Trustworthy Edge AI: Explainable MobileViT-S with Accurate Visual and Textual Reasoning

Shubham Limbachiya (Banner ID: 00138964)
Smit Patel (Banner ID: 001394365)
Course: AI Capstone Project (CSCI-5961-01)

Project repository (GitHub): https://github.com/SL-025/Explainable-AI-in-Edge-Deployed-Classification

Hugging Face Demo (Gradio Interface): https://huggingface.co/spaces/SL025/Trustworthy_Edge_AI

***Abstract*—Modern vision models such as convolutional neural networks and Vision Transformers achieve high accuracy on images but also behaves as black box for the predictions. This work presents an explainable version of MobileViT-S which is fine tuned on CIFAR-10 dataset with an interpretability pipeline designed for deployment on laptops and other constrained hardware. The model is trained with label smoothing and a warmup cosine learning rate schedule and the predictions are interpreted using Grad-CAM applied at the final feature layer. To assess the reliability of explanations we introduce an Interpretability Stability Score (ISS) that computes the Structural Similarity Index (SSIM) between Grad-CAM heatmaps for an original image and a slightly perturbed version of the same image. In addition, we generate structured natural language descriptions that summarize the region of attention, edge structure and color cues used by the model. Experiments on CIFAR-10 shows that the fine tuned MobileViT-S achieves approximately 95–96% validation accuracy while producing visually meaningful attention maps. At the same time ISS scores reveals that these explanations can be sensitive to small perturbations which highlights a gap between predictive performance and explanation stability. The resulting system provides an interpretable and stable vision model needed for trustworthy edge deployment.**

I. Introduction

A. Problem Definition

Deep learning has enabled major advances in computer vision with many architectures such as ResNet, MobileNet and Vision Transformers with very human level accuracy on various datasets. Even because of this progress most of these models operate as black box and only results in output class scores but do not provide clear insight which and how the parts of an image were focused for a decision. Which arises questions like:

How can we guarantee the model's predictions are interpretable and trustworthy?

How do we evaluate the stability of explanations across variations in input images?

This lack of transparency becomes problematic when models are deployed on edge devices such as phones, laptops or embedded systems and in sensitive environments where decisions may directly affect users.

B. Context

Edge scenarios impose additional constraints that models must be compact enough to run with limited memory and compute with high accuracy and robustness. MobileViT-S is a lightweight hybrid architecture that combines convolutional layers for local feature extraction with transformer blocks for global context while maintaining a small parameter footprint. So it is a promising backbone for edge deployment but in its original form MobileViT-S offers no built in mechanisms for interpretability evaluation and any kind of explanations which are human readable of how a prediction was made. As AI systems are increasingly used in various fields like Medical, Security, Automation it is very important to have a strong motivation to move beyond accuracy only reporting and to design systems in which reasoning patterns can be inspected, measured and communicated clearly. Describing brief explanations about the predictions made by the model to make it more trust worthy.

C. Motivation

However, in the field of explainable AI there are methods such as Grad-CAM that highlight regions of an image that contributed for predicting a class score which makes it possible to visualize where a model "looks" when making a decision. But this evaluation is based on only a single image at a time which has been used with a lack of proper explainability. We found that there is very limited work on quantifying whether the model can be trusted or not. In this project we focused on the specific problem of combining a compact MobileViT-S model with a pipeline that makes its decisions interpretable and provides a numerical score for the stability of its explanations under small image perturbations. The motivation is to get predictions with a numeric interpretability score which can show the trustworthiness of a prediction and a human understandable explanation which shows the exact understanding of model to provide a result by highlighting spatial features, region of focus and structure sensitivity of an image.

II. Related Work (similarities and differences)

A. Similar Work

Existing models of CNN and Vision Transformer backbones has several architectures such as MobileNet, ShuffleNet and

MobileViT. These models are designed for very high accuracy on prediction of any object with testing on various datasets. Where particularly the MobileViT has small convolutional blocks with transformer style self attention method to capture both local patterns and long dependencies at low parameter count which makes it best for using on edge devices. And predictions of those similar models have gradient based methods such as Grad-CAM, Lime, SHAP etc which are widely used to generate class specific heatmaps by combining feature activations from a layer with the gradients of the class logit with by using activation functions. Grad-CAM is a very good method for most of the CNNs and Vision Transformers and is also treated as a default tool for visualizing what regions influence a prediction. For image quality assessment to compare the activation maps under different conditions there are several proposed stability metrics that measure how much a prediction changes when the input is changed or when the model is retrained but all these takes huge space and are time consuming which are not suitable for edge device implementations.

B. *Differences and Contributions of Our Work*

The present work is some similar to the method of Grad-CAM and SSIM (Structural Similarity Index Metrics) stability analysis but differs in several ways that are specific and needed for an edge device object detection model.

First, instead of designing a new architecture we took a pretrained MobileViT-S model and fine-tune it on CIFAR-10 with some improvements by adding label smoothing and a warmup cosine learning rate schedule for very stable learning where the goal was to make a compact backbone that is realistic for deployment on edge devices like phones, laptops or embedded devices.

Second, we integrate Grad-CAM directly into this MobileViT-S model including the automatic selection of the final feature layer and consistent preprocessing which can be used in explanations to be generated with minimal additional overhead during inference.

Third, we used a similarity metrics SSIM (Structural Similarity Index Metrics) with Grad-CAM heatmaps to get a score named as Interpretability Stability Score (ISS) by computing SSIM between normalized Grad-CAM image (CAM_orig) and perturbed Grad-CAM image by adding rotations, brightness change and some gaussian noise (CAM_pert). This treats explanation stability as a first class quantitative metric that can be used alongside accuracy rather than just as an informal visual observation.

Fourth, along with the visual and metrics evaluation we enhanced explainability by building a rule based natural language explanation layer that analyzes the Grad-CAM map together with Sobel based filter for edge strength and color statistics in the attended region and then turns these into human readable descriptions stating the areas of images where the model focused and what kind of visual evidence it used for predicting an image. This mixture of a lightweight backbone like MobileVir-S, Grad-CAM an ISS stability metric with structured textual explanations within a single reproducible pipeline is not present in any prior work to our knowledge and directly targets the practical need for interpretable and stabile models in edge devices which are trustworthy and reliable.

III. Problem Definition and Solution Design

A. *Problem Definition*

Many deep learning models used for object detection have made computer vision extremely powerful with very high accuracy levels close to what humans can achieve. But even with this progress most of these models still behave like black boxes as they just gives us a label or a probability for a prediction and they do not show any reasons that why they made that decision or what parts of the image were focused while making prediction in natural human readable language. This becomes a real problem when we want to trust the model or understand how it will behave in new situations. This created questions that:

Can we see what the model is focusing on? Are its explanations reliable? And do those explanations stay consistent when the input image changes just a little?

