DEEP LEARNING: INDIVIDUAL PROJECT REPORT

TEAM: 2

NAME: MEDHASWETA SEN

TOPIC: EXPLAINABLITY AI(XAI)

Introduction:

Objective: Enhance understanding of complex machine learning models, often seen as "black boxes", through Explainable AI (XAI).

Background

- **Problem:** As machine learning models become more sophisticated, their internal processes become less transparent, making it difficult to understand their decision-making logic.
- **Significance:** It's increasingly important to comprehend how these models arrive at their conclusions. Relying solely on performance metrics like accuracy is insufficient; models can be misleading.
- **Solution:** XAI aims to make the model's behaviour understandable, indicating if any issues exist in data preparation or other stages.

Proposal

1. **XAI Demonstrations Across Machine Learning Models:**
    - Focus: Implement XAI techniques in various machine learning models, specifically in image classification.
    - Tools: Utilize base PyTorch for MLP and LSTM model architecture inspections on demo datasets.
2. **Leveraging Python XAI Utilities:**
    - Aim: Provide detailed explanations of internal workings of different XAI tools.

- Planned Utilities: GradCAM, Occlusion Sensitivity, RISE, Deconvnet, LIME, SHAP and Integrated Gradients.
- Outcome: Educate users on selecting the appropriate XAI utility for their specific applications.

3. **Development of a No-Code XAI Product:**
    - Target Audience: GWU community and others involved in image classification.
    - Features: Compatibility with TensorFlow and PyTorch, allowing users to upload any model (custom or pre-trained) with an observation for analysis.
    - Goal: Facilitate easy evaluation of model robustness, extending beyond standard test sets.

Impact

This project aims to provide a comprehensive and user-friendly approach to understanding and evaluating AI models, making advanced AI more accessible and trustworthy. By bridging the gap between complex AI algorithms and practical, understandable explanations, it fosters a deeper, more critical engagement with AI technology, crucial for its responsible and ethical application.

Description of Individual Work:

1. **FORGrad:**

    ForGrad represents a significant advancement in the field of explainable artificial intelligence (XAI), particularly in enhancing the clarity and interpretability of gradient-based attribution methods. This technique is crucial for researchers and practitioners who seek to understand the decision-making processes of complex machine learning models, especially in scenarios where the interpretability of such models is paramount.

    ForGrad applies a low-pass filter to attribution maps generated by gradient-based methods, such as Integrated Gradients or methods involving occlusion analysis. These methods are foundational in XAI, providing insights into how different input features influence the output of a model. However, a common challenge with these methods is the presence of high-frequency noise in the attribution maps, which can obscure the true contributions of input features and render the interpretation less reliable.

    ForGrad addresses this challenge by effectively filtering out the high-frequency components. The degree of filtering is controlled by a parameter known as sigma σ. This parameter determines the cutoff value for frequency retention, thereby influencing the

extent to which high-frequency components are filtered out. The range of σ is between 0 and N, where N is the dimension of the squared image. A σ value of 0 results in the elimination of all frequencies, while a value equal to N retains all frequencies. For optimal performance, especially in contexts like ImageNet image classifications, a default sigma value of around 15 is often recommended.

In practical applications, ForGrad can be seamlessly integrated with existing attribution methods. For instance, when applied in conjunction with methods like Occlusion or Integrated Gradients, ForGrad enhances the interpretability of the resulting attribution maps. It filters the explanations post-generation, leading to more interpretable and noise-free visual representations. This enhancement is particularly valuable in scenarios where the clarity of feature attribution directly impacts the utility and trustworthiness of the model's explanations.

