Fine Tuning & Interpretability Analysis of a Pretrained Vision Transformer (ViT)

Chitrashree Shankaranandha
MSCS2001-1 Artificial Intelligence
Mini Research Project

Overview

Motivation:
Interpretability in AI is key to building trust, debugging models, and ensuring fairness.

Main Idea:
Fine-tune a Vision Transformer (ViT) on CIFAR-10 and visualize its decision-making using Captum's Integrated Gradients.

Result:
Model accurately classifies CIFAR-10 images and highlights input regions contributing to decisions.

Literature Review

- **Interpretability Techniques**:
  - Feature Attribution: Integrated Gradients (IG)
  - Visual Attention: Attention Rollout for Transformers
- **Key Papers**:
  - "Explaining Explanations: An Overview of Interpretability of ML" (Doshi-Velez & Kim, 2017)
  - "Attention is All You Need" (Vaswani et al., 2017)
  - Captum Library Documentation (Facebook AI)

Approach

- Dataset: CIFAR-10 (10 classes, 60K images)
- Model: Pre-trained ViT from timm, fine-tuned on CIFAR-10
- Optimizer: Adam
- Framework: PyTorch, Captum, TensorBoard
- Batch Size: 8 (adjusted for GPU memory)
- Number of Epochs : 10
- Loss function : CrossEntropy (categorical)
- Visualization: Captum Integrated Gradients heatmaps

```
Epoch [10/10], Step [5100/6250], Loss: 0.0096, Accuracy: 99.89%
Epoch [10/10], Step [5200/6250], Loss: 0.0014, Accuracy: 99.90%
Epoch [10/10], Step [5300/6250], Loss: 0.0032, Accuracy: 99.89%
Epoch [10/10], Step [5400/6250], Loss: 0.0007, Accuracy: 99.90%
Epoch [10/10], Step [5500/6250], Loss: 0.0004, Accuracy: 99.90%
Epoch [10/10], Step [5600/6250], Loss: 0.0012, Accuracy: 99.90%
Epoch [10/10], Step [5700/6250], Loss: 0.0022, Accuracy: 99.90%
Epoch [10/10], Step [5800/6250], Loss: 0.0001, Accuracy: 99.90%
Epoch [10/10], Step [5900/6250], Loss: 0.0009, Accuracy: 99.90%
Epoch [10/10], Step [6000/6250], Loss: 0.0010, Accuracy: 99.90%
Epoch [10/10], Step [6100/6250], Loss: 0.0046, Accuracy: 99.90%
Epoch [10/10], Step [6200/6250], Loss: 0.0004, Accuracy: 99.90%
Epoch 10 Test Accuracy: 94.71%
Training complete. Best model saved.
```

Model Training and Dataset

```
# Load pre-trained ViT model from timm and adapt for CIFAR-10
model = timm.create_model('vit_base_patch16_224', pretrained=True)
model.head = nn.Linear(model.head.in_features, 10)  # CIFAR-10 has 10 classes
model = model.to(device)
```

CIFAR-10 Dataset:

- **Training set size** = 50,000 images
- **Test set size** = 10,000 images
- **Number of classes** = 10 -> ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

batch size = 8, so:

Images per Epoch:

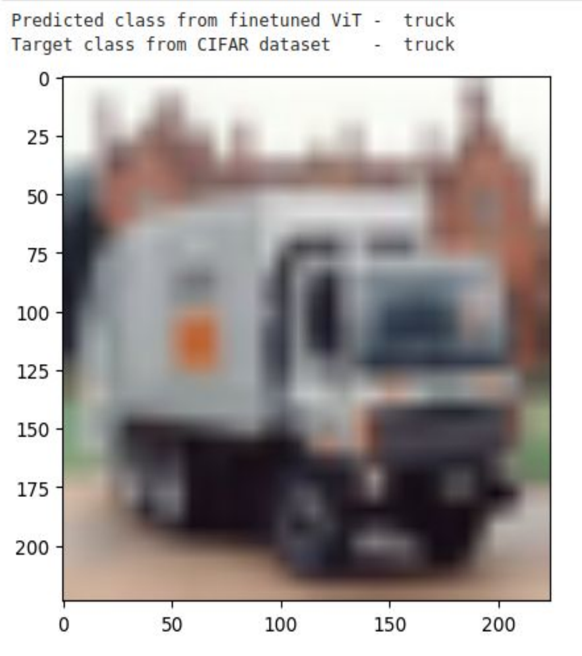
- **Training images per epoch** = 50,000
- **Batches per epoch** = 50,000 / 8 = **6,250 batches**

