FINAL YEAR PROJECT (FYP-II)

FINAL REPORT

F21-35-R-ViolenceXplainAR

FYP TEAM

Ms. Iqra Aziz	**18i-0491**
Ms. Tehreem Khawar	**18i-0489**
Ms. Hafsa	**18i-0405**

Supervised By

Mr. Hassan Mustafa

FAST School of Computing

National University of Computer and Emerging Sciences

Islamabad, Pakistan

2021

F21-35-R-ViolenceXplainAR

Students' Submission

F21-35-R-ViolenceXplainAR

Anti-Plagiarism Declaration

This is to declare that the above FYP report produced under the: Title: F21-35-R-ViolenceXplainAR is the sole contribution of the author(s) and no part hereof has been reproduced on as it is basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 17/12/2021

Student 1

Name: Iqra Aziz

Signature:_____

Student 2

Name: Tehreem Khawar

Signature:_____

Student 3

Name: Hafsa

Signature:_____

Supervisor (Faculty)

Name: Mr. Hassan Mustafa

Signature:_____

Authors' Declaration

We declare that the work presented in the report is our own, and has not been submitted/presented previously to any other institution or organization.

Executive Summary

The following report titled F21-35-R-ViolenceXplainAR consists of the details of all four iterations of our final year project on explainable AI and using it as a debugging framework for deep learning models. Our research will focus on applying Layer-wise Relevance Propagation (LRP) on various deep learning models and generating heatmaps to verify whether a generic debugging framework can be built for these models.

The first iteration consists of literature review of five most relevant papers along with references to other similar work, proposed strategy that we will be using i.e., the problem statement, research gap and models that we will be applying throughout the one-year FYP. Five research papers chosen are Driver activity recognition for intelligent vehicles, Explain and improve LRP-inference fine tuning image captioning models, resolving challenges in deep learning-based analyses of histopathological images using explanation methods, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, and A survey on Explainable Artificial Intelligence (XAI): Towards medical XAI.

In the second iteration we implemented LRP on a pre-trained aggressive behavior detection model for Explainability. We used pre-defined formulas that needed to be implemented on each layer of the deep learning model to perform explainability. We generated heatmaps for different frames of various videos which highlight the determining features in classification. LRP is then tested on a test set of frames from different videos and a confusion matrix is generated to evaluate the overall debugging ability of our LRP model. LRP was then applied on VGG and Alexnet model pretrained on imagenet dataset to see quality of heatmaps generated and determine if LRP formulas could be generically applied to all deep learning models.

In the third iteration, after we were sure that LRP technique could be applied to models of various depth, layer types and kernel sizes, we did some deterministic testing. To test how well the heatmaps could identify learned features of different deep learning models and verify whether it could successfully debug them, we tested and compared AlexNet and VGG by generating heatmaps for a variety of images from different classes. This clearly showed features each model had learned for each class and where it lacked, hence proving that heatmap generation through Layer-wise Relevance propagation could be effectively used as a debugging tool for deep learning models.

In the last iteration, we made a frontend using React to display the resultant heatmaps of AlexNet and VGG, therefore show casing the output and differences in features extracted more formally. Moreover, we analyzed results in the form of tables and confusion matrices and compiled all of the work in a finalized Research Paper draft.

Contents

Introduction

Problem Domain

Explainability in Neural Networks is one of the challenges in Deep Learning today. Neural Networks tend to act as a black box and it is oftentimes difficult to interpret how a neural network reached a particular decision. When we talk about real world applications, sometimes it is important to know the exact features that became the basis of a certain classification by the neural network and this can be achieved by Explainability. Some exceptional work has been done on Explainability of Image and Text classification. Different techniques have been used for this purpose of explaining classifications e.g. LRP (Layer-wise Relevance Propagation), CAM (Class Activation Mapping), SGLRP (SoftMax-Gradient LRP), GRAD-CAM (GradientWeighted CAM), CLRP (Contrastive LRP). These techniques have been successfully used for image and text classification as mentioned in (Samek et al., 2017) and (Bach et al., 2015).

However, we want to take this application of Explainable AI a notch up and use it as a debugging tool for deep learning models. Oftentimes, extensive amounts of energy, time and resources are wasted on training deep learning models and many a times, this training is fruitless as hyper parameters may not even be tuned accordingly. The element of transparency is hugely missing from training of deep leaning models and the black box problem intensifies once we start training models. Therefore, we need a better tool for debugging the learnability of these models and to verify we are going in the correct direction with regards to hyper parameters and feature maps.

Research Problem Statement

In our Final year project, we will be applying Layer-wise Relevance Propagation (LRP) on various deep learning models to verify and show that LRP can be used as a debugging framework for deep learning models. The generic automated system can be used to apply LRP and generate heatmaps for any given class of a model and helps in depicting the learnt features through the blue and red colored heatmap.

ITERATION 1

This iteration consists of literature reviewing i.e. reviewing high ranking papers to understand our problem domain better and show that our identified research gap is well recognized by work being done in similar areas. Following is the review of five most relevant research papers related to our work and framing the problem statement further.

Literature Review

Research Item # 1

The paper **"Explain and Improve : LRP-inference fine tuning image captioning models"** (Sun et al., 2022) aims at generating text descriptions from image representations. LRP, GradCam, Guided Grad-CAM(GG-CAM) and Guided Backpropagation(GBP) methods are used as these approaches provide high resolution image explanation for CNN models. This paper deals with the hallucination problem that is caused by language priors or visual misclassification which could be partially due to biases present in dataset. In this paper, object hallucination is reduced by LRP-inference fine tuning (LRP-IFT) strategy without additional annotations. It helps to de-bias Image Captioning Models (ICM) from frequently occurring object words which is achieved by implementing it on top of pre-trained ICM trained with Flickr30K and MSCOCO2017 datasets. Flickr30K dataset is prepared as per the Karpathy split. For MSCOCO2017, the original validation set is used as the offline test set and extract 5000 images from the training set as the validation set. The train/validation/test sets are with 110000/5000/5000 images. Vocabularies are built only on the training set. The words that appear less than 3 and 4 times are encoded as an unknown token for Flickr30K and MSCOCO2017, respectively, resulting in 9585 and 11026 vocabularies for the two datasets. In addition, LRP is used to design a re-weighting mechanism for context representation. A weight is calculated from LRP relevance scores which is used to up-scale the supporting features and down-scale the opposing ones. Then re-weighted context representation is used to predict the next word for fine tuning. This strategy resembles how we correct our cognition bias.

Moreover, the paper in discussion adopts the Encoder-Decoder approach to bridge the gap between image and text usually with CNN as image encoder and RNN as sentence decoder. For example, to caption a given image, it is first encoded with pre-trained CNNS and extract a visual feature which is decoded by LSTM augmented with an attention mechanism to generate context representation. The word predictor takes the context representation and the hidden state of decoder as inputs to predict the next word. Therefore, two image captioning models i.e., AdaLSTM and MH-FC are built. Ada-LSTM consists of adaptive attention mechanism(calculated with the additive attention) and LSTM followed by fully connected layer as word predictor. While MH-FC adopts multi head attention mechanism(calculated with scaled dot product) followed by fc layer as word predictor. These models are first trained with cross entropy loss and then they are further optimized with Self Critical Sequence Training algorithm

which optimizes non differentiable evaluation metrics. By comparison, it is seen that MH-FC achieves higher correctness than Ada-LSTM.