These concerns matter even more in everyday real world. Imagine a small vision model running on someone phone to help a visually impaired user identify objects around them. If the model's attention jumps to the wrong place because of a tiny lighting change then that person might lose trust in the system. Or consider a home security camera that classifies objects by its edges if its reasoning shifts unpredictably when someone walks by or the environment changes a bit then it could misclassify important features leading to wrong predictions.

The core problem what this project addresses is how to build a compact, edge friendly explanation model that not only performs well but also provides explanations that are easy to interpret and stable under realistic changes in the input. In other words the goal is to make the model's reasoning visible, reliable and trustworthy without slowing it down or redesigning its architecture.

B. *Solution Design*

To fulfill these goals we have designed a complete explainability pipeline on top of a fine tuned MobileViT-S Vision Transformer Model where the model was first trained on CIFAR-10 dataset of 10 classes with label smoothing and a warmup cosine learning rate schedule to achieve stable and high accuracy performance and also keeping the architecture compact enough for edge deployment.

During inference the Grad-CAM is applied on the final feature layer of MobileViT-S to produce a class specific heatmaps that highlights the regions which were most responsible for a prediction. Along with that to measure explanation stability the model generates a perturbed version of the input image by applying small rotations, brightness adjustments and Gaussian noise. Which then is used to compute a second Grad-CAM map and compares it to the original using the Structural Similarity Index (SSIM). This SSIM value becomes the Interpretability Stability Score (ISS) by using both the heatmaps of Grad-CAM which reflects how consistently the model focuses on the same visual features even with some changes of image.

Finally, to make the explanations understandable to non-experts we included a natural language system that analyzes the Grad-CAM heatmaps together with Sobel filtered edge detection and dominant color statistics. This produces human readable explanations describing where the model actually focused and what types of visuals (like shapes, edges, colors) guided the decision of the model. The work flow of the model has been displayed below in Figure 1.

IV. Methodology and Implementation Details

A. *Working of MobileViT-S Model*

All training and inference in this project are built around a MobileViT-S model implemented in PyTorch using the timm library with CIFAR-10 as the dataset. The full pipeline starts in `train_c10.py` with the function `make_cifar10_loaders` which downloads CIFAR-10 and prepares two DataLoader objects where: one for training and one for validation. Each image is resized from its original resolution to 32×32 pixels to 224×224 pixels to match the MobileViT-S input size. The preprocessing applies three steps in sequence: `transforms.Resize(224)`, `transforms.ToTensor()` and `transforms.Normalize(mean, std)` where the normalization is performed per channel for as

$$x^{(c)}_{\text{norm}} = \frac{x^{(c)} - \mu_c}{\sigma_c}$$

with CIFAR-10 statistics $\mu = (0.4914, 0.4822, 0.4465)$ and $\sigma = (0.2470, 0.2435, 0.2616)$. The output of this stage is a batch tensor of shape $1 \times 3 \times 224 \times 224$ which is ready to be fed into the model.

The model backbone is called itself is constructed via:

```
model = timm.create_model(
    "mobilevit_s",
    pretrained=True,
    num_classes=num_classes
    ).to(device)
```

and this call loads ImageNet pretrained MobileViT-S weights and replaces the classification head so that the final linear layer outputs logits for the ten CIFAR-10 classes.

Internally, MobileViT-S performs all feature extraction and attention. In our pipeline we treat it as a compact pretrained backbone whose interface is simply as $z = f_\theta(x)$ where x is the normalized image tensor and θ are the trainable parameters loaded from `best.pth`.

Training uses supervised learning with cross entropy loss and label smoothing. Conceptually, for a K-class problem with $K = 10$, if p_i is the softmax probability of class i and y is the true class index then the smoothed target distribution is

$$q_i = \begin{cases} 1 - \varepsilon & \text{if } i = y, \\ \varepsilon/(K-1) & \text{otherwise,} \end{cases}$$

and the loss becomes as:

$$\mathcal{L}_{LS} = -\sum_{i=1}^{K} q_i \log p_i$$

Here ε is the smoothing factor. This reduces overconfidence in the logits and makes the model less sensitive to noisy labels.

Optimization is handled by AdamW optimizer for two parameter groups which are the backbone and the head. In `train_c10.py` the head (classification layer) is given a larger learning rate (BASE_LR) while the backbone starts at BASE_LR $\times$ 0.1 which shows that the backbone is already pretrained and should be updated. The raw learning rates are then changed over time by a custom warmup cosine scheduler to prevent the model from exploding. Where, let s be the current global step, W the number of warmup steps and T the total number of steps. At each call to `sched.step()`, the scheduler computes a scale factor

$$\text{scale}(s) = \begin{cases} \frac{s}{W} & \text{if } s < W, \\ \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{s-W}{T-W}\right)\right) & \text{otherwise,} \end{cases}$$

and for each parameter group with base learning rate η_{base} sets

$$\eta(s) = \eta_{\text{base}} \cdot \text{scale}(s).$$

This means the effective learning rate starts near zero then rises linearly during warmup and then decays smoothly according to a cosine curve.

Each training step moved to the device and the optimizer gradients are reset and then the model is called as `logits = model(x)`. The function `topk_accuracy` computes top-1 and top-5 accuracies by checking whether the ground truth label appears in the top-k logits for each sample as:

$$\text{Top-k Acc} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\{y_n \in \text{TopK}(z_n)\}$$

where N is the batch size and z_n is the logit vector for sample n. Training loss, training accuracy and the current learning rate are logged to TensorBoard and to `metrics.csv` under `MVIT_C10/artifacts/` while model checkpoints `best.pth` and `last.pth` are saved in `MVIT_C10/checkpoints/` with associated metadata as (class_names, configuration, best validation accuracy).

The validation reports validation loss, top-1 and top-5 accuracy and this is the full training workflow where the pipeline performs computing of logits z and class probabilities via the softmax:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

and then selecting the most probable class with `logits.argmax(dim=1)`.