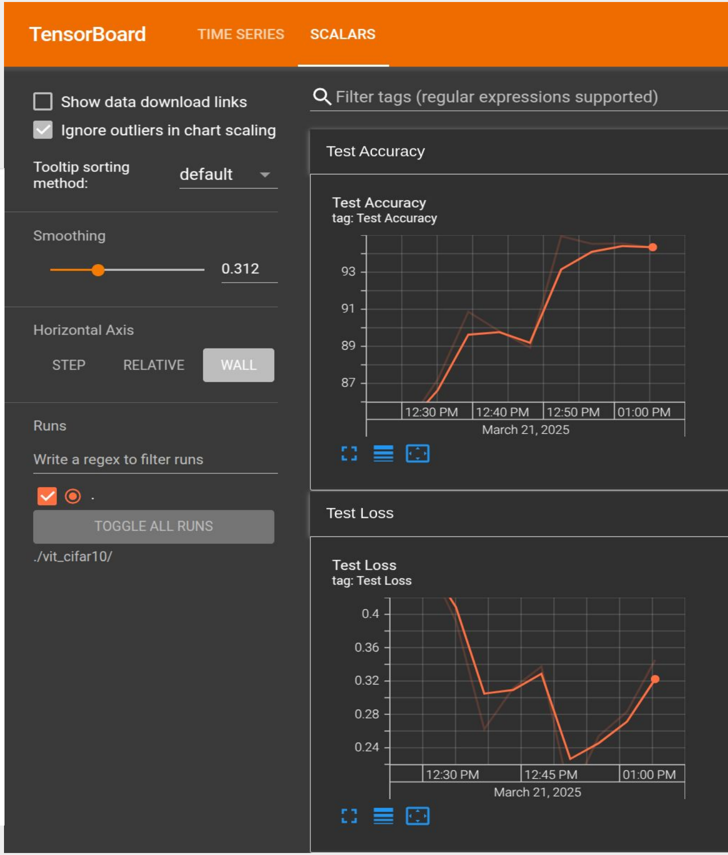
Total Over 10 Epochs:

- **Total training images served** = 50,000 × 10 = **500,000 images**
- **Total forward/backward passes** = 6,250 × 10 = **62,500 mini-batches**

Results, Loss, & Accuracy

- **Training Accuracy**: ~99.90 %
- **Test Accuracy**: ~94.71 %





Integrated Gradients and their usage for Interpretability

- **Integrated Gradients (IG)**
  - A feature attribution method that computes the contribution of each input feature (pixel) to the model's prediction.
  - It integrates gradients of the model's output with respect to the input along a path from a baseline (e.g., black image) to the actual input.

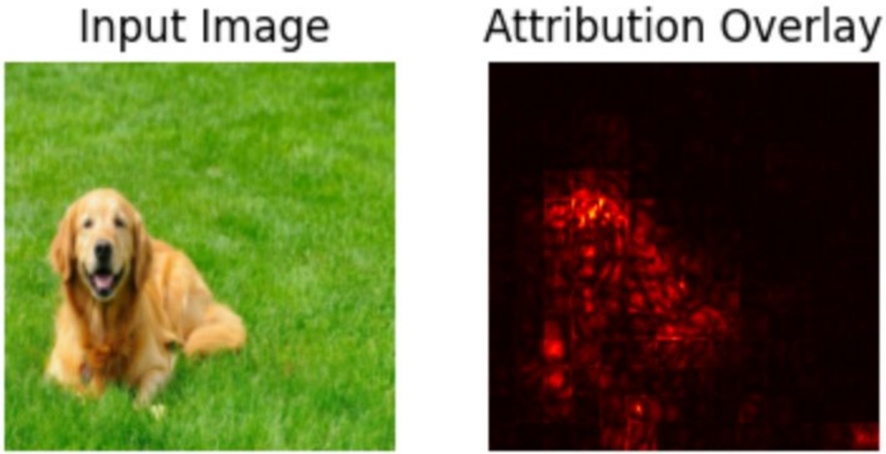
- **Why use IG for Interpretability?**
  - Provides a principled way to attribute the prediction to input features.
  - Helps visualize "what the model focused on" when making a decision.
  - Useful for debugging, bias detection, and model trust.