Furthermore, LRP assigns relevance score to every pixel of input image and every word of sequence input and provides explainability through backpropagation. Also Grad* methods are based on gradient back propagation. Finally, with extensive qualitative and quantitative experiments, it is demonstrated that explanation methods provide more interpretable information than attention, disentangle the contributions of the visual and linguistic information, and help to debug the image captioning models such as mining the reasons for the hallucination problem. So, it is concluded that the proposed LRP-inference fine-tuning strategy can successfully de-bias image captioning models, alleviate object hallucination and maintain sentence structure. Also it requires no additional annotations and training parameters. On the contrary, another challenging direction is Novel Object Captioning (NOC) and captioning with different styles. LRP re-weighting mechanism can be helpful for NOC. NOC aims to predict those object words that are unseen by the model during training. It also faces the challenge of unbalanced training data.

Lastly, as this research item describes that LRP is helpful for model debugging and remove object hallucinations, we can also use the same approach for debugging our model. Because LRP explicitly shows positive and negative evidence used by the model to make decisions. It also provides high resolution, pixel wise image explanation. In the second place, Grad-CAM and Guided Grad-CAM introduces an explanation method for video captioning model. They further adapt the method to image captioning model by slicing an image with grids to form a sequence of image patches, treated as video frames. They show that pixel-wise cross entropy loss can guide the model to make right decision using right reasons. So, we can extend this research and apply explainability of these methods on our Video Activity Recognition.

Research Item # 2

The paper **“Resolving challenges in deep learning-based analyses of histopathological images using explanation methods”** (Hägele et al., 2020) focuses on importance of Explainability methods in the medical domain specifically for medical imaging. Digital Pathology is a highly interdisciplinary field, where medical and machine learning professionals need to collaborate closely. The problem that arises is that machine learning experts involved usually have only basic level knowledge of medical domain and therefore there is a huge risk

of introducing biases unknowingly. The paper focuses on biases which are typically inherent in histopathological image data and investigates three types of biases namely: (1) biases which affect the entire dataset, (2) biases which are by chance correlated with class labels and (3) sampling biases. The reason this is such an important issue and needs to be discussed is because in clinical diagnostics, false predictions usually come at a high cost and can be life threatening. Therefore, it is absolutely essential to detect and remove hidden biases in the data that do not generalize to the generic patient case. Heatmaps can help detect biases without the necessity for labels, and debug the Neural network decisions (see whether they are consistent with medical expert decisions or not).

The research uses binary classification task of tumor tissue discrimination in publicly available hematoxylin-eosin-stained images of various tumor as input model for generating Explainability. The explanation method used is as Layer-wise Relevance Propagation (LRP) for generating high-resolution heatmaps for single input images because pixel-wise heatmaps are more flexible in resolution. Three different projects of TCGA Research Network have been used as input: skin melanoma (SKCM), breast carcinoma (BRCA), and lung carcinoma (LUAD). The contribution of this work to introduce explainable neural networks into a digital pathology workflow. First, regions of interest were chosen from images and annotated by a pathologist, then, image patches of 200 x 200 PX with corresponding binary labels (tumor vs. no tumor) were extracted from these large-area annotations. A pretrained Google Net from the Caffe Model Zoo was finetuned for each tumor entity. For the training and test set patient cases were split 80/20, while keeping the ratio of healthy and diseased cases constant. Publicly available Caffe implementation of Layer-wise Relevance Propagation (LRP) is used to provide pixel-level explanation heatmaps for the classification decision; where a relevance score is assigned to every pixel, indicating how much the individual pixel contributes to the classifier decision in favor of a particular class. Heatmaps have been produced for five experiments to help investigate three main forms of biases. Experiments include Feature visualization Class sampling ratio, Dataset bias, “Class correlated” bias, and Sample bias. The experiments help to investigate three main forms of biases. Strengths of this paper are that this type of research on explainable image classification is going to shape the future of medical imaging and AI based medical decisions. LRP which has flexible resolution has been used to produce high-resolution heatmaps.

Overall, two main insights are added through this research: (1) Explanation heatmaps not only enable domain experts to visually inspect learned feature, but also allow for quantitative

evaluations on cell level. (2) Besides, high-resolution heatmaps help to reveal and aid in subsequent removal of various types of latent biases in the data. Computing high-resolution heatmaps enabled the researchers to take the evaluation of the model from patch-level to the relevant level of cells to debug the neural network. The discrimination between tumor and healthy cells can often be guided by adjacent tissues and spatial relations to other cells. The paper proposes using patch sizes to be in a range where they include multiple cells and other morphological structures. In an attempt to improve on this research, another theoretical possibility, possibly better way of tackling the challenge of cell localization is to apply object detection algorithms from computer vision. Moreover, Grad-CAM heatmaps are also localized and can be an alternative to the use of LRP and may produce better resolution heatmaps.

Lastly, this paper paves the way for Explainability in Neural networks by demonstrating effectively a very important application of Explainability in Medical field. In our FYP, we propose to extend this type of research and apply LRP on Video Activity Recognition to get Explainability for model decision making which will help improve, debug and understand the Neural Network's decisions.

Research Item # 3

Many articles have suggested different measures and frameworks to capture interpretability, and the topic explainable artificial intelligence (XAI) has become a hotspot in ML research community. In medical field, failure is not an option because human lives are on the line. This requires greater interpretability/explainability, which often means we need to understand the mechanism underlying the algorithms. The survey paper **"A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI"** (Tjoa & Guan, 2020) provides a review on interpretabilities suggested by different research works and categorizes them. The different categories show different dimensions in interpretability research, from approaches that provide obviously interpretable information to the studies of complex patterns. By applying the same categorization to interpretability in medical research, it is hoped that clinicians and practitioners can subsequently approach these methods with caution and insights can be provided into interpretability with more considerations for medical practices.

This paper presents two major types of interpretability namely perceptive interpretability and interpretability by mathematical structures. Interpretabilities that can be humanly perceived are included in perceptive while mathematical structures are needed to reveal the mechanisms of ML and neural network algorithms. Dataset for perceptive interpretability include Medical data which comes in the form of traditional 2-D images or more complex formats such as NIFTI or

DCOM containing 3-D images with multiple modalities and even 4-D images which are timeevolving 3-D volumes. One of the most important subcategory of Perceptive is *Saliency* which explains the decision of an algorithm by assigning values that reflect the importance of input components in their contribution to that decision. Saliency methods via decomposition have been developed. Class activation map (CAM) has been a popular method to generate saliency maps that corresponds to discriminative features for classification. Also, LRP has been used to construct saliency maps for interpretability. It is also applicable for video processing. In short, the important scores are decomposed such that the sum of the scores in each layer will be equal to the output. These techniques are applied on various medical cases e.g., Grad-CAM is used for visualization of pleural effusion in a radiograph. CAM is also used for interpretability in brain tumor grading. LRP is applied on fMRI data from Human Connectome Project to generate heatmap visualization. Activation Optimization under the category of *Signal Methods* is another proposed strategy for interpretability. It includes finding input images that optimizes the activation of a neuron or a collection of neurons. The optimized input can emerge as something visually recognizable e.g., fuzzy combination of swirling patterns and hence interpretability can be easily applied.

