Erasmus+

COSI Computational Colour and Spectral Imaging

Master in Computational Colour and Spectral Imaging (COSI)

UNIVERSITY OF EASTERN FINLAND · UNIVERSIDAD DE GRANADA · UNIVERSITÉ DE LYON · UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE · NTNU Norwegian University of Science and Technology

Explainable AI for RGB to HyperSpectral CNN Models

Master Thesis Report

Presented by

Hamzeh Issa

and defended at the

Norwegian University of Science and Technology

September 2023

Academic Supervisor(s): Dr. Steven Yves Le Moan

Jury Committee:

1. Dr. Damian Muselet, University of Jean Monnet, France
2. Prof. Javier Hernández, University of Granada, Spain

Submission of the thesis: 10$^{\text{th}}$ August 2023
Day of the oral defense: 4$^{\text{th}}$ September 2023

Abstract

HyperSpectral Imaging (HSI) is a vital tool to many industries and fields. It is however financially expensive, time consuming, and in need of dedicated hardware. Lots of research was dedicated to find alternatives to traditional HSI systems. One of the most promising ones is RGB to Hyperspectral reconstruction. These models are usually CNNs that take in a single RGB image and estimate the hyperspectral image for the same scene (in the visible range). Such models can dramatically cut on costs and time needed to acquire a hyperspectral image given the availability and ease of acquiring RGB images.

However, to fully adopt such models, we need to establish trust in them (or distrust). To do that, we need to understand and explain how these models work on a fundamental level at least. This is especially the case because these models deal with a highly ill-posed problem of mapping only 3 RGB bands into a much larger number of bands (typically 31) to perform this estimation. Users do not have any evidence of how these models actually do that and how they are able to estimate the illuminant of the scene to avoid metameric effects and how they perform the 'one-to-many' mapping involved. In this thesis, we work on filling this major gap. We take 7 of the most prominent RGB to hyperspectral reconstruction models and apply many explainable AI (XAI) methods to try to understand how they work. We classify these models based on the different ways they perform the reconstruction. We establish points of failure where some or all models cannot perform as expected. We establish their spatial feature area in the input image. We try to find what kind of parameters and features they use and where in the network they use them. We present a theory on how they do illuminant estimation and present supporting evidences for that theory. Finally, we bring all tests together and try to break down these models into more simple sub-models that could be replicated by simpler explainable equivalents. We also introduce novel modifications to existing XAI methods that allows them to be used in any hyperspectral model explainability project in the future.

The outcomes of this work support that these models work in an intelligible manner. Meaning that they could be understood and equated by other explainable models. However, these models cannot be trusted all the time, since the work shows that they fail consistently under certain conditions. This work does not fully explain these models since some aspects are still unclear, but it does explain many important parts and paves the way for a clearer understanding of these networks.

Acknowledgment

To the One I cannot name, I owe everything I am.
To my parents, Nariman and Jamal, I owe everything I have.
To those close enough to read this, I owe everything I have become.
Thank you.

Acronyms

HSI: Hyper Spectral Imaging
RGB: Red Green Blue
SCI: Snapshot Compressive Imaging
CASSI: Coded Aperture Snapshot Spectral Imaging
XAI: Explainable Artificial Intelligence
MSA: Multi-headed Self Attention
SVM: Support Vector Machines
CNN: Convolutional Neural Network
CAM: Class Activation methods
MRAE: Mean Relative Absolute Error
RMSE: Root Mean Square Error
PSNR: Peak Signal-to-Noise Ratio
SAM: Spectral Angle Mapper

Contents

1 Introduction **1**
1.1 Contributions 4

2 Literature Review and Background **5**
2.1 Spectral Reconstruction from RGB 5
2.2 Explainable AI 14
2.2.1 Explainability as a concept 14
2.2.2 Different Types of Explainability 15
2.2.3 The Main Focus of This Work 18
2.2.4 Visual Explainability in Neural Networks 22
2.2.5 Feature Visualization 34
2.2.6 On Explainability for Regression Models 38

3 Methodology and Results **41**
3.1 Preparation 42
3.1.1 Error Metrics 42
3.1.2 The Dataset 44
3.1.3 The Models 45
3.2 Error Maps 46
3.3 Saliency Maps 49
3.3.1 Concept 49
3.3.2 Method 50
3.3.3 Results 51
3.3.4 Local Models 51
3.3.5 Partially Local Models 53
3.3.6 Global models 53
3.3.7 Edge saliency 55
3.3.8 Image Edge saliency 58
3.3.9 Single Image Analysis 62
3.3.10 Test Conclusion 62
3.4 Box Tests 64

3.4.1 Concept . . . 64
3.4.2 Results . . . 68
3.4.3 Test Conclusion . . . 70
3.5 Activation Maximizers . . . 70
3.5.1 Concept . . . 70
3.5.2 How it was applied . . . 74
3.5.3 Results . . . 77
3.5.4 Testing The Theory . . . 81
3.5.5 Test Conclusion . . . 86
3.6 Parameters Sensitivity . . . 90
3.6.1 The Parameters . . . 90
3.6.2 Concept . . . 91
3.6.3 Results . . . 92
3.6.4 Test Conclusion . . . 96
3.7 Illuminant and Color Tests . . . 97
3.7.1 Concept . . . 98
3.7.2 Results . . . 98

4 Discussion **101**
4.1 Trust, and Points of Failure . . . 101
4.2 Illuminant Estimation . . . 102
4.3 Different Models . . . 103
4.4 Big Picture . . . 104
4.5 What Was Achieved . . . 106
4.6 Conclusion . . . 106
4.7 Future Work . . . 107

A Appendix **109**
A.1 Activation Maximizers for Global Models . . . 109
A.2 More Error Maps . . . 109
A.3 Thin Edge Test . . . 112

Bibliography **115**

List of Figures **131**

List of Tables **137**

1 Introduction

HyperSpectral imaging (HSI) is a widely used technology in most applications and industries today. It proved to be highly effective in distinguishing various materials and components in a scene. Those components would have otherwise been very difficult to distinguish in a regular RGB image since the hyperspectral image provides a larger number of bands each with certain characteristics and at which only certain elements and compositions are absorbed. Applications of HSI spread to a large variety of fields including earth observation Patro et al. (2021); Mücke et al. (2019), food and agriculture Elmasry et al. (2012); Dale et al. (2013), medical applications Lu and Fei (2014); Calin et al. (2014); Fei (2019), anomaly detection Kang et al. (2017), tracking Van Nguyen et al. (2010), cultural heritage Picollo et al. (2020); Cucci et al. (2016), and many more.

All of these applications rely on HSI to produce accurate images of specific bands. When it comes to medical applications for example, having accurate images is vital to avoid mistakes in diagnosis or surgery planning. Accuracy here means having as close a representation of the spectra of imaged materials as possible, with as fine spectral and spatial resolution as possible. The same applies to all other fields with multiple levels of accuracy requirements.

Sometimes authors are not clear about their definition of HSI and the difference between it and multi-spectral imaging for example. So, for us to be clear from the start we adopt the definition provided by Christophe et al. (2005) who defined a benchmark for HSI. So, what characterizes an HSI system here is that it has a large number of bands (typically in the range from tens to hundreds of bands and larger than multi-spectral). Also, the width of these bands is narrow (between 10 to 20 nm) and that these bands have to be contiguous (touching or overlapping). These are the main distinctions between it and multi-spectral imaging. Figure 1.1 shows that difference visually. We also clarify here that the bands are within the visible range (400 - 700 nm) with a typical number of bands mostly used in literature = 31 bands. We will discuss this further in a later section.

However, a severe limitation of HSI is the expensive and time consuming requirements of its cameras. HSI cameras use diffracting elements and many other optical instrumentation that raise the price of a typical HSI camera to a level

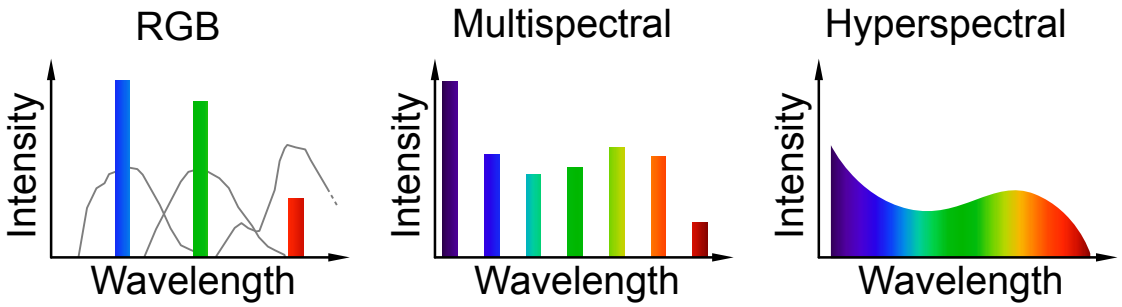


Figure 1.1: *Difference between RGB, multi-spectral imaging, and hyperspectral imaging Ozbulak (2019).*

where only professional industries can afford it. The price of an HSI camera varies significantly depending on the specification, but it is usually in the range of tens of thousands (USD) Salazar-Vazquez and Mendez-Vazquez (2020) which makes it expensive to applications with low cost requirements. Also, most available cameras require a high level of expertise or training to use them. All of that limits the availability of HSI and consequently limits the applications HSI can be used for.

Many works were dedicated to tackle this issue. Some would introduce low cost HSI systems that include certain compromises where performance is reduced in favor of the lower cost Stuart et al. (2021); Salazar-Vazquez and Mendez-Vazquez (2020). Other systems use alternative technologies that have a reduced price like snapshot compressive imaging (SCI) Wagadarikar et al. (2008). We will discuss this technology further in the literature review.

Of the most important and effective ways to deal with this issue is hyperspectral reconstruction from RGB images Chakrabarti and Zickler (2011). This is because RGB images are readily available and easy to acquire since they can be captured by virtually anyone with an RGB camera. If this method can be properly validated and accepted, it could even let go of the need of expensive HSI cameras and open the door for many more industries to use the full potential of this technology.

However, the problem of building a spectra (of typically 31 bands) from the 3 RGB bands is severely ill-posed. A large amount of information is lost when the full spectrum is sampled at only 3 bands, and that information is needed to reconstruct the spectra back again in this process. So, there is a need for learning an inverse function which might require extra information that would help the algorithm to do the main tasks of reconstruction. To perform reconstruction from RGB, the model needs to perform at least 2 main tasks Zhang et al. (2022):

The first is removing the effect of the illuminant, or equivalently, learning the illuminant. This is because RGB images naturally include the illuminant in their data, which heavily effects the RGB value. This illuminant has to be estimated so it can be discarded if we need to gain a true reflectance spectra of the material we

are imaging. Even if we only require the radiance spectra, the illuminant still has to be estimated to differentiate its effect from the effect of materials in the image and to be able to reconstruct these materials' spectra.

The second is up-sampling, or more precisely, the process of shaping the material spectra (distributed over 31 bands) from the 3 RGB bands. This involves a highly ill-posed interpolation where the shape of the spectra needs to be deduced from the 3 RGB values.

There are currently a significant number of models that achieve the RGB to hyperspectral reconstruction task. Most of these (and the best performing ones) are convolutional neural networks Arad et al. (2022). So, the natural question is can we trust these models to actually replace current HSI systems and hardware? This question is especially difficult since CNNs are black boxes. We do not understand how these models work, except for some minor details and features Li et al. (2020). As for the main tasks mentioned above, we do not know what is the 'workflow' within the network that performs these tasks. We do not even know what kind of parameters and features from the RGB image the networks use in the reconstruction.

This prevents us from trusting these models and from fully depending on them, since we cannot be sure they would work in all applications. We cannot know if they would work on another dataset or in other circumstances. We only have performance metrics (will be discussed later) to evaluate them. These metrics however do not provide any reliability evaluation that guarantees they would work on other conditions. To do that, we need to explain/understand these models and rationalize their workflow and the features they depend on. Then we can be closer to establishing trust in these models.

As we will see in the next chapter, there was never any work that explains CNN hyperspectral reconstruction models in general, and RGB to hyperspectral models specifically. In fact, there is barely any work on explaining CNNs that are not specialized in classification. This thesis is the first to work on this important issue and the first to address this large research gap. To do that, we use multiple explainable AI (XAI) methods and present new ones to study and explain these models. We also try to classify them based on the results we achieve and break them down into more explainable workflows.

This report will first give an overview of both RGB to hyperspectral models as well as explainable AI and how it can be used for our purpose in the 'Literature Review and Background' chapter. During that chapter we also establish the research question further since we first need to understand the concept of explainability in deeper detail and find where our research fits exactly. After that we go through our explainability methods and tests in the 'Methodology and Results' chapter. We combine both methodology and results for each test or method since we need the results of each test to justify using the other and keep track of the context since

this is mostly an investigative research. Finally, in the 'Discussion' chapter, we discuss the results in a general perspective and combine them together to have a bigger picture. We also present there our conclusion and possible future work.

Disclaimer: No software tool (like ChatGPT or Grammarly) was used in writing this report (even grammatical correction tools were not used). The use of ChatGPT was restricted to only making very limited miscellaneous programming functions (not the main parts of the code).

1.1 Contributions

The main contributions of this thesis can be summarized as follows:

- Establishing the details of the spatial area used by the RGB to hyperspectral reconstruction models and how it differs from one model to another while showing its impact on the model's workflow and performance.
- Classifying the models based on their spatial observed area into local, partially-local, and global models.
- Establishing points of failure for the majority of these models and where users cannot trust them to perform reconstruction.
- Presenting a theory on how these models perform illuminant estimation and correction within the network and providing multiple evidences for it (albeit non-conclusive).
- Examining the models' reaction to multiple important parameters and establishing where within each network these parameters are processed and how this is connected to each model's structure.
- Presenting a general workflow to a type of reconstruction models (local models) and breaking them down into more explainable equivalents.
- Presenting multiple novel modifications to explainability methods that can be used on these models as well as any regression based or reconstruction based model. These include a modification to the existing Guided Back-propagation approach Springenberg et al. (2014), a new feature visualization approach called Activation Maximizers, and a parameters sensitivity approach.
- Analysing illuminant and color sensitivity for these models and presenting ways on how to further study them.

2 Literature Review and Background

Our research is concerned with the interaction of 2 main fields of literature: the first is RGB to hyperspectral CNN models, which falls under more general super-resolution models (we will discuss where it fits in literature exactly). And the second is the field of explainable AI (XAI), more specifically, XAI concerned with the explainability of neural networks. In the following sections we will introduce the most prominent works done in these fields and see how our research topic is the result of an intersection of 2 branches of research representing each of these fields. We will discuss the general frameworks and their more specific cases until we find how the need for our research topic arises naturally from both the lack of research in that area and its vital role in the studies related to it.

2.1 Spectral Reconstruction from RGB

To address the issue of affordability and time constraints of conventional hyperspectral image acquisition, many efforts were directed to finding alternatives for using a fully-pledged hyperspectral camera. Of these alternatives was snapshot compressive imaging (SCI), where the hyperspectral information is compressed into a single 2D snapshot that includes both spatial and hyperspectral dimensions using different methods Cao et al. (2016); Du et al. (2009); Llull et al. (2013); Wagadarikar et al. (2008, 2009); Lin et al. (2014).

A type of SCI systems that uses coded apertures was especially popular Wagadarikar et al. (2008); Meng et al. (2020); Cai et al. (2022a). An example of such system is the Coded Aperture Snapshot Spectral Imaging (CASSI) shown in figure 2.1. As can be seen in the figure, the HSI block with width W and height H and number of wavelengths N_λ is first modulated (with an element-wise multiplication) by the coded aperture which corresponds to a physical mask M^*. The result $F^{'}$ is then passed through a disperser (prism-like) which has the effect of shearing the cube along the y-axis where d is the shifting step making $F^{''}$. Finally, we have the 2D compressed measurement snapshot captured at the detector array Y with size $H \times W + d(N_\lambda - 1))$. We will not go into detail here since this type of acquisition

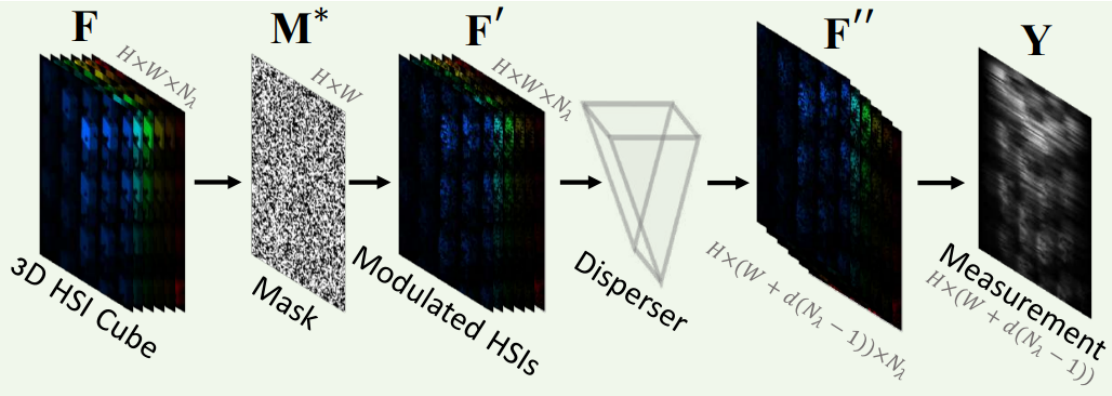


Figure 2.1: *Coded aperture snapshot spectral imaging (CASSI) Cai et al. (2022a).*

is not our target but more details can be found in the cited works.

Although SCI systems are considerably more affordable than conventional hyperspectral acquisition, they are still quite expensive for normal consumers since even SCI acquisition systems that are considered affordable, range in cost between 10K and 100K USD Cai et al. (2022b). That makes them less than optimal in tackling the time and cost issue of HSI.

Also, there is a category of models that depend on designing a specific system to be able to acquire the spectral image, and these systems are often complicated and hard to build Goel et al. (2015); Oh et al. (2016); Takatani et al. (2017). Although it is still using an RGB camera, these models require multiple cameras and/or multiple time-multiplexed illumination sources, and tube reflectors. All of them used to acquire the images under a large range of possible conditions that make up for the lost information inherent to the RGB capturing scheme compared to hyperspectral acquisition. Clearly, this requires a lot of preparation, knowledge, and instrumentation not readily available for consumers.

These limitations have given rise to models that directly reconstruct hyperspectral images from RGB ones based on the idea that hyperspectral images posses a much smaller intrinsic dimensionality than their number of bands, which means that they can be further compressed while maintaining the same information. So, with the right prior, the information gap between hyperspectral and RGB can be bridged Chakrabarti and Zickler (2011). Although this problem is highly ill-posed, since there is an inevitable loss of data expected with the RGB projection, many models are still able to perform reconstruction using learning based approaches since the mapping of such projections is highly non-linear. There are many conventional models that reconstruct hyperspectral images from RGB images depending on special priors and features in the hyperspectral domain Aeschbacher et al. (2017);

Arad and Ben-Shahar (2016); Jia et al. (2017); Parmar et al. (2008); Robles-Kelly (2015). These data-driven models use a variety of features that include sparsity of information between bands (high similarity between different bands) in hyperspectral images, dictionaries to correspond spectra and RGB components, or shallow learning models. However, these models are always restricted to certain types of data and do not generalize well to more representative datasets Cai et al. (2022b).

So, using the success of deep learning approaches in many image editing, restoration and reconstruction applications like in Cai et al. (2021); Deng et al. (2022); Hu et al. (2021a,b) and many others, research is going toward exploiting Convolutional Neural Networks CNNs for that task and to learn the underlying projection and mapping functions involved in the RGB to hyperspectral conversion problem.

One of the first breakthroughs in that regard was in developing end to end models specific for that task Galliani et al. (2017) along with Xiong et al. (2017) who developed a unified deep learning model HSCNN that can reconstruct hyperspectral images from RGB images as well as other compressive methods we mentioned before like SCI. This model achieved state of the art performance at the time in 2017 on the ICVL dataset Arad and Ben-Shahar (2016) and paved the way to other advancements in that field that modified it further. A problem with HSCNN was that it contained an explicit spectral up-sampling operation at the beginning of the network. This up-sampling is dependent on the existence of a spectral response function defined before hand that maps the integration of hyperspectral data into RGB images and has to be known in addition to the RGB images. This requirement combined with HSCNN's failure to improve its performance with increased growth and depth (it does not gain better performance with more data and more depth) limited the applications of HSCNN, until HSCNN+ was introduced Shi et al. (2018).

HSCNN+ along with a mid-way network HSCNN-R (R for residual) were introduced by the authors of HSCNN and provided multiple models as modifications to HSCNN. The authors replace the up-sampling stage with a convolutional layer relieving the model from the spectral response function requirement, then replacing regular convolutional layers with residual blocks forming HSCNN-R (Residual). Then, they modify the neural network further to include a fusion scheme and a densely connected structure with a network called HSCNN-D (Dense). The concatenation included in this new architecture is claimed to explicitly add information from the channels making it more suitable for reconstruction. That modification improved on the performance and achieved state of the art on the NTIRE dataset for the year 2018 Timofte et al. (2018). The architectures of HSCNN, HSCNN-R, and HSCNN-D are shown in figure 2.2.

These models defined a common practice for many other models after them that used the same idea of having convolutional layers with local and global residual

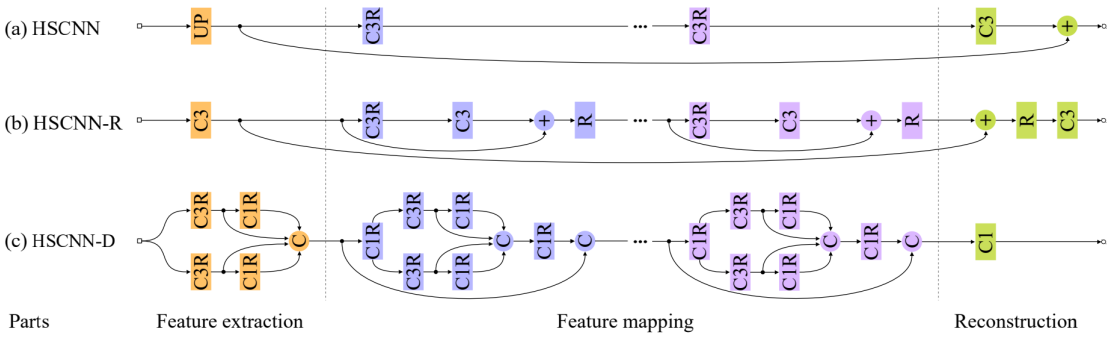


Figure 2.2: *Architectures of the baseline HSCNN (a), HSCNN-R (b), and HSCNN-D (c). Quoting from Xiong et al. (2017): The "C" with a rectangular block denotes a convolution, and the following "1" and "3" denote the kernel size (i.e., 1 × 1 and 3 × 3 respectively). The "R" represents a ReLU activation function. And the "C" with a circular block denotes a concatenation Xiong et al. (2017).*

connections to perform reconstruction.

Further advancements were done in Stiebel et al. (2018) where a U-Net architecture with some modifications was first introduced in this application. U-Nets are particularly interesting to our task because they have no specific pipeline for reconstruction. Unlike HSCNN+ models previously discussed (which had parts for each task like feature extraction, mapping, and reconstruction) the authors of the U-Net models made no effort to assign specific tasks to specific parts of the architecture which makes it a fully closed black box that depends solely on the learning of convolutional layers and generic capabilities of CNNs. This makes them in great need of some level of interpretation and shows that even generic unperturbed CNN models can learn the reconstruction task.

Another U-Net we refer to in this work as MultiScale was also used by Yan et al. (2018) where the U-Net was scaled and residual connections were taken at each scale until the bottleneck. These residual connections help maintain features from fading when the flow of the image goes near the bottleneck of the U. In other words, some features might be lost due to the strong compression near the bottleneck, so these residuals maintain these features and feed it to the upper-going part of the U. Figure 2.3 shows that architectures and the residuals connecting both arms of the U shape.

These two networks proved the applicability of U-Nets and auto-encoder blocks in the reconstruction task, but they had lower performance compared to other dedicated networks like HSCNN+. Also, Zhang et al. (2020) built a neural network that depends on learning pixel-wise mapping using function-mixture blocks that increase the flexibility of the network. This is important since this was the first

Figure 2.3: *The U-Net architecture used in Yan et al. (2018).*

network to have a variable receptive field that changes depending on context and spatial location. This is done by including multiple sub-nets in the architecture, where each sub-net has a different receptive field from the others, and then there is a mixing function block that chooses the combination of sub-nets and a corresponding receptive field for each pixel. That allows each pixel to have a specific receptive field based on the features around that pixel. Figure 2.4 shows the difference between typical fixed receptive fields (left) and adaptive receptive fields used in Zhang et al. (2020) (right) and how the chosen sub-nets differ depending on the pixel's features and surroundings, which correspond to a specific receptive field. All previous models had a fixed receptive field and learned a single mapping function for all pixels, while this one learns a specific mapping for each spatial pixel achieving better results. It is worth noting that this way of changing the receptive field is quite explained in the architecture. So, although this model is not explainable (like all the others) this added feature does not 'decrease' the level of explainability of this model compared to the others.

We need to clarify here that the U-Net we mean is not the typical classification network but the encoder-decoder architecture that compresses the input into the latent space (encoder part) then decompresses it in the decoder part (making the full U shape) to make either a similar result as the input (what is known as auto-encoder) or -like in our case- another version with modified features which is here the larger number of bands. There is an issue with models based on the encoder-decoder architecture which is the loss of information inherent to the deeper latent space when the network gets closer to the bottleneck especially with max pooling used as the down-scaling block for the size of the feature map. Although

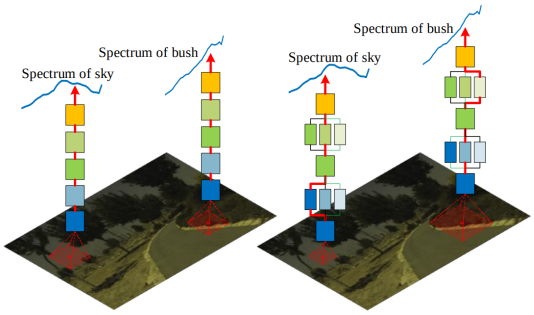


Figure 2.4: *Regular receptive fields (left) and adaptive receptive fields used in Zhang et al. (2020) (right).*

most of these models use residuals connecting the encoder to the decoder to tackle that, the encoder path still loses a lot of information until it reaches the bottleneck where most of the convolutions are done (because of the compact features and efficient computation). This means that lots of the original pixel information could be lost in such architectures Zhao et al. (2020).

Further modifications mainly included adding architecture modifications along with specific blocks for tackling problems with previous models. One of these issues was the difference in importance between each of the spectral bands and that models treated all of the bands as the same regardless of the amount of information in them. This is tackled in the Adaptive Weighted Attention Network (AWAN) Li et al. (2020) where dual residual attention blocks were added along with a trainable adaptive weighted channel attention that learned weights for each channel and multiplied them by the original channels giving them different importance scores, making the network work better with channel wise dependencies. Also, this was the first work that tried tackling the limitation of long range dependencies, since all networks so far were spatially local and constricted to a local receptive field around the pixel being reconstructed. This was suspected by the authors to be limiting the capability of the network since it limits the amount of information the network can access. Although models might not specifically need long range dependencies for reconstruction, the authors proved that employing them can improve performance, which makes sense since the model will have access to more spatial information while reconstructing a given pixel.

This was addressed using a patch level second order non-local module. Although it did improve the performance, that solution did not achieve full spatial globality since the receptive field was increased but to a certain extent. Also, second order modules require large amounts of memory Cai et al. (2022b).