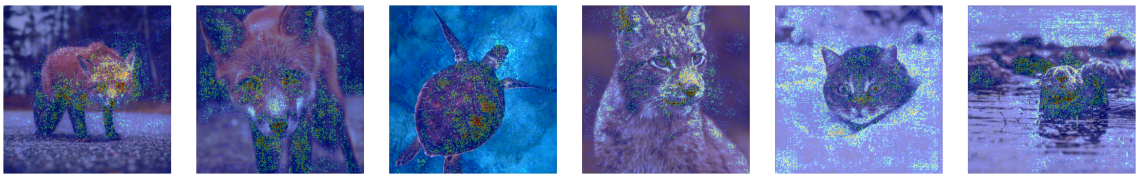
Furthermore, ForGrad's implementation as a function allows for flexibility and ease of use. The function accepts a tensor of explanations and a sigma value, and it is designed to work with explanations that are at least 3D (batch, height, width). The sigma parameter serves as the bandwidth of the low-pass filter, where a higher sigma value corresponds to the retention of more frequencies.

In summary, ForGrad is a versatile and powerful tool in the XAI toolkit, elevating the interpretability of gradient-based attribution methods. By mitigating the impact of high-frequency noise, it enables clearer and more reliable insights into the decision-making processes of AI models, particularly in complex and critical applications.

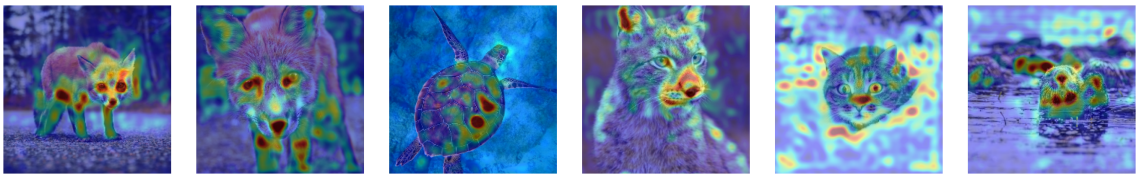
EXAMPLES:

ORIGINAL IMAGE:

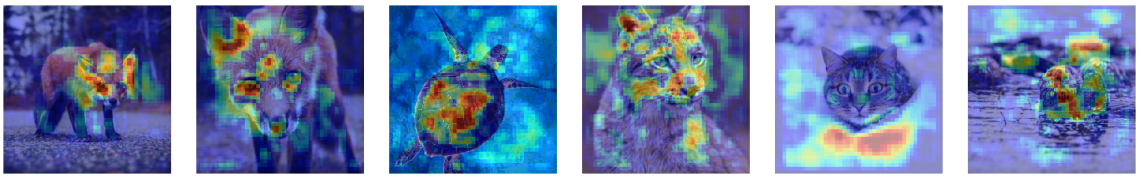
INTEGRATED GRADIENT:



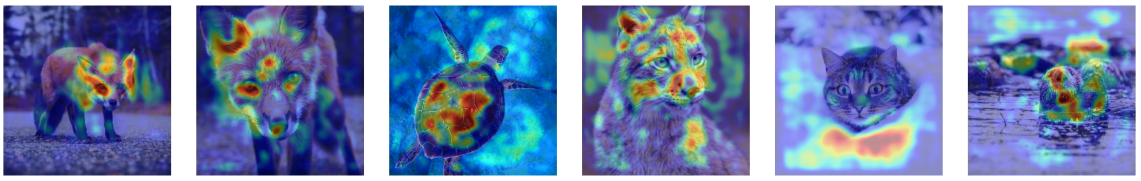
INTEGRATED GRADIENT WITH FORGrad:



OCCLUSION :



OCCLUSION WITH FORGrad:



Since we have used occlusion sensitivity and integrated gradient in our StreamLit app, that's the ones I am concentrating on. As expected, after applying ForGrad, the explanations provided by every methods are much less noise.

2. **L2X (Learn to Explain) on MNIST:**

   L2X (Learning to Explain) is an innovative approach in the domain of explainable artificial intelligence (XAI), particularly applied to the MNIST dataset for image classification. This method diverges from traditional gradient-based explanation techniques by employing a distinct explanation model trained separately from the predictive model. The core advantage of L2X lies in its ability to generate explanations rapidly once the explanation model is trained. However, the quality of these explanations is heavily dependent on the explanation model itself, influenced by factors such as the network structure and training parameters.