Future research and application may benefit from a practice of consciously and consistently extracting interpretable information for further processing, and the process should be systematically documented for good dissemination. Currently, with feature selections and extractions focused on improving accuracy and performance, there are still vast unexplored opportunities in interpretability research.

Since DNNs are still not clearly explained, interpretable algorithms deployed in medical practices, human supervision is still necessary. Interpretability information should be considered nothing more than complementary support for the medical practices before there is a robust way to handle interpretability. We will extend this research and try to achieve interpretability in activity recognition on our model which can then also be used in medical tasks as well.

Research Item # 4

The paper **"Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI"** (Barredo Arrieta et al., 2020) is an overview of the existing contributions in the field of Explainable AI, including the prospect toward what is yet to be reached. Recent last years have witnessed the rise of opaque decision systems such as Deep Neural Networks for complex computations. Explanations supporting the output of a model are crucial as these models affect human lives, e.g., in precision medicine, where experts

require far more information from the model than a simple binary prediction for supporting their diagnosis. Interpretability helps reduce biases in decision making, debug models, provide truthful causality in model reasoning, and helps in production of explainable models that aid human understanding while maintaining a high level of learning performance.

The paper makes a clear distinction among models that show some interpretability by design (transparent models), and those that can be explained by means of external XAI techniques (post-hoc explainability). Within transparency, three levels are contemplated: algorithmic transparency, decomposability and simulatability. One example of a transparent model is Linear/logistic regression used to predict category for binary or continuous data. It is among the simplest classification models in supervised learning, however to maintain decomposability and simulatability, its size must be limited, and the variables used must be understandable by their users and therefore it is not so fitting for complex inputs. Another model is Decision Trees which are hierarchical structures for decision making. Despite their high level of transparency, they are not preferred because of their poor generalization properties. The K-Nearest Neighbors model predicts the class of a test sample by voting the classes of its K nearest neighbors. This simple model has the ability to inspect the reasons by which a new sample has been classified inside a group empowers interpretability.

Moreover, the Post-hoc explainability includes models that are not readily interpretable by design and aims to enhance their interpretability, with text explanations, visual explanations, feature relevance explanations and explanation by simplification along with other techniques. Post-hoc explainability in shallow ML models may use KNN, Decision Trees, Tree ensembles or Support Vector Machines. Tree ensembles are amongst the most accurate ML models and provide an improvement for decision trees' generalizations. They combine different trees to obtain an aggregated prediction. While effective against overfitting, the combination of models makes the interpretation of the overall ensemble more complex and so explanation by simplification and feature relevance are used for their explainability. SVM models are more complex than tree ensembles, with a more opaque structure. They construct hyper-planes in a high or infinite-dimensional space, which can be used for classification. Techniques for posthoc explainability are explanation by simplification, local explanations, visualizations and explanations by example.

Lastly, Explainability for deep neural networks include Post-hoc local explanations and feature relevance techniques. The models for which explainability has been proposed are multi-layer neural networks, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Multi-layer neural networks have the ability to infer complex relations among variables, however trust is still an issue so explainability is required. Model simplification approaches,

feature relevance estimators, text explanations, local explanations and model visualizations are used for this purpose. DeepRED algorithm instead extends the decompositional approach to rule extraction by adding more decision trees and rules. Simplification of multilayer neural networks is more complex, so explaining these models by feature relevance methods has become progressively more popular. Taylor decomposition and DeepLift are few methods that back propagate and find relevance scores for input features. CNNs consist of convolutional and pooling layers to automatically learn increasingly higher level features. At the end of the sequence, one or multiple fully connected layers are used to map the output features map into scores. This structure entails extremely complex internal relations that are very difficult to explain. Fortunately, explainability for CNNs is easier, as the human cognitive skills favor the understanding of visual data. In a forward pass, each CNN layer outputs a feature map with strong and soft activations. For understanding the decision process, Deconvnet is used which reconstructs the maximum activations from the feature maps. A cleaner visualization than Deconvnet can be obtained by using a guided back propagation method or adding global average pooling layer before the last convolutional layer. Some authors proposed visualizing the contribution to the prediction of each single pixel of the input image in the form of a heatmap with Layer wise Relevance propagation (LRP) or Gradientweighted Class Activation Mapping (Grad-CAM). Visualization mixed with feature relevance methods are the most adopted approaches to explainability in CNNs. RNNs have lately been used extensively for predictive problems defined over inherently sequential data. RNNs are able to retrieve time-dependent relationships by retaining knowledge in the neuron as another parametric characteristic that can be learned from data. Feature relevance and local explanations are proposed approaches for explanation. Interpretable mimic learning distillation method is proposed for the LSTM networks, so that interpretable features are learned by fitting gradient boosting trees to the trained LSTM network under focus.

The paper emphasizes developing XAI into a Responsible AI. Perhaps the most crucial challenge is the trade-off between model interpretability and model accuracy. It is easier to perform explainability on simpler models but gets harder for complex models. Secondly it is important to ensure confidentiality while performing explainability. Certain vocabulary and general metrics should be defined to make XAI understandable for people who aren’t much familiar with the technicalities of XAI. As efforts are continuously being made to remove the compromise between model accuracy and interpretability it is also exposing the system to risk and that is what we need to tackle with in the near future.

Deep Learning is the model family where most research has been concentrated in recent times and they have become central for most of the recent literature on XAI. Our final year project also focuses on this domain. The review this paper provides is very extensive: text and image classification explainability have been a major focus, however a major research gap we see is Explainability for video activity recognition. Our focus will be applying LRP and CAM for high resolution heatmap generation and Class activation to the state of the art CNN model based classification of Violence.

Research Item # 5

Activity recognition is an emerging area when we talk about neural networks. Detecting human activity can help solve problems and minimize human efforts. One such example is detecting driver activity as mentioned in the paper **"Driver Activity Recognition for Intelligent Vehicles: A Deep Learning Approach"** (Xing et al., 2019). The aim of this paper was to develop a driver behavior monitoring system for intelligent vehicles that focuses on driver's activity and decides whether the driver is ready to take control of the car or not (in case he is using mobile phone). This will help reduce road accidents.

The proposed strategy first takes raw images consisting of seven driver activities (four normal driver activities and three secondary tasks performed by ten drivers) as input and performs segmentation using a Gaussian Mixture Model to find the region of interest (the driver in our case). Finally the images are passed to a CNN model for classification. The CNN models used for activity recognition were *Alex Net, Google Net* and *ResNet50*. They all got good results when they were applied on the ImageNet dataset (a large-scale dataset, which contains more than 15 million high-resolution annotated natural images of over 22,000 categories). Alex Net has five convolutional layers and 3 fully connected neural network layers having pooling and non-linearity layer in between. Google Net was chosen for not only its depth but also because it has an Inception architecture which takes in less parameters and increases sparsity among the layers. Similarly ResNet50 avoids model degradation of deep neural networks by using Residual Networks which use identity maps and copy layers from shallower models to achieve this purpose. The CNN models were fine-tuned using transfer learning. The original CNN models were maintained to provide enhanced feature extraction and only the last layers were replaced by new fully connected layers and a SoftMax layer to train the model according to driver behavior recognition and also find the probabilities of the seven activities performed by the drivers.