Interpretability Analysis using Integrated Gradients

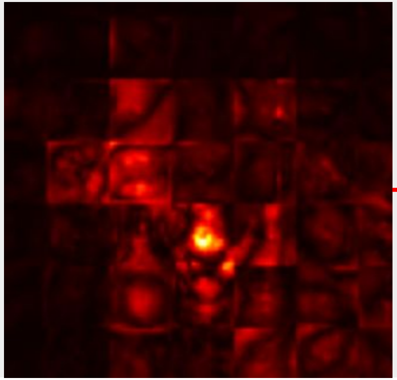
```
show_attributions(images[index], attributions)
print("Predicted class from finetuned ViT - ", train_dataset.classes[predicted_class[index].item()])
print("Target class from CIFAR dataset    - ", train_dataset.classes[labels[index].item()])
```


```
Predicted class from finetuned ViT -   horse
Target class from CIFAR dataset    -   horse
```


```
Predicted class from finetuned ViT -   truck
Target class from CIFAR dataset    -   truck
```

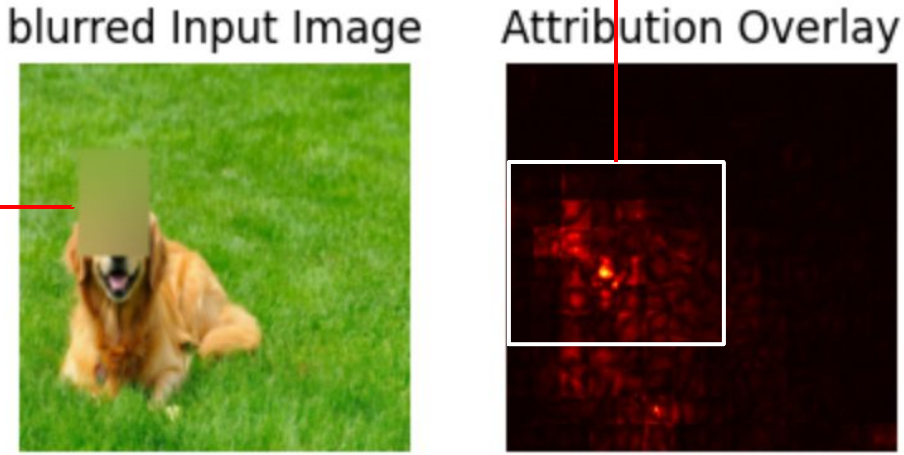



```
Predicted class from finetuned ViT -   dog
Target class from CIFAR dataset    -   dog
```

