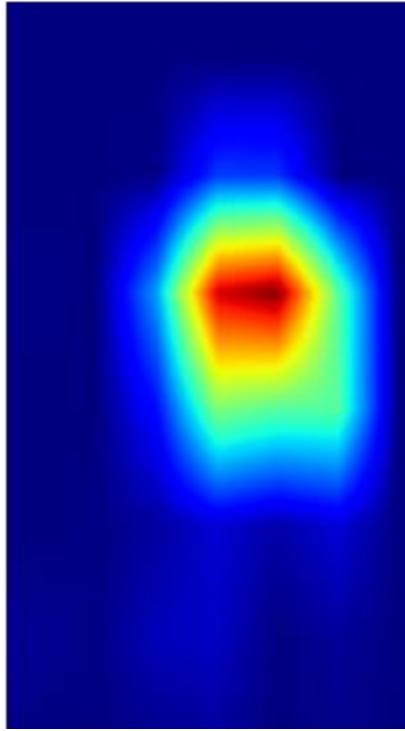
- The heatmap highlights the dog's face, especially the eyes and snout.
- Model attends to the most discriminative regions for dog classification.
- This is a successful explanation: behavior matches human intuition.



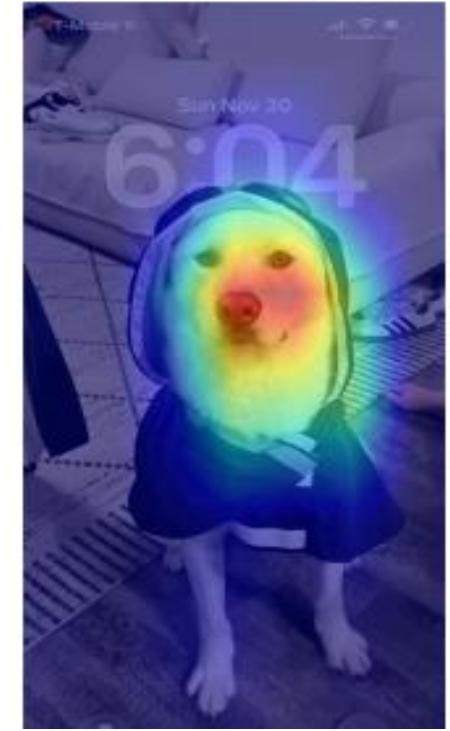
Original



Grad-CAM Heatmap

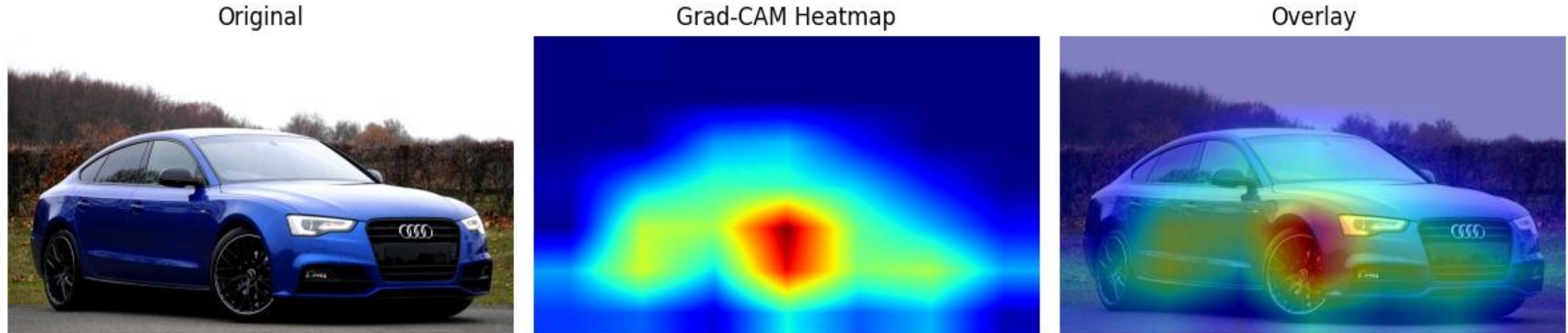


Overlay



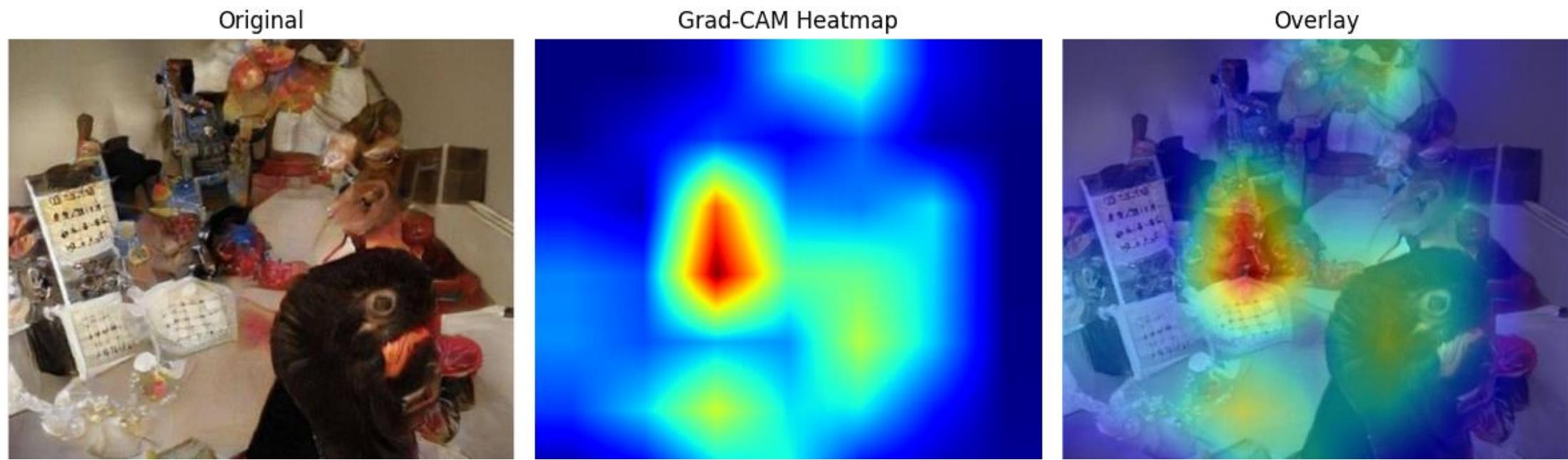
# RESULTS: MY DOG (PERSONAL PHOTO)

- The model still focuses on the face and upper body even though this is not an ImageNet training image.
- Shows good generalization to personal photos.
- Some background activation may appear depending on lighting or pose. (For example, the side of the rain cap)



## RESULTS: CAR IMAGE

- Heatmap activation clusters around wheels, headlights, and body edges.
  - CNN focuses on structural features that help classify vehicles.
  - This aligns with how humans identify cars, another strong, meaningful explanation.
-



## RESULTS: AMBIGIOUS IMAGE / FAILURE CASE

- The model's attention is inconsistent or misplaced; heatmap may focus on background, shadows, or random textures.
- This indicates model uncertainty and lack of discriminative features.
- Great example of Grad-CAM exposing when a model is “guessing.”

---

# DISCUSSION & LIMITATIONS

- Grad-CAM is post-hoc, so explanations approximate, not perfect.
  - Sensitive to choice of convolutional layer.
  - Heatmaps sometimes highlight irrelevant regions.
  - Not suitable for non-CNN architectures unless adapted