The results were generated based on segmented and raw images as the input to CNN model. It was found that AlexNet gave the highest accuracy (91.4%) with segmented images. Google Net gave 87.5% and ResNet50 gave 83% accuracy. It was seen that raw images provided less accuracy for all models. Apart from this Alex Net also took minimum time for training. The results were also compared with other feature extraction methods such as FC7+ANN, PHOG+SVM, OP+ANN, OPsCNN and GMMsCNN. However, the proposed strategy out performed all other methods.

From this we can see how neural networks are able to identify human activity. From this paper we can say that the convolutional layers and pooling layers can help with feature extraction and determining the probabilities of driver behaviors. However, we do see some challenges like all the models gave more accurate results in case of secondary activities. They gave larger inaccurate results for primary driver activities as they do not involve significant movement. This means that the model could be trained better to identify eye gaze to see if the driver is looking in the rear view mirror. We know that secondary activities were easier to identify but we do not know the exact determining frames and that is what we are going to find out in our project. We will be providing explainability for such activity recognition models using LRP and CAM to tell how they identified particular activities.

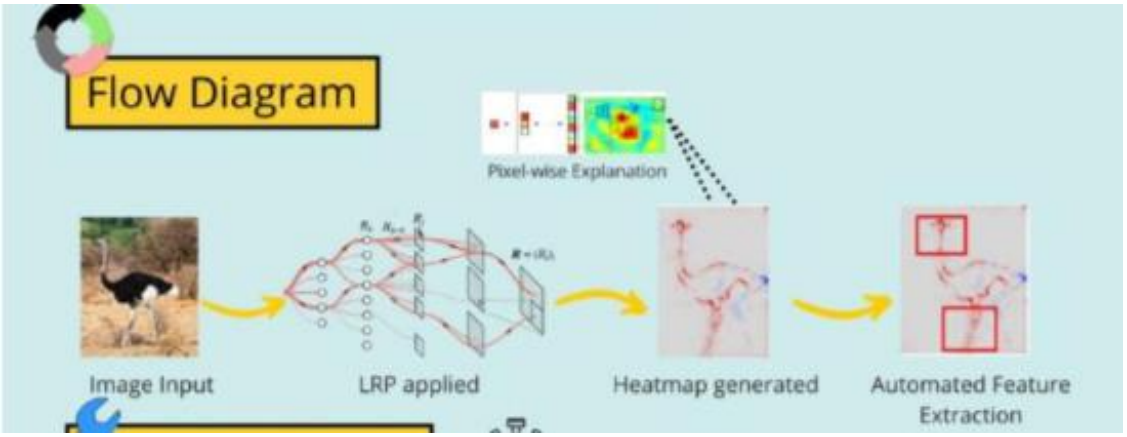
Proposed Approach

To solve the issue of no explainability while training, we propose this project and our debugging framework: **DeepBug**. Our approach for constructing a generic debugging framework for deep learning models would consist of multiple interdependent steps that we'll be carrying out throughout the four iterations of this project.

Firstly, we'll take a pre-trained model and for testing, we pass images through the pre-trained weights of the model. In the forward pass, we'll gather activations at each layer and once we have the final layer activation, we start back propagating relevances at each layer using Layer-wise Relevance Propagation (LRP) and its rules and formulas for various types of layers. In short, once classification is done in the forward pass, we will apply LRP in backward pass for explainability through heatmaps. LRP works by back propagating relevant neurons to the lower layers. These neurons are selected by certain pre-defined propagation rules for each layer. The output of our model will be heatmaps. The red part of heatmaps will show the regions of the images that have bigger contribution towards the classification and are the more prominent extracted features by the models.

The 3 models we are testing with are: 7 layered Aggressive behavior detection model trained on data from Marium Memorial Hospital, 20 layered AlexNet pretrained on ImageNet dataset, and lastly, 38 layered VGG-16 model pretrained on Imagenet dataset. We'll start off by explainability of Aggressive behavior detection model, and move towards VGG and AlexNet later. Since AlexNet and VGG are both trained on same dataset and serve the same classification purpose, with major difference in layers, they would give a better side by side comparison of the debugging nature of the LRP based framework. Our main focus will be AlexNet and VGG.

For each model, to see how well the models have learnt and extracted features, we can test them on a variety of input images. For each, we'll generate heatmaps and see which portions are highlighted red. However, to make it easier to use, we propose that once a model is fully trained, a best and worst image from the test dataset is used to see how well the heatmaps depict learnt features. Since we want that this framework also be used at train time, the debugging model can also be used during training. After every few epochs, best and worst image heatmaps can be generated to see how well the model has learnt features and where it lacks. Hyper parameters can be tuned accordingly and a lot of futile training time can be saved.



Conclusion Iteration1

In this iteration, we have successfully done literature review of most relevant high ranking papers. In next iteration, we will be using LRP for explainability and debugging of deep learning models. We will compare our results with the existing metrics of explainability of image and text classification and see how well our technique will be as a debugging framework for deep learning models.

ITERATION 2

In iteration 2 of our final year project, we had to apply Layer-wise Relevance Propagation (LRP) on a Violence Detection Model, AlexNet and VGG-16 to generate heatmaps for single images. LRP is then tested on a variety of images for all models and a confusion matrix is generated to evaluate the overall debugging ability of our LRP model.

Pre-processing

Model

Models used as input for our LRP based debugging framework are: Aggressive Behavior Detection model, VGG-16, AlexNet.

First, we are using a pre-trained **Aggressive Behavior Detection model** as the input for our LRP implementation. The model is trained on 17,000 images from Maryam Memorial Hospital CCTV's footage available on Kaggle along with other images used for similar projects. These images contain 10,000 examples of aggressive behavior and 7,000 examples of non-aggressive behavior contributing to the only two classes we have in the model. The model although trained on images is tested on videos which are all roughly under 2 minutes long.

Second, we use **VGG-16** which is a widely used CNN architecture for ImageNet dataset. This architecture is used for deep learning image classification problems. It is 16 layers deep including 13 convolutional, 3 fully connected and 12 ReLu and Dropout layers. VGG uses large kernel-sized filters (sizes 11 and 5 in the first and second convolutional layers, respectively) with multiple (3×3) kernel-sized filters, one after another which is a huge improvement on AlexNet. The input dimensions of the architecture are fixed to the image size, (244 × 244). In a pre-processing step the mean RGB value is subtracted from each pixel in an image. After the pre-processing is complete the images are passed to a stack of convolutional layers with small receptive-field filters of size (3×3). All the hidden layers for the VGG network are followed by the ReLu activation function.

Lastly, we use **AlexNet** which is based on convolutional neural networks. It is also trained on ImageNet dataset and very similar to VGG-16. AlexNet architecture is a conv layer followed by pooling layer, normalization, conv-pool-norm, and then a few more conv layers, a pooling layer, and then several fully connected layers afterwards. There are five of these conv layers, and two fully connected layers before the final fully connected layer going to the output classes. AlexNet was trained on ImageNet, with inputs at a size 227 x 227 x 3 images. In the end, there are a couple of fully connected layers of size 4096 and finally, the last layer, is FC going to the

softmax, which is going to the 1000 ImageNet classes. This architecture is the first use of the ReLu non-linearity.