B. *Working of the Explanation Pipeline*

Once the MobileViT-S model is trained the second part of the methodology focuses on explaining its predictions. This explanation module is implemented in `iss_gradcam.py` and runs on top of the trained model loaded from `best.pth`

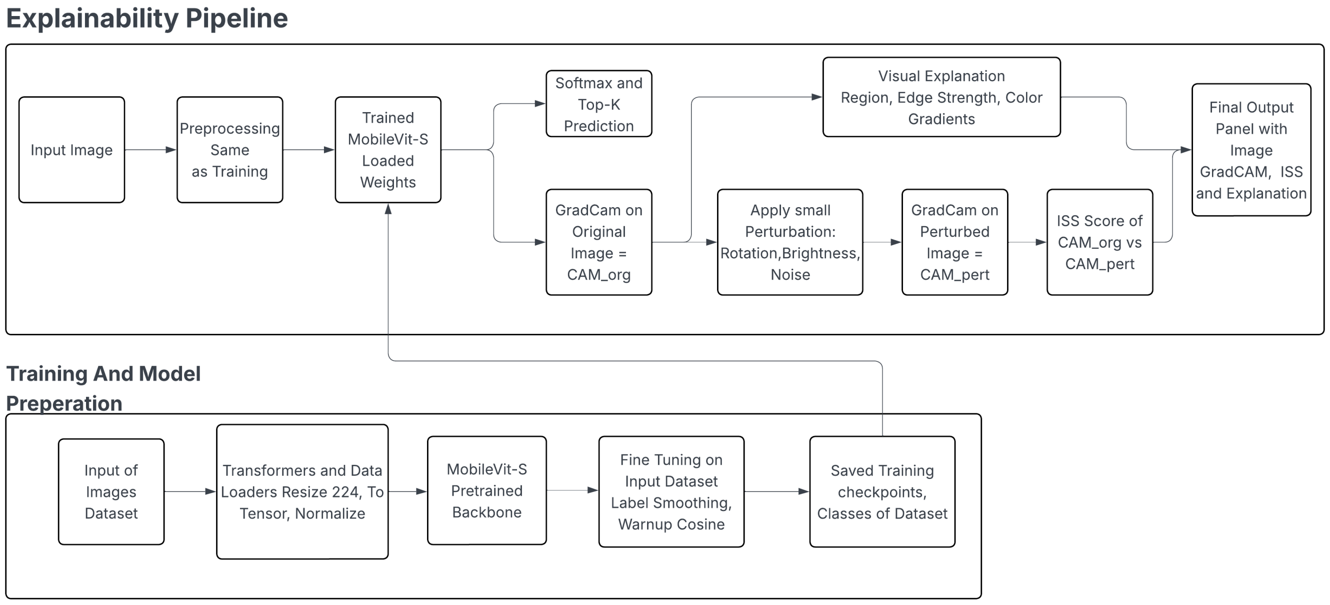

Explainability Pipeline
Input Image
Preprocessing Same as Training
Trained MobileVit-S Loaded Weights
Softmax and Top-K Prediction
GradCam on Original Image = CAM_org
Visual Explanation Region, Edge Strength, Color Gradients
Apply small Perturbation: Rotation,Brightness, Noise
GradCam on Perturbed Image = CAM_pert
ISS Score of CAM_org vs CAM_pert
Final Output Panel with Image GradCAM, ISS and Explanation
Training And Model Preperation
Input of Images Dataset
Transformers and Data Loaders Resize 224, To Tensor, Normalize
MobileVit-S Pretrained Backbone
Fine Tuning on Input Dataset Label Smoothing, Warnup Cosine
Saved Training checkpoints, Classes of Dataset

Fig. 1. Overview of the proposed explainability pipeline added with the backbone as MobileVit-S model to give explanations and ISS score to check interpretability of the prediction.

and the class names from `classes.txt`. It has three main components: Grad-CAM heatmaps, an Interpretability Stability Score (ISS) derived from SSIM and a template based natural language explanation generator.

1) Grad-CAM in Our Model: Initially the GradCAM receives the trained model and a target layer which is automatically chosen by scanning through the modules to find the last `nn.Conv2d` layer via a helper that behaves like:

```
target_layer = find_last_conv(model)
```

On this layer GradCAM registers a forward hook and a full backward hook where the forward hook saves the feature map activations across channels A_k and the backward hook saves the gradients $\frac{\partial y_c}{\partial A_{k,ij}}$ of the predicted class score y_c with respect to each spatial location (i, j) in each channel. When GradCAM is called with the logits from a forward pass and backward pass is proceeds as follows:

The raw Grad-CAM heatmap is then computed as a weighted sum of the feature maps:

$$\alpha_k = \frac{1}{H \cdot W} \sum_{i,j} \frac{\partial y_c}{\partial A_{k,ij}}, \qquad \text{CAM} = \text{ReLU}\left(\sum_k \alpha_k A_k\right)$$

This produces a single channel activation map where high values correspond to spatial positions that contributed strongly to the target class. The heatmap is then min–max normalized to the range $[0, 1]$,

$$\text{CAM}_{\text{norm}} = \frac{\text{CAM} - \min(\text{CAM})}{\max(\text{CAM}) - \min(\text{CAM}) + 10^{-6}},$$

and upsampled to the input resolution 224×224 before overlaying it on the original image which later shows representative overlays for samples such as cat, horse, automobile and truck,

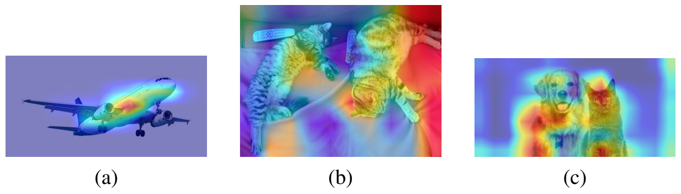


Fig. 2. Example Grad-CAM overlays for different CIFAR-10 images.

illustrating what parts of the image MobileViT-S relies on for each prediction.

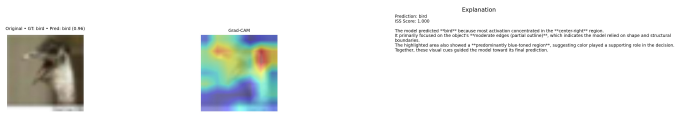
2) Interpretability Stability Score (ISS): The Interpretability Stability Score is designed to measure how robust a Grad-CAM explanation is under small, realistic perturbations of the input. This is implemented through two key functions: `apply_small_perturbation` and `compute_iss`.

Given a normalized input tensor of shape $(1 \times 3 \times 224 \times 224)$ and `apply_small_perturbation` first converts it back to a PIL image:

```
pil = transforms.ToPILImage()(img_tensor)
```

It then applies three types of mild perturbations which are a random rotation, a brightness shift and additive Gaussian noise. Concretely, the function samples an angle θ uniformly in $[-3°, 3°]$ and rotates the image, then it adjusts brightness by a factor β drawn from $[0.9, 1.1]$ and the perturbed image is then converted back to a tensor x_{pert} in $[0, 1]$ and pixel wise Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with standard deviation σ is added.