   In the specific implementation for image data, as seen in the `omnixai.explainers.vision.agnostic.l2x` module, a default explanation model is provided. Users have the flexibility to implement alternative models by adhering to the same interface. This approach aligns with the principles outlined in the paper "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation" by Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan.

To illustrate the application of L2X with MNIST, the process begins with a simple convolutional neural network (CNN) design, comprising two convolutional layers and one dense hidden layer. This model structure is tailored for the digit recognition task within the MNIST dataset.

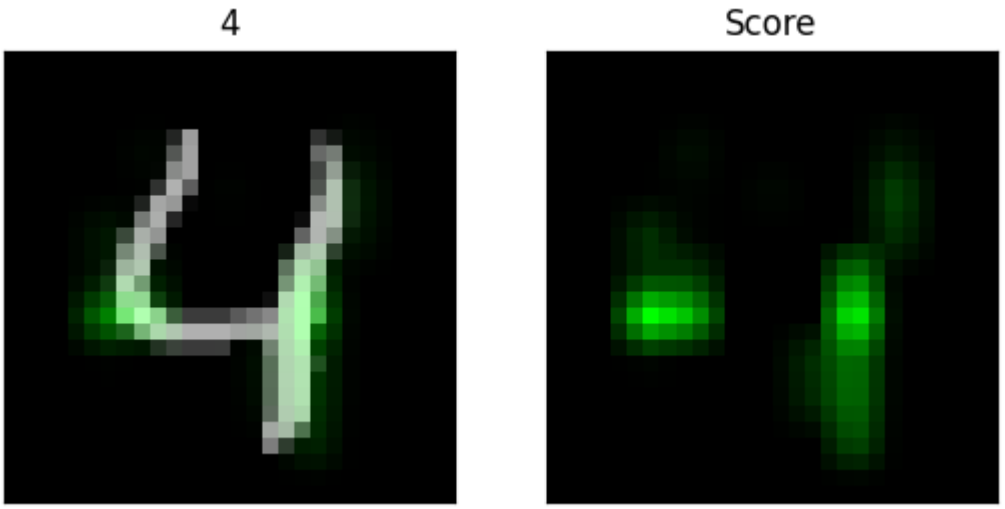
The MNIST dataset is loaded, and `Image` objects from the `omnixai` library are used to represent batches of images. This representation is beneficial as `Image` can be constructed from a numpy array or a Pillow image, making it versatile for handling different image formats.

For the prediction function, the CNN model is employed. The images are preprocessed using a transform pipeline and then fed into the model. The prediction function outputs class probabilities or logits for classification tasks.

Training the CNN involves standard procedures with an Adam optimiser and cross-entropy loss function. The model's performance is evaluated on the test dataset, yielding accuracy metrics for each digit class.

When initialising the `L2XImage` explainer, several parameters are set, including the training data (`train_imgs`) and the prediction function (`predict_function`). The `L2XImage` explainer employs two PyTorch models for estimating $P(S|X)$ and $Q(X_S)$ in L2X, where S represents the subset of features deemed relevant for explanation.

Once trained, the explainer's `explain` method is invoked to generate explanations for the classification task. These explanations can be visualised , which provides an intuitive way to understand which parts of the image most significantly influenced the model's predictions.



3. **Model Interpretability with Integrated Gradients:**

Integrated Gradients is a powerful technique used in the realm of machine learning to shed light on the decision-making processes of complex models, especially in tasks related to Computer Vision. In simpler terms, it helps us understand why a machine learning model makes a particular prediction by identifying the critical factors or features that influence that decision.

To grasp the essence of Integrated Gradients, let's start with the broader context of attribution methods. Attribution methods, in the world of machine learning, are like detectives trying to figure out which clues led to a specific outcome. In the case of Computer Vision, their mission is to highlight the pixels in an image that have the most significant impact on the model's output.

Integrated Gradients belongs to a category of attribution methods known as back-propagation methods. These methods, alongside perturbation-based techniques, are pivotal in understanding model decisions. What sets back-propagation methods apart is their reliance on the model's internal weights. Essentially, they backtrack from the model's final prediction to the input data, following the path of significance.