Technology

The models were trained on Python using PyTorch to make the CNN model. We are embedding a .pth file of the model in our LRP implementation i.e. we are using the trained weights of the models as an input for our LRP based debugging framework.

Neural Network

The Aggressive behavior detection model used consists of seven layers; four 2D convolutional layers followed by two fully connected layers and finally a max pooling layer that gives the maximum class output.

```
CustomCNN(
  (conv1): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=128, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=2, bias=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

This network acts like a binary classification model amongst the aggressive and non-aggressive classes. It is a sequential model where the input is eventually downsized to extract the high level features using the 2D convolutional layers. For this purpose, each convolutional layer uses different number of filters to extract those pixels of the frame which seem to have the most prominent elements in the frame. The outputs from the convolutional layers is flattened and passed to the fully connected layers. These fully connected layers are then used for the classification of the frames into the two classes (aggressive and non-aggressive). Before the final classification, the output of the fully connected layers is passed on to a max pooling layer for the down sampling of the input to only show the pixel given the most weightage. The input video is passed into the network frame by frame and each frame is evaluated for aggressive behavior. The frames generated are 224 * 224 along with 3 RGB channels so the total size of the passed frame becomes 3*224*224. After successful propagation it declares a frame as either aggressive or non-aggressive. This labeling is done for all the frames of the video. The final output of this model training is same as the input video with each frame individually labelled.

Architecture of 8 layered AlexNet model pre-trained on ImageNet dataset is as follows:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Architecture of 16 layered VGG-16 model pre-trained on ImageNet dataset is as follows:

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Model Testing and Performance

The aggressive behavior detection model was trained over 30 epochs at which it gave the accuracy of 85% on the validation set. For the purpose of testing, different videos were passed to the model and resultant labelled videos were generated. While testing, we noticed, that the model over fits on epochs greater than or equal to 30 but under fits on epochs less than 30.

VGG16 is object detection and classification algorithm, pre-trained on ImageNet dataset over 150 epochs and is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

AlexNet was born out of the need to improve the results of the ImageNet challenge. This was one of the first Deep convolutional networks to achieve considerable accuracy on the 2012 ImageNet LSVRC-2012 challenge with an accuracy of 84.7% over 90 epochs.

Explainability for the Model

Motivation

Explainable Artificial Intelligence can be used to identify why models classified images in a particular class and what features had more weight in this classification. For this purpose, we are using Layer-wise Relevance Propagation (LRP) to generate heatmaps for frames that are suspected to contain aggressive activity.

Since the heatmap shows exactly which part of the image or which features contribute most to the classification, our LRP model acts as a **debugging framework** and can show whether the models reached correct results or not, what features they learnt along the way, how well they are trained and whether better training and feature extraction can be done or not. In this way, previous deep learning models may be able to identify where the issue occurred in training and correct it by doing subsequent changes in model's layers, activations, and dropouts etc. Moreover, our LRP model will help **improve** the activity recognition models—through explainability any faults in the training of activity recognition may be identified and improved. Lastly, this work helps provide **assurity** of correct results, because instead of just providing a classification through a black box, explainability makes sure to tell the viewer why the classifier or neural network reached a particular decision by generating heatmaps and therefore, provides guaranteed correct results.

Processing of LRP

Once the Aggressive behavior model is trained, we get a file with .pth extension. This model.pth file contains all the specifications of the model used i.e., the layers, features extracted at each layer, and most importantly weights and biases for the trained model.

To begin our implementation, we first run a test video for which we want to generate heatmaps using the following command:

```
%run test.py --model sports.pth --label-bin lb.pkl --input nonFight1.mp4 --output test.mp4
```

The video is then broken down into frames, and each frame is marked as aggressive or nonaggressive. For each frame a (1x2) matrix is generated which contains scores for both aggressive and non-aggressive classes e.g. (3.332359,-2.34532). The final label for the frame depends on the maximum score in this array.

As input for our heatmap generation, we pick out the frame with maximum aggressive score and store the frame separately. This frame is then passed through the trained model and activations at each layer are calculated. In our model.pth file, we have the weights and biases for each layer along with the input and output dimensions which are used to calculate the above mentioned activations. These Activations are stored for use in the later relevance calculations in backward pass. The dimensions of the activations at each layer are given below.

```
print("Activation of Input image x: ", A[0].shape)
Activation of Input image x:  torch.Size([1, 3, 224, 224])

print("Activation of Layer 1: ", A[1].shape)
Activation of Layer 1:  torch.Size([1, 16, 220, 220])

print("Activation of Layer 2: ", A[2].shape)
Activation of Layer 2:  torch.Size([1, 32, 216, 216])

print("Activation of Layer 3: ", A[3].shape)
Activation of Layer 3:  torch.Size([1, 64, 214, 214])

print("Activation of Layer 4: ", A[4].shape)
Activation of Layer 4:  torch.Size([1, 128, 210, 210])

print("Activation of Layer 5: ", A[5].shape)
Activation of Layer 5:  torch.Size([1, 256])

print("Activation of Layer 6: ", A[6].shape)
Activation of Layer 6:  torch.Size([1, 2])

print("Activation of Layer 7: ", A[7].shape)
Activation of Layer 7:  torch.Size([1, 2])
```

At the last fully connected layer we get an activation score of (1x2) similar to the one we initially had for the maximum frame e.g. (7.28392,-828783). Since our focus is on the image parts that contribute to aggressiveness, we use a mask that makes the non-aggressive score 0. The resultant matrix will look something like this: (7.28392, 0).

This Activation matrix is then passed to the model in the backward propagation. For Relevance calculations at each layer we have a defined set of formulas.

Epsilon Rule (LRP-ϵ)

$$R_j = \sum_k \frac{a_j \rho(w_{jk})}{\epsilon + \sum_{0,j} a_j \rho(w_{jk})} R_k$$

A first enhancement of the basic LRP-0 rule consists of adding a small positive term ϵ in the denominator. The role of ϵ is to absorb some relevance when the contributions to the activation of neuron k are weak or contradictory. As ϵ becomes larger, only the most salient explanation factors survive the absorption. This typically leads to explanations that are sparser in terms of input features and less noisy. (Montavon et al., 2019)

The computation of this propagation rule can be decomposed in four steps:

$$\forall_k: \; z_k = \epsilon + \sum_{0,j} a_j \cdot \rho(w_{jk}) \qquad \text{(forward pass)}$$
$$\forall_k: \; s_k = R_k / z_k \qquad \text{(element-wise division)}$$
$$\forall_j: \; c_j = \sum_k \rho(w_{jk}) \cdot s_k \qquad \text{(backward pass)}$$
$$\forall_j: \; R_j = a_j \, c_j \qquad \text{(element-wise product)}$$

The code for the implementation of these mathematical equations is as follows:

```
for l in range(1,L)[::-1]:
    A[l] = (A[l].data).requires_grad_(True)
    if l <= 2:       rho = lambda p: p + 0.25*p.clamp(min=0); incr = lambda z: z+1e-9
    if 3 <= l <= 4: rho = lambda p: p;                      incr = lambda z: z+1e-9+0.25*((z**2).mean()**.5).data
    if l >= 5:       rho = lambda p: p;                      incr = lambda z: z+1e-9

    z = incr(utils.newlayer(layers[l],rho).forward(A[l]))  # step 1
    s = (R[l+1]/z).data                                    # step 2
    (z*s).sum().backward(); c = A[l].grad                  # step 3
    R[l] = (A[l]*c).data                                   # step 4
```

The model has 4 convolutional layer and then 2 fully connected layers.

For layer 5 and 6 which are fully connected we used:

```
rho = lambda p: p
incr = lambda z: z+1e-9
```

For layer 3 and 4, which are convolutional, the formula used is as follows:

```
rho = lambda p: p
incr = lambda z: z+1e-9+0.25*((z**2).mean()**.5).data
```

For layer 1st and 2nd convolutional layer, formula used is:

```
rho = lambda p: p + 0.25*p.clamp(min=0)
incr = lambda z: z+1e-9
```

On the 0th layer from where we passed the image, we will regenerate the image in the form of a heatmap. This particularly highlights the targeted aggressive image portions. The equations and code used for 0th layer are given below:

```
A[0] = (A[0].data).requires_grad_(True)

lb = (A[0].data*0+(0-mean)/std).requires_grad_(True)
hb = (A[0].data*0+(1-mean)/std).requires_grad_(True)

z = layers[0].forward(A[0]) + 1e-9                                  # step 1 (a)
z -= utils.newlayer(layers[0],lambda p: p.clamp(min=0)).forward(lb) # step 1 (b)
z -= utils.newlayer(layers[0],lambda p: p.clamp(max=0)).forward(hb) # step 1 (c)
s = (R[1]/z).data                                                   # step 2
(z*s).sum().backward(); c,cp,cm = A[0].grad,lb.grad,hb.grad         # step 3
R[0] = (A[0]*c+lb*cp+hb*cm).data                                    # step 4
```

Evaluation of the LRP Explainability Model

Heatmap Evaluation of Aggressive Behavior Detection(ABD) Model

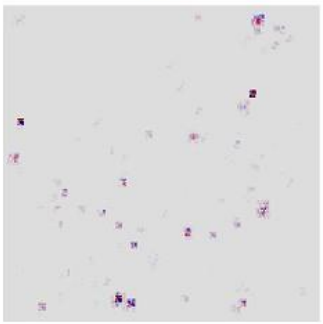
According to the research paper (Hägele et al., 2020), trained experts visually evaluated the heatmaps generated through LRP to check if the model had made accurate highlights. So, in our project we used a similar approach for the purpose of evaluating the resultant heatmaps.

For confirmation of our results, we collected a test set comprising of around 500 short video clips. From all these videos, we extracted the frames that had maximum probability of aggressive behavior. Then, we hand labelled the exact portions of the maximum frames that contained aggression. Also, we noted down the measurements of the labelled portion which would be used later for comparison with heatmaps generated by the LRP model. In this way, we could support our experimentation and results through comparison with hand labelled images.

LRP Model Results and Analysis of ABD Model

When we tested on the 500 videos test set and generated heatmaps, we reached the conclusion that the chosen aggressive behavior model may be overfitting. Since the number of layers in the classification model are low, and once trained over 30 epochs it gets over trained on the training set, therefore the model does not perform well when shown completely new data result in attempts to overfit. Hence, when passed through our debugging framework, the model did not perform well which indicates that there may have been flaws in the initial training – leading to the overfitting on test set.