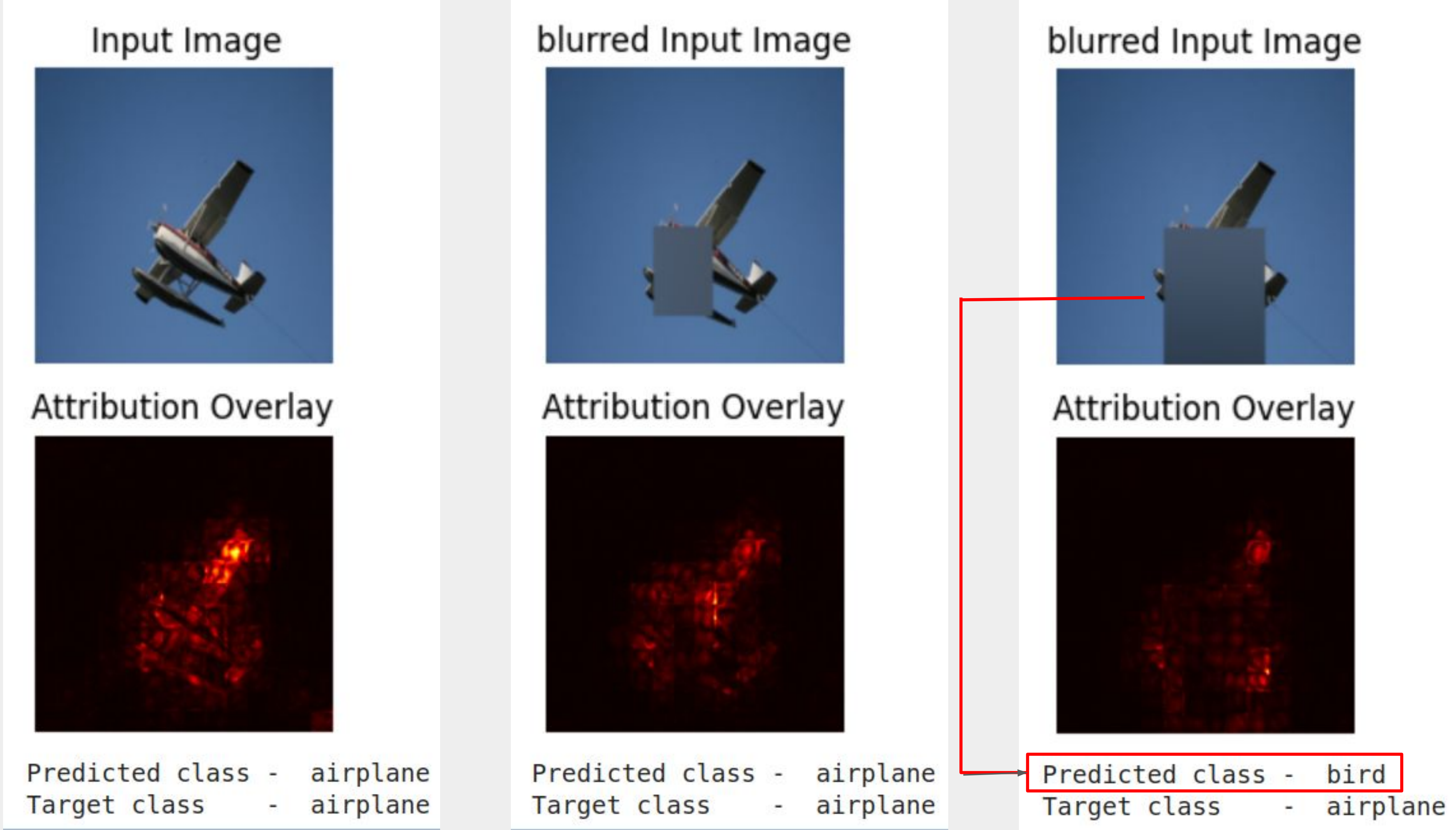


Adversarial attack (perturbation)

```
# Perturbation: Blur a region

img_blurred = img.copy()

blurred_region = img.crop((180, 180, 400, 400)).filter(ImageFilter.GaussianBlur(radius=100))

img_blurred.paste(blurred_region, (180, 180))
```




```
Predicted class from finetuned ViT -   deer
Target class from CIFAR dataset    -   dog
```


Input Image
Attribution Overlay
Predicted class - airplane
Target class - airplane
blurred Input Image
Attribution Overlay
Predicted class - airplane
Target class - airplane
blurred Input Image
Attribution Overlay
Predicted class - bird
Target class - airplane

Demo & Resources

GitHub Link: https://github.com/ChitrashreeShankaranandha/ai2025-spring-sofia/tree/main

YouTube Demo: https://youtu.be/_20NqslWbRM

Tools Used: PyTorch, timm, Captum, TensorBoard

Future Work: Add attention rollout, explore COCO dataset, scale to ViT-Large

Thank you ! ..

- Chitrashree Shankaranandha