Now, let's dive into the heart of Integrated Gradients. Imagine we have an image, and we want to understand why our model classified it the way it did. Instead of just looking at the image's gradient, Integrated Gradients takes a more sophisticated approach. It calculates the gradients not just at a single point but along a path from a starting point (called the baseline) to the image we're interested in.

Here's a breakdown of the key elements:

1. **Baseline State**: The journey begins with a baseline state, often representing the complete absence of features. Think of it as the "zero" state where nothing relevant is happening.
2. **The Path**: Integrated Gradients computes gradients at various points along a straight-line path from this baseline state to the input image. These gradients tell us how sensitive the model's prediction is to changes along this path.
3. **Averaging**: Unlike basic gradient calculations, Integrated Gradients doesn't stop at just one point. It averages the gradient values across the entire path. This helps provide a more comprehensive view of feature importance.

Mathematically, Integrated Gradients can be represented as an integral:

$$IG(x) = (x - \bar{x}) \cdot \int_0^1 \partial Sc(\bar{x} + \alpha(x - \bar{x}))/\partial x \, d\alpha$$

In this equation, x represents the image we want to explain, $\bar{x}$ is the baseline state, c is the class of interest, and Sc is the unnormalised class score.

Now, why is Integrated Gradients so valuable? It helps us understand not just the "what" but the "why" behind a model's decisions. It uncovers which parts of an image or data contributed most to a particular prediction. This insight can be used for debugging models, improving their transparency, and even identifying biases or unexpected behaviour.

In practice, when using Integrated Gradients, we'll need to consider a few parameters, such as the output layer, batch size, steps, and baseline value. These parameters help fine-tune the method to suit our specific use case and computational resources.

In summary, Integrated Gradients is a method that brings transparency and interpretability to machine learning models, especially in tasks like image recognition. By tracing the path of significance from a baseline state to the input data, it helps us understand why models make certain decisions, making it an invaluable tool in the world of explainable AI.

```
output_layer=-1,batch_size=16,steps=50,baseline_value=0
```

```
output_layer = -1, batch_size = 16, steps = 50, baseline_values = 0.5
```

4. **RISE (Randomised Input Sampling for Explanation of Black-box Models):**

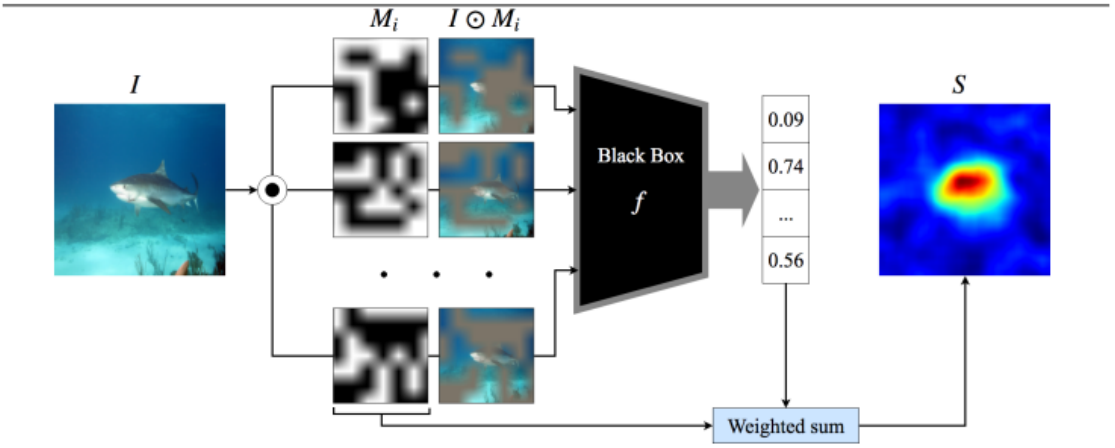


Figure 3: Overview of RISE: Input image I is element-wise multiplied with random masks M_i and the masked images are fed to the base model. The saliency map is a linear combination of the masks where the weights come from the score of the target class corresponding to the respective masked inputs.