```
utils.heatmap(R[0][0].sum(axis=0).cpu().detach().numpy(),4,4)
```



It is obvious from the heatmap that it gets confused while obtaining the exact pixels for aggression and highlighted some parts as blue and some as red. Therefore, it is proved that the

model is overfitting and is giving poor results on test data while it works perfectly fine with training data.

To confirm our results and findings of how well the aggressive behavior classification model was trained, we are using Confusion Matrix. This confusion matrix can be used as a guidance for further debugging the classification model.

Once heatmaps were generated for the maximum frames of the videos in the test set, it was observed that the heatmaps were highlighting parts of the images either different than the one we hand labelled in the same frames, some red patches(Aggressive) were smaller than expected and some were larger. Whereas, some images were entirely made red. All this shows that the model overfits because it is behaving entirely unexpectedly on test set. In our findings, we compare the actual aggressive part of image dimensions in hand labelled model with the LRP model generated heatmap highlighted aggressive regions and many test cases we found out a mismatch between the two. This shows flaws in the model training and accuracy.

Furthermore, through the test set we tried to evaluate the classification model's performance by heatmap generation on varying videos and images. Firstly, we tested the LRP framework on multiple different aggressive frames, for which the heatmaps generated were heavily flawed. Secondly, we used images similar to the ones in training set, only with minor changes e.g. keeping the same image background and removing only one of the persons in a frame with 2 people fighting—as the image background contributes to a major portion of the image, and while training, the model was extracting some feature with higher relevances, the heat map generated may still classify the frame as aggressive due to a high portion of the image that the model has previously learnt. This type of mis-classification on the basis of rigid learning is called overfitting and in the model we are using as our input, it is seen to be very prevalent.

For result evaluation two kinds of Confusion matrices were constructed. Confusion Matrix is used for evaluating the performance of a classification model. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. It basically consists of True Positive, True Negative, False Positive and False Negative values. False positive and false negative are also referred as error 1 and error 2 respectively.
Considering our dataset and model, these above mentioned errors are important in findings. These errors will actually support our finding that the trained model is overfitting and behaves poorly on unseen dataset. Until now, we're calculating our results and findings visually but for next iteration, we'll improve our findings and results and would present them properly after their comparison with metrics of LRP on images.

We are constructing two types of Confusion matrices, one is micro Confusion matrix and other is macro. In micro confusion matrix, we assigned a weight to the correct portion our heatmap constructed or highlighted for each individual image in the test set i.e., percentage of correctness in each frame, whereas in the macro confusion matrix, the normal average over the whole dataset is constructed i.e. how many images were wrong and how many were right.

The confusion matrix distinguishes the output of our data into four different categories; True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). According to our model all the frames which actually had aggressive behavior and the model also detected falls under TP. Where we get non-aggressive behavior and the model is also detecting nonaggressive is said to be a TN. For FP, a frame which is actually non-aggressive and the model detects aggressive. Similarly, in FN a frame is actually aggressive and but the model gives nonaggressive. In our LRP model which is intended to be a debugging framework, False Positive and False Negative are given more importance as they truly determine if the model is making correct predictions or not.

Confusion matrix:

	Actual	
Predicted	TP	TN
	FP	FN

Heatmap Evaluation of AlexNet & VGG-16

Again, since according to the research paper (Hägele et al., 2020), trained experts visually evaluated the heatmaps generated through LRP to check if the model had made accurate highlights. So, in our project we used a similar approach for the purpose of evaluating the resultant heatmaps.

Since Alexnet and VGG-16 are both trained on ImageNet dataset, we took them as base models for our debugging framework, so that a better comparison of the resultant heatmaps could be done. Furthermore, the Aggressive behavior detection model we were testing was created by university students, however VGG-16 and Alexnet are both renowned, well known and used classification architectures and if we tested our framework on them, it would give us more realistic and better grasp of the results as we already know model training and testing accuracies and datasets quite well.

Therefore, we took both these models and applied Layer-wise relevance propagation on them as well. Alterations in backpropagation formula applications on different layers and changes in dimensions while back propagating had to be done separately for both models due to differences in architecture and layers in both models. Once LRP was successfully applied to both models, and we had resultant heatmaps, we drew the conclusion that LRP is a generic technique that could be tailored and applied to any deep learning model and therefore, would be quite effective as a generic debugging framework for deep learning models.

Moreover, we roughly tested both the models by generating 30 heatmaps of different classes for each and calculating micro and macro scores for comparison. In macro scores, score 1 was given to model which correctly highlighted true class in final heatmap and 0 otherwise. In micro scores, the scale was 0-3, how well the model highlighted the true class. This evaluation was done by visual analysis by the team members. The resultant matrix is shown below:

MODELS	MICRO SCORE	MACRO SCORE
ALEXNET MODEL	21/90	0/30
VGG MODEL	69/90	30/30

Conclusion Iteration2

In this iteration we implemented LRP on a pre-trained aggressive behavior detection, VGG-16 and AlexNet model for explainability and debugging purposes. We used pre-defined formulas that needed to be implemented on each layer of our model to perform explainability. We have generated heatmaps for different images which highlight the determining features. However, we are still trying to optimize the finding and generate better quality results. We will keep on fine-tuning our debugging framework and optimizing ways to debug the models through this framework in the next iterations. Since, we have now discovered that the aggressive behavior detection model is not trained well and gives inaccurate results, we will be discarding it from further testing and will use only AlexNet and VGG-16 for formal testing in next iteration.

Iteration 3

At the end of iteration 2, we had made a full Layer-wise Relevance Propagation based testing and debugging framework that generated heatmaps as an end result for various models. Layer-wise Relevance Propagation was applied on VGG and AlexNet. Both these models are trained on ImageNet dataset and it was a good starting point to see the differences in their learning capabilities through heat map generations.

We found out through the informal testing we had done in iteration2 that the performance of VGG was far better than AlexNet in terms of blue and red parts in the heatmaps. The informal testing constituted of visually analyzing parts of the heat maps and scoring it based on whether it highlighted more prominent features of classes with red and least prominent features with blue.

The testing we had performed in iteration 2 was an informal methodology where we visually analyzed heat maps generated by both Deep learning and classified them into pass/fail categories to measure the overall performance of the models. We used an overall dataset of 30 images just to get a basic idea of the differences in heat maps.

Testing Debugging Framework for Deep Learning Models

In this iteration, we will be doing a more formal testing with a bigger dataset on both Alexnet and VGG-16. We will produce heat maps for different variations of the input data. Next we will find patterns in this data and make informed decisions as to why the model learns different patterns and how it makes classification predictions. This information can then be used to evaluate the overall performance of a model as it has been thoroughly tested on variations of training data. At the end we will compare performance of both Alexnet and VGG.

We applied LRP on AlexNet and VGG-16 to compare and debug their results along with understanding them as feature extraction models which resulted in the classification of an image.

Data set

The dataset we have collected is variations of the training classes of ImageNet dataset. For example, if a certain class in the dataset was bus and the model has been trained on images of bus, then in this phase, we will be testing on different image variations. We collected image

dataset of different classes such as Ostrich, Volcano, Car, Speedboat and Dog with total of 500 images and 100 of each class.

Dataset variations

In order to achieve best results, we added certain variations in our dataset. For that, every class was further divided into several sub classes such as Day, Night, Front, Animated, Black and White, and many more. We did this to see whether the models were able to classify certain features in different types of environment or not. Some variations were as follows:

1. Front images
2. Images on angles
3. Back images
4. Images in dark
5. Images in light
6. Sketches
7. Animations
8. Clear background
9. Cluttered background

The reason for introducing these variations in the test dataset is to particularly test against any biases in the training set and see how well the model has been trained for a particular class in the dataset. The model may be very good at predicting a bus but only of a certain type i.e. colored image from front. However, we suspect that the models will perform poorly when given images of the same class i.e. bus but with variations in angles, colors, or background. To test this, we collected a dataset of about 100 images for each class, produced heat maps, decided a margin for success, calculated evaluation matrices i.e. combined scores and judged overall performance.