Finally, each channel c of this perturbed tensor is normalized using the same CIFAR-10 mean and standard deviation as in training which gives a perturbed normalized tensor $\tilde{x}$ that can



Prediction: bird
ISS Score: 1.000

The model predicted **bird** because most activation concentrated in the **center-right** region.
It primarily focused on the object's **moderate edges (partial outline)**, which indicates the model relied on shape and structural boundaries.
The highlighted area also showed a **predominantly blue-toned region**, suggesting color played a supporting role in the decision.
Together, these visual cues guided the model toward its final prediction.

Fig. 3. Paired Original and Perturbed Grad-CAM Maps with ISS Values.

be passed through the model in exactly the same way as the original input.

Thus, for a given image Grad-CAM is computed on both the original input and the perturbed input producing two heatmaps: CAM_{orig} and CAM_{pert} respectively. Inside `compute_iss` both are first converted to NumPy arrays of type `float32` and individually normalized to using min–max normalization.

These normalized maps denoted as x and y, are then compared using the Structural Similarity Index (SSIM) from `skimage.metrics.structural_similarity` with `data_range=1.0`. Where conceptually the SSIM between two images x and y is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where μ_x and μ_y are the mean intensities of x and y, σ_x^2 and σ_y^2 are their variances. σ_{xy} is the covariance and C_1, C_2 are small constants that stabilize the division. In our context the SSIM is applied locally and aggregated by the library implementation, giving a single scalar in $[0, 1]$ which stays between 0 and 1 for normalized Grad-CAM maps. We define the Interpretability Stability Score as

$$\text{ISS} = \text{SSIM}(\text{CAM}_{\text{orig}}^{\text{norm}}, \text{CAM}_{\text{pert}}^{\text{norm}}).$$

An ISS close to 1 means that the predicted structure is nearly unchanged under the perturbation while a small ISS indicates that the explanation is unstable even if the predicted class might remain the same. The ISS is computed per image and printed in the explanation panel for example an ISS of 0.014 or 0.003 indicates strong sensitivity of the heatmap to small rotations and brightness shifts, whereas higher values near to 1 reflect more stable attention patterns. Paired original and perturbed Grad-CAM maps along with their ISS values are later shown in Figure 3.

3) (C) Natural Language Explanations: The final component of the explanation module is a small rule based system that converts the Grad-CAM heatmap into a short natural language explanation. The main functions involved are `detect_cam_region`, `detect_edge_strength`, `detect_color_feature` and the explanation template that combines their outputs.

The function `detect_cam_region(cam)` takes the single channel Grad-CAM map and first converts it to a NumPy array. It finds the location of the maximum activation using: `y, x = np.unravel_index(cam.argmax(), cam.shape)`.

Given with the height H and width W of the heatmap then it assigns the row index to one of three bands: "upper", "center", or "lower" and the column index to "left", "middle", or "right". The final region descriptor is a string such as "upper-left", "center-middle", or "lower-right". This provides a good description of where in the image the model concentrated.

The function `detect_edge_strength(img, cam)` measures how much the explanation focuses on structural details. It first converts the input image tensor back to an 8-bit RGB NumPy array then converts it to grayscale and then uses Sobel filters in the horizontal and vertical directions:

```
sobelx = cv2.Sobel(gray, cv2.CV_32F, 1, 0)

sobely = cv2.Sobel(gray, cv2.CV_32F, 0, 1)

  edges = np.abs(sobelx) + np.abs(sobely)
```

The Grad-CAM map is resized to match the edge map and a scalar edge score is computed as the mean of `edges * cam_resized` which creates good edges in the regions where Grad-CAM is strong. This score is then divided into three qualitative categories:

if the score is greater than 80, the function returns "strong edges (clear object boundaries)",

if it is greater than 40, it returns "moderate edges (partial outline)",

otherwise it returns "weak edges (shape not very important)".

This gives a human-readable description of whether the model is paying attention to contours and shapes.

The function `detect_color_feature(img, cam)` focuses on color. It resizes the Grad-CAM map to the image size and thresholds it at its own mean value to obtain a binary mask of "high-activation" pixels and then extracts the corresponding RGB pixels from the image. Depending on which component is largest it returns "predominantly red-toned region", "predominantly green-toned region", "predominantly blue-toned region", or "mixed color region" if no single channel dominates.

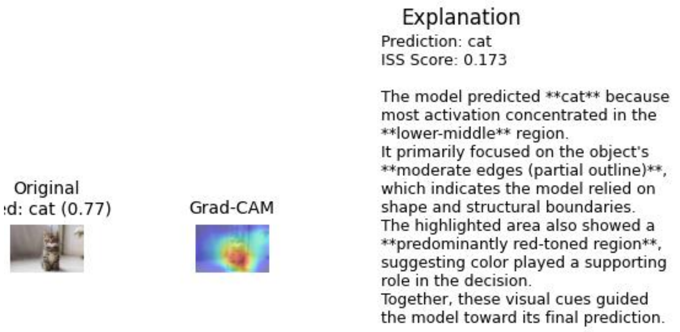
These three descriptors of spatial region, edge pattern and color emphasis are then combined with the predicted class name into a template string such as:

In the output the script also annotates the ISS value producing an explanation that states the predicted class, the ISS score and a structured description of where the model looked, how strong the edges were and which colors were important.

For example, for a correctly classified cat image where Grad-CAM is concentrated in the center-middle region, edge strength is moderate and color analysis yields a mixed region, the explanation might say that the model focused on the central region, relied on moderate edges around the head and body, and used a mix of colors such as fur and background.

Together, these two parts of the MobileViT-S training and the explanation pipeline we made a complete system that starts from raw CIFAR-10 images and ends with class predictions, Grad-CAM heatmaps, an ISS stability score and natural language explanations.

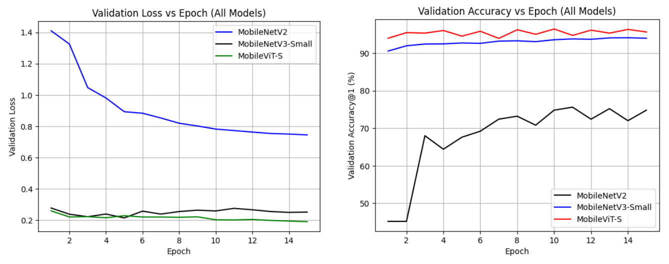
MobileViT-S is designed for low resource environments and our implementation was focused on this property throughout training and inference. The final trained checkpoint (`best.pth`) is compact in storage size which helps in making it feasible to deploy directly on an edge device or remotely



Explanation
Prediction: cat
ISS Score: 0.173

The model predicted **cat** because most activation concentrated in the **lower-middle** region.
It primarily focused on the object's **moderate edges (partial outline)**, which indicates the model relied on shape and structural boundaries.
The highlighted area also showed a **predominantly red-toned region**, suggesting color played a supporting role in the decision.
Together, these visual cues guided the model toward its final prediction.