Other similar ideas were introduced in the Hierarchical Regression Network (HRNET) Zhao et al. (2020). This neural network also uses a global attention block

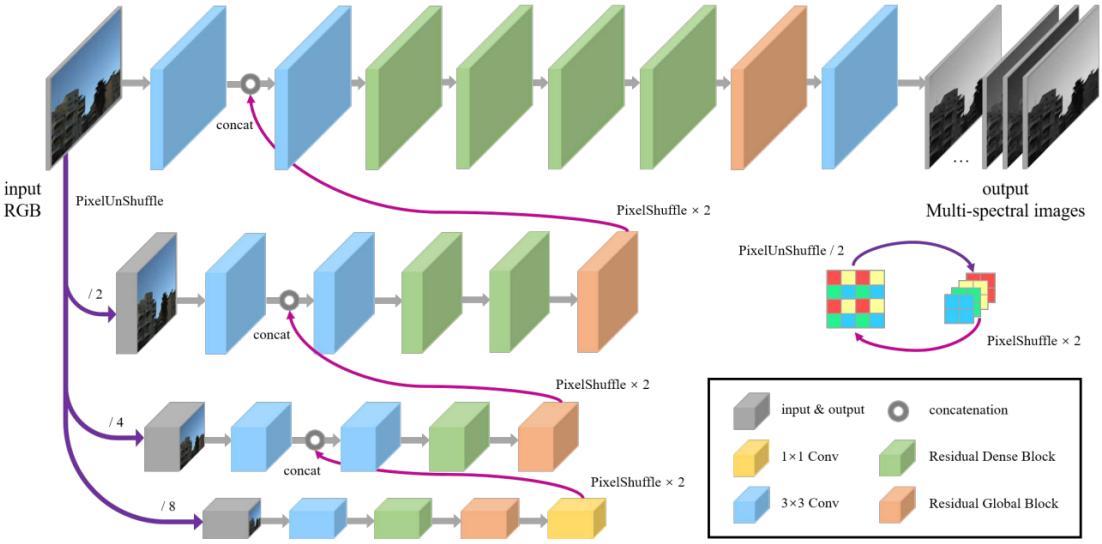


Figure 2.5: *HRNET Architecture with its 4-level hierarchy Zhao et al. (2020).*

that can increase the receptive field. It also has dense residual blocks for artifact removal in real world images. This network -shown in figure 2.5- consists of a 4-level hierarchy where each level is a division by 2 on the spatial dimension of the input done by a pixel shuffle block. This block was introduced to replace max pooling or convolution by stride blocks commonly used to do the downsizing. More specifically, it was made to address the problems with information loss common in U-Net architectures mentioned before. The pixel shuffle block interleaves neighbouring pixels into different (smaller in spatial size) feature maps, conserving the information of all pixels. For the reverse operation, the pixel un-shuffle block does the opposite, merging small feature maps into a large one. With these improvements, the HRNET won the first place in the NTIRE challenge 2020 for the real-world images track and ranked 3rd place for the clean images track Lugmayr et al. (2020). The clean image track has images with clear camera function applied to their reflectance spectra, without any camera noise models or real-life considerations. That relieves the images from the noise, jitter, and lens problems that accompany real world images. In later issues of the NTIRE dataset like that of 2022 Arad et al. (2022), that track was removed since most models started performing far better on it than on real images as it became not much of a challenge for advanced models. Also because the real world track is much more realistic and useful.

An important advancement was made in Cai et al. (2022a) with the Mask-guided Spectral-wise Transformer (MST), which is a model used for reconstruction from snapshot compressive imaging (SCI) images (discussed before), then the same model was adapted to direct hyperspectral reconstruction from RGB images which

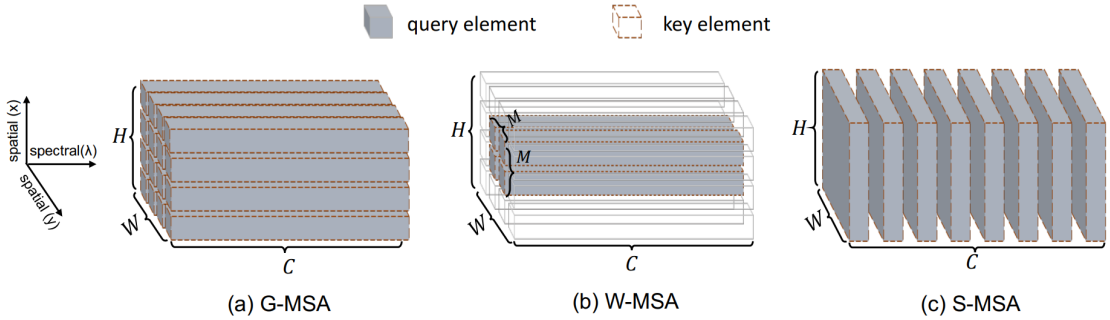


Figure 2.6: *(a) Global Multi-headed Self Attention (G-MSA). (b) Window MSA (W-MSA). (c) Spectral MSA (S-MSA) Cai et al. (2022b).*

was called Multi-stage Spectral-wise Transformer MST++ Cai et al. (2022b) where 2 problems in existing models were discussed: the first is that these models do not capture long range dependencies, since as we discussed, most models have a local spatial receptive field. While the second is their inability to exploit the self similarity or correlation evident between hyperspectral channels since although HSIs are spatially sparse, they are highly self similar in the spectral dimension. Of course, not all bands are correlating to all others, but some bands have high correlation with each other. Although this feature was exploited in works like Arad and Ben-Shahar (2016), it was not used in deep CNNs since most CNNs that use attention blocks have *spatial* attention blocks. This neural network uses transformers and spectral wise multi-headed self attention blocks.

Figure 2.6 taken from Cai et al. (2022b) shows 3 types of Multi-headed Self Attention (MSA). The first one (a) is Global MSA (G-MSA) where all pixels are sampled as both query and key elements. The second (b) is Window MSA (W-MSA) where self attention is only calculated inside spatial windows. The third (c), -which is the one used by the authors- is Spectral MSA (S-MSA) where the attention is calculated along the spectral dimension (each channel/band is treated as a token).

To put it simply, the same idea of attention blocks was used, but instead of having the attention applied to the image in the spatial dimension, it was applied in the spectral dimension (spectral wise), while also keeping the residual connections similar to the ones used in previous models. This produces a spectral wise calculation and a global receptive field reaching long range spatial dependencies, with a linear complexity requirement. This achieved much higher accuracy with less parameters required in the network, achieving state of the art for the year 2022 on the NTIRE challenge. It is also the current state of the art as of writing this text.

Also, some models were adapted from other tasks like reconstruction from SCI to reconstruction from RGB since the output is the same (reconstructed spectra).

One of the most successful adaptations is High-resolution Dual-domain Learning (HDNET) Hu et al. (2022). This network employed both spectral and spatial attention blocks which also theoretically allows it to get long range dependencies. However, these spectral attention blocks are different from the ones used in MST and MST++ in the sense that they only contribute half the attention (the other half is spatial attention). This model focuses on obtaining high spatial resolution spectral images.

Another model worth mentioning is Half Instance Normalization Network (HINET) Chen et al. (2021) which was adapted from an image restoration model by the same name. This model also employs an encoder decoder architecture, but it has special half instance normalization blocks in 5 levels of encoder scales. Instance normalization is similar to batch normalization but applied for each instance individually, instead of a whole batch. The authors argue it gives better performance with low level tasks (tasks where the value of each pixel in the image matters significantly). This is because there is a high variance between small patches in input images. This is the case for both image restoration and RGB to hyperspectral reconstruction.

Instance normalization in this case can maintain the same normalization procedure in both training and testing. It can also adjust the mean and variance of features without being affected by the batch dimensions (which is the case in batch normalization). They introduce half instance normalization blocks where half the input is normalized and half is kept. The reason it is cut in half is because the authors find with experimentation that this keeps features in shallow layers intact. As for the decoder part, they use a residual block that simply has a residual connection. Figure 2.7 shows the half instance normalization block (HIN block) and the residual block (Res block). Five of each are used in the encoder and decoder respectively.

To sum up this section, there are many works that try to tackle the issue of cost and time requirements of HSI, and many of those apply HSI reconstruction from RGB images which is the most promising way since it has little to no cost on the end user. However, these models seem to vary in their architectures, dependencies, and performance. Most models though have similar concepts that include either having an encoder-decoder structure, having residual connections in an otherwise sequential model, or having extra blocks and connections that do a specific task. To try to explain these models, we need to take care of their architecture and their context. Because although some parts of some models are dedicated for a specific task which is explainable, non of these explainable parts actually does the main reconstruction task. In the following section we define our concept of explainability and go through works of literature that tried to explain other models from other applications and see how we can use them to our application.

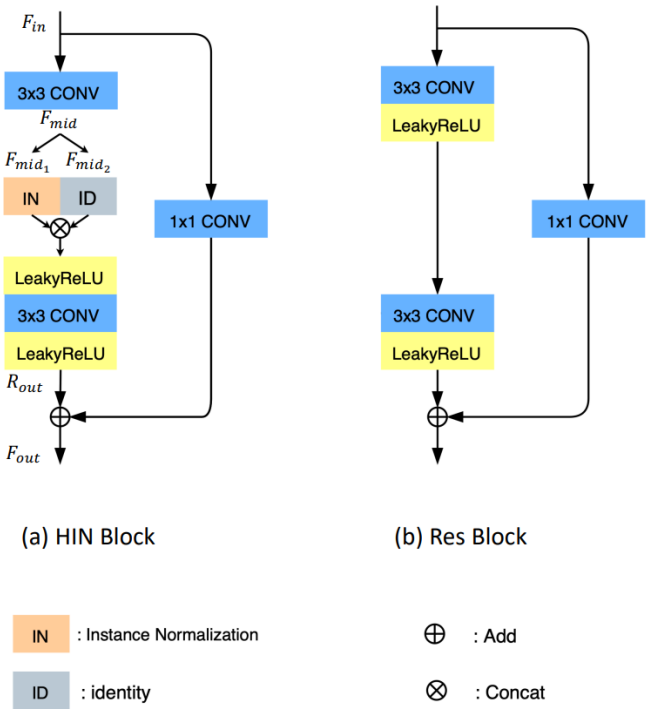


Figure 2.7: *(a) Half Instance Normalization Block (HIN). (b) Residual Block (Res). Chen et al. (2021).*

2.2 Explainable AI

2.2.1 Explainability as a concept

When it comes to explaining how a certain AI works, or in our case how a neural network works, many studies have been done to achieve that task. However, the idea of "explainability" is an elusive one, since these works do not all agree on what must be "explained" for a certain AI system. They do not even agree on what explainability or interpretability is (as discussed by Lipton (2016)). Most works proceed to try to explain or interpret a model without defining what it means to explain the model and what is required to do to say that the model is "explainable". That means we have to approach this topic carefully and move step by step from defining what we mean by explainability in this scope to how it could be applied safely for our purposes.

Lipton (2016) focuses on this problem in the general sense (for models of any kind) and shows the different types of what researchers call interpretability. Some works define it or use it as any type of information that is useful to show how a model works like Lou et al. (2013) where explainability is a property of the model which

can be called "transparent" in the sense that the model's workings are visible and can be comprehended while other models that can not be comprehended are called "black-boxes". However, it is pointed out that this division lacks accuracy since the level of how "visible" a model must be in order to be considered transparent is not defined. For example, a model can be visible, but too complex to comprehend even if it was fully visualized. Other papers define explainability as anything useful about the model (regardless if it explains it fully or not) like Kim (2015) and Huysmans et al. (2011). So in that case, any sort of information extracted about the inner workings of the model is considered explainability. Other papers work on explaining the output or the predictions of the model without explaining the model itself like Krening et al. (2016); McAuley and Leskovec (2013); Van der Maaten and Hinton (2008); Wang et al. (2016). These are sometimes called "post-hoc" interpretations, but not always, as we will discuss in the next section where we present some post-hoc interpretations and how it is inaccurate to define this type of interpretability as post-hoc. Also, some works consider a model to be interpretable if it was interpretable to humans Caruana et al. (2015a), which is also not always true since the human brain is also considered a black box. That is because although we can provide reasoning on why we made a certain decision or a prediction, we still know very little about the exact mechanism our brains followed to achieve that decision (the neuron connections in the brain) Kim et al. (2015).

All that portrays the different aspects to what an explainable model could be. It also shows the different levels of explainability one could have for a certain model. These levels depend on the need for interpretability, which could be a type of information, or establishing trust, or understanding the model, etc. The least of which would be to gain any information about the model, and the most would be to have a fully transparent and explainable model with all its under-workings visible and comprehensible. Another aspect that is very important in XAI is establishing trust with the model by the user, which is in many cases a satisfactory level of interpretability.

Here we also point out the trade-off between performance which correlates with complexity, and explainability. This trade-off was established by many works Simonyan et al. (2014); Eberle et al. (2020); Schnake et al. (2021); Gupta et al. (2023). This can be seen in figure 2.8, where CNNs are considered the most complex and the least interpretable.

2.2.2 Different Types of Explainability

Letzgus et al. (2021) performed a comprehensive study on the various types of interpretability for models which can be divided based on multiple categories. These include explainability for an overall dataset (global explainability) Nguyen et al.

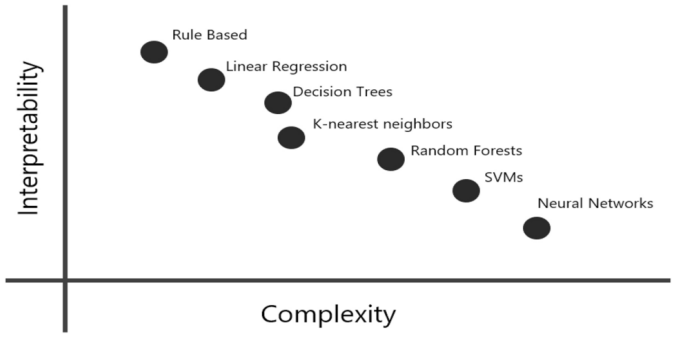


Figure 2.8: *Trade-off between complexity and explainability in machine learning Gupta et al. (2023).*

(2016c); Lapuschkin et al. (2019); Simonyan et al. (2014) and explainability for a specific sample (local) Bach et al. (2015); Baehrens et al. (2010); Ribeiro et al. (2016b). These refer to finding an explanation for why the model performed in such a way for that specific sample by relating specific features in that sample to the model's decision. Or, for global explainability, why the model performed in such a way for an entire dataset, by looking at the features of the dataset as a whole and trying to connect it to the model's architecture. We will see how both ways are valid and usable in different cases. For example, global explainability gives insight on the main workflow of the model, and how it processes inputs in general. On the other hand, local explainability tells us about where the model fails and gives more details on how the model reacts to specific inputs.

On the other hand, some methods perform the explanation based on individual input features, meaning that the explainability method reacts to only one feature (like a change in hue in the RGB image in our case), which is clearly limited. Of these works are Baehrens et al. (2010); Bach et al. (2015); Strumbelj and Kononenko (2010); Sundararajan et al. (2017a). Others perform it based on various combinations of these features which gives more insight on the model's workflow like Caruana et al. (2015b); Eberle et al. (2020); Schnake et al. (2021).

Finally, perhaps the most important distinction is based on whether the methods require some modifications to the model (also known as ant-hoc methods). These methods cannot interpret a model without first making some modifications to it, either in its structure or input Brendel and Bethge (2019); Sundararajan et al. (2017b). These methods also require retraining the model if they perform changes to it which would add to its complication and might -if not done carefully- change the model in a fundamental way which will make the model we are actually explaining different from the one we tried to explain in the first place and that could yield misleading results Letzgus et al. (2021).

Other methods are called post-hoc which means they perform an explainability

study on an already existing model and do not require any changes to the model but perform some sort of explanation to it after it was built and trained. Although these methods usually work on the predictions or the output of the model, they can still provide information on the inner-workings of the model Došilović et al. (2018); Samek et al. (2017); Zhang and Zhu (2018).

Some works found that a more practical approach to find a fitting definition depending on the model would be to classify interpretability -and more specifically transparency- into 3 different classes based on how the model performs in comparison to a human brain Selvaraju et al. (2017).

The first class is composed of models that are still not efficient enough at their tasks that humans are still doing a much better job at that task, in which case we cannot deploy such models reliably. Examples of such models are visual question answering Antol et al. (2015); Zitnick et al. (2016) and visual storytelling Huang et al. (2016). In that case, the aim of interpretability would be to asses where and why the model fails, and to try to find a solution to improve the performance of the model.

The second class is when the model is comparable to human performance like models concerned with the popular task of image classification (when there are enough training examples) Karpathy (2014); Horak and Sablatnig (2019); Maier et al. (2019). In that case the aim would be to establish trust and make sure that the model is not basing its success on false assumptions that might break later if deployed to real life situations. This is the case for most models that interpretability was applied to Zhang and Zhu (2018).

The third class is when the model is exceeding human performance significantly, like game models such as Go and chess Silver et al. (2016); Hölldobler et al. (2017), in which case the aim is to find ways for the model to help humans learn more about the task. This is referred to as machine teaching Johns et al. (2015); Selvaraju et al. (2017).

In the case of spectral reconstruction, we are well in the second class. The models are performing well, not as well as a human, but they are comparable, as will be shown in later sections. It is important to note here that what we mean by a human doing the task of spectral reconstruction well, we mean that humans can easily figure out which material every object is just by looking at it (in RGB). And although humans cannot draw the exact spectra of that material, this is not due to the human's incapability of doing so, but due to the human not having the time to memorize all the spectra of all materials. But, if a human theoretically took the time to do so, they would perform reconstruction better than current CNN models.

So, the main aim of interpretability in our case would be to establish trust in the model and see if we can safely depend on it and deploy it. In a more specific sense, we need to know if we can trust it enough to use it instead of using a

spectrometer to measure the spectra of a certain scene. That is especially the case if the application is very sensitive and the model needs to be trustworthy like in order to avoid consequential wrong decisions (eg: medicine Lu and Fei (2014)) or avoid economic and intellectual losses (eg: agricultural and earth observation Dale et al. (2013); Elmasry et al. (2012); Patro et al. (2021); Mücke et al. (2019))

2.2.3 The Main Focus of This Work

Here we try to use the previous sections to draw a more clear picture of what we are trying to do in this thesis in more precise form. We mentioned that we are trying to explain RGB to Hyperspectral reconstruction models, but as we can see from the previous section, this is not yet a well founded question since it leaves a lot of uncertainty.

We use the works of Lipton (2016) and Millan and Achard (2020) to divide models in the following classes in terms of their need for explainability:

2.2.3.1 Class A: Transparent and Simple

These are models that are fully transparent and completely explained and comprehensible to a human being. Examples of that is simple linear models like say $y = 2x + 3$. All the parameters of these models are clear and the functionality of how it works is also clear. These are usually very simple but could also be more complicated to a certain degree. These models do not need us to apply any explainability methods since they are already explainable.

2.2.3.2 Class B: Transparent but Complex

These models are more complicated, but they are not complete black boxes. Examples of these models are non-linear separators and Support Vector Machines (SVMs), or even simple CNNs. In such models, we know how they work and their general principles, but we do not necessarily understand the full depth of parameters they use. The parameters they use could be too many for a human to comprehend.

However, that does not mean they are in need of explaining. For example, we know that if we train a simple NN on data that relates personal income to the number of cars used for single person (given that the data is enough and valid), that NN will learn the correlation between both parameters and work as expected. If the network does not work as expected however, then we immediately question the network structure, the validity of the dataset, or training, etc. The important part here is that we do not question that the network might not inherently work (we know it should, given the right data and network/training parameters).

That is because: *first* we know that there is a correlation function between the input and the output. *Second*, The task is simple enough that the workflow of the model should be clear and understandable. Like a separation line between 2 classes, the function (shape) of that line could be complex, but we understand that it is simply a line found that separates these 2 classes. So, these models have their inherent level of explainability (if they are applied and/or trained correctly). This is a level of explainability that most works mentioned in this section agree that it is satisfactory in most applications since the workflow is clear enough to understand how it works even if not all parameters are accounted for Millan and Achard (2020); Lipton (2016).

We have to point out that such models could use explaining if they are not working, as a tool for debugging and finding what is going wrong and why the model fails. Although in that case the models need explaining, this kind of explaining is only to find the right hyper-parameters for the model or find problems with the dataset/input. It does not try to uncover the workflow of the model since that workflow is clear.

Now, we will discuss 2 other classes (C & D) of models that are even more complex and need more explaining. but we mention here that in most cases of explaining, we consider classes A & B to be explained and satisfactory. So, in our quest to explain more complex models (C & D), what we do is to try to break them down into a series of smaller (A and/or B) models that is equivalent to the full workflow of the complex model.

2.2.3.3 Class C: Black Boxes

These models are much more complex that previously mentioned ones in the way that their workflow is obscure. For example, a CNN that classifies cats from dogs in image inputs falls under that class. This is because we do not really know how the network makes its decision. We only know how we *expect* the the network to make its decision (using distinguishing features of cats and dogs). However, the model is a black box and we cannot tell if it is actually doing that or using other -possibly irrelevant- features.

Most neural networks currently in literature including the vast majority of classification and segmentation CNNs fall under that class Becker et al. (2018); Robnik-Šikonja et al. (2011); Robnik-Šikonja and Kononenko (2008); Montavon et al. (2017). Of course, this class of models need explaining since their workflow is completely obscure and we only know how we expect them to work, not how they actually work. For that, most explainability methods mentioned in this work are directed towards that kind of networks, especially image classification networks Egmont-Petersen et al. (2002); Wiley and Lucas (2018); Montavon et al. (2018).

As we mentioned, most works employ methods in the aim of breaking down such

complex black boxes into a series of more explainable parts (of classes A and B). And since reaching class A is very difficult (because class A needs the model to be fully comprehensible), we usually try to break down class C into a series of simpler class B models. In fact, we see many works including that in their architecture, like models that have a feature extraction block in their architecture. We can think of that feature extraction block as a class B model within the larger class C model Robnik-Šikonja and Kononenko (2008). Of course, whether that block actually does what the authors claim it does (feature extraction) is another problem.

In the case of cat and dog classifiers for example (a class C model), we can break it down to parts of the network that detect the ears, and parts that detect the eyes, and mouth and so on. Each of these detectors can be thought of as a class B model, since we know what it does and how it does it (training) but we do not know the exact parameters for it. So, breaking down that classifier into these class B sub-models helps us understand its general workflow and provides more trust in the model.

2.2.3.4 Class D

These models take it a step further than class C. That is because we do not even have any expectation for how they work. In the previous example of cat/dog classifiers, we expect the model to distinguish each class by its defining features (ears, eyes, etc), so if it is doing that then the model is on the right track. However, for class D models, we do not even know what these features are. We as humans are not entirely sure how the task is done (even if our brains are able to do it).

The most important example here for such models is our case: RGB to hyperspectral reconstruction models. We as humans can distinguish different materials under a variety of lighting conditions, so we can perform (theoretically) that reconstruction task. However, what features did we really use in that task, how did our brains manage to distinguish different illuminants for example? This is unclear in most cases, especially that we mostly use our memory, and senses like touch, and context to perform that task. Such inputs are not available to the network. Another similar example of class D models are spatial super-resolution models Dong et al. (2014b,a).

Also as we will see in the next chapter, the network only looks at a limited spatial area in the input image. So, the context provided is very limited and yet it still does the reconstruction task. All of this begs the question of *how do we expect the network to be working* in the first place.

This emphasizes the importance of our task, and the need for explaining such models. Again, explaining class D models could be thought of as breaking them down into a sequence of class B models. This is especially difficult for class D since we do not have a clear picture of what these sub-models could be. All classes

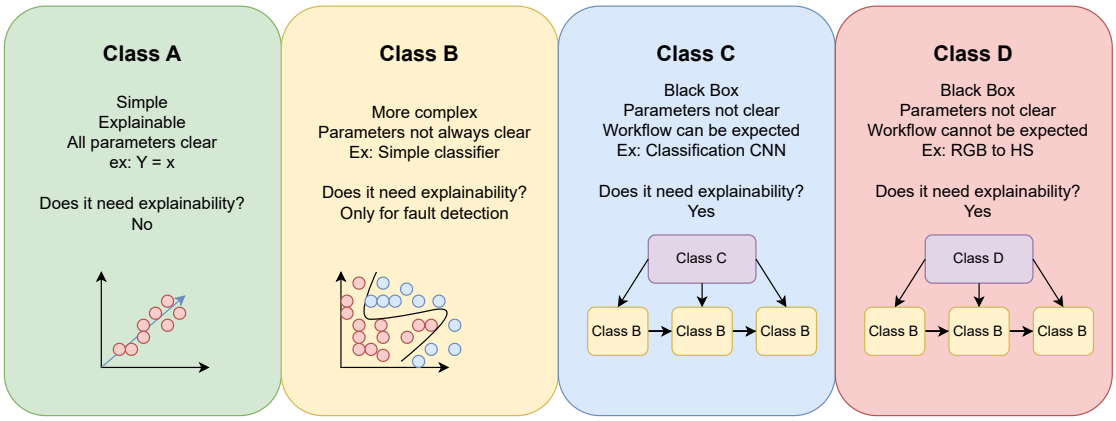


Figure 2.9: *Different classes of models in terms of explainability need.*

mentioned are shown in figure 2.9.

2.2.3.5 Important Detail

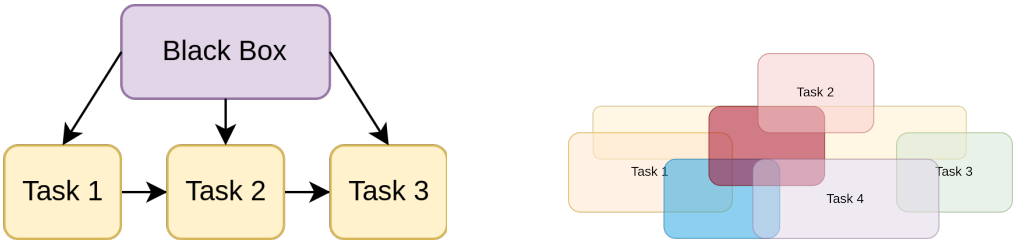
Now, we have to note here that what we mentioned about breaking down classes (C & D) into a 'series' of class B sub-nets is not a realistic description. In reality, these sub-models are almost never a 'series' but an overlapped parts of the network, each doing a task, or many tasks. This is inherent in the way CNNs work and how they are optimized Hou et al. (2022); Wang et al. (2022); Olah et al. (2017); Carter et al. (2019).

Because of that, when we break a class (C or D) into class B sub-nets, these sub-nets are usually the 'explainable equivalent' of the real sub-net. This means that this equivalent will not be exactly as the original sub-net (since that cannot be separated from the original network) but will serve as an equivalent that helps us understand how it works and even try to replicate it with an explainable model. The sequential and overlapping sub-net distinction is shown in figure 2.10.

2.2.3.6 The Aim in Conclusion

So, our aim in this work is to try to break down the main functionality of the RGB to hyperspectral models (which are class D models) into sub-models of type B. We cannot for sure know that we can interpret all the workflow of these models, but we will extract as much information as possible (as many class B sub-nets as possible). That will give a better look at how these models work and perhaps provide a level of user trust, or awareness.

The way we do that is by using explainability methods and conducting tests that will shed light on how some parts of the models work. As mentioned, these



(a) *Sequential (unrealistic).* **(b)** *Overlapping parts (more realistic).*

Figure 2.10: *Breaking down a black box (Class C or D) into smaller task specific sub-nets (class A or B). The sequential case being unrealistic but equivalent representation of the more realistic overlapping nature of sub-nets doing specific tasks.*

sub-nets will be intertwined, and will be an explainable equivalent of the real sub-nets, and not necessarily how the original sub-net is.

In the next subsections, we will dive deeper into these methods, especially post-hoc methods because they are the most useful. We will see what they provide exactly and how they compare to each other in more detail. Of these post-hoc methods, some of the most useful are visual methods that provide a visual representation that helps with explaining the model, the output, or both. This will be the focus of the upcoming subsection.