Pre-Processing of Dataset

After completing our test dataset, we did some pre-processing of the images such as:

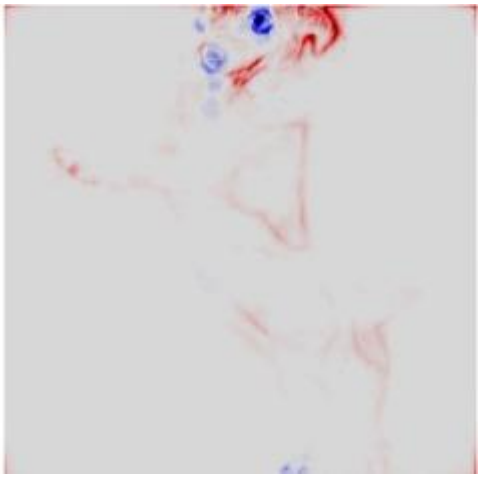
In first step i.e., read image, we store the path to our image dataset into a variable then we created a function to load folders containing images to arrays by importing the necessary libraries.

In next step i.e., resize image, we created a function called resize that just receives the images as a parameter and resized it to 200*200. In order to visualize the change, we saved these resized images into a new folder. We needed to resize our image dataset because some images were captured by a camera and they varied in size, therefore we established a base size for all images that went as an input into our algorithm.

Heatmap Generation and Feature Extraction

After data collection, we passed each image of the collected dataset through both models i.e. AlexNet and VGG. As already mentioned, VGG had produced much better in informal testing so it was important to produce heat maps for a larger dataset and analyze performance more formally to reach a solid conclusion based on concrete evident. For each image, heat maps were produced with both models. The method for heat map production remains same i.e. generation of activations at each layer of the model for the input image in the forward pass, and regeneration of the heat map in the backward pass. Technique for heat map production was Layer-wise Relevance production.

Additionally, in this iteration, we automated the whole process of heatmap generation by the deep learning models for all images of any particular class. You just have to provide the path of image folder of any particular class from ImageNet dataset, and all images in that class folder are automatically loaded into the algorithm, will go through different functions such as calculating activations and relevances, and all heatmaps are generated in a single loop as a result. We stored these heatmaps in another folder for better visualization. Shown below is a heatmap of dog class from AlexNet model:



Here, red represents the dominant features that the model has learnt thus showing positive relevance and blue represents the features that have negative relevance and aren't contributing towards the prediction.

Finding Patterns

Once heat maps for all images of the collected dataset were produced through both models, the next task was to find patterns. For a certain class e.g. bus and heat maps for all its variations i.e. front, back, dark, light – we wanted to find similarities, differences in the heat map regions highlighted with red and blue. Moreover, we wanted to see which parts were given more weight in the activation map vectors e.g. wheels, length, front mirrors and reason for these patterns.

Once these patterns are found, you can actually explain the process of image classification and why the machine makes certain decisions. The basis for classification and weightage to image portions can be more clearly observed once we analyzed the patterns in produced heat maps. Based on these observations, we learn whether the performance of the model is good or poor and where it needs to improve. Thus, our debugging framework actually ranks the performance of existing state of the art image classification models based on their heat maps and their decision making abilities.

Features extracted through VGG-16 model on car class:

Car Class – VGG16

CORRECTLY CLASSIFIED	
Tires	38/100
Headlight	18/100
Window	21/100
Roof	11/100
Windscreen	10/100
Backlight	10/100
Side-mirror	07/100
Number plate	08/100
Car seat	05/100
Bonnet	03/100

INCORRECTLY CLASSIFIED	
Clutered Background	10/100
Pixelated Heatmaps	05/100
Sharp close edges	03/100
Roof	07/100

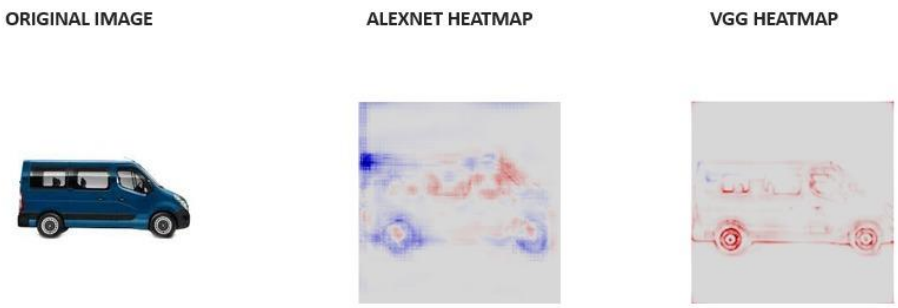
Features extracted through AlexNet model on car class:

Car Class - AlexNet

CORRECTLY CLASSIFIED	
Tires	10/100
Headlight	11/100
Window	04/100
Roof	10/100
Windscreen	05/100
Backlight	11/100
Side-mirror	05/100
Car door	03/100
Car seat	04/100
Bonnet	06/100

INCORRECTLY CLASSIFIED	
Clutered Background	13/100
Pixelated Heatmaps	35/100
Roof	07/100

Once we had heatmaps, we clearly observed that there were major differences in performances of both the models. As AlexNet was not making correct predictions and heatmaps were completely thrown off but VGG was trying to map exact feature variants onto the image to produce a much more accurate heatmap. Let's take an example of ostrich class in which its features like face, neck, legs, wings helped to classify it correctly. In case of AlexNet, heatmaps were quite blotchy and premature for most of the images and sometimes it got ostrich confused with other animals and objects. While in case of VGG, error rate was comparatively less and extracted a much wider and accurate range of features, even its nostrils were contributing to the correct predictions. This model is not 100% accurate and did make incorrect predictions on some images having darker background, sharp edged objects, etc.



Through our debugging framework, we came to conclusion that models lacked training on variety of background or darker images. So this technique will help up make quick decisions

on how training is to be done, what features are important, if hyper parameters need tuning or biases needs to be removed, etc.

Results and Discussion

Our results had some remarkable patterns that if looked into more deeply could be a source of improvement for both VGG and AlexNet's classification accuracies.

To elaborate, often times, the models predicted the images very accurately but only if they had certain features like RGB coloring, front angle, sharp edges, light background, daylight etc. Therefore, if we showed the model images belonging to the same class but with variations like side angle, dark, light edges, sketches, no color – the performance reduces manifold. This could indicate that the model was either trained on a biased dataset or was not giving due weight to the most important feature which is edges. Edges are the one feature that define any classification problem but the models seemed to be giving more relevance to other features like color, background and picture mode. Therefore, the predictions were not as accurate with variation images.

This is a novel idea and it can be a breakthrough in the field of AI and Deep Learning because the existing techniques of hyper parameter tuning and training often took a lot of time, and yet sometimes when it comes down to debugging a model, you never know if you are on the correct path or not. This explainability debugging technique is transparent and hence provides us with immediate feedback on what's happening inside the black box.

Applications of LRP debugging framework

This model can be used at training time to better understand the learning done by the deep learning models and introduce improvements at the end of a few epochs if required. The heat map analysis technique can be used along with train/ validation set accuracies to mold model's performance and prevent learning with a bias.

As with the passage of time, AI is taking over the world and since we can't always trust its predictions. So it is necessary to generate a debugging model for AI which gives us transparency and an edge to trust AI decisions. Nowadays, data scientists spent days or weeks on training of their models and then get to know what their model has learnt, what hyper parameters needs to be tuned, what changes does training dataset require. And then they have to repeat the whole process again from scratch to do retraining of their model, thus ended up wasting a lot of time and resources.

Conclusion and Future Work

DeepBug is a novel idea and it can be a breakthrough in the field of AI and Deep Learning because the existing techniques of hyper parameter tuning and training often took a lot of time, and yet sometimes when it comes down to debugging a model, you never know if you are on the correct path or not. Existing techniques to debug the trained models are OLS regression, Gaussian process, directly adjust model parameters, improve training data, influence training process, etc. All these techniques require re-training of your model which consumes and wastes a lot of resources and time. But our solution of explainability debugging technique is transparent and hence provides us with immediate feedback on what's happening inside the black box, thus end up saving a lot of time and resources.

For future work, DeepBug can be converted into an automated system to generate a generic heatmaps for any given class instead of visualizing each image individually to see learning of the model. For this purpose, image datasets need to be pre-processed by resizing them to same base and moving the object in center at same coordinates in all images. And when these images get through the function of generating heatmaps, all resultant heatmaps will be stacked over each other to generate a generic heatmap highlighting all the features having positive relevance.

Papers Referenced

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, *10*(7), e0130140. https://doi.org/10.1371/journal.pone.0130140

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Hägele, M., Seegerer, P., Lapuschkin, S., Bockmayr, M., Samek, W., Klauschen, F., Müller, K.-R., & Binder, A. (2020). Resolving challenges in deep learning-based analyses of histopathological images using explanation methods. *Scientific Reports*, *10*(1), 6423. https://doi.org/10.1038/s41598-020-62724-2

Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. *ArXiv:1708.08296 [Cs*,

Stat]. http://arxiv.org/abs/1708.08296

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (n.d.). *GradCAM: Visual Explanations From Deep Networks via Gradient-Based Localization*. 9.

Sun, J., Lapuschkin, S., Samek, W., & Binder, A. (2022). Explain and improve: LRP-inference fine-tuning for image captioning models. *Information Fusion*, *77*, 233–246. https://doi.org/10.1016/j.inffus.2021.07.008

Tjoa, E., & Guan, C. (2020). A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/TNNLS.2020.3027314

Xing, Y., Lv, C., Wang, H., Cao, D., Velenis, E., & Wang, F.-Y. (2019). Driver Activity Recognition for Intelligent Vehicles: A Deep Learning Approach. *IEEE Transactions on Vehicular Technology*, *68*(6), 5379–5390. https://doi.org/10.1109/TVT.2019.2908425

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K.-R. (2019). Layer-Wise Relevance Propagation: An Overview. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Eds.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Vol. 11700, pp. 193–209). Springer International Publishing. https://doi.org/10.1007/978-3-03028954-6_10