Fig. 4. Complete Explanation Panel: Image, Grad-CAM, ISS and Text) shows such a combined output for a single image, illustrating how the numerical and visual signals are translated into human-readable language.



(a) Lowest validation loss indicating stronger feature representation (b) MVIT achieves highest and most stable accuracy

Fig. 5. Comparison of MobileNetV2, MobileNetV3-Small, and MobileViT-S over 15 epochs.

TABLE I
PARAMETER, MEMORY AND FLOPS COMPARISON FOR MOBILENETV2, MOBILENETV3-SMALL AND MOBILEVIT-S.

Model	Parameters (Billions)	Memory (MB)	GFLOPs
MobileNetV2	3.4	5.8	0.3
MobileNetV3-Small	4.5	8.5	0.06
MobileViT-S	5.6	18.86	1.6

on a cloud endpoint that edge devices can use as a query. Because the model's forward pass, Grad-CAM generation, ISS computation and natural-language explanation do not require GPU-specific operations the entire inference pipeline can run efficiently on CPU only environments.

In practice, the model weights can be hosted on a cloud server and an edge device only needs to transmit an input image and receive back the predicted class, Grad-CAM heatmap, ISS score and explanation text. This architecture helps in not worrying about storage and hosting the model locally while still enabling real time interpretability. To validate this, we have deployed the MobileViT-S model on a HuggingFace Gradio interface confirming that inference and explanation generation can be served directly from the cloud with negligible local storage and minimal latency. This demonstrates that the explainability pipeline is compatible with real edge device integration where storage, compute and memory budgets are limited.

V. EXPERIMENTAL EVALUATION: SETTINGS AND RESULTS

A. Experimental Setup

All experiments in this project are based on image classification using CIFAR-10 Dataset but we have changed the architecture and explainability pipeline evert time. We started with a MobileNetV2 baseline on a smaller subset of classes then moved to MobileNetV3-Small and finally decided to stay on with MobileViT-S as the main backbone. In all stages the workflow followed the same pattern: define the dataset and transforms, train the model with a fixed set of hyperparameters, log metrics to CSV and TensorBoard and then generate Grad-CAM–based explanations on selected images.

For the final MobileViT-S model the training is performed on CIFAR-10 images which contains 50,000 training images and 10,000 test images of size 32×32 with 10 different classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck).

Functionalities are: `BATCH_SIZE = 128`, `EPOCHS = 15`, `BASE_LR = 1e-3`, `LABEL_SMOOTH = 0.1`, `WARMUP_E = 5`, `FREEZE_HEADONLY_E = 3`. Training is carried out on a CPU. GPU can be used to speedup the training.

B. Baseline Models: MobileNetV2 and MobileNetV3

We started the project with simple setup using MobileNetV2 on a reduced set of classes. In this first model we used a smaller subset of the dataset (5 visually distinct classes) to quickly test whether a lightweight backbone plus Grad-CAM could give reasonable explanations. The training loop, logging and Grad-CAM visualization were structurally similar where images were resized and normalized MobileNetV2 was loaded with pretrained weights the classification head was adapted to the number of classes and crossentropy loss was used for optimization.

From this baseline, we moved to MobileNetV3-Small to better match the "edge-device" goal. MobileNetV3-Small is designed to be even more efficient than V2 with fewer parameters and lower FLOPs. In our workflow the MobileNetV3-Small was trained on more classes with similar preprocessing and Grad-CAM applied on its last convolutional layer. For both MobileNetV2 and MobileNetV3, we recorded training and validation accuracy curves, training and validation loss curves and measured CPU inference latency and parameter counts.

The key observations from this phase can be summarized in three types of plots/tables:

Accuracy and Loss Curves: For both MobileNetV2 and MobileNetV3 the training curves showed stable convergence but there was a visible gap between them and the final MobileViT-S curves. MobileNetV2 and MobileNetV3 reached good accuracy on the subset and on CIFAR-10, but did not match the performance and stability of MobileViT-S.

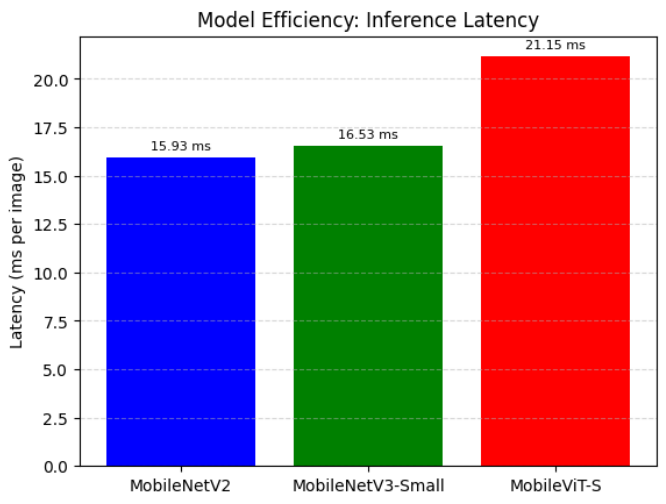
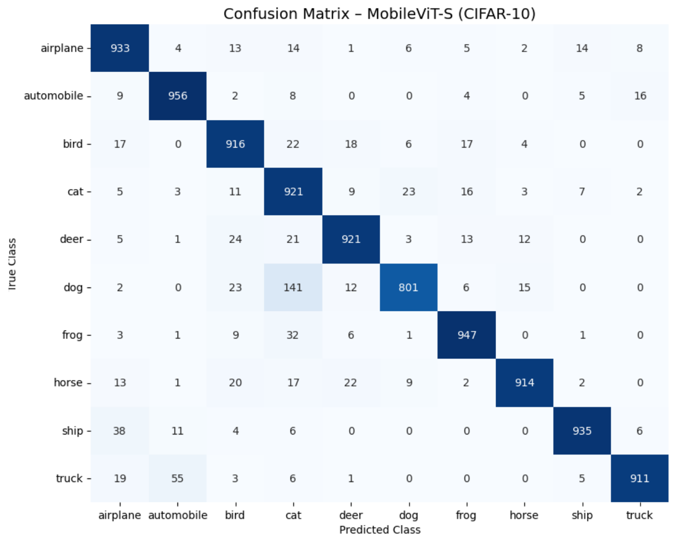


Fig. 6. Inference Latency Comparison across MobileNetV2, MobileNetV3-Small, and MobileViT-S. MobileNetV3 achieves the lowest latency, while MobileViT-S offers a balanced trade-off between latency and interpretability.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	933	4	13	14	1	6	5	2	14	8
automobile	9	956	2	8	0	0	4	0	5	16
bird	17	0	916	22	18	6	17	4	0	0
cat	5	3	11	921	9	23	16	3	7	2
deer	5	1	24	21	921	3	13	12	0	0
dog	2	0	23	141	12	801	6	15	0	0
frog	3	1	9	32	6	1	947	0	1	0
horse	13	1	20	17	22	9	2	914	2	0
ship	38	11	4	6	0	0	0	0	935	6
truck	19	55	3	6	1	0	0	0	5	911

Fig. 7. Confusion Matrix of MobileViT-S: accurately classifies most classes, but shows some confusions in similar categories like Dog-Cat.