2.2.4 Visual Explainability in Neural Networks

Now we turn to the more specific case of explainability for neural network models. Ant-hoc operations would require us to either build a new explainable model (which would defy our purpose to explain current operational models), or to change the model to be explainable (which would have a similar effect). Because of that, and since our main goal is to establish trust and explainability in current used models, we need to use post-hoc methods. They are also proven to be the most used and the most practical in that field because they are *first* the easiest to interpret by humans, and *second* they can provide implicit and unexpected information on the model, unlike other methods that only provide the specific information we expect from them Zhang and Zhu (2018). This will become more clear once we show examples of these methods.

For neural networks, and more specifically for convolutional neural networks used on image data (which is our main interest) these methods usually try to relate the output or the prediction of the model to the input in a way that highlights visually which parts of the input the network found most useful in its prediction. The result of that is a heat-map for the importance of the pixel called Saliency

Maps, or Attribution Maps Selvaraju et al. (2017); Chen et al. (2020); Kolekar et al. (2022); Mundhenk et al. (2019); Millan and Achard (2020); Simonyan et al. (2013); Mahapatra et al. (2017); Mahendran and Vedaldi (2016); O'Shea et al. (2022); Shrikumar et al. (2017); Springenberg et al. (2014); Bach et al. (2015); Letzgus et al. (2021).

Another way to visualize how the model perceived the input is to generate reference maps (atlases) that show how different parts of the network are more concerned with a specific type of inputs (they are activated when a certain shape or color is entered to that network). These are called "Activation Atlases" Carter et al. (2019); Buhrmester et al. (2021) and they provide an effective way to visualize entire layers in the network as an atlas of small images each shows what kind of features that particular "area" in the network is activated from. These methods can also be applied to more than one input since sometimes the method can be pooled over a whole dataset as we mentioned about global and local explainability Nguyen et al. (2016c); Simonyan et al. (2014).

Both saliency maps and activation atlases provide representations that help explain where the network is looking and how the input is treated. This can greatly help in explaining which features the network is looking at. More importantly, it helps explain why a network would yield a false output or prediction. For example, in simple terms, looking at the saliency map of a classification network that classifies cats from dogs, one would expect the network to focus on features that relate to the cat or a dog in the input image, such as eyes, nose, ears, whiskers, and so on. If the network focuses on these features, then we can establish some level of trust that it is doing the job as we expect it to do Millan and Achard (2020). However, if the network is focusing on unrelated features like unrelated features in the background or the color of the animal, then we cannot trust such a network even if it had high accuracy on a particular dataset since its success would probably be due to the limitation of the dataset, and it will probably fail on another dataset. That shows the flaws either in the dataset, which could have biased or unbalanced data (for example the cats are all orange and the dogs are all gray so the network would learn the color as the defining feature) or that there is a fault in the training like over-fitting to some particular set of attributes, or a problem in the network itself, like the architecture being unable to capture the more representative features Carter et al. (2019).

If a saliency map is provided given a particular input (an image in this case) of a network, that saliency map only serves to localize (spatially) where the network was looking. Further analysis is needed to find which exact features of the network the saliency map is actually capturing. So, in some applications, spatial localization of the activation saliency could be an important first step, but not enough to find what the network is actually "focusing on" since that could be undefined in

the spatial space but in another feature space like color space or frequency space Baehrens et al. (2010); Selvaraju et al. (2017). It could also be that the model is focusing on many things and that makes it difficult to interpret even if a saliency map is given.

Since we have RGB images as input, one could wonder if we can apply a saliency map to each of these color channels. Although we can create an RGB saliency map, the RGB channels of that saliency map will not correspond to the RGB channels of the input, since the saliency information will get mixed up in the process of creating the saliency map. This will become clear once we introduce how saliency maps are created and the many ways to create them.

So far we established the general idea of how some methods could explain how a CNN looks at the input (how it was activated based on that input), but that could be done in a number of ways. We now go through the many methods used to generate these saliency maps. As for activation atlases, they are quite exclusively used in the main paper describing them Carter et al. (2019).

Methods to generate saliency (or attribution) maps can be divided into 3 categories: *Perturbation-based*, *Class activation based*, and *Gradient-based*. We will review the literature of each of them in detail and find which ones would be most suitable to our purposes and how they still need to be modified to match our exact focus.

2.2.4.1 Perturbation based

Perturbation (or sometimes referred to as attack based) methods aim to explain the most important part of an input to the decision made by the network by perturbing that input in some way that should make it harder for the network to do the prediction (attacking the network, by altering the input) and monitoring how the network deals with that perturbation Zeiler and Fergus (2014); Millan and Achard (2020). So, basically we disturb the input in a known manner, then look at the output to see how it was affected. If the input was an image that was disturbed by occluding or adding noise to some of its parts, then if the occluded parts were important to the prediction, the output will change accordingly. However, if the output remains the same, then we can deduce -arguably- that the network does not depend on that part of the image in making its predictions. The opposite could also be true if the prediction changed in response to the perturbation introduced in the input. An example of perturbation can be seen in figure 2.11

This is what is done in Zeiler and Fergus (2014) where parts of the image are occluded with a gray area and then that area is increased until the accuracy of the classification decreases significantly. The portions of the image that are kept right before this dramatic drop in accuracy can be said to be the most salient parts of the image for that model. In object detection networks, they are expected to

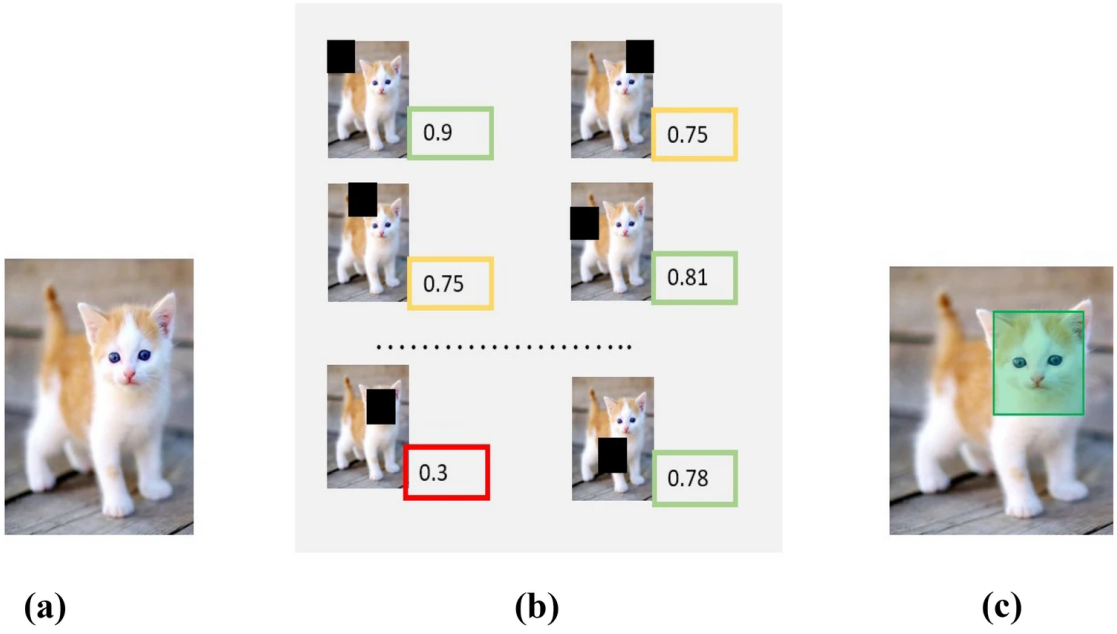


Figure 2.11: *Perturbation based approaches: (a) Original image. (b) Perturbed dataset. (c) Attribution map based on the most sensitive parts of the image. Gupta et al. (2023)*

be the objects themselves, while in classification networks, they are expected to be defining features of the the designated class. A sub-modular optimization is performed to produce the linear local model approximation.

Another method worth mentioning is called Local Interpretable Model Agnostic Explanation (LIME) Ribeiro et al. (2016a). This method is based on perturbation of the input, but merely uses it to construct an approximation for a linear model local around the prediction that could be explained. It is also demonstrated to work on any classifier (not just neural networks). This is known as 'model agnostic'. The advantage of LIME is mainly that it provides a very simple and intuitive approximation to any model. Also, it assigns weights to features based on their contribution to the prediction. That way, we can tell which features are more important easily. An example of what LIME can do is shown in figure 2.12, where the Inception model is tested on the original input image (left) and 3 top predicted classes for that image. It can be seen that the results are very descriptive and show the reasoning behind the model's choice in the input image.

However, LIME has a local scope, meaning that it can only provide explanations to specific predictions, not for an entire model. Also, since it changes the model, it affects its performance in exchange for explainability (ant-hoc). Although this is a useful trade off in many cases, we try to only use post-hoc models in this work to

Figure 2.12: *LIME applied on Google Inception model showing the original image (left) and saliency maps for the top 3 classes (electric guitar, acoustic guitar, Labrador) respectively Ribeiro et al. (2016a).*

keep the models intact and try to explain how they work without changing them.

It is important to note here that sometimes these methods cannot work properly if the task the network is trying to solve requires it to find global features (spatially global) all around the input image. In which case the localization for this kind of methods will not be flexible enough to capture the global or scattered nature of the features. Also, they require a lot of time and computational costs since they require to re run the inference on orders of magnitude the size of the dataset to be tested, depending on how accurate the saliency maps are required to be Arrieta et al. (2020); Došilović et al. (2018). Here, by 'accurate' saliency maps, we mean that they have a fine spatial resolution that shows clearly where the spatial area of interest for the model starts and where it ends along with any change in its saliency in the input image.

2.2.4.2 Class Activation Based

In the context of classification networks, class activation methods produce saliency maps called Class Activation Maps (CAM) Zhou et al. (2016). These maps highlight the spatial pixels used in making the classification choice. These methods were introduced with Zhou et al. (2016) and updated by many later works like Grad-CAM Selvaraju et al. (2017), U-CAM Patro et al. (2019), Smooth Grad-CAM Omeiza et al. (2019), and many other related works Kwaśniewska et al. (2017); Zhou et al. (2018); Pinciroli Vago et al. (2021); Morbidelli et al. (2020); Fu et al. (2020); Wang et al. (2020a); Sattarzadeh et al. (2021); Song et al. (2021); Naidu et al. (2020); Byun and Lee (2022); Wang et al. (2020b).

Class activation methods started with the basic CAM model Zhou et al. (2016) which first chooses the class to be activated, say class 3 out of 10 classes. Then it would zero all other classes and keep the target class (3 in our example) as one. If the model has a fully connected layer in the end, it is replaced with a convolutional

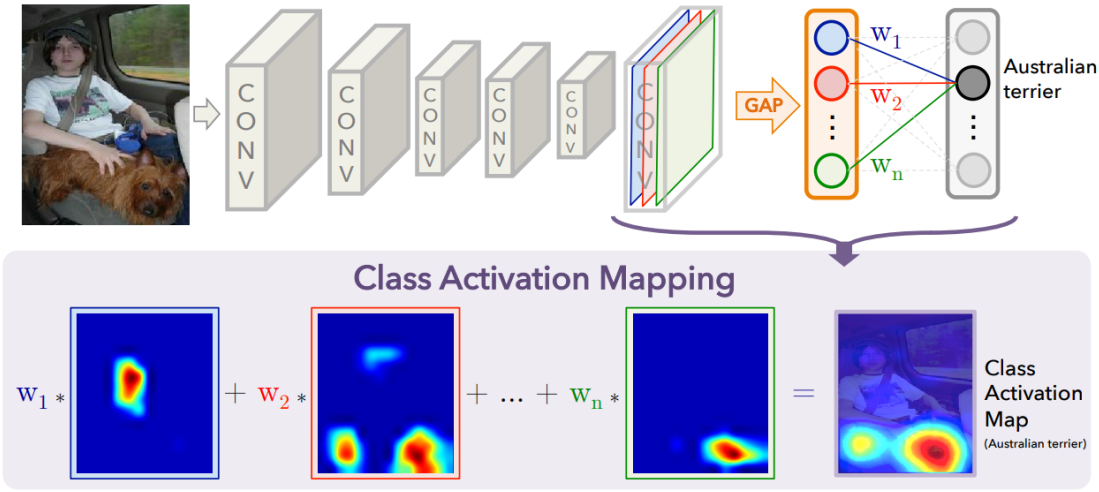


Figure 2.13: *CAM workflow and example Zhou et al. (2016).*

layer and a global average pooling layer. This is because fully connected layers -unlike convolutional layers- destroy valuable spatial information needed by CAM to keep track of the spatial saliency.

Then, it uses the global average pooling layer to back propagate the output class into activations in the layers in a backward pass. These activations are then averaged and sent into a Relu activation function to remove negatives. This results in a spatial heatmap resembling the activations the model did when choosing a specific class for that specific input. Figure 2.13 shows the workflow of CAM, and how the chosen class is mapped back to the input through convolutional layers.

That does not work however if the model has fully connected layers (since they need to be replaced) which is very common in many state of the art models, and replacing them will greatly hinder the performance. That makes an unwanted trade-off between performance and explainability.

To address this, Selvaraju et al. (2017) introduced Grad-CAM which updates on the previous CAM method by using gradients of the target class (passing through the last convolutional layer in the model) as weights applied to the activations of the convolutional layers in the backward pass. This makes the CAM method applicable to a larger number of models. Examples of Grad-CAM can be seen in figure 2.14 for the VGG-16 model. The heatmap-like saliency map is here added on top of the original image to give better representation. We note here that we will use the same input image (original images in figure 2.14) and the same model (VGG-16) in the next examples showcasing other methods.

Later methods updated on it with some modifications, but they all have the same characteristics: *First*, the saliency map produced (as an activation heatmap)

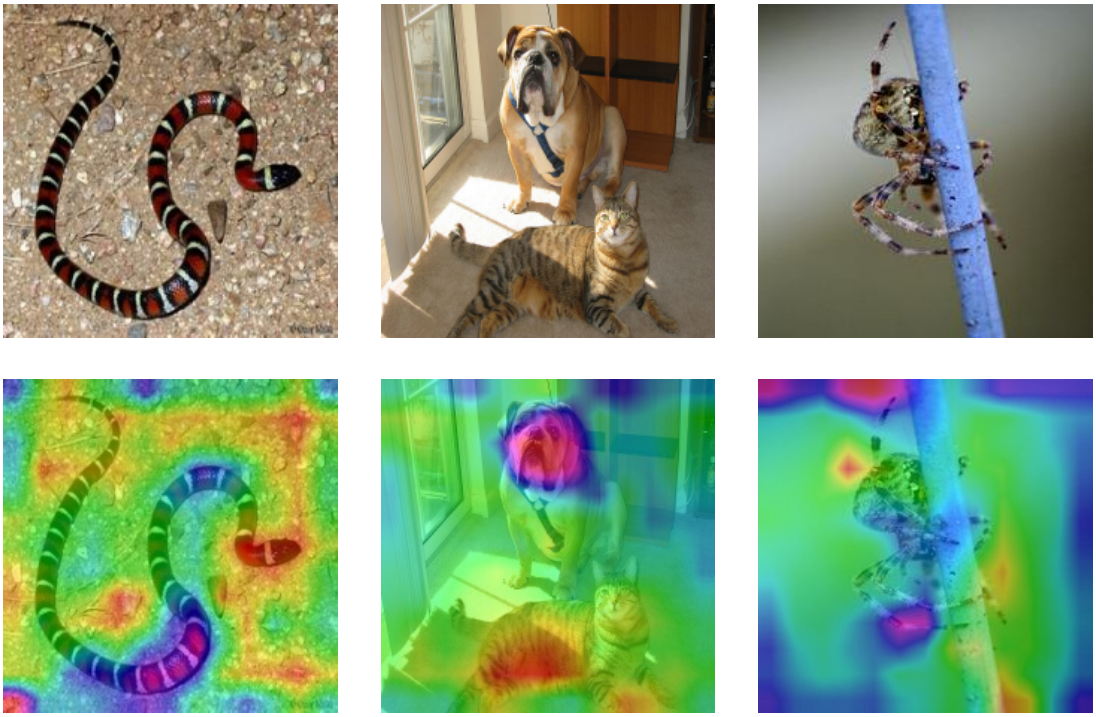


Figure 2.14: *Grad-CAM saliency maps examples (bottom) on top of original image (top) for classes (King Snake, Mastiff, Spider) left to right respectively Selvaraju et al. (2017); Ozbulak (2019). Most salient areas are red, then blue, then yellow, then green.*

is very coarse and displays the general attention points in the input image. *Second*, they all depend on the class as a starting point to the activation backward flow which make their result class-dependent and class-discriminative.

These models were very favourable over the perturbation based models since they are much less computationally costly. In principle and in most of these methods, only one forward pass and one backward pass is enough to generate the saliency map. Also, the class-discriminative feature is sometimes desirable when the user is interested in finding the explanation for a specific class Millan and Achard (2020); Selvaraju et al. (2017).

2.2.4.3 Gradient based

These methods started mainly with Simonyan et al. (2013) where the gradient of the output score function (not of the class, but of the last convolutional activation) was proposed as the main drive for the backward propagation (the method is called

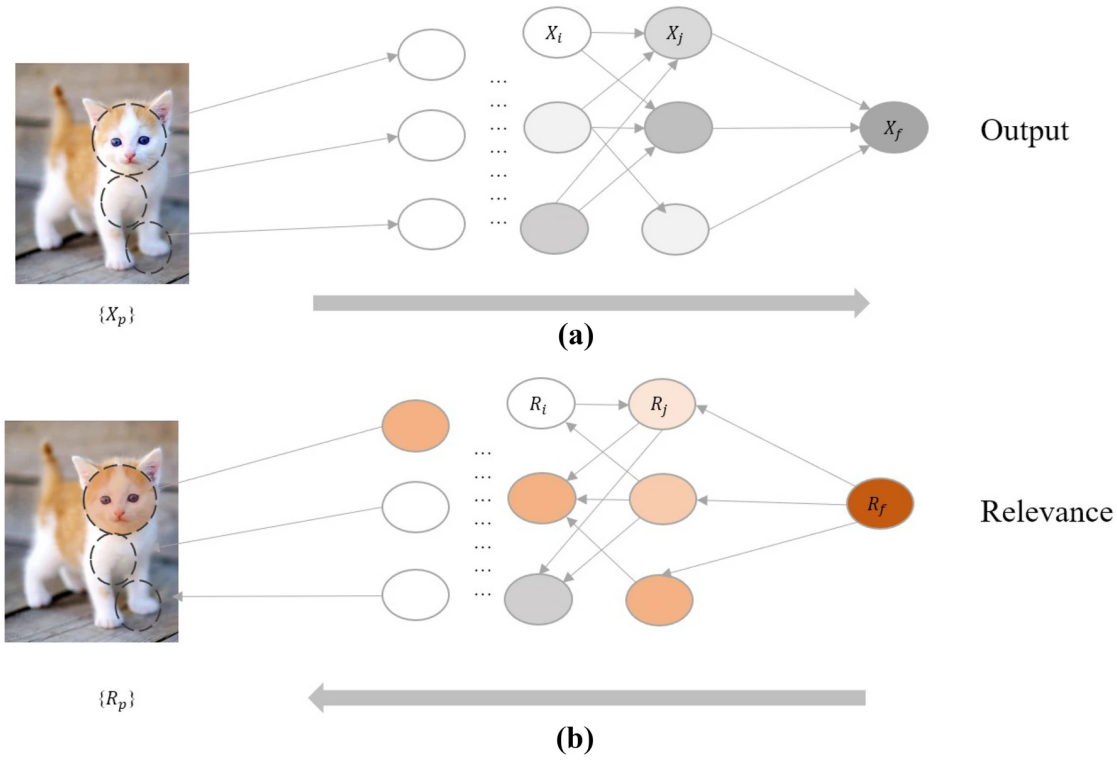


Figure 2.15: *Gradient Based methods: (a) Forward pass. (b) Backward pass. Gupta et al. (2023)*

back-propagation). This propagation would follow the gradients from the output backwards in the model until the first convolutional layer making an activation map on the input stage. The forward and backward passes of a generic gradient based approach are illustrated in figure 2.15.

These maps were much finer than the class activation ones, but they were very noisy. Examples of back-propagation saliency maps are shown in the top part of figure 2.16. It can be seen that the maps are more detailed than the CAM based heat-maps, but they are extremely noisy.

Later updates like deconvolution Zeiler and Fergus (2014) proposes a change in how Relu non-linearity is handled in classic back-propagation by discarding negative activations in the backward pass which makes the activation maps more focused on positive activations and have better resolution and less noise. Deconvolution uses "switches" in the forward pass to record the maximum values in the max-pooling layers. That way, these switches hold the locations of the pixels chosen by the max pooling layers and allow to retrieve them once we need to go backward in the network. Figure 2.17 shows how the maximum locations of the pixels are stored in the switches.

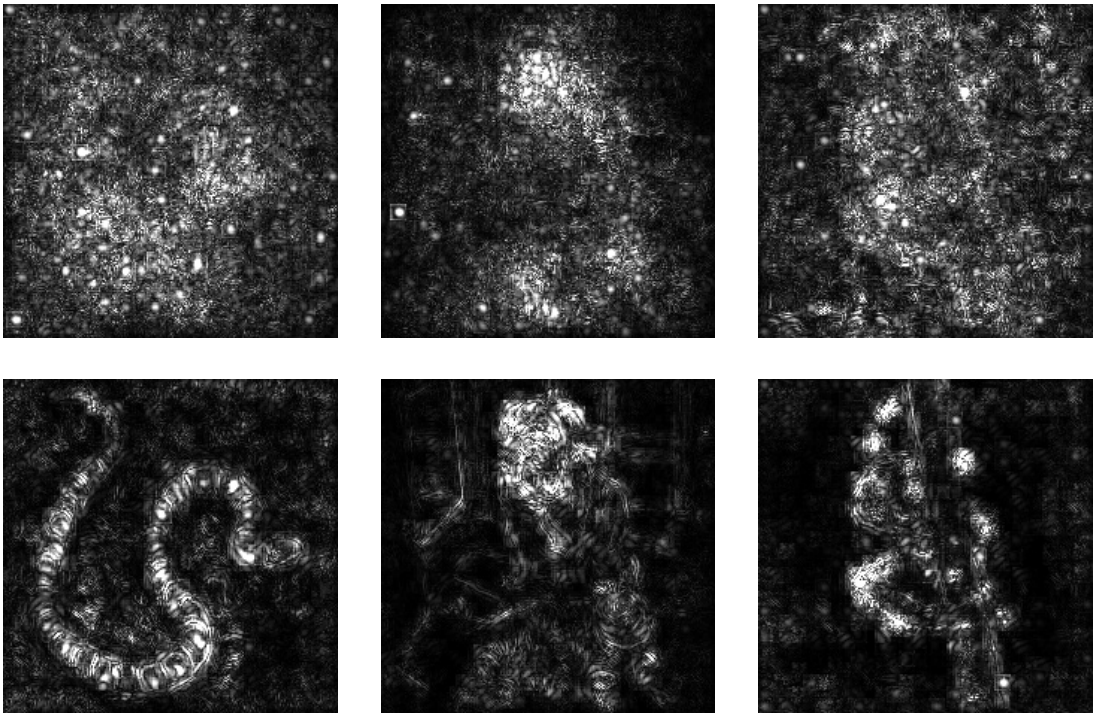


Figure 2.16: *(Top) Back-propagation examples, (bottom) Guided Back-propagation examples. Original images are shown in figure 2.14. Simonyan et al. (2013); Springenberg et al. (2014); Ozbulak (2019).*

Then, we use these values while going through the network in reverse to allow un-pooling (pooling in reverse that expands the spatial dimensionality). That way we can go through the whole network including pooling layers and finally reconstruct the activation map on the input. This update allows us to perform saliency maps explainability on networks that have pooling layers which was not possible before in previous methods.

A further key update was made by Springenberg et al. (2014) who argued that CNN models do not really need pooling layers and fully connected layers. The authors prove mathematically that these layers can be replaced by convolution layers with stride (as used by many models that do not have pooling). It also gets state of the art performance by replacing the fully connected and the pooling layers of existing architectures by convolutional layers. Of course, the network needs to be retrained in that case which was the downside of this method.

Once that was established, they showed how that replacement could make networks much more visualizable and explainable with back-propagation which only worked on convolutional layers at the time. Then they introduce an update

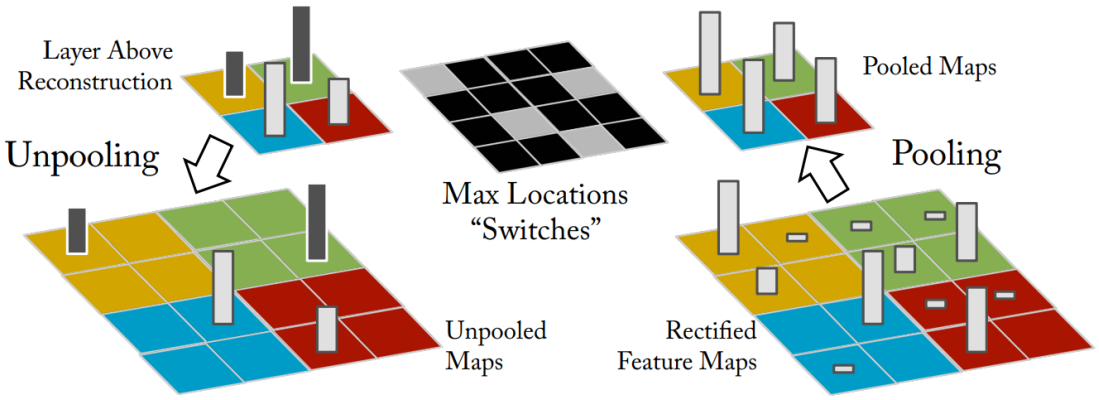


Figure 2.17: *The concept of switches and unpooling in Deconvolution Zeiler and Fergus (2014).*

to deconvolution and classic back-propagation where negative activations were removed on both the backward and the forward pass making the activation map much more fine and accurate, and at the same time it will not need switches since there are no pooling layers to begin with.

This method was called Guided back-propagation and was later used as a benchmark for how back-propagation was applied because of its high fidelity maps. The bottom part of figure 2.16 shows the great improvement of guided back-propagation over classical back-propagation (top part).

After that, guided back-propagation was used in combination with other methods, and was adapted to work with pooling layers without replacing them. This is because the main idea works even with the existence of pooling and fully connected layers in principleMundhenk et al. (2019).

A work done by Mahendran and Vedaldi (2016) introduces a general architecture for deconvolution networks designed for reversal and visualization, and shows how back-propagation can be considered as a special instance of that architecture. It also shows how the structure of deconvolution and the localization of back-propagation can be mixed to produce a more local structured visualization architecture called DeSaliNet, and shows how such a network can further be used (in its reversed form) as a semantic segmentation network.

It is important to note that unlike class activation maps, gradient based methods do not provide local importance. That means that the activation maps do not have specific areas on the input image where the pixels that contributed to the output are highlighted. Instead, they have a global activation map of the input where the intensity of the pixel in that activation map represents the importance of that pixel in making the decision of the output (chosen class in classification networks for example). So, where the class activation map looks like a coarse heatmap locally

locating the important parts of the image, gradient based activation maps provide a full map of the whole input (non-local) but with finer resolution.

One problem that sometimes occurs is that the output score function does not always represent the importance of the pixels. This is because the score function could get saturated around some pixels and then it would have a small gradient Sundararajan et al. (2016). This was fixed by taking each input pixel at a time and calculating the global saliency of that pixel by performing back propagation from all the output back to each input each pixel at a time. This was done in Shrikumar et al. (2017) and the method was called Deep Learning Important features (DeepLIFt).

A similar approach was used by Bach et al. (2015) with a method called Layer-wise Relevance Propagation (LRP) which decomposes the input into pixels to separate the contribution of each input pixel in the decision class. Using a specific back propagation rule, the output's "relevance" is reflected backwards for each layer separately, eventually building the activation map which focuses on the pixels most responsible for the target class. This method (as well as DeepLIFt) depend on the target class, but it should not be confused with class-discriminative methods since it only *depends* on the target class, but we cannot separate the contributions made by the input to other classes completely. The top part of figure 2.18 shows the results of applying LRP to the same images on layer 7 of the VGG-16 model.