Rise is an attribution method designed primarily for computer vision tasks, offering insights into why a machine learning model makes specific decisions when processing images. It has been seamlessly integrated into the Xplique framework, which provides tools for explaining and interpreting machine learning models.

At its core, Rise falls under the umbrella of attribution methods, which aim to shed light on the critical factors or variables that influence a model's decision-making process. In the context of computer vision, this typically involves identifying the pixels within an input image that have the most significant impact on the model's output or prediction.

Rise, however, stands out as a perturbation-based attribution method, which places it in one of two major categories of attribution methods. Unlike back-propagation methods, perturbation-based methods focus on perturbing or modifying the input data and observing how these perturbations affect the model's output. The underlying assumption is that perturbing important input elements will result in more substantial changes in the model's output, thus highlighting their significance.

Rise employs a unique approach to attribution, distinct from gradient-based methods or other perturbation-based techniques. Here's a closer look at how the Rise method operates:

1. **Binary Masks**: Rise generates binary masks that cover various portions of an input image. These binary masks act as a way to either hide or reveal specific pixels within the image.
2. **Scoring Pixel Influence**: For each pixel in the image, Rise calculates a score. This score is determined by observing how the model's output changes when that particular pixel is either masked (hidden) or unmasked (revealed). Essentially, Rise assesses the influence of each pixel on the model's decision.
3. **Averaging Scores**: To provide a comprehensive explanation, Rise takes the mean of the scores obtained for each pixel. This aggregation of scores allows Rise to identify which pixels, on average, have the most significant impact on the model's decision.

Mathematically, this can be expressed as follows:

$\phi c(x) = 1/\mathbb{E}[M] \Sigma(fc(x \odot Mi)Mi)$

In this equation:

- $\phi c(x)$ represents the explanation score for input x and class c.
- Mi is the i-th random mask.
- $fc(x)$ is the class c score obtained from the model f for input x.

Visualising Rise Masks

To help our understand the concept of Rise masks better, consider the analogy of a grid. Rise generates these random masks by following a grid pattern, where each grid cell can be set to either 0 or 1, indicating whether the corresponding pixel is hidden or revealed. This grid-based approach allows Rise to explore various combinations of pixel masks to understand their collective impact.

Hyperparameters in Rise

To effectively use the Rise method, we need to be aware of and tune several hyperparameters. These hyperparameters significantly affect the quality and efficiency of our attribution results. Some crucial hyperparameters in Rise include:

1. **batch_size**: Determines the number of perturbed samples processed simultaneously. It should align with our hardware capabilities and image size.
2. **nb_samples**: Specifies the number of masks generated for Monte Carlo sampling. Finding the right balance is essential, as too few masks result in noisy explanations, while too many increase computation time.
3. **grid_size**: Sets the size of the grid used to generate masks. Smaller values create smoother explanations, while larger values increase precision.
4. **preservation_probability**: Controls the percentage of non-masked pixels in each mask. It affects the quality of the explanation but not computation time.

Tuning Rise Hyperparameters

To achieve the best results with Rise, we'll need to fine-tune these hyperparameters based on our specific use case. Here are some tips for tuning them effectively:

- Match the batch size with the one used during our model's training to ensure efficient memory usage.
- Experiment with the number of samples (nb_samples) to strike a balance between explanation precision and computation time. Larger input sizes typically require more samples.
- Adapt the grid size (grid_size) to our image dimensions and the size of elements of interest. Start with a default value of 7 and adjust from there.
- Modify the preservation probability (preservation_probability) based on our goals. Increase it to highlight more elements simultaneously or decrease it for a more focused explanation.

```
nb_samples=20000, grid_size=13, preservation_probability=0.5
```

```
nb_samples_list = 100, grid_size = 13, preservation_probability = 0.5
```

```
nb_samples_list = 20000, grid_size = 13, preservation_probability = 0.5
```

```
nb_samples = 10000, grid_sizes = 2, preservation_probability = 0.5
```

5. **Class Activation Maps for Object Detection with Faster RCNN :**

Object detection in computer vision is a demanding task, and achieving precise results involves locating objects in images and accurately classifying them. Class Activation Maps (CAM) are a popular tool for highlighting important image regions for classification. However, when applied to object detection, they face unique challenges due to the need to work with complex, multi-object scenes.