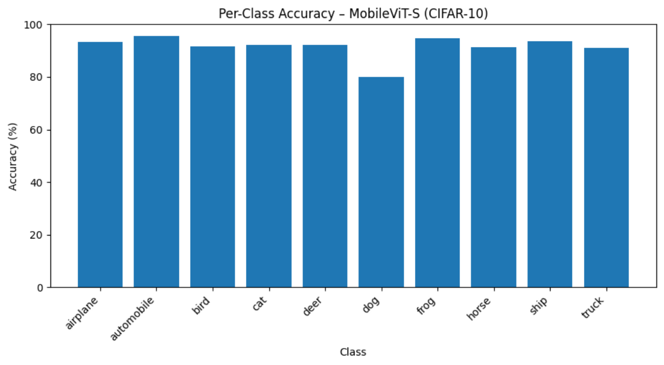


Fig. 8. MobileViT-S Per Class Accuracy. It maintains strong performance across all ten CIFAR-10 classes.

Parameter and FLOPs Table: (Table I) – A small model-comparison table lists number of parameters and approximate GFLOPs for MobileNetV2, MobileNetV3-Small and MobileViT-S. MobileNetV2 and V3 are lighter in GFLOPs than standard large models, but MobileViT-S offers a better trade-off between capacity and explainability while still remaining compact.

Inference Latency: (Figure 6) – A bar chart or line plot compares CPU inference time and Grad-CAM generation time for MobileNetV2, MobileNetV3 and MobileViT-S. While all three models are within a reasonable range for edge-like constraints, MobileNetV3 often has the lowest latency, and MobileViT-S trades slightly higher latency for better interpretability and stability.

This baseline phase was important: it confirmed that simple CNN-style models can be efficient, but it also showed that we needed a backbone that is more naturally aligned with transformer-style attention while still being edge-friendly. That motivated the shift to MobileViT-S.

C. Comparative Analysis Across Backbones

Putting all models together, the experiments showed that:

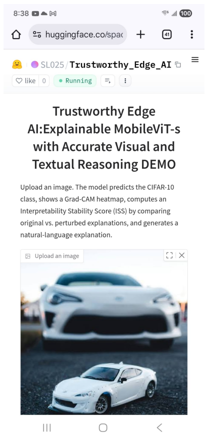
MobileNetV2 (5-class) was a useful starting point to set up the training, logging, Grad-CAM and basic visualization pipeline. It confirmed that a compact CNN can give meaningful heatmaps but did not fully explains like the transformers.

MobileNetV3-Small actually improved efficiency and retained good accuracy. However, when we compared Grad-CAM behavior and the qualitative stability of explanations across perturbations, MobileNetV3's attention sometimes appeared noisier and less structurally aligned than what we later observed from MobileViT-S.

MobileViT-S delivered the best overall trade-off and strong CIFAR-10 accuracy, semantically meaningful Grad-CAM heatmaps and a stable well defined results for computing ISS and generating human readable explanations.

In addition to the standard training results, we also wanted to see whether our model could realistically run in an edge device setting. To test this, we deployed the trained MobileViT-S checkpoint on a HuggingFace Gradio interface and used it as a cloud based inference system. Instead of storing the model on the device and an edge client simply uploads an image and the server returns the prediction, Grad-CAM heatmap, ISS score and natural language explanation all in real time. This experiment showed that the entire pipeline, including Grad-CAM and ISS calculations runs smoothly even without a GPU which is often a major concern for Vision Transformers.

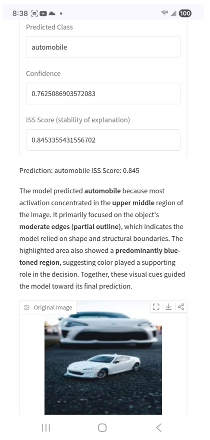
The responses were fast and consistent and the storage footprint on the device remained essentially zero. This deployment proves that our explainability system is not just a controlled lab prototype but can genuinely operate in practical edge based scenarios where low memory and limited compute resources are real constraints.



Trustworthy Edge AI:Explainable MobileViT-s with Accurate Visual and Textual Reasoning DEMO

Upload an image. The model predicts the CIFAR-10 class, shows a Grad-CAM heatmap, computes an Interpretability Stability Score (ISS) by comparing original vs. perturbed explanations, and generates a natural-language explanation.

(a)



Predicted Class: automobile

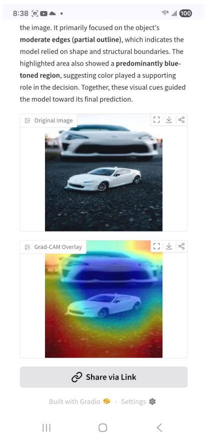
Confidence: 0.7625086903572083

ISS Score (stability of explanation): 0.8453355431556702

Prediction: automobile ISS Score: 0.845

The model predicted **automobile** because most activation concentrated in the **upper middle** region of the image. It primarily focused on the object's **moderate edges (partial outline)**, which indicates the model relied on shape and structural boundaries. The highlighted area also showed a **predominantly blue-toned region**, suggesting color played a supporting role in the decision. Together, these visual cues guided the model toward its final prediction.

(b)



the image. It primarily focused on the object's **moderate edges (partial outline)**, which indicates the model relied on shape and structural boundaries. The highlighted area also showed a **predominantly blue-toned region**, suggesting color played a supporting role in the decision. Together, these visual cues guided the model toward its final prediction.

(c)

Fig. 9. Hugging Face (Gradio) interface running our MobileViT-S explainability pipeline on a mobile edge device, showing input upload, prediction with ISS, and Grad-CAM explanation.