Another proposal was made to simply multiply the gradient activation map with the input to improve their quality (Gradient x input methods) Shrikumar et al. (2016). The bottom part of figure 2.18 shows the results of this method to the same examples. This method was later adapted as a common practice with all methods.

Also, in Sundararajan et al. (2017a), the authors started the idea of integrated gradients, where the gradient is not calculated from the output directly, but calculated from the integration along a path between the input and a chosen baseline. That integration (approximated as a summation) is done for each dimension in the input. Figure 2.19 shows the results of this method on the same examples.

Also, Smilkov et al. (2017) proposed to smooth the gradient with a Gaussian kernel to improve the quality of the activation maps. This is done to address the issue that the gradient is sometimes very noisy because the input changes rapidly in local areas. It essentially removes the noise in the activation maps by adding noise to the output gradient.

Some methods combined class activation based with gradient based methods to get the fine activation maps that gradient based methods provide and adding to that the class-discriminative and locality features that class activation methods provide. This was done in Guided Grad-CAM Selvaraju et al. (2017) where grad-CAM was used to produce a class-discriminative coarse map, then guided

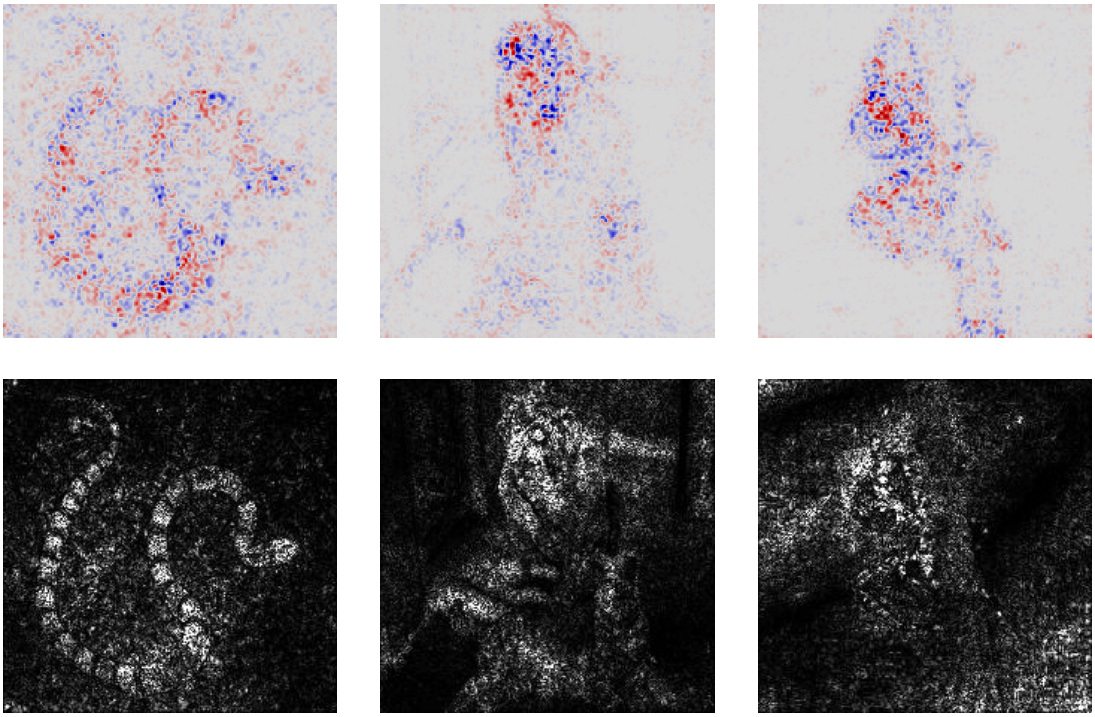


Figure 2.18: *(Top) LRP applied to layer 7 examples, (bottom) (Grad x input) examples. Original images and target classes are shown and mentioned in figure 2.14. Bach et al. (2015); Shrikumar et al. (2016); Ozbulak (2019).*

Figure 2.19: *Integrated Gradients example. Original images and target classes are shown and mentioned in figure 2.14. Sundararajan et al. (2017a); Ozbulak (2019).*

back-propagation was used to increase the accuracy and produce a fine resolution, local, and class-discriminative activation map at the same time with a slightly more

computationally expensive process.

Finally, in Mundhenk et al. (2019), a more efficient way of making the same kind of fine resolution maps was proposed by applying a statistical calculation of gradient-related information using a specific simplification called Saliency Map Order Equivalent (SMOE scale) to replace the guided back-propagation operation. That calculation is only applied to layers where the spatial scale of the feature map is changed (with a convolutional stride or with a pooling layer). That combined with Grad-CAM yielded very good results with much less computational cost.

To summarize, we described a wide range of saliency map methods. Each with its advantages and issues. In our case however, we need a method that does not depend on having a class. This already excludes many methods like CAM based methods. Of the best performing methods mentioned that does not depend on a class is Guided Back-propagation. It can be seen in the figures that it provides fine resolution saliency maps (arguably the finest) and has lots of developments and updates that make it work with most models. For that, we will use Guided Back-propagation in our work, but with some modifications that will be mentioned later. Now, we turn to a different type of explainability methods that give more insight on the inner parts of models.

2.2.5 Feature Visualization

All the methods described so far use some way to extract information from the black box of the CNN without actually peeking inside it. The main result out of all the methods described so far is a saliency map (even perturbation based methods end up giving us a map that depicts which parts of the input the network is mostly interested in, which is still under the definition of a saliency map).

There are significant limitations in saliency maps however. Most importantly, they depend on the single spatial pixel as the primary unit of attribution Olah et al. (2018). This is not true in the default case since pixel information in CNN is always intertwined and used in groups, and they are not resilient to visual shifts or transformations. Also, with the exception of methods including the output class like Grad-CAM, most of these methods have no connection to the prediction of the network, meaning that they do not provide any information about why the model made a specific prediction (output) and not another. Finally, saliency maps only allow to see the attribution on one end of the network (attribution on the input image). Although some works like Mundhenk et al. (2019) did develop a way to extract saliency maps at different stages of the network, they are still limited to places in the architecture where a scale change occurs, because they depend on the change to calculate the scale (SMOE scale mentioned in the previous subsection). Also, they will be found to create highly similar and uninformative saliency maps

for different layers but with only more low frequency images the closer we go towards the output of the network (which is to be expected).

So, they do not allow a deeper look into the hidden layers of the network, and they do not allow any visualization of the intertwined neurons and feature maps deep inside the network. That is because they only visualize a very limited aspect of the overall functionality of the network Olah et al. (2017).

This does not mean that saliency maps are not good visualization tools, they are proven to provide invaluable insight on the network's behaviour and even identify faults with the network as all the previous works we discussed so far have proven. This only points out that CNNs are still too complicated to be explained based on their input. Saliency maps allow us to easily find problems with how the network provides its predictions (if the network is focusing on features unrelated to the actual representative features of the class it gives). So, they work well to find where the network fails, but if the network does not show any obvious failure, or if we need to really understand how it works inside the black box, then we need to dig deeper Olah et al. (2018).

This is where feature visualization comes in. The basic idea outlined in Erhan et al. (2009) is to provide a visualization of a feature vector somewhere inside the CNN by trying to generate a stimuli that is optimized to activate that specific feature using an iterative optimization method. Let's say that we would like to visualize a feature somewhere in a neuron in a hidden convolutional layer in the trained network. We extract that feature vector, then we try to maximize it by running a gray image or an image of random noise as input. Then, we tweak that input iteratively while observing the output to optimize the input image until the output (the feature we are trying to visualize) is maximally activated. The tweaking is usually done by gradient ascent or any custom optimizer.

This is done by using the derivative nature of convolutional layers that connect the input to the the neuron we are trying to visualize. This allows us to optimize a loss function, which is the difference between the output of the input image we provide and the activation of the neuron we are visualizing.

When the optimization is complete, the image we generate represents features which that neuron is trained to process. It is what the neuron maximally activates in response to. Figure 2.20 shows this process as applied by Olah et al. (2017) on the 11th channel in the mixed4a layer of the GoogleNet Szegedy et al. (2015) model. We can see that we start from a random noise image that is then iteratively tweaked until we reach a feature visualization after a number of iterations. In the figure, that visualization seems to resemble a pattern of connected nodes, which suggests the layer is specialized in recognizing patterns similar to it.

Of course, this can be done on any combination of neurons rather than a single neuron for which it would be very hard for a human to observe the interactions of

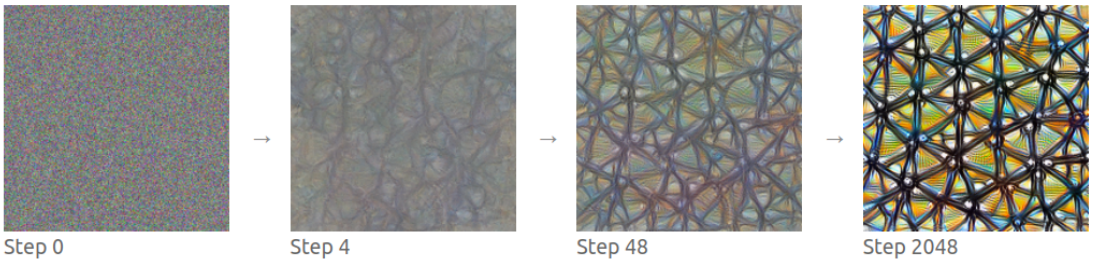
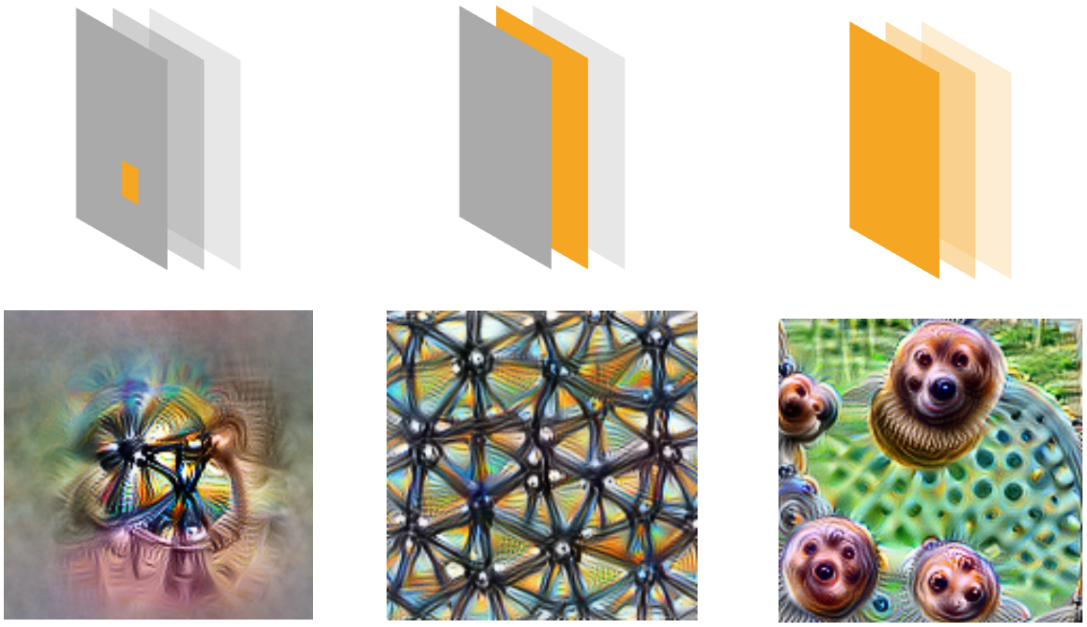


Figure 2.20: *Feature visualization for the 11th channel in the mixed4a layer of GoogleNet done by Olah et al. (2017).*



(a) *Neuron.* **(b)** *Channel.* **(c)** *Layer.*

Figure 2.21: *Different possible objectives (top) and their respective feature visualization (bottom) Olah et al. (2017).*

the large number of neurons in the network. So, the optimization is usually done on the feature activation to an "objective" which is a number of neurons grouped together according to a certain criteria. This criteria could be to visualize a patch in a channel (2D section of a layer), or certain channels within a layer, or to visualize a layer as a whole. Class logits and class probabilities (for classification networks before and after the softmax respectively) could be used to create examples of the output class Olah et al. (2017). This is illustrated in figure 2.21 where visualizations are made for multiple objectives in the GoogleNet model.

Of course, these objectives are just ways to group visualizations into some form that makes it easy for humans to understand, but do not necessarily reflect the real way neurons interact in a network. For example, neurons in channel A could be more connected semantically (highly correlated) to neurons in channel B than other neurons in channel A. In that case, channel based objectives would not be the best to visualize different groups of neurons. Even the idea of using groups of neurons as basis activation vectors is not necessarily the best to define more interpretable vectors rather than other vectors with other directions in this space. Some works found that random directions could be just as interpretable Szegedy et al. (2013) while other more recent works found that in the usual case, it could be more interpretable to visualize in the direction of the basis vectors rather than random directions Bau et al. (2017).

Either way, it seems there are many options for grouping that could be explored, which resulted in dedicated algorithms for finding and clustering such groups into optimal blocks where each one represents a feature visualization objective. These algorithms are mostly based on matrix factorization Lee and Seung (2000); Mnih and Salakhutdinov (2007); Olah et al. (2018) and they are not in the scope of this study, but they are sometimes used by researchers to achieve valid objects for feature visualization. What is important to know is that groups resulting from matrix factorization are not aligned naturally with spatial dimensions of layers. Instead, they are a combination of multiple channels and multiple spatial locations. To understand this better, we refer to figure 2.10 which shows the same concept but from the perspective of tasks in the CNN.

However, simply perturbing the input until the activation is maximized is a naive approach that leads to an image full of high frequency edges and patterns that merely activate the targeted objective severely but without actually providing any helpful visualization Olah et al. (2017). For that reason, such optimization needs to be regularized using one of many regularization methods used by researchers on this topic. These include frequency penalization which directly tries to reduce high frequencies in the generated image Nguyen et al. (2015); Mahendran and Vedaldi (2015) and transformation robustness which tries to generate images immune to visual transformations like rotation, scaling and translation Mordvintsev et al. (2015); Montavon et al. (2018).

Learned priors can also be used for regularization, in which case we learn a model on the real data to extract these visualizations, which is equivalent to searching through the datasets for images that activate the targeted objective. These produce the most realistic results and sometimes used as dedicated generative models rather than an explainability method Mordvintsev et al. (2015); Nguyen et al. (2016a, 2017).

Although the abstraction provided by matrix factorization helped gain insight

on the network as a whole, these visualizations are for one input image only. So, to gain a visualization over an entire dataset, further works were done to find a generalization form using multiple dimensionality reduction techniques like t-SNE Van der Maaten and Hinton (2008) and UMAP McInnes et al. (2018). One of the uses of t-SNE in that regard was in Nguyen et al. (2016b) where the generated images where clustered in a t-SNE map. Another great way of visualizing entire datasets was done with activation atlases Carter et al. (2019) which maps whole datasets into an atlas for each layer in the CNN where each part of the map corresponds to similar feature visualizations and the size of each generated image corresponds to the activation intensity.

These methods have been proven to give insightful visualizations that can explain what each part of the network was working on, what features it was supposed to detect, and how these features progress throughout the network from general edges and patterns in the beginning of the network to more whole parts and features of known objects in the middle of most classification networks and ultimately to full objects and creatures towards the output.

They helped researchers gain a deeper understanding about the capabilities and limitations of many classification networks. But that is exactly the catch in this discussion. They are only made for *classification* convolutional networks since those depend on known features in the input to classify a certain output as one object or another. It is clear that merely applying such methods out of the box to spectral reconstruction models would be naive at best, and meaningless at worst. This is what we will be discussing in the next 2 sections.

2.2.6 On Explainability for Regression Models

All the methods explained so far are designed for classification networks only. Their developers did not have any other kind of models in mind. That clearly presents a challenge since our task in this thesis is not a classification task, and the models we have to explain are reconstruction models in various forms and their output is not even discrete or limited by a small number of possible outputs. It is in fact closer to being a regression model than a classification one.

The work done in Letzgus et al. (2021) addresses this issue in the general sense of explainable AI (not just CNNs). It first shows how most explainability work was done towards classification and only very few were done for regression, and even those done for regression were developed for very specific tasks and no generic framework was introduced. However, it shows mathematically and by examples that some explainability methods could be adapted and applied correctly to regression problems if a number of considerations were taken care of:

1. The measurement unit used by the model should be preserved throughout

the explanation system. The unit here is simply the unit of the output. For example, if we have a model that inputs the years of experience and outputs the salary, the input unit is in years and the output is in euro (for example). These units have to be preserved throughout the explaining process.

Since in regression problems, units are important in quantifying the output, and so when an explanation model is applied, it has to preserve the units to retain a meaningful outcome. This is done when the explainability method is deemed "conservative" which means that it does not tamper with the unit of the original model. In the case of spectral reconstruction, this can be done with strictly visual methods since they retain the closest equivalent of a unit in that scope which is the intensity of a visual pixel.

2. In some models like temperature prediction models, the output could be more meaningful if we have a reference value (say 20 degrees). That way the output can be measured relative to that reference value (5 degrees above the reference 20 degrees for example). The paper argues that this adds more context to the explainability prediction. However, this only applies for models that output a measurement quantity (like temperature prediction or sales estimation) and does not apply to models like spectral reconstruction since these produce a spectral image that does not need a reference value to ground it.

3. If retraining is applied, like in Ant-Hoc methods, then some considerations need to be taken care of with both the data and the model. These considerations are mentioned in the paper. There are many details to that point, but we will not go through it in detail since the methods we are using are Post-Hoc and require no retraining or modifications to the original model, which is the desired case in the first place.

Many methods from the previously mentioned visual explanation methods conform to these considerations, so in principle, they could be used for spectral reconstruction. However, some methods that depend on having a target class like class-discriminative methods such as class activation maps would not be the first choice since they are built on a principle that does not exist in the scope of spectral reconstruction. Adapting them not to depend on a target class would simply reduce them to methods very similar to gradient or perturbation based methods Millan and Achard (2020).

Perturbation based methods are computationally intensive and, as we will describe later, could sometimes be made redundant with using gradient based methods in the particular case of spectral reconstruction models. However, perturbation methods could be a very good way to validate what other explainability methods

theorize. For example, if a gradient method showed visually that the network focuses on pixels with a certain attribute, we can test that theory by employing an adversarial attack (perturbation) by modifying the dataset to have that attribute in a controlled manner and test if the theory gradient methods provided us can be generalized or even quantified. So, although perturbation methods can be very time consuming and inefficient at *finding* meaningful correlations in the network, they can be invaluable at *verifying* them.

Therefore, gradient based methods as well as feature visualization seem to be the first candidates to use for our purpose, but we still need to narrow down and adapt these methods to our specific case as will be explained in later sections.

2.2.6.1 Final Note

It is clear at this point that no previous published work was directed towards explicitly interpreting or establishing trust for hyperspectral reconstruction CNN models, or any spectral reconstruction models for that matter to the best of our knowledge, which makes this work the first to touch on that area.

3 Methodology and Results

In this chapter, we will discuss the various methods used to explain multiple CNN hyperspectral reconstruction models. Each method, or test, focuses on acquiring a certain type of information that reveals specific attributes about the model. We will examine how the models differ from each other on a fundamental level and how we can classify them into different types based on the most prominent varying attributes.

Furthermore, we will try to find which fundamental information these models use to perform the reconstruction. We will need to test the most 'basic' of these models, so that we find the main building blocks for analyzing more advanced models. We will describe each interpretability method or test, why is it used, how it was adapted for this case, what to expect from it, the results it yielded, and their meaning to the models they were applied to.

Some of these tests are generally used like the ones mentioned in the literature review and applied to investigate general information about the models (investigative methods). Others are new and specifically crafted to follow up on leads provided by the investigative methods. We will start with the investigative general methods, and move from there following the interpretations they provide to get as much of a full picture as possible. This is why we decided to put both methods and results grouped together for each method in one chapter, since we need the results of one test to justify using the other and understand it better.

All of the methods were applied to a large variety of the most representative, and highest performing hyperspectral reconstruction models in literature. These include the following models listed in table 3.1 with their performance metrics shown. The models are ordered based on their overall performance from lowest performing (on the top) to highest performing (on the bottom). These metrics were extracted by testing all the models on the NTIRE 2022 challenge dataset for spectral recovery from RGB Arad et al. (2022), which is a benchmark for testing such models. Throughout this work, we will be using that dataset as our main testing target since most models in literature, and all the models discussed in this work, were trained on that dataset. Information about the error metrics used in table 3.1 are discussed in the next section. More information on the dataset is

Table 3.1: *Table depicting the hyperspectral reconstruction models tested and used throughout this work ordered by their overall performance.*

Model/Metric	MRAE	RMSE	PSNR	SAM
MultiScaleYan et al. (2018)	1.9465	0.1542	18.37	0.2548
HSCNN+Shi et al. (2018)	0.3814	0.0588	26.36	0.1042
HRNETZhao et al. (2020)	0.3476	0.0550	26.89	0.1014
HDNETHu et al. (2022)	0.2048	0.0317	32.13	0.0973
HINETChen et al. (2021)	0.2032	0.0303	32.51	0.0907
MSTCai et al. (2022b)	0.1772	0.0256	33.90	0.0909
MST++Cai et al. (2022b)	0.1645	0.0248	34.32	0.0835

introduced in the section after that.

3.1 Preparation

Before we begin with the tests, we discuss here some information about the error metrics used, the dataset, and the models.

3.1.1 Error Metrics

There are many error metrics that can be calculated. We choose those most common in literature and the most applicable for spectral error calculation. All models tested (table 3.1) used at least 2 of these metrics in their published work. These include the following metrics:

3.1.1.1 MRAE

Mean Relative Absolute Error (MRAE) is commonly used in hyperspectral reconstruction model evaluation. It is simple to calculate and measures the average distance between prediction values (predicted spectra in this case) and the ground truth spectra. Then, it is divided by the average of the ground truth values. It uses the following equation:

$$MRAE = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \frac{|Y_i - Y_i'|}{|Y_i|}$$

Where N is the number of data points. In our case, it is the number of bands in the reconstructed spectra. We choose this value to be 31 corresponding to 31 reconstructed bands. This is because it is the value used in most works in literature and it is what all tested models have been trained on. Y_i is the ground truth value at band i and Y_i' is the reconstructed value at band i. MRAE is sensitive to extreme (outlier) and small values.

Of course, this metric, like all other metrics we will discuss calculates the error for each spatial pixel (which has a spectra of 31 bands). That error is then averaged across all pixels to get it for a full image, and then averaged over all images to get the total error for the whole dataset leading to the final number shown in table 3.1.

3.1.1.2 RMSE

Root Mean Square Error (RMSE) is another popular metric that measures the average squared difference between the true values and predictions. It is defined by the following equation:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(Y_i - Y_i')^2}{N}}$$

Where N, Y_i, Y_i' and i are the same as defined previously. RMSE is also sensitive to outliers since it squares their values, but it is useful to evaluate the overall magnitude of the error.

3.1.1.3 PSNR

Peak Signal-to-Noise Ratio (PSNR) is a metric commonly used to evaluate signals in image and video processing applications and it is used as well to evaluate hyperspectral reconstruction. It measures how much the peak signal amplitude is higher than the noise signal which gives an idea of the reliability of the reconstructed spectra compared to noise. It differs from other error metrics in that the high PSNR value means a better signal (it is a performance metric) and it is quantified in decibels (dB) since it is a unit-less ratio. It depends on the Mean Square Error (MSE) to calculate the noise value and is defined by the following equation:

$$PSNR = 10 \log \left(\frac{Max^2}{MSE}\right)$$

$$MSE = \sum_{i=1}^{N} \frac{(Y_i - Y_i')^2}{N}$$

Where Max is the largest possible value for the spectra (for normalized spectra that value is 1). The rest of the symbols are the same as previously defined.

3.1.1.4 SAM

Spectral Angle Mapper (SAM) is one of the most useful metrics in spectral imaging and remote sensing. This is because it measures the similarity (or more precisely the dissimilarity) between two spectral signatures by calculating the angle between them in spectral feature space. It is agnostic to the absolute values (which are measured by the other error metrics) and is sensitive to the shape and signature of the generated spectra signal. It is calculated using the following equation:

$$SAM = \arccos\left(\frac{Y.Y'}{||Y|| * ||Y'||}\right)$$

Where Y, Y' are the ground truth spectra and the reconstructed spectra respectively. Their dot product is computed in the nominator to measure the similarity between them and denominator is for normalization, then the arccos is taken to compute the angle.

3.1.2 The Dataset

The dataset we will mainly use in this work is the NTIRE 2022 challenge dataset Arad et al. (2022). This is because most RGB to hyperspectral models (including all models to be tested in this work) were trained and tested on this dataset. It consists of 1000 hyperspectral images and their RGB counterparts with 50 validation and 50 test set images. The hyperspectral images were captured using a 204 band Specim IQ camera. The bands were resampled to 31 bands in the visible range (400 - 700 nm) with a 10 nm difference between each band and the next.

The RGB images were calculated from the hyperspectral ones using a camera response function. That function implements a realistic noise model. The exposure settings for the camera are done using an algorithm (defined by the authors and available online). However, the parameters this algorithm uses are unknown to the models using the dataset. The camera response function is shown in figure 3.1. The error metrics used in the challenge evaluation were MRAE and RMSE which we also use in our work. We mainly use the test set of 50 images since the models were not trained on these images.

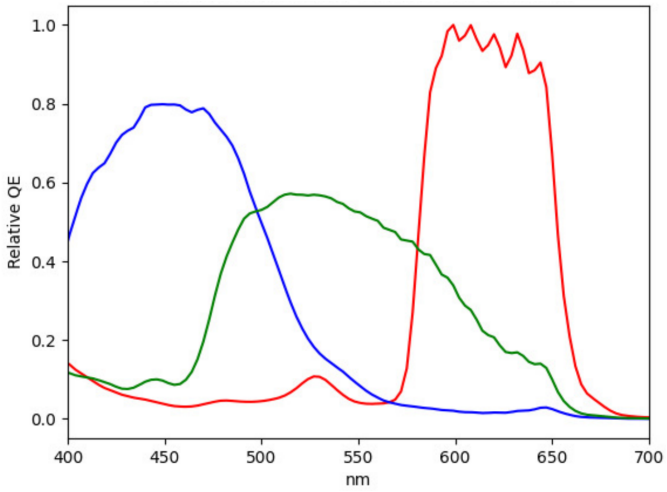


Figure 3.1: *Camera function adopted by the NTIRE 2022 dataset and challenge Arad et al. (2022).*

For some parts of the work we also use the reflectance dataset provided by Foster et al. (2007). This is because for some parts we need a reflectance images instead of the radiance images provided by NTIRE.

3.1.3 The Models

All models used were discussed in the literature review. The reason they were chosen is based on the following criteria:

1. These models include the most distinguishable milestones in performance in RGB to hyperspectral reconstruction. Meaning that on their release, some of them provided state of the art performance for that category (benchmarked by the NTIRE challenge from 2018 until 2022 Timofte et al. (2018), Lugmayr et al. (2020), Arad et al. (2022)).

2. These models include a variety of the different architectures and structures, including cascaded convolutions with residuals (like HSCNN+, HRNET, and HINET). U-Net models (like MultiScale). Spatial and spectral attention blocks (like HDNET). Full Spectral attention (like MST++).

3. Some of these models were made specifically for this task (like HSCNN+, HRNET, MST++, and MultiScale). Others were adapted from other similar tasks like Spatial super resolution (HINET) and SCI (MST and HDNET).

So, these models provide a large variety on different aspects which makes them a good representation to RGB to hyperspectral reconstruction models in general. They were also trained and tested on the same dataset and with same metrics we are using which helps in providing a fair comparison.

Now, we start with the tests which include: error maps, saliency maps, box tests, activation maximizers, and parameters sensitivity.