  - Does not guarantee “true” model reasoning, but gives useful insights.
-

---

# CONCLUSION

- Grad-CAM is a simple, powerful tool for CNN interpretability.
  - Visual heatmaps help validate correct behavior and detect failure cases.
  - Works well for everyday images, including custom photos.
  - Future work: Grad-CAM++, Vision Transformer explanations, Integrated Gradients.
-

---

# REFERENCES

- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). *Grad-CAM: Visual explanations from deep networks via gradient-based localization*. ICCV, 618–626. <https://doi.org/10.1109/ICCV.2017.74>
  - Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “*Why should I trust you?*” Explaining the predictions of any classifier. KDD, 1135–1144. <https://doi.org/10.1145/2939672.2939778>
  - Lundberg, S. M., & Lee, S.-I. (2017). *A unified approach to interpreting model predictions*. NeurIPS 30.
  - He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. CVPR, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
  - Paszke, A., Gross, S., Massa, F., et al. (2019). *PyTorch: An imperative style, high-performance deep learning library*. NeurIPS 32. <https://pytorch.org>
  - Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). *A survey of methods for explaining black box models*. ACM Computing Surveys, 51(5), 93.
  - He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. CVPR, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
  - Sofia University. (2024). *Lecture 9: Interpretability & Explainability*. MSCS 2201 – Artificial Intelligence.
-