VI. Discussion, Limitations and Ethical Considerations

A. Discussion

The experimental results show that MobileViT-S is a strong compact backbone that achieves high classification accuracy with a lightweight interpretability pipeline across a wide range of CIFAR-10 samples and Grad-CAM heatmaps indicate that the model focuses on meaningful regions like the body of an animal or of a vehicle which suggests that MobileViT-S is capturing structural and shape based features rather than just depending purely on background of images. However, the Interpretability Stability Score (ISS) makes it clear that the model's internal reasoning is stable as many images show ISS values as low as 0.003 or 0.02 which means that even very small perturbations like a $\pm 3^\circ$ rotation or a little brightness shift can cause the attention map to shift significantly. But, these fluctuations occur even when the model's top-1 prediction remains correct which highlights an important thing that: the model can be right but for reasons that are not consistent. This gap between accuracy and interpretability is an important outcome of the study and demonstrates why relying only on accuracy is insufficient when designing AI systems for edge deployment where real world environmental variations are inevitable.

And, talking about its usage on Edge devices then the weights or the checkpoints of the model trained which have knowledge it received while training is not much very high in storage, so it can be used or called from could to just use it and then convert it to probabilities and then compare it with the image uploaded by user, Which can give real time explanations on edge devices without using much storage on devices.

The natural language explanations generated by the system also provide valuable insights as they generally align with the heatmaps and correctly describe the activation region, edge structure and color features. So, these explanations can be used as a statement to be used for an AI to state trustworthiness on the model itself for a human. These shows that the combination of Grad-CAM, ISS and structured explanations does offers a meaningful insights into the model's reasoning and also reveals areas of more stability, better understanding and stronger explainability which is a very important expect to built trust on an AI system.

B. Limitations

Although, with the strengths of this pipeline there are several limitations which we came through the model from a careful analysis of models behavior, implementation and outputs are.

First, the ISS scores shows that MobileViT-S is sensitive to even mild perturbations which indicates that the spatial structure of its attention is not strongly anchored and this shows underfitting. The low ISS values also tells an important limitation of Grad-CAM itself because it only relies on gradients of a chosen feature layer.

Another limitation is from the perturbation design which includes only rotation, brightness change and Gaussian noise. While in real world there can be many changes to an image like occlusion, viewpoint changes, motion blur, environmental shadows or partial object visibility which are not captured here for now.

In addition to that, CIFAR-10 dataset is very small small (32×32 images) and does not fully represent the diversity or complexity of real world datasets. MobileViT-S appears powerful may be because the CIFAR-10 is relatively simple but increasing the resolution or moving to very huge datasets would expose the further instability.

The natural-language component also has some limitations as it depends on fixed templates and simple heuristics (like Sobel edges, dominant color, peak activation region). While these rules provide clear interpretability they cannot capture more patterns of the images which might the model uses leading to explanations that sometimes feel very oversimplified or repetitive.

C. Ethical Considerations

From an ethical perspective the goals of this work was to focus on principles of transparency, explainability and user trust mainly in edge device environments where model behavior is difficult to check after deployment. However, there are several ethical risks which must be acknowledged.

First, there might be unstable explanations like very low ISS scores can lead users to misinterpret the confidence or reliability of a system. A model that is "correct but unstable" may still be giving false assumptions of reliability which creates a mismatch between system behavior and user expectations.

Second, interpretability methods such as Grad-CAM can be misused to justify decisions because heatmaps give an appearance of transparency where developers or organizations may present them as evidence of fair or robust decision making even when stability metrics (ISS) reveal inconsistency or unreliability.

Third, the template based natural language explanations are helpful but can be incorrect in semantic reasoning where they may be wrong in describing edges or colors may give users the impression that the model understands high level concepts rather than low level visual correlations which can make user to over trust on the model.

Additionally, CIFAR-10 is not a data of sensitive domains such as healthcare, surveillance or public safety so using this kind of model in those sensitive areas can give incorrect outputs and validated and high training should be needed for that. And if this system were adapted for those sensitive data then personal data captured by edge cameras can arise some ethical concerns such as privacy, fairness, accountability and misclassification impact may become critical.

References

[1] S. Mehta and M. Rastegari, "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer," *arXiv:2110.02178*, 2021. Available at: https://arxiv.org/abs/2110.02178.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018. Available at: https://arxiv.org/abs/1801.04381.

[3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, *et al.*, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2019. Available at: https://arxiv.org/abs/1905.02244.

[4] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017. Available at: https://arxiv.org/abs/1610.02391.

[5] J. Nilsson and T. Akenine-Möller, "Understanding SSIM," *arXiv:2006.13846*, 2020. Available at: https://arxiv.org/abs/2006.13846.

Appendix A
Reproducibility Checklist

This project can be fully reproduced using the training, inference, Grad-CAM and ISS scripts. All experiments use Python 3.10 with the exact library versions listed in `requirements.txt`. The dataset (CIFAR-10) is downloaded automatically and no external data is required.

A. Environment Setup

Install dependencies locally:

```
pip install -r requirements.txt
```

Or use Docker:

```
docker build -t mvit_explain .
and
docker run -v $(pwd)/src:/app/src -v
$(pwd)/data:/app/data mvit_explain
```

B. Training the MobileViT-S Model

Run:

```
python train_c10.py
```

Training artifacts will appear under:

```
src/MVIT_C10/artifacts/
    best.pth
    last.pth
    metrics.csv
    classes.txt
    config.json
```

To change the learning rate, modify the value inside `train_c10.py`:

```
"BASE_LR": 0.001
```

C. Reproducing Figures

Training and validation plots, Grad-CAM panels, and ISS examples can be regenerated by running the experiment scripts in the repository:

```
python experiments/plot_metrics.py
python experiments/gradcam_examples.py
python experiments/iss_examples.py
```

Generating Grad-CAM + ISS

For a single image:

```
python iss_gradcam_oneimage.py
```

Docker:

```
docker run -v $(pwd)/src:/app/src \
           -v $(pwd)/data:/app/data \
           mvit_explain python
           iss_gradcam_oneimage.py
```

For the full CIFAR-10 test set:

```
python iss_gradcam.py
```

Docker:

```
docker run -v $(pwd)/src:/app/src \
           -v $(pwd)/data:/app/data \
           mvit_explain python
           iss_gradcam.py
```

For the full CIFAR-10 test set:

```
python iss_gradcam.py
```

Docker:

```
docker run -v $(pwd)/src:/app/src -v
           $(pwd)/data:/app/data \
           mvit_explain python
           iss_gradcam.py
```

Both scripts load `best.pth`, apply the same preprocessing used during training, compute Grad-CAM maps, generate the perturbed heatmap, and calculate the ISS score using SSIM.

D. Cloud / Edge Deployment

To reproduce the online version (Gradio):

```
python app.py
```

Appendix B
Team Contribution Statement

This project was completed through collaboration of both two team members Shubham and Smit and both contributed in different components that made this final system to work as a complete, reproducible and interpretable MobileViT-S pipeline.