3.2 Error Maps

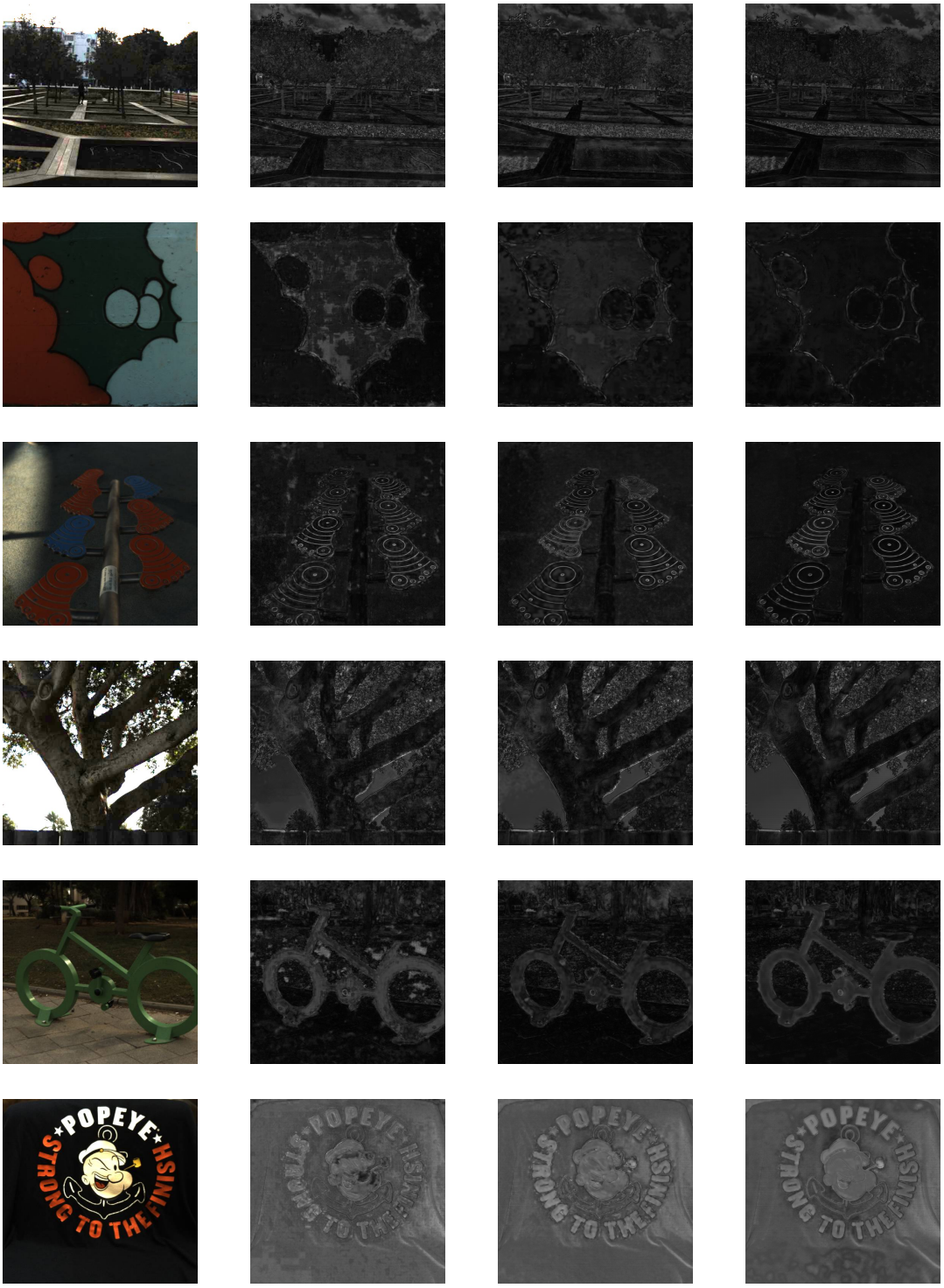
Error maps are a useful tool that can serve as a starting point for our investigation. They can show which spatial areas the models perform the best on, and on which areas they perform the worst. They are spatial heat maps that represent the error value at each spatial pixel calculated between the output spectra of the model and the ground truth spectra at each spatial pixel.

We calculated error maps to help us understand where the models have weaknesses in the everyday images taken from the NTIRE 2022 dataset. It is important to note that these are just examples since the whole dataset cannot be presented here. They do however help establish patterns that can be clearly seen in the error maps and that show consistency throughout different error metrics and different models.

The error metrics used in these error maps are the same defined in the Error Metrics section. All error metrics showed consistency in their resulting maps, so we only show here the error maps for the SAM metric for abstraction. The issue with error maps is that they are made for individual images and they need to be examined as such. That takes a lot of time to examine a large dataset. However, we show here only a few examples that give some insight on the particular areas these models fail in particular in the spatial domain in figure 3.2. More error maps can be found in the appendix.

The maps shown in figure 3.2 show a high intensity (white) at pixels with high error and low intensity (black) at pixels with low error. The darker the error map, the better the reconstruction quality. The error shown in these maps is Spectral Angle Mapper (SAM), but the same behaviour was seen in all other error metrics discussed before. These maps highlight some of the most prominent areas where the models work and where they fail. The whole dataset was examined, and since we cannot display all of it here we show a few examples that are sufficient to display the main ideas these error maps tell us about the models. They were chosen because they portray the general representation of looking at the whole dataset. Also, the error maps were generated for all models, but we show here 3 models that represent the rest since they are at the lower end of overall performance (HSCNN+), mid performance (HDNET) and best performance (MST++).

It is perhaps more helpful for us to look at areas of failure first, since these areas should have features that make it harder for the model to interpret them. This makes them valuable in understanding how the model works. Looking at the images, we can conclude the following points of failure:



(a) *Original.* (b) *HSCNN+.* (c) *HDNET.* (d) *MST++.*

Figure 3.2: *Example error maps for images from NTIRE for some of the models showcasing a variety of scenes with various characteristics. The bottom image shows a case of vague setup where errors are high. Some original images' brightness was adjusted for better visualization.*

1. The most clear points of failure are pixels where the camera sensor saturates like areas of relative high intensity that the dynamic range of the camera cannot perceive. This can be most seen in the areas of overcast sky in the first and the fourth image (from the top). These areas clearly have high error in all tested models even the most well performing ones (like MST++). This is understandable since the camera simply cannot get the color of these pixels due to saturation. And if the network does not have context of it being 'saturated sky' then it will falsely assume it is some other object and mistake its color for something else. Also, even if the network managed to figure out it is a saturated pixel, it cannot really reconstruct its spectra with the little information a saturated pixel provides (and the surrounding pixels).

2. Then we have edges especially thin edges of objects. This can be seen in all images especially the second, third, and fifth image. These edges do not have a lot of surrounding similar pixels around them which makes them very distinctive and so, very difficult for the network to reconstruct. This is because there is little information that can be gained from the pixels around them. Although this can be fixed if the model could look at the image as a whole and find similar pixels at the same edge. The reason this is not the case will be investigated and shown in the next section which will provide explanation to that phenomena.

3. Also, clearly seen in the last image, all models struggle with images that do not have a clear setting (outdoor/indoor) and that have very vague material. Even to a human, the last image is challenging to interpret where it is. By only seeing the curves and looking closely, one can see that it is an image printed on a shirt, but it is very vague which material that is. Even if the material can be deduced by the model, it will fail in detecting the illuminant since the setting is very unclear. In the rest of the images the setting is mostly outdoors, which has a common illuminant that the model is very familiar with (especially that most NTIRE images are shot outdoors under the sun). So, the models are well trained on outdoor illuminants, but we find it struggling with indoor images that have much more varying illuminants and shadows.

More points of failure could be seen in the images but the ones listed above are the most consistent across all models and all metrics so we will only list those for now. From these 3 points, the first and the third do not show limitations of the model as much as they show limitations of the setup. The models cannot perform well (inherently) with saturated images and images with no clear setup. However, they are worth pointing out since they give the user some guidelines on where to use such models. They cannot be used if the image has any saturation and it is preferred that they are used with outdoor clear set up images (unless the model was trained on a different dataset that had another dominant setup).

As for the second point, it does show a limitation with the models that can help

us understand it. Although these edge pixels do not have as much information as large areas with the same pixels, the model could still be trained to figure out that it is an edge. In fact, the performance with these edges can be seen to increase from low end models (like HSCNN+) to high end models (like MST++) as seen in figure 3.2. So, it is worth investigating how these models look at spatial features in the image and why they fail especially with edges and especially on low end models. For that, we have our next section that will use saliency maps to investigate how these models treat spatial information in the input image.

3.3 Saliency Maps

3.3.1 Concept

As mentioned earlier, saliency maps provide an image depicting which parts of the input image the model found most interesting (was activated the most by it). It is a useful tool to establish what kind of features the model looks at when making a certain decision on the input image, and so provides a good starting point for investigating hyperspectral reconstruction networks.

From the large variety of saliency maps discussed before, many depend on having a class (like Class Activation Maps or CAM methods). We cannot use such methods since spectral reconstruction models are not classifiers, so they do not have any output class. Other methods like deconvolution and back propagation Zeiler and Fergus (2014); Simonyan et al. (2013) could be applied in our case since they do not require a class. They instead back track the convolutions from the output back to the input regardless what form the output is as along as it has gradients (an output of a convolution layer). This is the case in hyperspectral reconstruction models, where usually the last layer is a n-kernel convolutional layer that maps its input into n feature maps representing the n bands of the hyperspectral image. In the dominant number of works, n is taken to be 31 bands, but that number is more of a common practice and not a requirement.

From these methods left, the choice was made for guided back-propagation Springenberg et al. (2014). This is because it provides the most fine resolution saliency maps from the methods applicable to use (discussed in the literature review section). Figure 3.3 shows some examples of applying guided back-propagation on one of the models (HSCNN+). For other kinds of models like classification networks, this is what a saliency map would look like. It highlights important parts, but it would focus on the features that made the network choose a certain output.

However, in this case, we do not have a firm expectation that the reconstruction CNN will focus on the eyes and ears of a dog for example to classify a dog, or

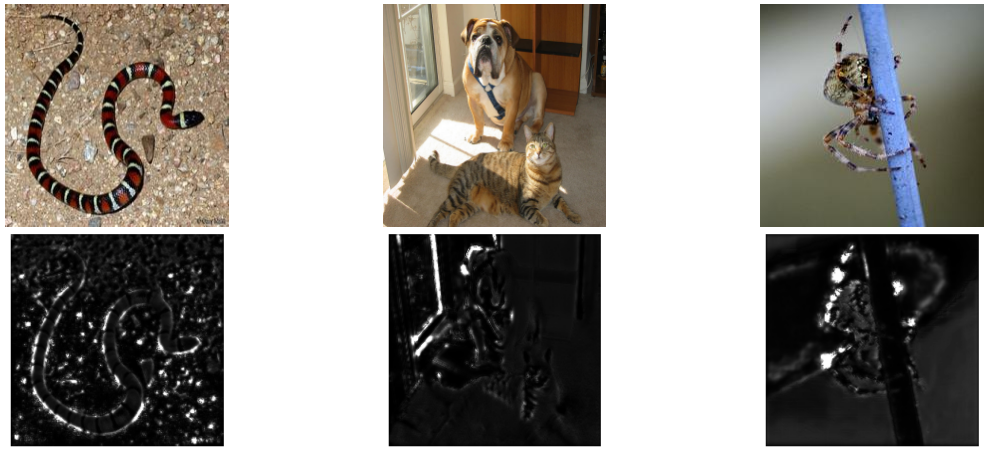


Figure 3.3: *Examples of Guided back-propagation applied directly to HSCNN+.*

same cats because it is not a classifier. Instead, the saliency is highlighting various parts of the image since what we see is a composite of saliencies taken on each reconstructed pixel. This means that regular saliency maps (like the ones shown in figure 3.3) do not portray the intended meaning for reconstruction CNNs (or any regression or transformer network for that matter). This is because these models do not just make one decision for each image, but they reconstruct a spectra for each pixel.

Although some pixels activate the network more than others, as can be seen by the figure, such maps do not give us the saliency for the decision (in this case spectra) made for *each pixel*. Just as saliency maps for classification networks look at the saliency for each class, in reconstruction networks, we need to look at the saliency for each reconstructed spectra (each spatial pixel).

Now, each reconstructed spectra consists of a $Single\,Spatial\,Pixel \times n\,bands$ represented by the feature maps of the output. So, to get a meaningful saliency map, we need to look at the saliency of each single spatial pixel back-propagated all the way from the output and projected back into the input. This carries a large resemblance to the concept of the receptive field, but the subtle difference is that this would yield the salient pixels *within* the global receptive field of the network. That tells us which spatial pixels of the input were important in reconstructing the spectra of that single pixel on the output, which provides more meaningful saliency maps for this case.

3.3.2 Method

This was applied by zeroing out all spatial pixels in the output feature maps except for one (the target pixel), leaving only 31 non-zero values (the output spectra of

the target pixel). Then applying guided back-propagation, and building a saliency map projected on the input. This was done for each model and each image in the NTIRE 2022 dataset (resized to 224 x 224). The target pixels were chosen randomly (10 random pixels for each of the 50 test images), but then centered at the middle so the total effect of all saliencies of all pixels of all images can be seen. The saliency maps were then averaged to get one saliency map for each model. The pixels chosen were random but on the condition that they are no less than 50 pixels away from the edge of the image.

Original Guided Back-propagation was modified by customizing it to each model. This is because each model had a different structure and non-typical activation functions (like Gelu in MST++). So, we made sure the flow of back-propagation followed the right paths to the input by manually examining all layers and activation functions in each model, then modifying the path accordingly.

3.3.3 Results

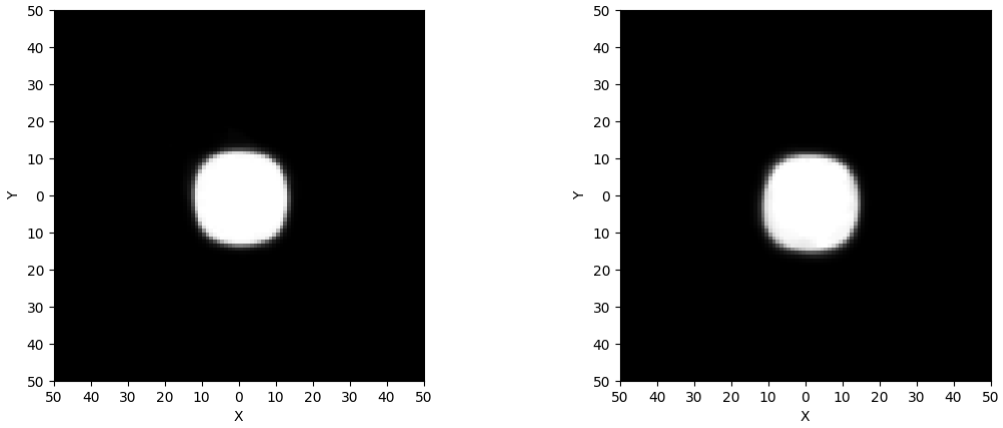
The results can be seen in figures 3.4, 3.5, and 3.6 where each saliency map is displayed as a flat 2D gray image image showing the saliency area for local models in figures 3.4 and 3.5. However, for global saliency models (3.6), we chose to keep a hue graded image and keep the full size of the image to notice the saliency spread around the entire image. It will be clear why that is needed soon. In all images, the saliency values are normalized to a 0 to 1 range to make it easier to follow and compare. The x and y axis are in pixels around the target pixel and the saliency area.

As in the figures, we found that models are best divided into 3 types depending on the spatial spread of their saliency. There are local, partially local, and global models:

3.3.4 Local Models

In figure 3.4, the saliency maps for the models HSCNN+ and HRNET are shown. It can be seen that these maps are highly local. The observed area is specific around the target pixel (here centered on the middle of the image), which seems to be simply the receptive field of the network. These models seem to make no attempt to expand the receptive field (and consequently the saliency) of the model to a global reach around the image, or to capture long range dependencies.

It can also be seen from the maps that the saliency is uniform across all the observed area, which means that the model treats all pixels in the observed area as equally valuable and makes no discrimination based on the location of the observed pixel or its proximity to the target pixel (as long as it is within the observed area).



(a) *HSCNN+ model random saliency map.*

(b) *HRNET model random saliency map.*

Figure 3.4: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*

This means that these models look at this specific local area around the target pixel when performing the reconstruction of that target pixel. Of course, this is quite limiting to the amount of information and features the model actually uses. This is because looking at such a limited area might hinder the reconstruction process or at least limit its potential.

However, although these models are of the least performing on our list of tested models, they still seem to perform relatively well. They outperform all non-CNN models and even outperform some old CNN models (like Arad and Ben-Shahar (2016); Galliani et al. (2017)), which was shown by the authors in the papers introducing these models Shi et al. (2018); Zhao et al. (2020). This means that although the area observed by the model is limited, it is (in most cases) sufficient to reconstruct the spectra.

Also, this aligns with the results shown by the error maps. More specifically, the failure to reconstruct pixels at edges. Since these models can only look at narrow areas, they cannot have enough information if the pixel is an edge, since then it will even have less information in that area. That is why edge pixels have a relatively low reconstruction accuracy especially from these 2 models that have completely local saliency.

Most importantly, this also means that there are features in this limited area that the models can utilize to reconstruct that spectra. Even without looking at the rest of the image, these features seem to provide enough information for reconstruction. This is counter to the way humans would perform reconstruction

for example. When we try to figure out the material of an object, we usually do so by knowing the context and full picture. We do not look at a very narrow area around the target. So, there are valuable features that the model can detect in these limited areas. Finding these features, and how these models use them is important to our task.

3.3.5 Partially Local Models

That said, if we look at figure 3.5, we can see that some models (like the MultiScale and the HINET models) have different saliencies that reach slightly more into the image than local models. Also, The saliency of these models is not uniform, since it can be seen that for HINET especially, the further the observed pixel is from the target pixel, the less saliency it has in a gradual decrease that starts towards the edge of the observed area. A similar -but less spread out- behaviour can be seen for the MultiScale network.

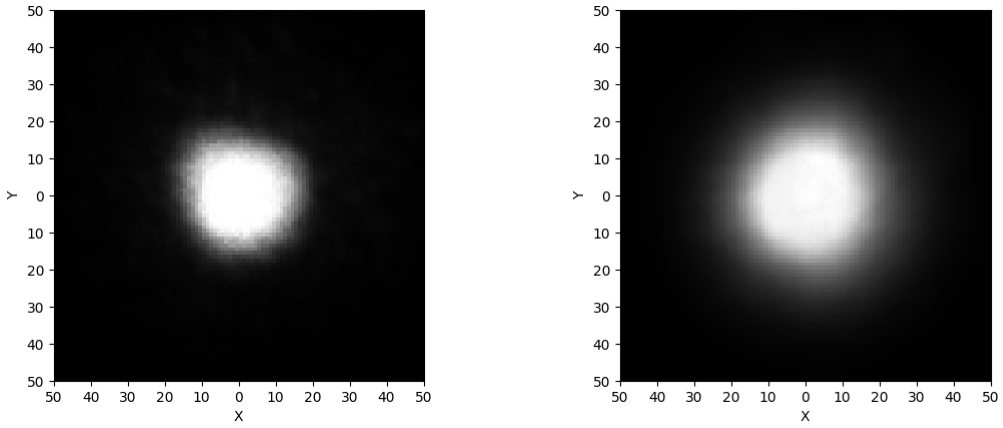
The area (in pixels) of the observed pixels for all the models can be seen in table 3.2. That area was calculated after thresholding the saliency values above 0.1 after performing the 0 to 1 normalization mentioned before. It can be seen (looking at the 'Random' column of the table only for now) that partially local models have higher saliency area than completely local models. This means including more features in the reconstruction process.

This also corresponds with the architectures of these models. For example, the MultiScale network is basically a U-Net with residuals, which theoretically allows it to have a receptive field larger than the observed area. However, the parts actually used from that large field are relatively small (see figure 3.5a) which means that the network is not using its full potential in extracting and using global dependencies.

3.3.6 Global models

In figure 3.6, global models are shown where the saliency is global across most or all of the image. This means that the model uses all the image in the reconstruction of each pixel, and uses long range dependencies, allowing it to capture information across a larger area and collect more features that could be beneficial to the reconstruction. This is why we chose to keep the image in its full size (224 x 224) in this figure to show these global saliencies.

However, it is still giving a significantly higher saliency to the area surrounding the target pixel, although that area is a bit smaller. These models seem to find a compromise between giving importance to the area around the target pixel and other areas, while also maintaining a gradual decrease in saliency the further a pixel is from the target.



(a) *MultiScale model random saliency map.*

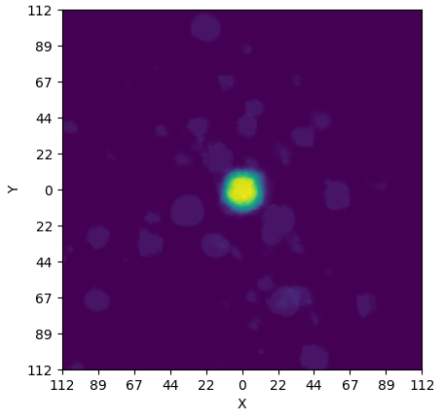
(b) *HINET model random saliency map.*

Figure 3.5: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*

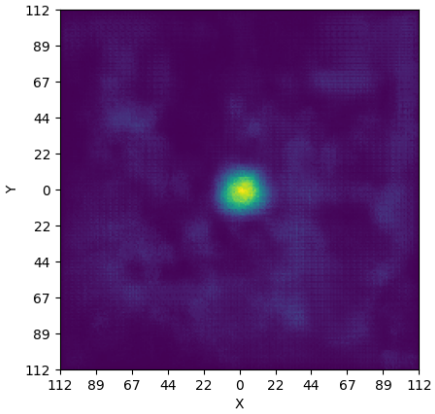
It is however worth discussing each model of this category separately since although they all possess global saliency, they have very different ways of using it. For example, although HDNET has saliency reaching to the edges of the image, that saliency is not at all uniform or even consistent across the image. As can be seen in 3.6a. It has peaks spread in various locations around the image. We cannot deduce direct correlation between these locations and specific features here because this saliency map is an average of all saliency maps of the whole dataset, so it is not limited to specific features. It could also be that these saliencies spread around the image are more the result of the network behaviour than a reaction to specific image features.

Looking back at the performance metrics, MST and MST++ perform the best relative to all other models. This cannot be directly connected to them having large saliency since HDNET also has a large saliency but does not perform as well. However, as we mentioned, HDNET's saliency is highly random compared to the more consistent one in MST and MST++.

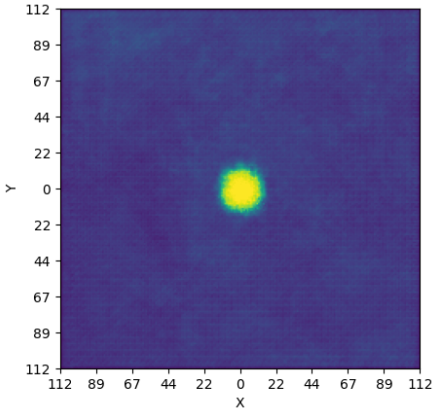
This aligns with the architecture of HDNET (discussed before) which includes both spatial and spectral attention blocks. This allows for long range dependencies but not as much as the full spectral blocks used in MST and MST++ since it only contributes to half the attention. Also, HDNET has the smallest saliency point (saliency just around the target pixel) in all tested models. It covers a relatively small area and decreases almost right after the target pixel. It can also be seen



(a) *HDNET random saliency map.*



(b) *MST random saliency map.*



(c) *MST++ random saliency map.*

Figure 3.6: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. Full size images are displayed to show global saliency.*

that MST++ has a significantly stronger global saliency than any other model (it is also uniform), which correlates also with its high performance. It is a similar case with MST but on a smaller scale.

3.3.7 Edge saliency

Since error maps provided the insight that all models are significantly limited in areas of the image with edges, it is worth exploring the saliency at these areas specifically to check if the models (any of the models) react to the existence of an edge in the image considering this is where the models mostly fail in the

Table 3.2: *Table depicting the area of saliency (observed area) in pixels with a threshold of 0.1 (after 0 to 1 normalization) for all tested models on multiple types of target pixels.*

Model/Target	Random	Edge	Flat	Image Edge
MultiScale	1316	1348	1326	993
HSCNN+	627	654	634	583
HRNET	649	683	671	579
HDNET	733	741	727	555
HINET	2123	2218	2101	1504
MST	2227	2241	2236	1563
MST++	50081	50127	50097	35733

reconstruction. To test that, we first used a canny edge filter to detect strong edge pixels in the image as shown in figure 3.7. A high threshold was used to get the strongest edges only (larger than 0.6 for normalized values between 0 and 1). Then, we again used the modified guided back-propagation (same as before) to generate saliency maps but this time only for those edge pixels (randomly chosen from the set of edge pixels).

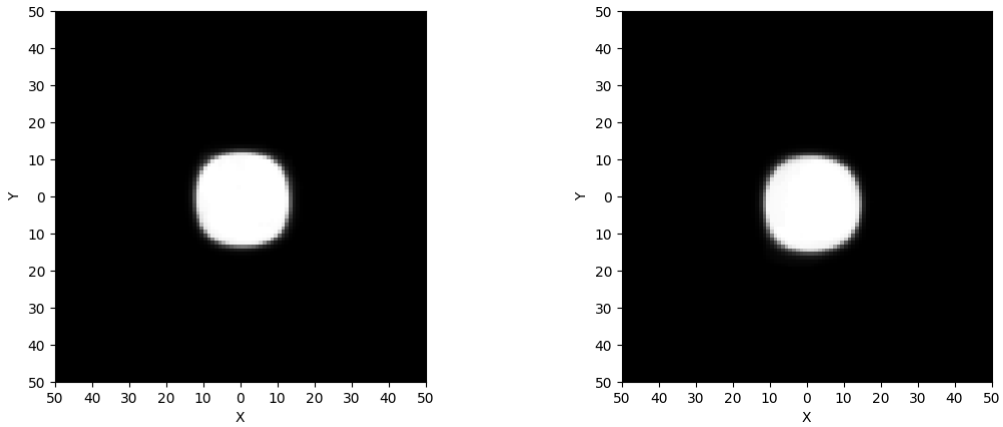
Results for that are shown in figures 3.8, 3.9 3.10. As can been in the figures, there is no real distinct difference between saliency maps for edge pixels and saliency maps for randomly chosen pixels (previously discussed). None of the networks seem to recognize edge pixels or treat them differently than any other pixel. As mentioned before, there is an inherent difficulty in reconstructing spectra for edge pixels. The aim of this test was to check if any of the networks tackles this by extending (or compressing) the range of the observed area when met with an edge.

However, this does not seem to be the case for randomly chosen edge pixels. This does not exclude the possibility that the network can shape its saliency depending on the texture and features around the target pixel, since these images are pooled over an entire dataset and do not show individual images and their detailed textures.

Also, table 3.3 shows the Spearman correlation between saliency maps targeted at randomly chosen pixels with edge pixels (first column). It can be seen that the correlation (which has a maximum of 1 and a minimum of 0) is very high between

(a) *Original image.* **(b)** *Edges detected.*

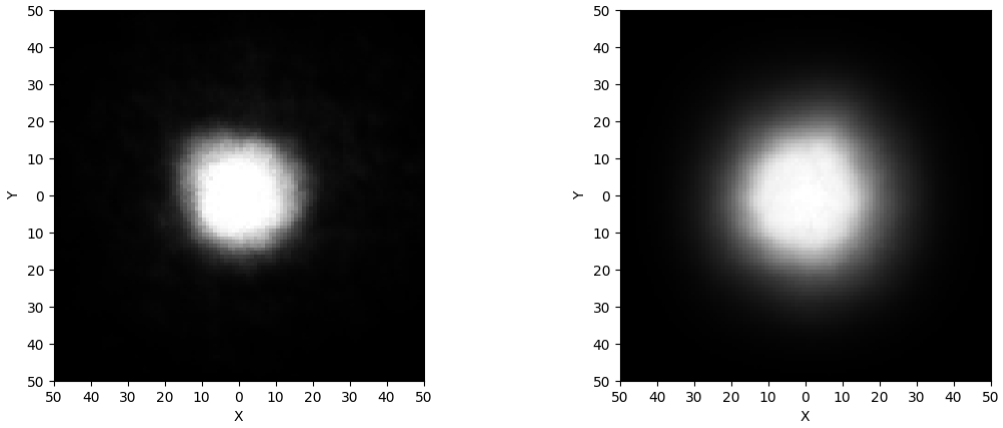
Figure 3.7: *Detected edges using Canny edge filter to isolate them and generate their saliency maps.*



(a) *HSCNN+ model edge saliency map.* **(b)** *HRNET model edge saliency map.*

Figure 3.8: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*

the two. This numerically supports and quantifies that these models do not have any different saliencies for edge pixels. Further numerical support is the area of these saliencies shown in table 3.2 (under Edge and Flat columns) which show very similar areas to random pixels, although they are slightly larger.



