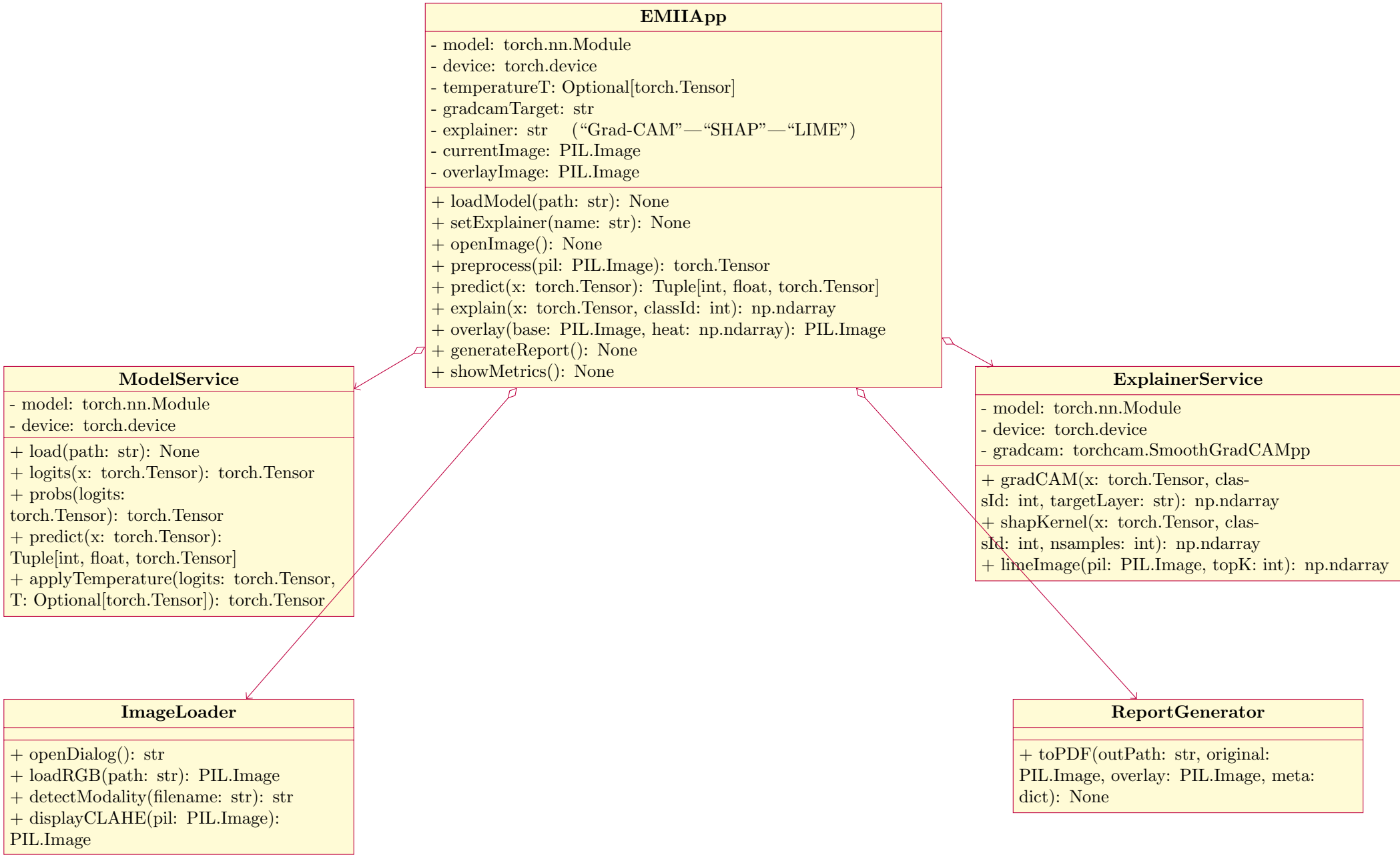
EMIIApp
- model: torch.nn.Module
- device: torch.device
- temperatureT: Optional[torch.Tensor]
- gradcamTarget: str
- explainer: str ("Grad-CAM"—"SHAP"—"LIME")
- currentImage: PIL.Image
- overlayImage: PIL.Image
+ loadModel(path: str): None
+ setExplainer(name: str): None
+ openImage(): None
+ preprocess(pil: PIL.Image): torch.Tensor
+ predict(x: torch.Tensor): Tuple[int, float, torch.Tensor]
+ explain(x: torch.Tensor, classId: int): np.ndarray
+ overlay(base: PIL.Image, heat: np.ndarray): PIL.Image
+ generateReport(): None
+ showMetrics(): None
ModelService
- model: torch.nn.Module
- device: torch.device
+ load(path: str): None
+ logits(x: torch.Tensor): torch.Tensor
+ probs(logits:
torch.Tensor): torch.Tensor
+ predict(x: torch.Tensor):
Tuple[int, float, torch.Tensor]
+ applyTemperature(logits: torch.Tensor,
T: Optional[torch.Tensor]): torch.Tensor
ExplainerService
- model: torch.nn.Module
- device: torch.device
- gradcam: torchcam.SmoothGradCAMpp
+ gradCAM(x: torch.Tensor, clas-
sId: int, targetLayer: str): np.ndarray
+ shapKernel(x: torch.Tensor, clas-
sId: int, nsamples: int): np.ndarray
+ limeImage(pil: PIL.Image, topK: int): np.ndarray
ImageLoader
+ openDialog(): str
+ loadRGB(path: str): PIL.Image
+ detectModality(filename: str): str
+ displayCLAHE(pil: PIL.Image):
PIL.Image
ReportGenerator
+ toPDF(outPath: str, original:
PIL.Image, overlay: PIL.Image, meta:
dict): None