Shubham focused mainly on all models development and experimentation tasks including preparing the CIFAR-10 dataset, training and testing each of the backbone models (which were MobileNetV2, MobileNetV3 and MobileViT-S) with tuning their relevant hyperparameters and generating the full set of performance plots such as accuracy curves, loss curves, inference comparisons, parameter tables and training metrics. He also done the cloud deployment and testing and implementing the HuggingFace Gradio interface and validating that the models Grad-CAM and ISS pipeline work correctly as an edge device accessible cloud service.

Smit was responsible for the interpretability and explanation components of the project. This includes implementing and refining the full Grad-CAM pipeline for all models with designing and selecting the perturbation based method used to compute ISS scores, applying SSIM to compare original and perturbed heatmaps and extending the system with the natural language explanation module using Sobel filters and also built the Dockerization workflow to make the entire project reproducible in isolated environments and prepared all explainability visualizations used in the report.

Both team members contributed equally on to GitHub repository with full maintenance, reproducibility documentation, debugging and the final report. The conceptual design of the pipeline, weekly Scrum discussions, analysis of results and writing revisions were also done with contribution of both together throughout the project.

Individual Reflection – Shubham Limbachiya

In this project my main responsibilities was of building models, training it and evaluating all the backbone models. I personally implemented the full CIFAR-10 training pipeline for MobileNetV2, MobileNetV3-Small and the MobileViT-S. This included setting up the dataset loaders, preprocessing of image for transformations, adding the label smoothing, configuring the warmup cosine learning rate scheduler and then monitoring the optimization process. I also generated all of the training and validation plots and accuracy curves, loss curves, confusion matrices, per class accuracy charts and also computed the FLOPs and parameter comparisons with latency of each model to compare their usability. Later, I focused on deploying the final MobileViT-S explainability system on a HuggingFace Gradio interface to test whether the model pipeline could realistically run in an edge device or not.

Skills Learned

This project helped in several ways as I was not at all aware about Vision Transformer so technically I gained confidence in understanding the working and training of Vision Transformers. Also got to know that why the certain ways like warmup schedules or label smoothing stabilizes the learning. I learned how to compare models not just by accuracy but by efficiency, latency and interpretability in our case which is something I had not aware about before. Deploying on HuggingFace gave me a confidence that cloud inference and CPU optimized pipelines can be easily used on cloud for any edge devices. Beyond the coding I also leadned many things in scientific writing, documenting experiments and organizing results so they can be reproduced by others by using Latex.

Challenges and How I Overcame Them

A big challenge was training MobileViT-S entirely on CPU. It required so much time and careful tuning as it makes mistakes so much and can waste many of hours of work. Continuously checking that training remained stable and reproducible took multiple iterations. I also struggled initially with making the inference pipeline responsive enough for deployment on edge device before deployingon HuggingFace. Through debugging, simplifying parts of the code and coordinating closely with Smit on model loading logic, the explainability which was the main part of this project took many efforts and was not at all easy to do but these issues were resolved one step at a time and we made through it.

Areas I Would Improve With More Time

If I had more time I would push the project toward larger datasets like CIFAR-100 or Tiny-ImageNet to test how stable MobileViT-S remains when images become more complex. I would also explore quantization or ONNX conversion so the model could run even faster on real edge hardware. Our professor suggested us many good points like focusing on medicalImages which is very much important in todays world which can be my next goal to be focused on and I will try my best to upgrade this whole pipeline for a variety of datasets. I can also try exploring the ISS metric with more realistic perturbations of motion blur would and other variations of an image and also make the evaluation stronger with various mode models.

Reflection on Team Collaboration

What I liked the most in our collaboration is how we distributed the two main tasks the training and explainability to each other so that no one have more burden. I handled the modeling and experiments, while Smit focused on explanations and the interpretability pipeline. This division made our workflow very efficient and allowed each of us to learn without blocking each other. I especially liked that whenever one of us faced an issue whether it was model instability or Grad-CAM problems we always helped and debugged together instead of working and solving alone. The thing which i liked the least is difficulty in synchronizing files across versions but we even improved that in the project with time. Overall, the collaboration was very smooth, balanced and our support to each other and guidance from professor made it worked..

Individual Reflection – Smit Patel

My main contribution to this project was building the entire interpretability and explanation system on top of the trained MobileViT-S backbone. I implemented Grad-CAM from scratch for all three models by using various references and figured out how to hook into MobileViT-S's final convolutional layer and designed the perturbation pipeline that rotates, brightens and adds noise to inputs. I then implemented the computation of the Interpretability Stability Score (ISS) using SSIM and built the natural language explanation module that detects the region of attention, edge strength and color features. I also created the Docker environment so the project could be reproduced easily on any machine. Many of the figures used in the report Grad-CAM overlays, ISS comparisons, explanation panels were also made by me.

Skills Learned

This project taught me a lot about how interpretability algorithms actually function inside neural networks. I learned how PyTorch hooks actually work in behind and how gradients flow inside hybrid CNN to Transformer architectures and how stability metrics like SSIM behave when explanations are noisy or inconsistent. Building the explanation generator helped me practice structured reasoning and rule based NLP and Dockerization gave me hands on experience with reproducible machine learning environments. The project helped me in learning both my engineering skills and new interpretability methods.

Challenges and How I Overcame Them

One of the hardest parts was getting Grad-CAM to work correctly with MobileViT-S. Unlike simpler CNNs the architecture is not straightforward so identifying the right layer and making sure gradients were captured properly required careful debugging. Designing the ISS system was also challenging way too much because perturbation made the scores meaningless while too little didn't reveal any kind of instability. After several trials I reached on a balanced set of perturbations that revealed meaningful differences. Coordinating with Shubham's training pipeline also presented some challenges when model checkpoints werec hanged or he trained the models again but regular communication helped us stay together.

Areas I would Improve With More Time

I would have given more time, I would expand the interpretability section to compare multiple methods such as Attention Rollout or Integrated Gradients to see whether some techniques provide more stable or insightful explanations. I would also improve the natural language system to not only templates but create more flexible and context aware explanations. Finally, I would explore a multi level perturbation ISS score that can check image for various types of angles and situations.

Reflection on Team Collaboration

The best part of our collaboration was how our contributions for various tasks was divided priorly. This clear separation helped ua in a way that we could both work independently but still move forward together. I also appreciate that whenever one of us got stuck the other helped with any problem faced. If I had to name something I liked the least, it would probably be just the occasional delay caused by large model files or long training times but as that is a technical limitation than a collaboration issue which we should have figured out earlier. But overall everything was fine, whole project was filled with various problems but was very worth of it.