(a) *MultiScale edge saliency map.* **(b)** *HINET model edge saliency map.*

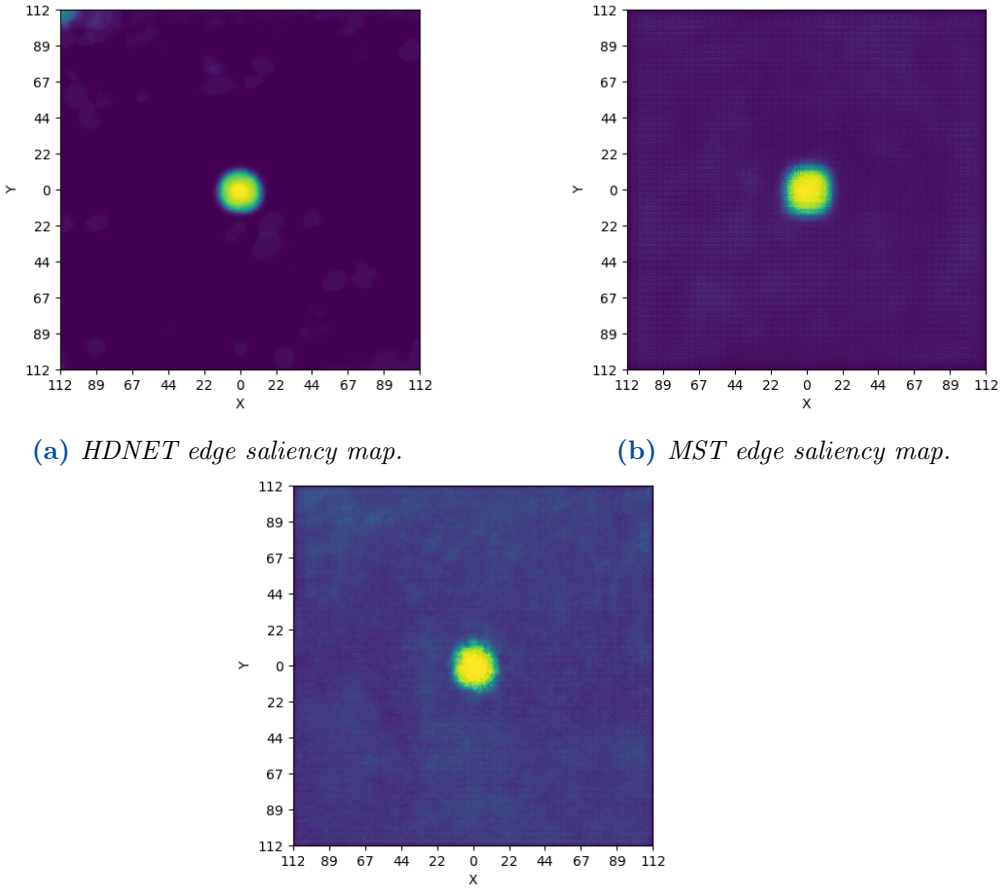
Figure 3.9: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*

Table 3.3: *Table depicting Spearman's correlation between saliency maps targeted at randomly chosen pixels and edge pixels and pixels falling on the image edge for all tested models.*

Model	Random/Edge	Random/Image Edge
MultiScale	0.951	0.896
HSCNN+	0.994	0.986
HRNET	0.998	0.985
HDNET	0.975	0.789
HINET	0.990	0.860
MST	0.976	0.735
MST++	0.960	0.873

3.3.8 Image Edge saliency

One last possible candidate that could alter the saliency is the edge of the image itself (rather than the edges of objects in the image like the previous section). This comes from the suspicion that the networks might (through training) develop awareness that such pixels have inherently less information since almost half the typical observed area is not there. Another possibility is that the models behave differently with pixels falling on the very edge of the image due to architecture characteristics specific to the model.



(a) *HDNET edge saliency map.*

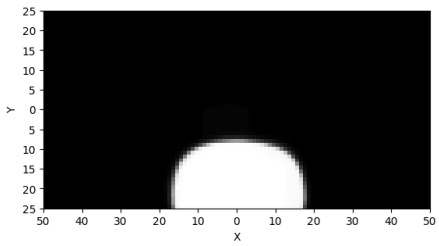
(b) *MST edge saliency map.*

(c) *MST++ edge saliency map.*

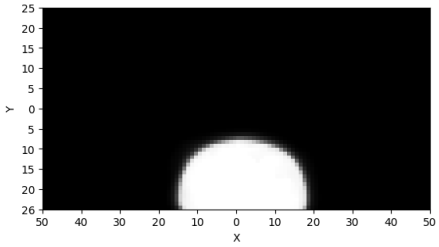
Figure 3.10: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency, in flat and 3D for edge pixels (full size images are displayed to show global saliency) averaged over all the NTIRE 2022 images.*

Either way, it is worth studying the saliencies at such pixels. We first point out that all pixels in the previous sub sections (random and object edge pixels) were chosen on the condition that they are no less than 50 pixels away from the image edge to make sure they are not part of the image edge category so we can study those separately here. In this subsection, we present saliency maps for pixels randomly chosen but with the condition that they fall on the very edge of the image (for example: pixel 100, 223 for an image of size 224 x 224). Figures 3.11, 3.12, and 3.13 show the results for that.

If the models would have the same saliency for image edge pixels as random or object edge pixels, we would expect them to have half the saliency on the edges of

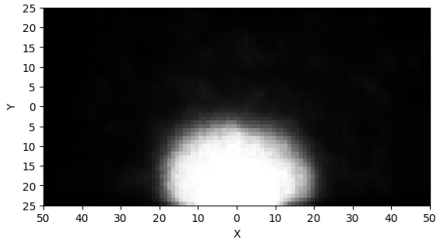


(a) *HSCNN+ image edge saliency map.*

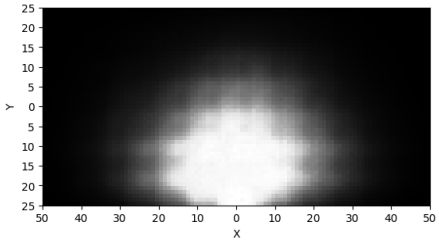


(b) *HRNET image edge saliency map.*

Figure 3.11: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for image edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*



(a) *MultiScale image edge saliency.*



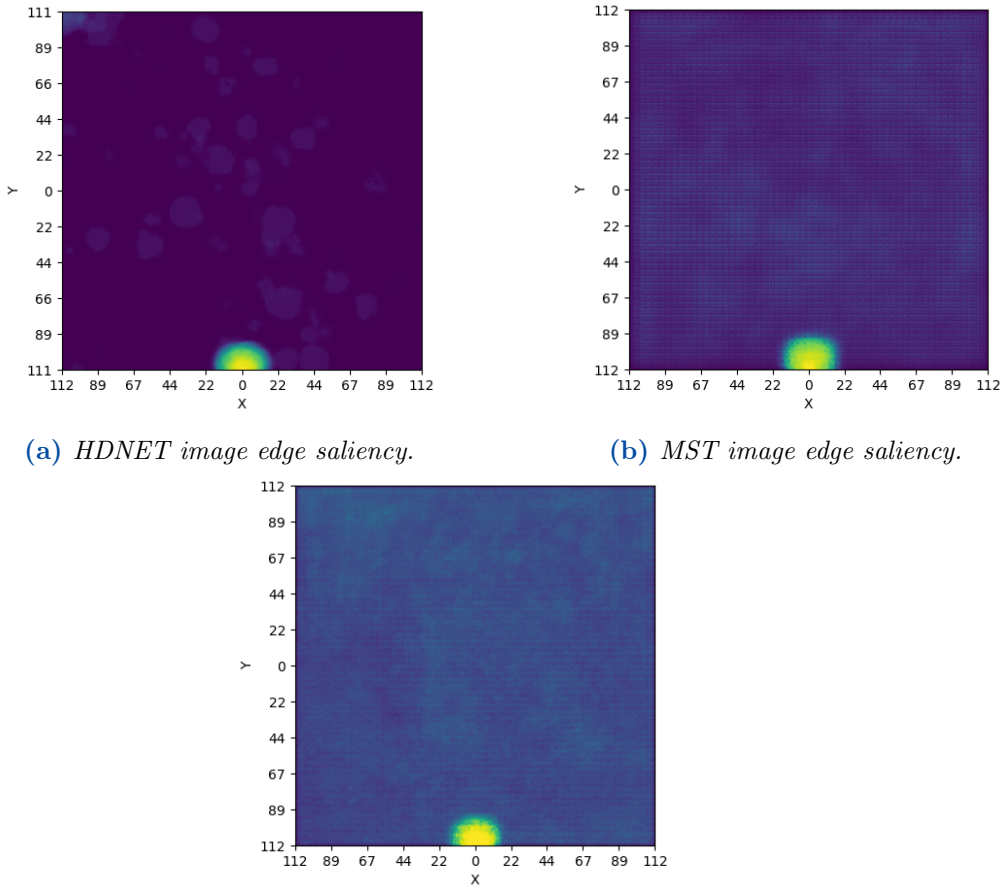
(b) *HINET image edge saliency map.*

Figure 3.12: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for image edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images.*

the image (half the saliency of a normal pixel).

This is the case in 3.11a for the HSCNN+ model. However, this is not the case for all models, since it can be seen that most other models have more-than-half the typical saliency observed in random pixels. This is quantified in table 3.2 under the image edge column where it can be seen that the area is more than half that of the random or edge or flat pixels. This can also be seen in the images shown although it is not very clear since the change is subtle.

However, it is clear for the MultiScale and the HINET models in figure 3.12 where the saliency is not merely halved but 'lifted' further inside the image. Table 3.3 shows also that the Spearman correlation with random pixels is less than that of edge pixels and with significant difference for some models like HDNET, HINET, MST, and MST++. This means there is also a difference in the shape of the saliency when the pixel is on the edge of the image (it gets slightly 'squashed') and expands which allows it to get more information from the image than it would with a typical halved saliency.



(a) *HDNET image edge saliency.*

(b) *MST image edge saliency.*

(c) *MST++ image edge saliency.*

Figure 3.13: *Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency, in flat and 3D for image edge pixels (full size images are displayed to show global saliency) averaged over all the NTIRE 2022 images.*

However, although this change is noticeable and consistent for all models (except HSCNN+), it is not large especially to some models, but it still suggests the models' capability to adjust their saliency in some cases. The point of these investigations is to figure out if these models treat any parts of the image differently, and from the results, we can see that they do that for the *edge of the image* but not the *edges of objects within the image*. Applying a similar change to object edge pixels could help the network tackle the issue of bad performance for such pixels and potentially improve the model's performance.

3.3.9 Single Image Analysis

So far, we looked at pooled average saliencies across an entire dataset. This was important to give validity to the data and show patterns emerging from all the dataset. However, it does not allow us to observe detailed shapes of these saliencies and how they work on particular images. This is what this section is about.

We look at figure 3.14, which shows some examples of individual saliency maps (for single images). The top examples are for HINET and it can be seen how the saliency is adapting to the texture and information surrounding the target pixel (here it is x = 100, y = 200). The image with the snake shows how the saliency follows the body of the snake. The same can be seen for the body of the cat in the second image. In the spider image, the pixel does not coincide any specific body or texture (it is on the background) so it seems to have a more uniform round saliency. The same behaviour for HINET was also noticed in HDNET.

As for HSCNN+ (middle image), the saliency is almost constant regardless of the features or the object around the target pixel. It is varying very mildly between the 3 examples (the same applies to all examples). This highlights that some models can adapt their saliency while others cannot. Similar to HSCNN+, MultiScale and HRNET also have constant saliencies.

Finally, for MST++ (last row), the saliency is global and seems to also follow certain objects (like the body of the snake and the spider). That gives the model the ability to choose collected features from the most prominent objects in the image. A similar behaviour was also seen in MST (but on a smaller scale).

3.3.10 Test Conclusion

Completely local models depend fully on that area surrounding the target pixel. Also, all models have the largest saliency in that area. Because of that, it can be concluded that to at least all the tested models, this area is key to perform the reconstruction. The size of that area is clearly reflected on the performance of these models. Furthermore, the shape of that area (uniform or gradually decreasing) seems to also differ from one model to another while also affecting the performance.

Having global saliency around the entire image in a uniform manner seems to help the model perform better, making state of the art models like MST++. For example, the only modification that MST++ has over MST's architecture, was to increase the feature extraction in long range dependencies Cai et al. (2022b). This highlights the importance of extracting global information from the image.

However, it is important to note that to merely perform reconstruction, the model *does not need* such long range dependencies. Completely local models can perform well without it, which means that the fundamental information strictly

Figure 3.14: *Examples of saliency maps on a single pixel ($x = 100$, $y = 200$) for single images for HINET (top), HSCNN+ (mid), and MST++ (bottom).*

required for reconstruction lies in that limited local area. Of course, here we are talking about what is required from the models. Instead, it could be that a model is using wrong information (or inherently limited information) to do its task, which would mean that the model cannot be trusted to perform well in all cases. However, we need to investigate what kind of information that is and how the model uses it to be able to judge its validity. This is part of what this work will continue to investigate.

As for edges, there does not seem to be any change in the way models treat edges of objects in the image. However, the models do expand the saliency if the

pixel is on the edge of the image. The shape of the saliency also changes in that case. Looking at saliency maps for individual images shows that some models follow defined features in the image, while others simply have a box or a circle highlighting the observed area.

3.4 Box Tests

3.4.1 Concept

Now that it is established that hyperspectral reconstruction models mostly depend on a local area around the target pixel, we need to identify which parts of that local area it is using. And what kind of features it is using within that area. For example, although the saliency map shows that the model uses that area, does that mean that it depends on all of that area? Or does it only need parts of it and uses the rest to gain more accuracy (like global models but on a different scale)? Also, is it actually using the features and information around the target pixel, and if so, to what extent? To answer these questions we start with what we call the Box Tests. These tests are of a perturbation and adversarial nature since they limit and control the amount of information available to the model. Here are the 3 types of Box tests and how they could help gain a better understanding of what these models actually use and to what extent:

The first looks at a certain pixel in the image (being reconstructed, target pixel). And removes the information around that pixel (with a black box). Then gradually increases the area around it giving the model more information about the surroundings of the target pixel, meanwhile monitoring the reconstruction error of that target pixel. Figure 3.15a shows the starting point of that test. Only a small area of 5 x 5 pixels is provided while the rest of the observed area is a black box. Then, the area available is increased gradually until we provide full information (normal image) similar to that in figure 3.15b with a large box.

The second looks at a certain pixel in the image (being reconstructed, target pixel). And removes the information around that pixel (with a black box). Then gradually increases the area around it. However, instead of giving it the real image data, it replicates the same pixel in the surrounding area (painting with the same value, a box around the pixel), meanwhile monitoring the reconstruction error of that target pixel. Figure 3.15c shows the starting point of that test. Only a small area of 5 x 5 pixels is provided while the rest of the observed area is a black box. Then, the area around the target pixel is replaced with replicas of the target pixel (in the example it is a reddish color) until we reach a large replicated box similar to that in figure 3.15d with a large box.

The third looks at a certain pixel in the image (being reconstructed, target pixel). It does not remove any data, but replicates the same pixel in the surrounding area (painting with the same value, a box around the pixel). Basically sacrificing real image data with fake replicas of the target pixel and increasing it gradually, meanwhile monitoring the reconstruction error of that target pixel. Figure 3.15c shows the starting point of that test. Only a small area of 5 x 5 pixels has the replicas so it starts with an image similar to the real image. Then, the area around the target pixel is replaced with replicas of the target pixel (in the example it is a reddish color) until we reach a large replicated box similar to that in figure 3.15d with a large box.

The idea is to compare these tests relative to each other and how the model performs on each one. If the model uses features in the observed area like shapes and grain, it needs the real image data to perform well. In that case test 1 should witness an increase in performance with more area given since it gives more of the real image data the larger the box is (replaces black pixels with real image data with increased box size). However, test 3 should witness a decrease in performance with more area given since it replaces real image data with replicas of the target pixel. This is the expected and most probable case and we call it Case A.

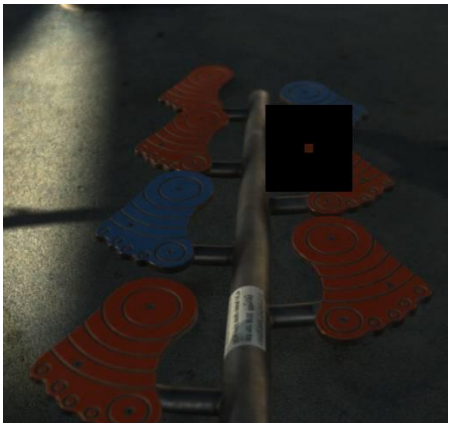
The reason we chose to replace real image data with replicas of the target pixel is to test if the models can make use of that color (that of the target pixel) in the reconstruction and how much does it depend on it compared to other features in the surroundings of the target pixel.

This is because the target pixel's RGB data offers the closest estimate on the real RGB of the target pixel. If the model simply learns a mapping function between the RGB values (after color correction) and the spectra, then it could use that RGB value to map it to a spectra. If that was the case, all the model has to do is estimate the illuminant and discard it, then it will directly map the clean RGB values of the target pixel to a spectra it learned. So, it could be that there is heavy dependency on that target pixel RGB value, and we wanted to use these tests to validate that as well (two birds with one stone).

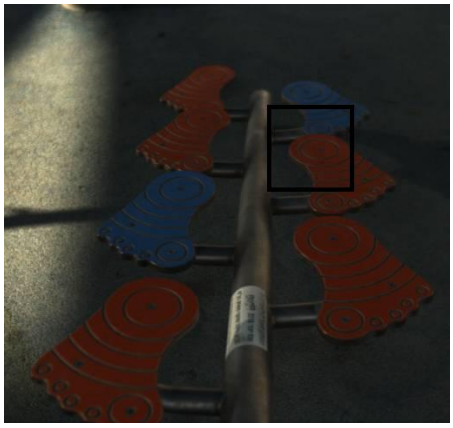
This is where test 2 comes in play. It replaces black pixels with replicas of the target pixel, so it would evaluate how well can the model perform with only that RGB value, and provide insight on whether the model needs a lot of observed-area-features and shapes or not.

If the model uses shapes and features, but also uses the RGB value of the target pixel, then one would expect test 2 to witness an increase in performance that is proportional to how much the model depends on that RGB value. We call this Case A1 since it adds on Case A and is not mutually exclusive with it.

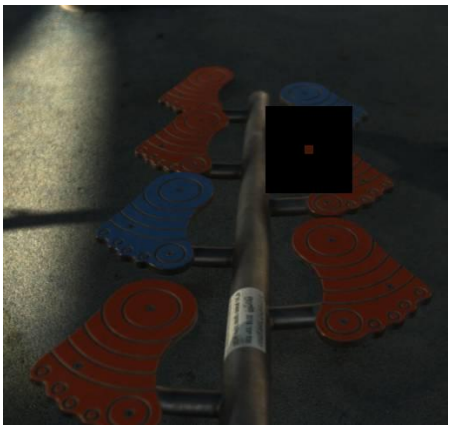
If it does not depend on it at all, then it should not have any increase in performance since that value will be equivalent to a black pixel. We call this Case



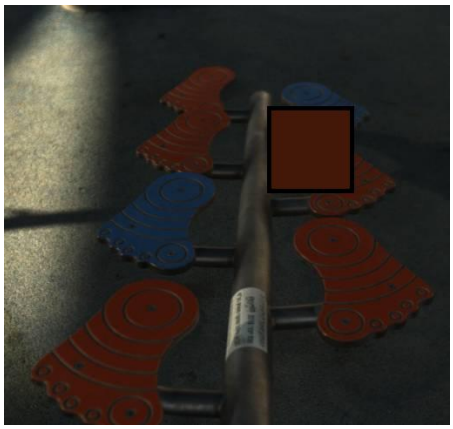
(a) *Test 1 small box.*



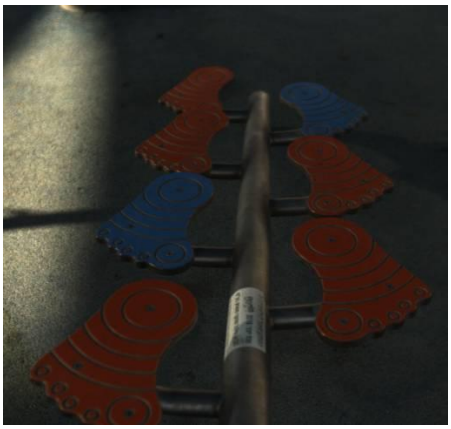
(b) *Test 1 Large box.*



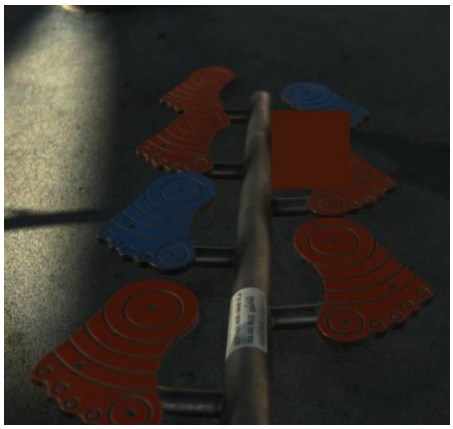
(c) *Test 2 small box.*



(d) *Test 2 Large box.*

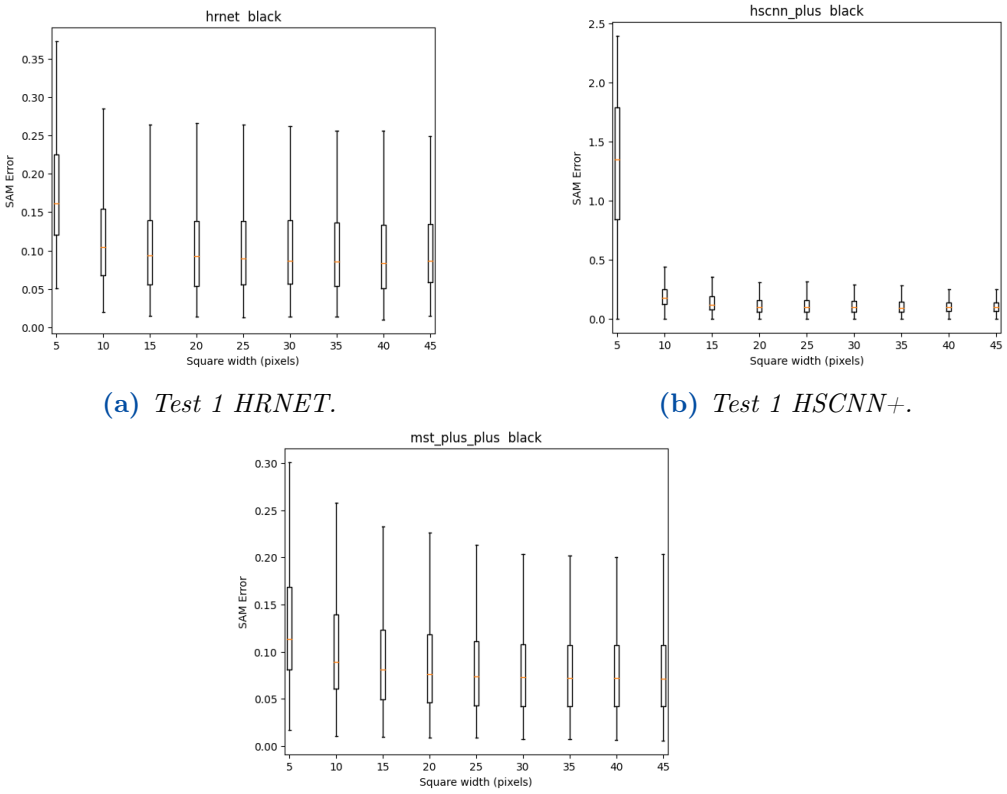


(e) *Test 3 small box.*



(f) *Test 3 Large box.*

Figure 3.15: *Examples showing the images fed to the models in each one of the box tests and their varying size from small to large.*



(a) *Test 1 HRNET.*

(b) *Test 1 HSCNN+.*

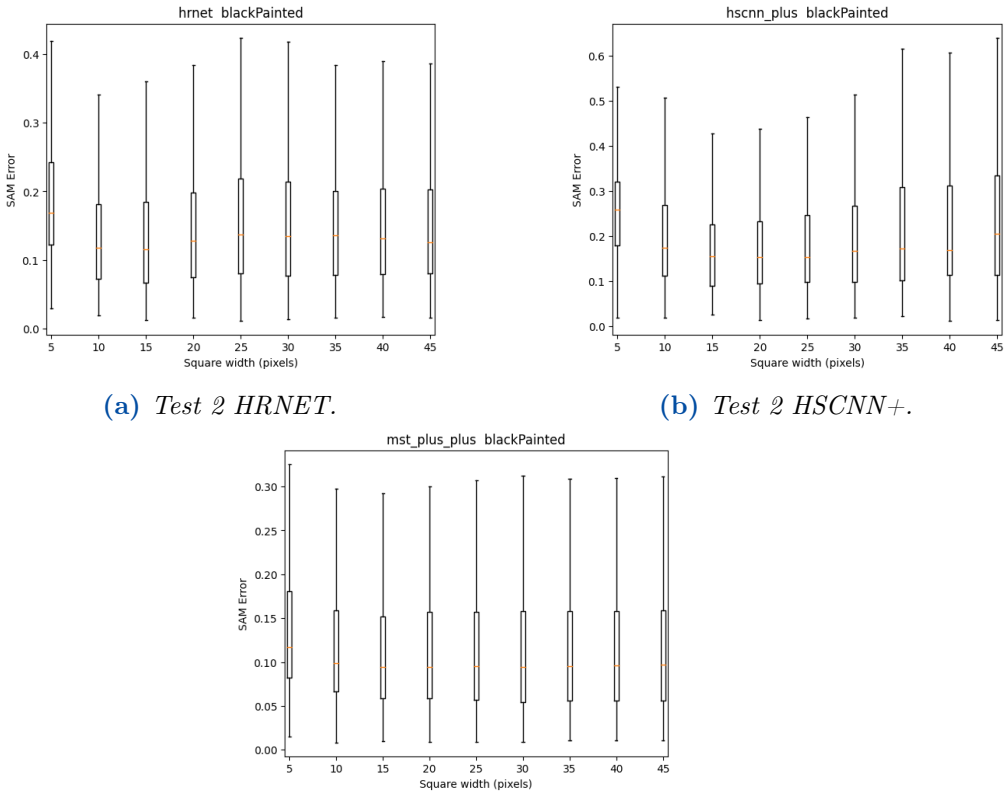
(c) *Test 1 MST++.*

Figure 3.16: *Results of test 1 for some of the models.*

A2. If the model does not depend on features but only depends on the (RGB+color correction) and then direct mapping, then test 3 would witness and increase in performance for regular images that have true RGB values. This is Case B which is a highly improbable case, but excluding it goes a long way to gain some level of assurance that we are on the right track.