Gradient-Free CAM Methods for Object Detection:

In object detection scenarios, the usual CAM methods that rely on gradients may not be suitable because they can't easily differentiate between objects or efficiently handle complex object detection pipelines. Instead, we turn to gradient-free CAM methods, two of which stand out:

- **EigenCAM**: EigenCAM is a lightning-fast method for identifying crucial image regions, making it ideal for scenarios where speed is essential, such as object detection. However, it may not excel at distinguishing between different object categories within an image, which is acceptable in many object detection applications.

- **AblationCAM**: AblationCAM, on the other hand, is a more sophisticated gradient-free CAM method. It's particularly valuable when the goal is to achieve fine-grained class discrimination. This is crucial in object detection when we want to differentiate between various object categories present in the same image. AblationCAM achieves this by selectively "ablating" or deactivating specific regions in the image and observing how these changes affect the model's output. It assigns higher importance to areas that significantly influence the model's ability to classify objects.

The Role of the FPN Backbone:

In the context of object detection models like Faster R-CNN, the Feature Pyramid Network (FPN) backbone plays a vital role. It generates essential activations that are used to locate and classify objects within an image. When applying CAM methods, we focus on this FPN backbone to compute meaningful activations.

Custom Reshape Transform:

Working with tensors of varying spatial sizes generated by the FPN backbone can be a technical challenge in object detection. To address this, a custom "reshape" transform called `fasterrcnn_reshape_transform` is introduced. This transform aggregates differently sized tensors into a common shape, ensuring effective utilization of the activations.

Custom Target Function for CAM:

For CAM methods to be effective in object detection, it's crucial to define a custom "target" function. This function determines what aspect of the bounding boxes to maximize when generating the CAM. In object detection, this could involve maximizing bounding box scores, matching object categories, or considering properties like width or height. The introduces a target function called `FasterRCNNBoxScoreTarget`, which calculates a score for each bounding box based on factors like Intersection over Union (IoU) and category matching.

Adjusting the `ratio_channels_to_ablate` Parameter:

An intriguing feature discussed is the ability to modify the number of layers to ablate in AblationCAM. This flexibility is governed by the `ratio_channels_to_ablate` parameter, which allows for the adjustment of the proportion of channels (activations) to ablate. Fine-tuning this parameter provides a balance between computation speed and the level of detail captured in the CAM. For example, setting it to 0.1 means ablating 10% of the channels, potentially speeding up computation while still capturing essential information. Conversely, setting it to 0.01 reduces ablation to just 1% of the channels, resulting in a more detailed but computationally intensive CAM.

In summary, this exploration demonstrates the adaptability of gradient-free CAM methods, such as EigenCAM and AblationCAM, to address the specific challenges and goals of object

detection projects. These methods provide valuable insights into detected objects within images while considering computational efficiency, making them powerful tools in the field of computer vision.

6. **StreamLit App:**

Due to the 4minutes per person time limit only the Integrated Gradients and RISE methods were included by me in the main steam lit demo. and I have also made the XAI for object detection page. Note these demo provides a no code platform for playing around with the parameters of the various XAI tools.

Summary and Conclusions:

This project contributes to the field of XAI by implementing and explaining various XAI techniques across different machine learning models. It empowers users to understand complex AI models and make informed decisions. The StreamLit app further simplifies the application of XAI methods, promoting transparency, accountability, and responsible AI development. Overall, this project enhances our ability to harness the power of AI while ensuring its ethical and responsible use.

References:

1. https://deel-ai.github.io/xplique/latest/
2. https://github.com/salesforce/OmniXAI

Percentage of Borrowed Work:

LINES OF CODE FROM INTERNET: 525

LINES OF CODE FROM INTERNET I MODIFIED: 150

LINES OF CODE I WROTE: 225

PERCENTAGE: 50%