We applied these tests with all error metrics mentioned before and on all models. We performed the tests on 10 random target pixels for each image and on all the images of the NTIRE 2022 dataset. Each one of these 10 pixels in each image had a box drawn around it with size of 5x5 (smallest) to 45x45 (largest) with steps of 5 making a total of 9 boxes each with a different size for each pixel for each image for each test in each model. Clearly this was a computationally extensive process which was expected from a perturbation based method.

The reason for choosing size 5 as a minimum is simply because starting with 0 or 1 would give a very high error in tests 1 and 2, and would be redundant in test 3 (because it would be a normal unperturbed image). As for the upper size of 45, it was chosen from the saliency maps discussed before since that was the average



(a) *Test 2 HRNET.*

(b) *Test 2 HSCNN+.*

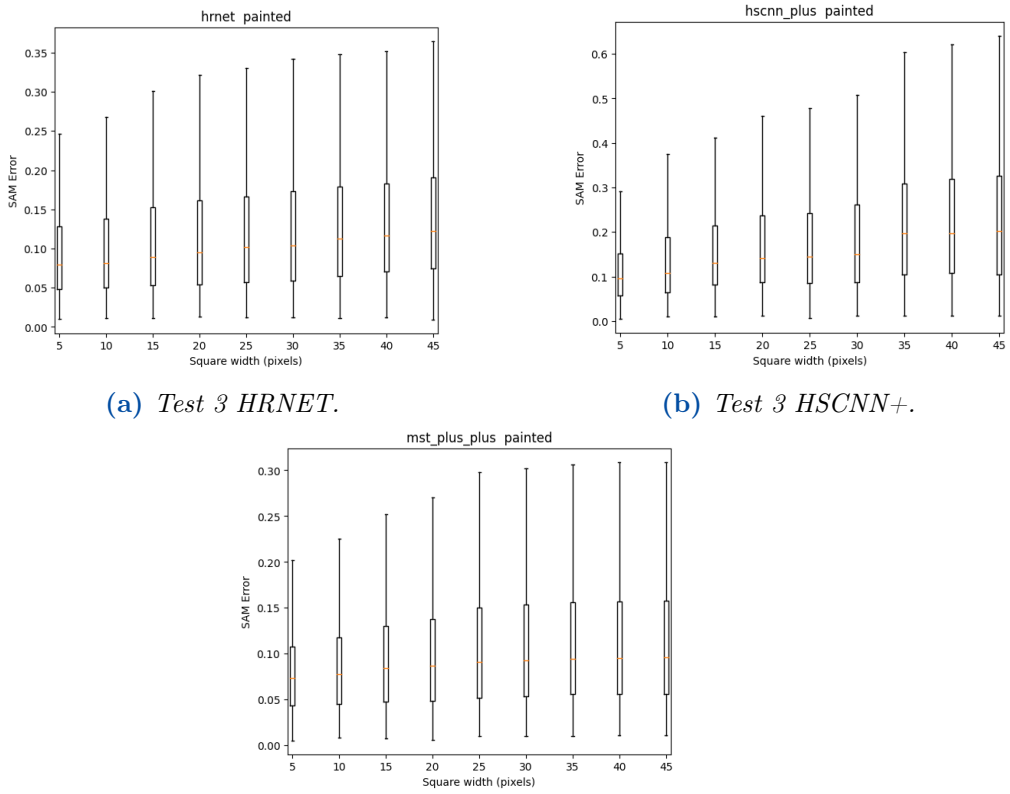
(c) *Test 2 MST++.*

Figure 3.17: *Results of test 2 for some of the models.*

width of the saliency area. Also, it was noticed that all tests saturate after that width which is natural since the model only uses that area anyway. All models were tested but we show here 3 of them only for abstraction since the rest had the same patterns and behaviour just with varying scales depending on the model's performance which was already established. All error metrics had similar results so we only show SAM here as a representative to the rest and for abstraction.

3.4.2 Results

Now, looking at the results for test 1 shown in figure 3.16, we can see the SAM error decreases with the increase of the box size which is normal because it gets more information. However, the interesting part is the scale at which that error drops. The most significant error drop happens between size 0 and 5 (not shown in the images because it is trivial) but the second most significant drop happens between sizes 5 and 10 pixels especially in HSCNN+. After that, the error drops gradually.



(a) *Test 3 HRNET.*

(b) *Test 3 HSCNN+.*

(c) *Test 3 MST++.*

Figure 3.18: *Results of test 3 for some of the models.*

This implies an even smaller area of high importance to the HSCNN's performance which is the area of approximately 10x10 around the target pixel. That is where most of the model's information comes from even though it is looking at the whole saliency/observed 45x45 area. However, this only applies to HSCNN+.

Looking at test 3 in figure 3.18, we can exclude case B since the models show consistent increase in error with more replicas of the RGB value on the cost of real image data and features. So now it is either case A1 or A2 depending on the outcome of test 2. Looking at test 2 shown in figure 3.17, we can see there is first a decrease in error that varies between each model and the other, then the error increases again.

This preliminary decrease in error shows that the model does use the RGB value to some extent. The increase in error after acts as further proof that the model depends on the surrounding features as well (when the features get to the point where they could be misleading, the error started to increase again). However, both the increase and the decrease of the error is very slight and cannot be said to be significant, but the pattern is quite consistent across all models and all images.

So, it at least shows that the models use the RGB value of the target pixel as well as the features in the observed area surrounding it, both to some extent in the reconstruction process (case A1).

3.4.3 Test Conclusion

This is reassuring since it shows that the models use feature information of both colors as well as textures/shapes in the observed area surrounding the target pixel. However, it still does not uncover the way these features are used. Also, these tests quantify how the error changes with increased information given to the models. Although there is a pattern clear in the results, the significance of that error change is not established here. This is because the change in that error seems to be highly gradual and not highly consistent or significant. More work has to be done here on establishing how significant that change in the error really is.

3.5 Activation Maximizers

3.5.1 Concept

As mentioned in the literature review section, many works used the concept of feature visualization, which aims to visualize what each part of the model is looking for in the input. That 'part' of the model could be a single neuron, or a group of neurons, or a whole channel, or a full layer, or any combination of those. That combination is known as *objective*. That objective is the target of feature visualization in a sense that the visualization will encompass not only what this target is doing, but how is that target is being used in the context of the whole network. For example, if the objective is a kernel, feature visualization tells us how the network uses that kernel in its place from the start of the network up to that kernel.

Details regarding how they work were discussed in the literature review. However, we also discussed that these methods were directed towards classification networks. With regular classification networks that work on regular images and classify for example the animal showing in the picture between cat or dog. The network (given that it is trained well and working properly) would look for features unique for cats and others unique for dogs and try to classify the image based on these features. So, if we apply feature visualization in that case, we would expect the results to be visual features understandable to us humans. This is because the features the network would be looking for in that case are visually identifiable, like the ears or eyes of a dog, or a certain visual pattern or texture. This is shown in figure

3.19. The yellow box shows the workflow of feature visualization introduced in Olah et al. (2017) being applied to an objective (in this case a kernel in a layer) in a classification network (GoogleNet, layer mixed4a kernel 143). The input is a noise image that was optimized iteratively to activate that part of the network. The output of that optimization provides meaningful results (what seems to be eyes and dog fur) meaning that this part of the network is specialized in detecting dog eyes and fur and any patterns that look like it. This is feature visualization. Now, this was later used as a building block for activation atlases that identify what each part of the network is specialized in detecting. This is shown in the top part of figure 3.19.

However, if we apply the same procedure and concept directly to a hyperspectral reconstruction network, what would we expect the features to be? It cannot be showing a visual pattern or an identifiable object like the eye of a cat, simply because the network is agnostic to such features. The process of reconstruction does not depend on such features as we so far established. It even depends (for local baseline models) on local features around the reconstructed pixel. To verify that, the same feature visualization procedure used in Olah et al. (2017) was applied to the 64 kernels of some of the layers of the hyperspectral models tested. This is shown in the red box of figure 3.19. It can be seen that the output does not provide much intelligible information. It changes the colors slightly but the rest is still very noisy and hard to understand. All other results for other models and other layers are very similar. Just different random colors like the ones shown in the mid right part of the figure.

It can be seen that no clear features can be discerned using that method. This is to be expected since we expect these networks to look for much more subtle and local features like color, texture, shade. Also, for each pixel reconstruction, only the observed area around it is processed.

We can see that various colors are already showing in the feature visualization output, because a feature like color and hue is easily discernible, but not much else can be seen, and even these colors are very vague and not meaningful since they do not have any context. A kernel can be looking for the color green for example, but we cannot know the object or feature that color is for. Is it the color of a particular object, or the illuminant, or just a general sensitivity to the color green?

To deal with that issue, we propose a modification to the procedure followed by Olah et al. (2017). First, instead of having the input as a randomly generated noise image, we propose the input to be an actual image. An RGB image from the NTIRE 2022 test set. This makes sure that whatever changes the optimization process applies to the images will have a contextual meaning that can be understood by us humans. This is because it is not generating a complete feature visualization, but instead modifying an image to have the most interesting features highlighted

while maintaining the context of the image.

This is not to be confused with activation atlases which also use a real input image. However, in activation atlases, that input image is only studied using generated feature visualization building blocks as seen in the top part of the diagram. In our case, we use that image as the input to the iterative optimization process itself.

Second, instead of letting the optimization process reach saturation (finding the image with the maximum activation), we restrict the optimization with a special type of regularization. Regularization is generally used in feature visualization (discussed in the literature review) and a lot of work was done to come up with good regularisation methods to make sure the image does not turn into pure high frequency patterns.

However, the proposed regularization is different than the ones mentioned previously since it is directed towards keeping part of the features of the original image intact. The idea is to keep the parts of the original image that are already interesting to the objective (it is activated by them) while modifying the parts that are not as interesting (the objective was not activated by them).

That way, we can find new features while maintaining the context provided by the original image. This is seen in the bottom part of figure 3.19 where our method was applied on the 10th kernel of the 5th and 15th blocks (top and bottom blue boxes respectively) of the HSCNN+ model. These blocks consist of multiple convolutional layers, so here we are applying the method to the last convolutional layer of each block. There is a total of 30 blocks in HSCNN+. We can see how more information can be gained than with feature visualization since we have context information maintained from the original image. We can see that backgrounds are replaced meaning they are not the main objective of these kernels. Also, the dog and the cat are colored differently than the rest of the image, meaning a sensitivity to these colors by these kernels. Grass is replaced by a bluish hue by the block 5 kernel (top) but kept intact by the block 15 kernel (bottom). Eventually, most kernels in a layer (here 64 kernels) are collected together in a collage to showcase that layer's features (the collage on the bottom right part of the figure).

Simply put: for each kernel, parts of the image that are 'interesting' meaning that they activate that kernel, are kept intact. Meanwhile, parts of the image that are not 'interesting' meaning that they do not activate that kernel strongly, are replaced by a color or a pattern or a shape that does.

Regularization is done by ensuring that the overall 'modification' to the original image does not exceed a threshold α. Overall modification is calculated by the following simple equation:

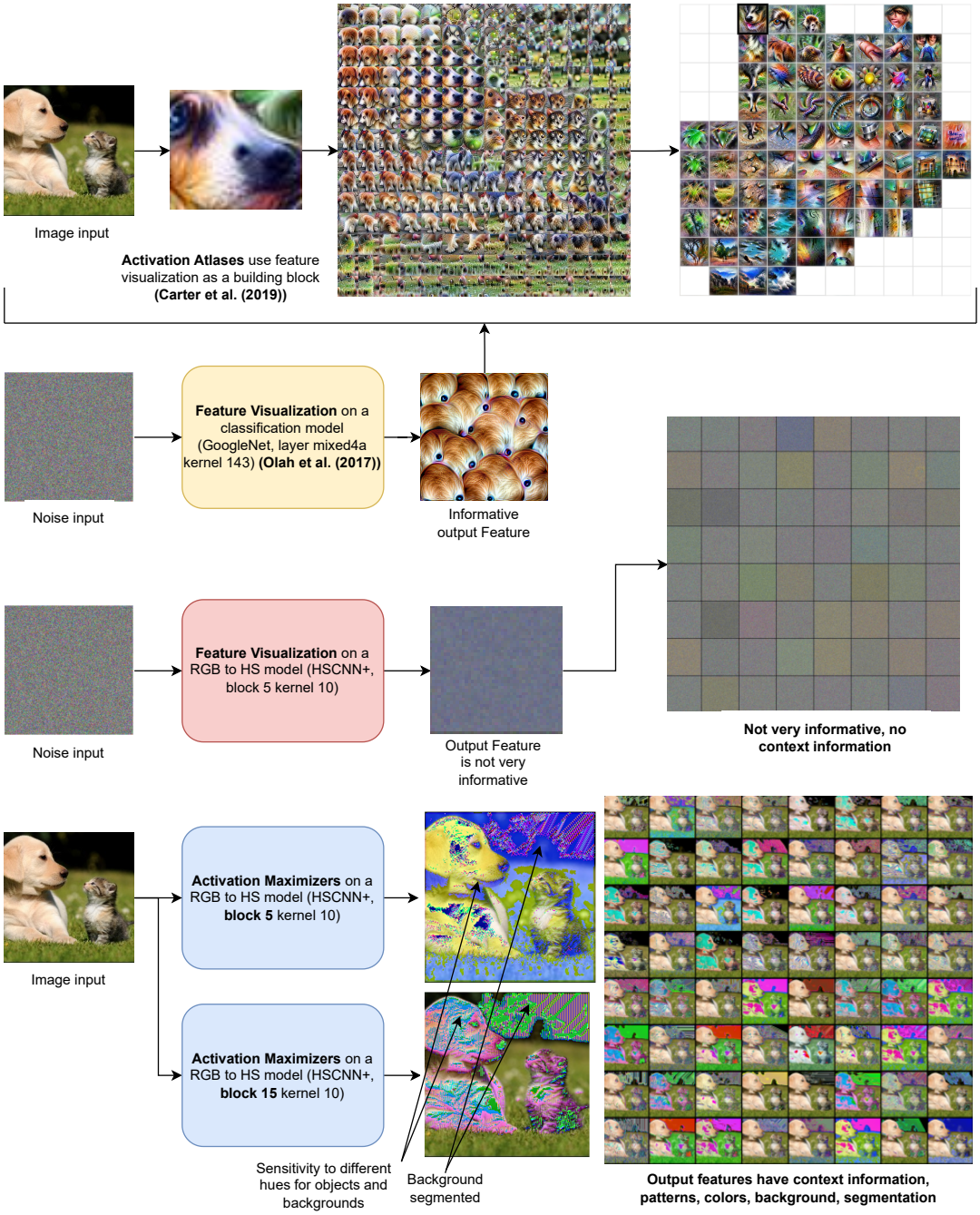


Figure 3.19: *Difference between feature visualization Olah et al. (2017), Activation Atlases Carter et al. (2019), and the proposed method (Activation Maximizers).*

$$\left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{|O_i - M_i|}{|O_i|} < \alpha$$

Where O_i, M_i is the original image and the modified image respectively (both RGB with normalized values of 0 to 1), flattened into a single dimension array of length N and α is the threshold for modification. This equation merely calculates the average difference between the original and the modified image and makes sure it does not exceed α which was found to produce best results if it was 0.6.

Furthermore, the regularization method used in Montavon et al. (2018) was also used on the modified pixels. It applies general transformations to the generated images (rotation, scaling, etc) and makes sure the generated images are robust to these effects. In this case however, these transformations are not applied to the full generated image (original + modified) but to the modified parts only since we would like to keep the original parts visible and not transformed beyond recognition. This particular method was used because it showed the best results for methods that do not require priors or training as shown by works that evaluated such methods Carter et al. (2019); Olah et al. (2018, 2017). This is a new method that we call *Activation Maximizers*, since it focuses on maximizing specific activations for specific objectives under certain regularization rules.

3.5.2 How it was applied

To apply such visualization, feature extraction was used. Feature extractors are segments of the main network made in a way that the output of a feature extractor is the activation of a certain layer of the main network. For example, if a network has 5 convolutional layers, and we need to peek into the output of the 3rd layer, we construct a feature extractor that has replicas of the first 3 layers and monitor its output. An example of feature extractors are shown in figure 3.21.

Of course, it is not that simple with complex networks that have many residuals and skip connections. Choosing the right layer that has complete and meaningful information is also an important part of the process since many layers in complex networks depend on concatenation and could only provide the results of other layers in other parts of the network.

Also, although the concept of feature extractors may sound similar to the concept of hooks which is available in deep learning platforms like Pytorch, it is not the same. Hooks merely extract the activation of a layer in a network while that network is being run. However, that output activation cannot be back-propagated directly through the sub-network back to the input in multiple iterations. This is because the activations do not change the architecture and maintains one big network and the calculation graph is constructed across the whole network.

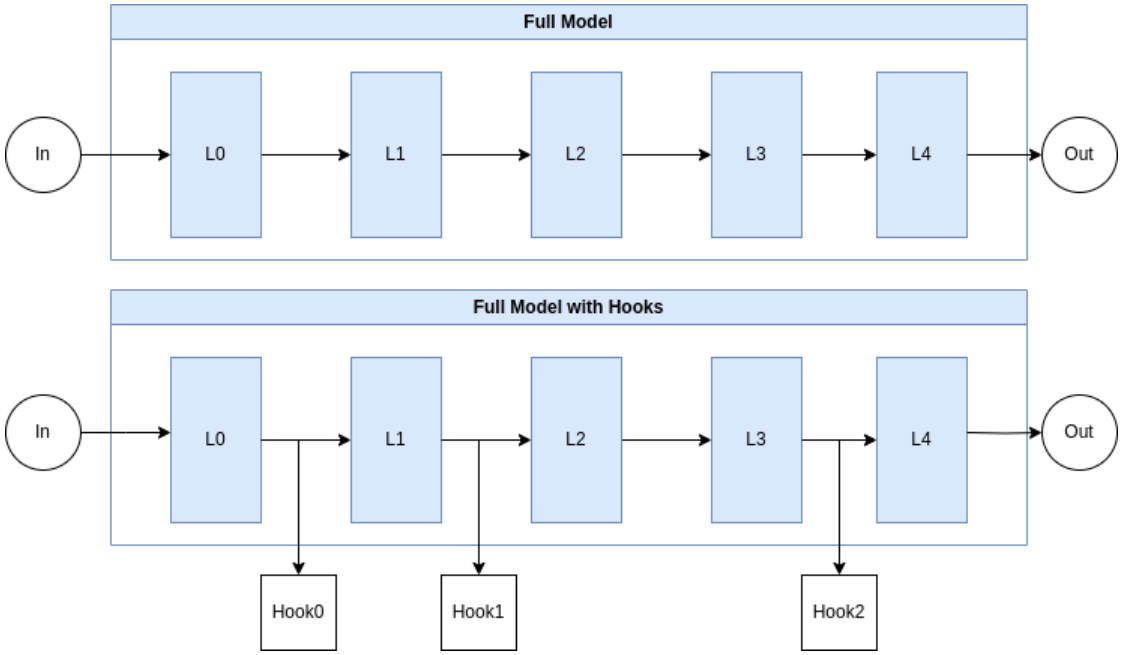


Figure 3.20: *Hooks applied to a model. The model keeps its original graph.*

If we merely want to extract activations, then hooks would be perfect since they are fast and do not require a lot of memory (we will use hooks in other parts of this work where we extract activations). However, in this case, we also want to optimize the input based on the extracted activations of each monitored layer. So, each feature extractor will have different weights than the others at the same location because the values will pass through different paths. That means that we need a separate calculation graph for each feature extractor which holds these weights and computes the back-propagation back to the input for each iteration.

Figures 3.20 and 3.21 show the difference between hooks and feature extractors (on an arbitrary example model) and how only feature extractors (acting as sub-models) are capable of holding a separate computational graph that allows for optimization of the input based on the output of a particular layer from the main model.

All models tested were already trained on NTIRE and their weights are preserved (frozen) throughout the whole process. After defining multiple feature extractors from the model on the desired locations, and making sure they keep the trained weights the model had, we run each of the images (also from NTIRE but from the test set) on each of the feature extractors.

We extract the output of that feature extractor, and we use gradient ascent to maximize that output. This output is the activation of a certain kernel in a certain layer averaged over all the pixels in the input. Then, we back-propagate

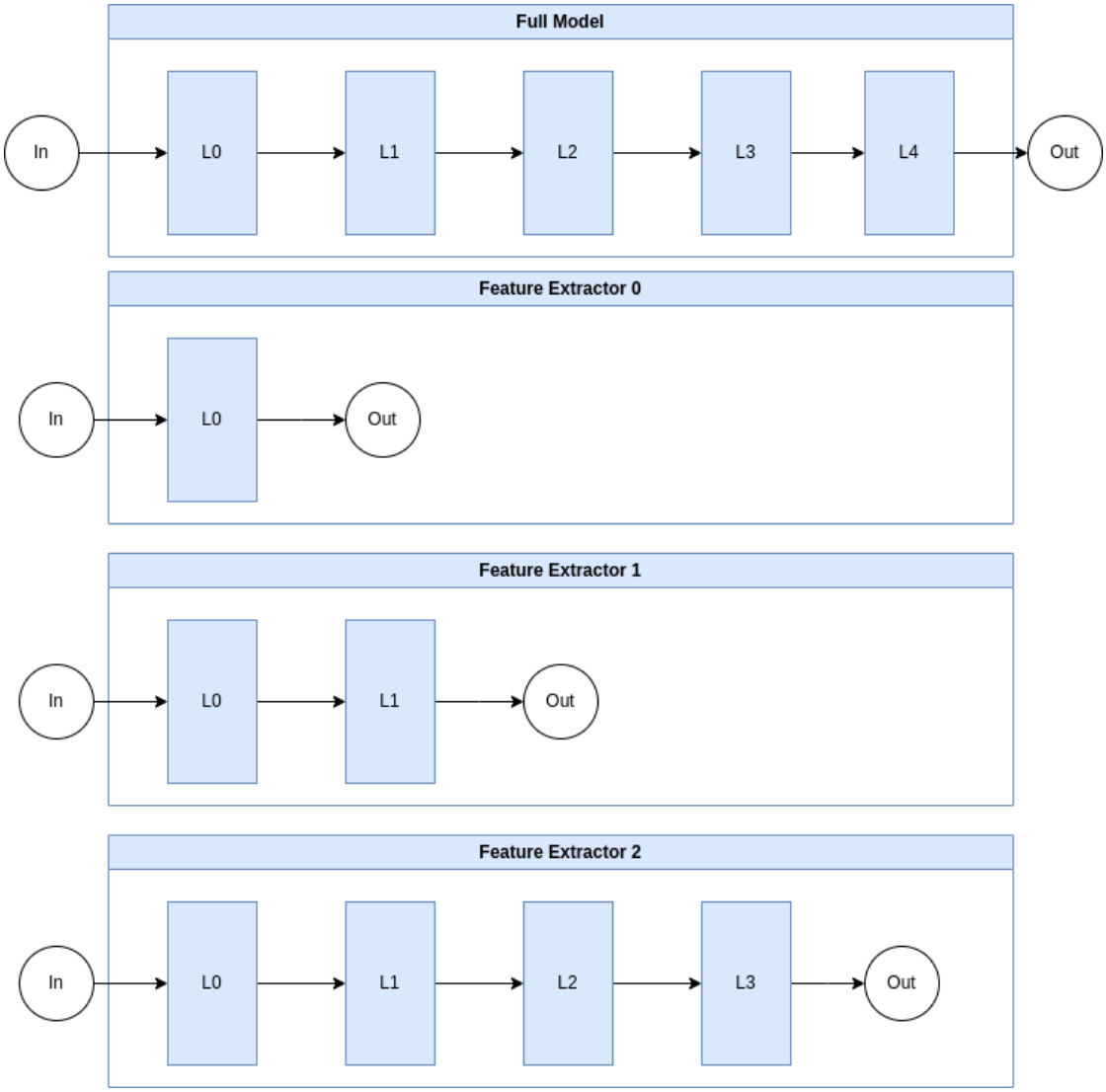


Figure 3.21: *Feature Extractors as sub-models each with a different graph.*

back to the input and use a learning rate of 1 to adjust the input based on the activation. We repeat the process for 50 iterations while all the time applying the regularization mentioned before.

This is applied to each of the 64 kernels in each desired layer (the objective) for each image. The result is 64 variations of each of the NTIRE test set images for each layer representing what each of the 64 kernels in that layer are dedicated to detect.

Not all models had 64 kernels at all tested layers, so the number 64 was changed to 32 sometimes depending on the targeted layer.

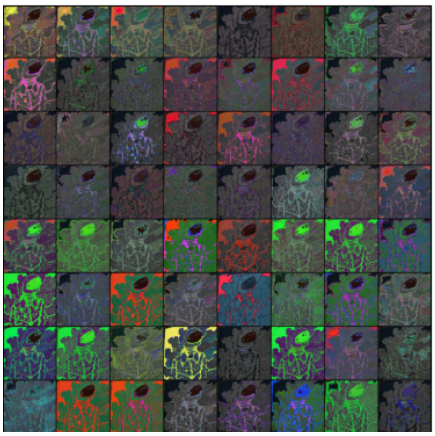
Figure 3.22: *Original images for the examples used in figures 3.23 and 3.24.*

Of course, it is clear that such computation will take a lot of time especially on heavy models which is the case for most spectral reconstruction models. for all 50 images of the NTIRE test set, this took a maximum of about 22 hours for 8 layers on the heaviest tested model (HSCNN+ and HRNET) on a 24 Gb RTX 4090 GPU running Pytorch with Cuda. If more layers are required, more time will be needed to run them.

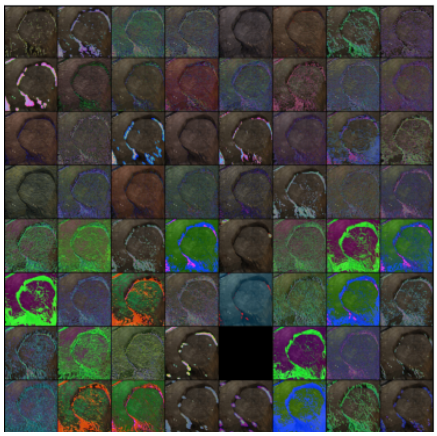
3.5.3 Results

Applying this to some of the layers of the HSCNN+ model for example yields the results shown in figure 3.23. Original images (from the NTIRE 2022 test dataset) for the examples used are shown in figure 3.22. In the figure, visualizations for 64 kernels (convolutional filters) that comprise each of 3 layers in different areas of the network are displayed. The first is the input block of the HSCNN+ model, representing the beginning of the model. The second, is the convolutional layer at the end of the 15th block out of 30 total main blocks that make up the HSCNN+ model representing the rough middle of the model. And finally, the near end of the model is represented with the 29th block. Two examples from the NTIRE dataset are used to give an idea of the generation. The rest of the dataset was tested and had similar results but not displayed here to save space.

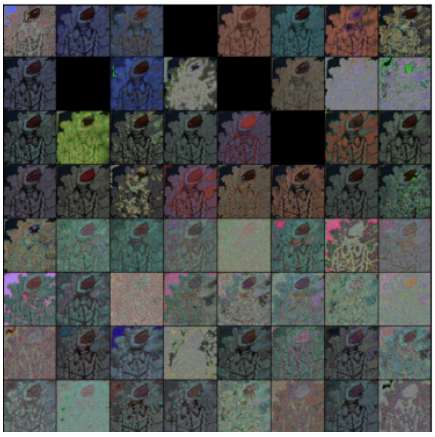
Each of these images represents what that particular kernel in that particular layer is optimized to detect. The features appearing in the generated image represent the features that kernel was trained to detect and activate at. This figure is a starting point to give an idea of what a network detects at certain areas.



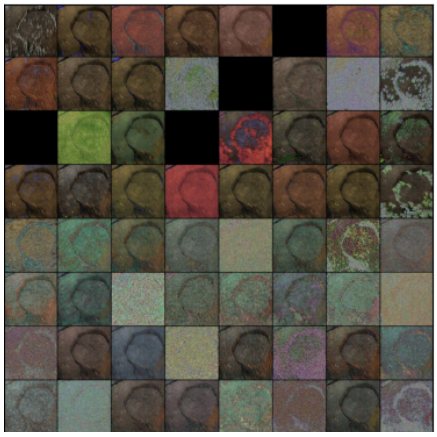
(a) *Input block example 1.*



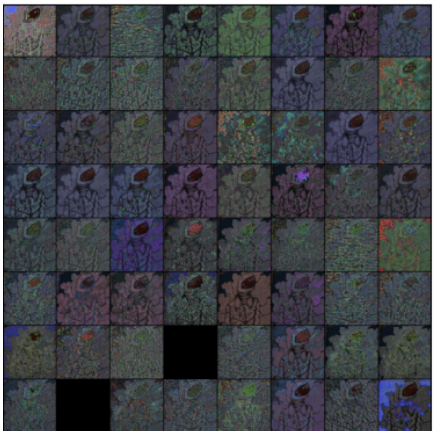
(b) *Input block example 2.*



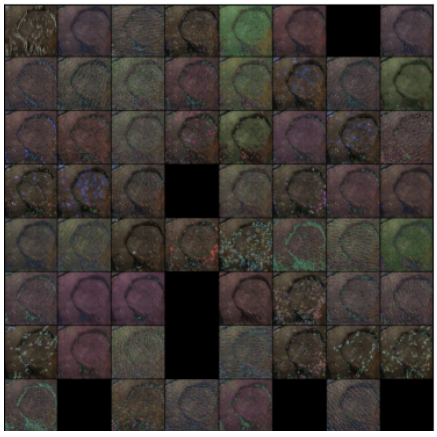
(c) *After block 15/30, example 1.*



(d) *After block 15/30, example 2.*

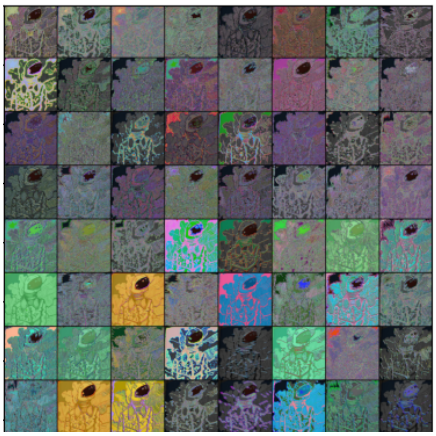


(e) *After block 29/30, example 1.*

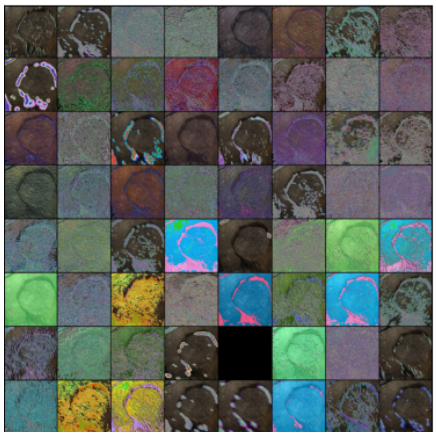


(f) *After block 29/30, example 2.*

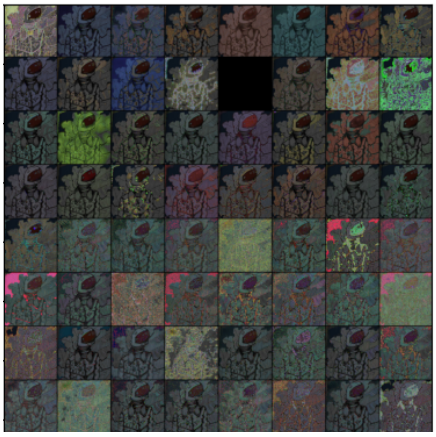
Figure 3.23: *Activation Maximizers output of 2 examples on 64 kernels for the HSCNN+ model at 3 stages: input block, after block 15/30, and after block 29/30.*



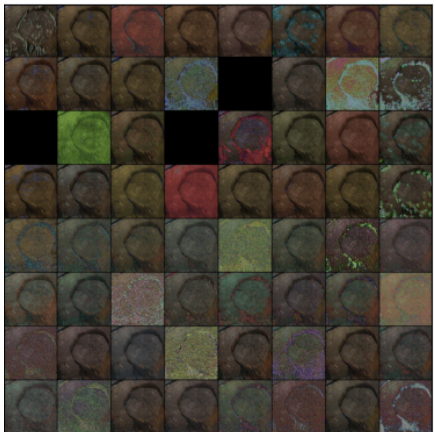
(a) *Mainstream layer 2/8 example 1.*



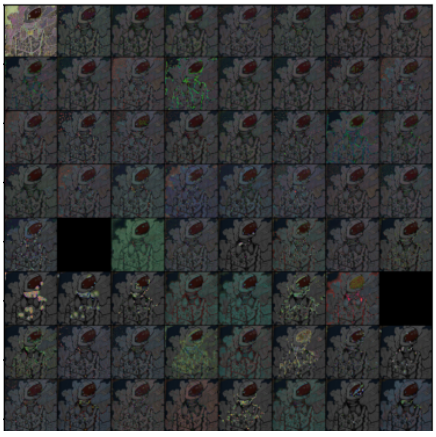
(b) *Mainstream layer 2/8 example 2.*



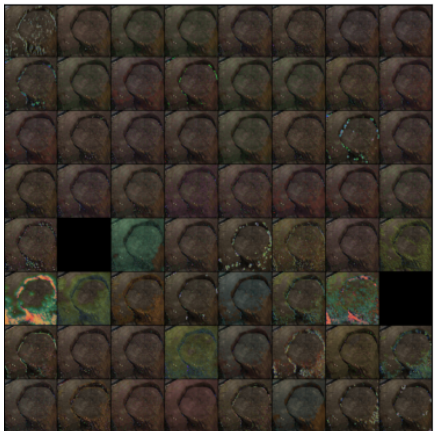
(c) *Mainstream layer 5/8 example 1.*



(d) *Mainstream layer 5/8 example 2.*



(e) *Mainstream layer 8/8 example 1.*



(f) *Mainstream layer 8/8 example 2.*

Figure 3.24: *Activation Maximizers output of 2 examples on 64 kernels for the HRNET model at 3 stages: mainstream layers 2, 5 and 8 (last mainstream layer).*

Before we go into the details, we should note that these generated images do not fully encompass all what the network does (even at the layers they are generated) since they *first*, do not cover all the aspects of the layer. *Second*, not all of them are necessarily used by the model as a main component in the reconstruction process. *Third*, as we mentioned in the feature visualization section in the literature review, the objective used here (which is individual kernels in layers) does not necessarily portray the main building blocks of the networks (although it probably does). This is because that objective could be anything from multiple layers to specific parts of a layer or a mixture of both, so we should expect some mingling in the results we have here.

Either way, these images still give a strong idea of what the network does at that location and what kind of priorities it has for each feature. Also, the black images show kernels where the gradient was zeroed out and they can be ignored.

In the beginning (the input block represented by figures 3.23a and 3.23b), we can see that most kernels are activated by strong colors and edges. Some are activated by background objects while others are activated by foreground objects. Some kernels just detect a certain color or hue while others detect shapes. Also, objects within the image are segmented or thresholded.

It is important to note that this block is not the very first layer in the model but comes after a few layers (in this case 3 convolutional layers that make out the input block). This means that the generated images shown here are not merely the filter response of each kernel, (this can be concluded simply by observing the weights of the filter) but they portray a cascade of convolutional layers' interactions with each other at the beginning of the model.

Moving towards the middle of the network, specifically after block 15 shown in figures 3.23c and 3.23d, we can see that the images are much less saturated and no longer have the strong and sharp features detected in the beginning. Instead, they show depictions of the original image, but with what seems to be a variety of general hues and lighting. That lighting still shows seemingly strong and seemingly unrealistic colors at this point, but there is a general inclination towards colors similar to that of the original image.

Reaching towards the end of the network, in figures 3.23e and 3.23f, we can see that these colors are now even more realistic and depict what seems to be the original image but under a variety of lighting conditions or possible illuminants similar to the original illuminant portrayed by the global hue of the original image.

These observations are consistent across the whole NTIRE dataset and also the reflectance dataset provided in Foster et al. (2007). The same observations are also seen in the HINET network, and HRNET (all the local and some of the semi-local models).

Also, although we only show 3 layers here to save space, the actual test was

done to around 8 layers in each model depending on the model's complexity. With relatively simple models like the MultiScale model, only 7 layers were monitored (accounting for each scale variation), while with complex and large ones like HSCNN+, 10 layers were monitored. And for all tested layers for local models, the pattern was the same.

Figure 3.24 shows a highly similar behaviour in HRNET for 3 of its mainstream layers 2, 5, and 8 (the last mainstream layer). As mentioned before, HRNET has a hierarchy at its beginning using pixel shuffle and unshuffle. The layers chosen here skip that part since it only applies to this particular model, whereas the rest of it is typical convolutional layers (layers in the mainstream) and they resemble other similar reconstruction models.

So we check at the first layer after the hierarchy (layer 2/8), then the mid (5/8), then the last (8/8) and we notice a striking resemblance in behaviour to what we observe in the other local models.

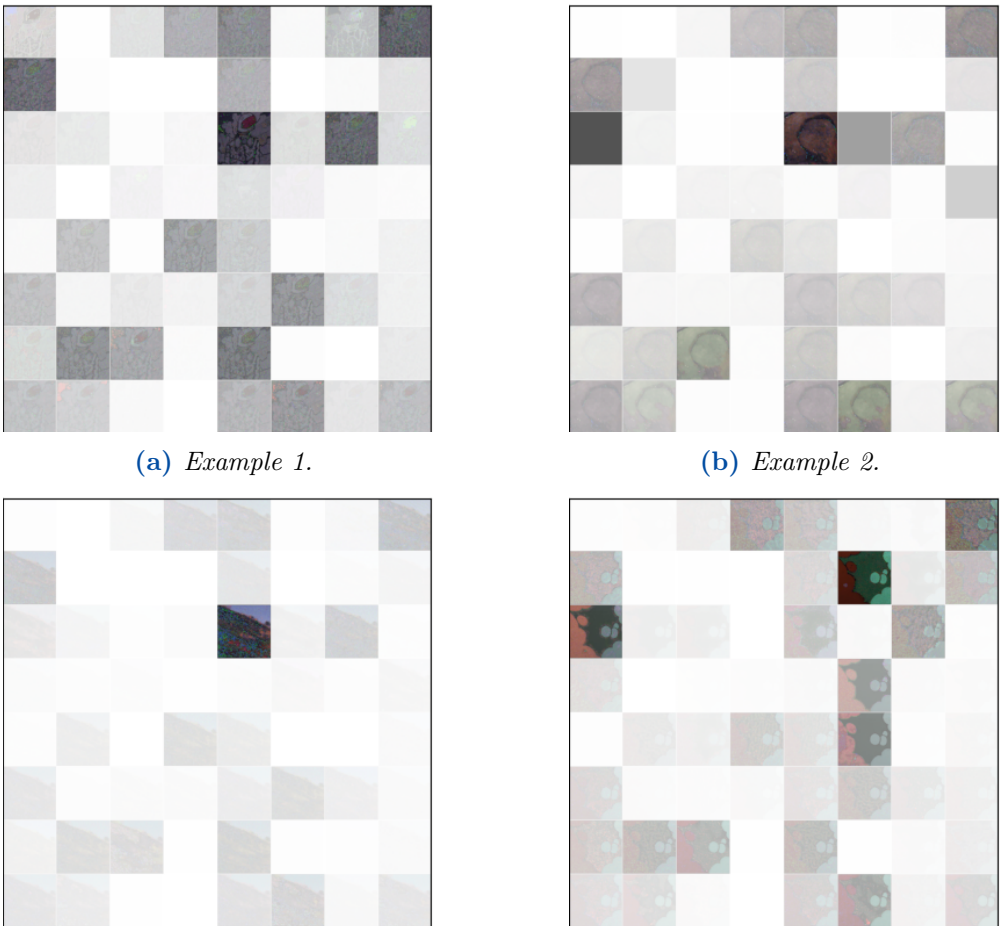
Note that this pattern is *only observed in local and semi local models, which have local and semi-local saliencies*. These include HSCNN+, HRNET, and HINET. Unfortunately, other more global networks did not show such behaviour and their results were highly random and difficult to interpret (an example of that can be seen in the appendix). Not only that these models are local, but they also have high similarities in their general architecture, since they all depend on cascaded blocks of convolutional layers with attention connections both locally and globally across the architecture, whereas other models tested have a different type of architecture like U-Net, and spectral attention and so on.

Since this can be seen as a consistent pattern across the local models, we can theorize that the model tries to generate different illuminants towards the end of the network, where each part of the network (represented here by the different kernels) generates a possible illuminant. Then the model settles on which one activates the most based on its training. This hints on the general direction the model takes when tackling the problem of illuminants. We can also see that these 'generated illuminants' are not far from the original image, which means the model tries to generate them depending on the original with a certain learned distribution. Of course, this is only an assumption for now.

3.5.4 Testing The Theory

If that assumption is true, we would expect the model to 'pick' one of these illuminants towards the end of the network by having strong activations at one of them, or showing a scarcity of activations for most of them. To test for that, the following was applied.

Using the same feature extractors, we also extract the activation from the first



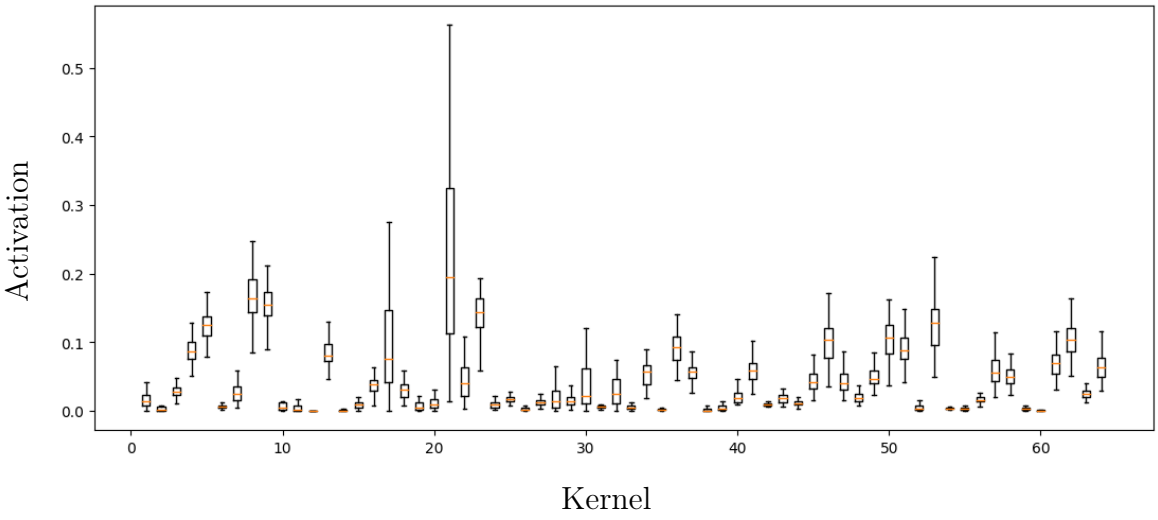
(a) *Example 1.* (b) *Example 2.*

(c) *Example 3.* (d) *Example 4.*

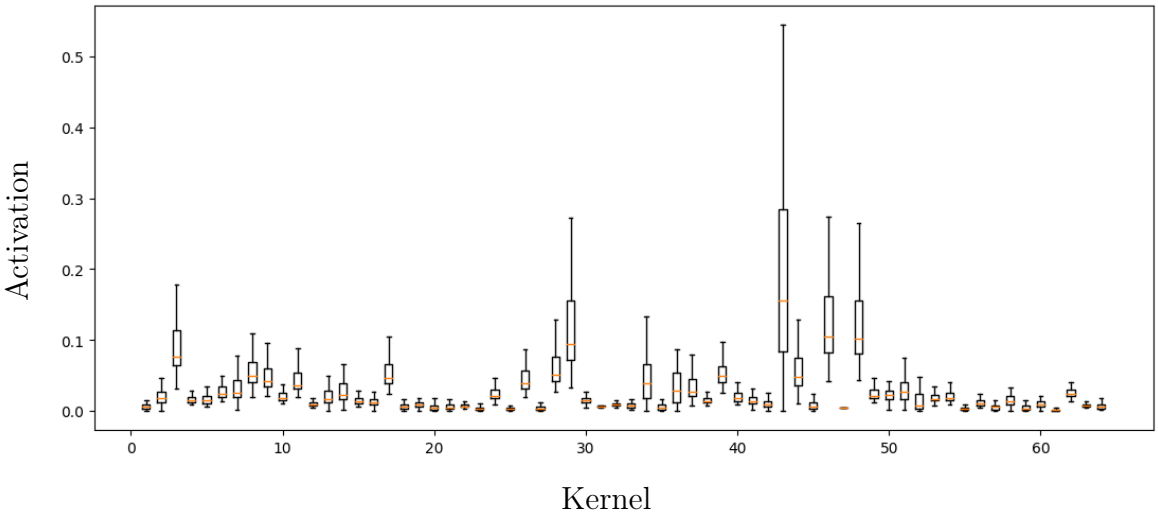
Figure 3.25: *Generated visualizations of 4 examples on 64 kernels for the HSCNN+ model all after block 29/30. with the activation added as alpha channel. In each example, only 1 or 2 kernels are activated significantly.*

iteration which was not disturbed by the optimization process. This activation gives the natural activation of the network at that layer for each kernel to the original image (as in a regular test setup).

Now, we apply that activation as an alpha channel to the already generated images we have. The activation is first normalized to have values from 0 to 1 then an alpha channel was added to the images where the value of that channel was the values of the normalized activations. This will give a low alpha value (high transparency) to images that were weakly activated, and high alpha values (low transparency) to images that were strongly activated. The results for that are shown in figure 3.25.



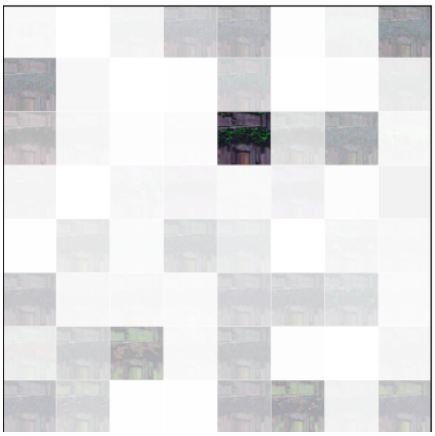
(a) *Activations after the 29/30 block for the HSCNN+ model.*



(b) *Activations for the last layer of the HRNET model.*

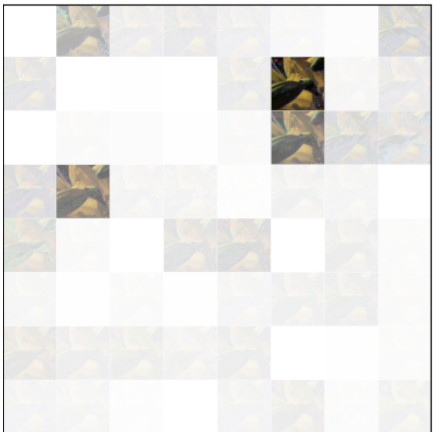
Figure 3.26: *Normalized activations for 2 local models on their last layers across the whole NTIRE dataset showing only particular kernels being strongly activated in each model.*

It can be seen in the figure that only 1 or 2 kernels are being strongly activated in the layer at the end of the network (other layers have more random activations and not shown here to save space). This behaviour is consistent across all datasets and all local models tested.

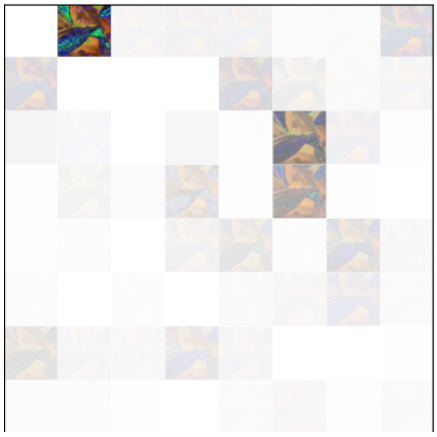


(a) *D65 illuminant example 1.*

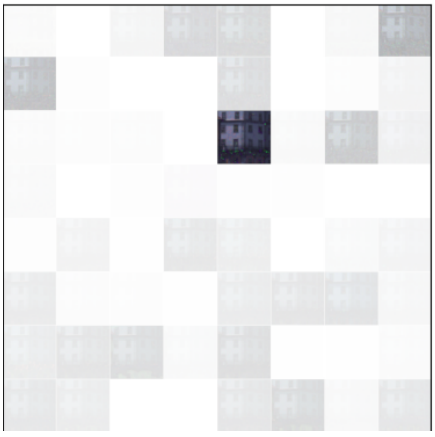
(b) *A illuminant example 1.*



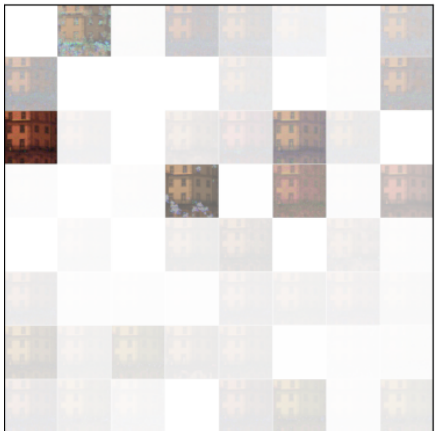
(c) *D65 illuminant example 2.*



(d) *A illuminant example 2.*



(e) *D65 illuminant example 3.*



(f) *A illuminant example 3.*

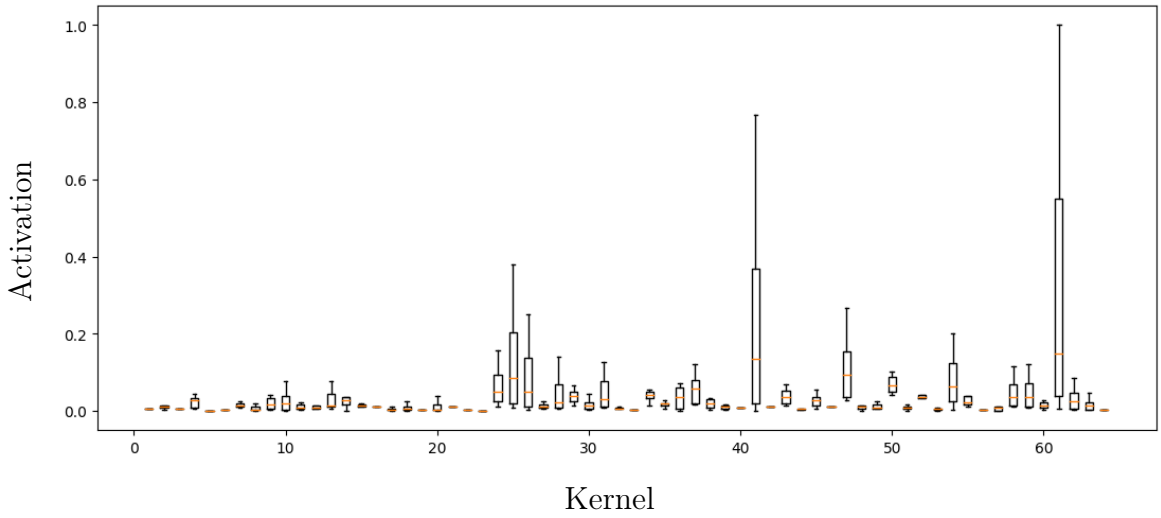
Figure 3.27: *Generated visualizations of 3 examples with 2 different illuminants (D65 and A) with the activations applied as the alpha channel from the last layer of the HRNET and the HSCNN+ models*

Also, to pool all the results of the dataset, we show activations averaged over all images in the dataset (all for the same layer of the same model) in figure 3.26 which further proves the same point for both models (HSCNN+ and HRNET). The same applies to HINET. The reason we keep these pooled results kernel wise is to show 2 1. how one or 2 kernels only are activated significantly, and 2. to show the consistency of that chosen kernel throughout all images and all pixels for the single model (it only changes by changing the model or the illuminant).

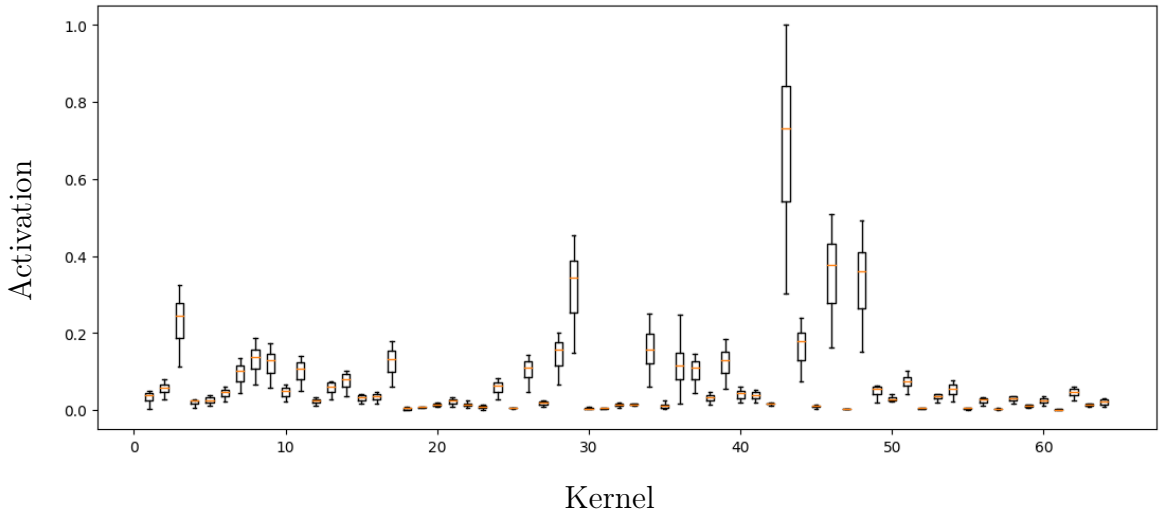
Another thing that could test the theory is to change the illuminant and see if the chosen activation changes with it to prove its connection to that illuminant. For that, we used the reflectance Manchester dataset provided by Foster et al. (2007) since it has reflectances instead of the radiances in NTIRE.

Since this dataset provides us with reflectances, we can simply multiply that reflectance by various illuminants to get RGB images with different illuminants. So, we applied that using two illuminants (D65 and A) to see how the activations change with each illuminant while keeping everything else (in the reflectance spectra) fixed. That was done to all of the images in the dataset. Again, the same behaviour was seen in all images, but we only show a couple of examples to save space in figure 3.27. For a more numerical representation, figures 3.28 and 3.29 show these activations for all the images for the HRNET and the HSCNN+ models respectively (also HINET was tested but the results are not shown to save space but it has similar results). It can be clearly seen that the kernel chosen is different when the illuminant is different. Also, the kernel activated does not change regardless of the change of the image which further supports the assumption. This is further support to the theory that the network 'chooses' an illuminant from the generated ones in these end layers based on what it learned.

Also, some images consistently choose a different kernel for any illuminant. These images that have the same kernel(s) activated usually have similar lighting or are of similar scenery like outdoor images. To test that, we isolated outdoor images which have mostly the same illuminant (natural sunlight) from the dataset and monitored their activations for each model. The results for both models are shown in figure 3.30 which shows even more discrepancy between a single kernel and the rest of the kernels. This is because in these images the illuminant is mostly constant, while the opposite is the case for indoor images which could have various illuminants. This is shown in figure 3.31 for both models. Notice how the spread of the activations decreases in outdoor images and increases for indoor images. The maximum value for the activations is around 0.5 for all images, 0.7 for outdoor images, and around 0.4 for indoor images (since the activations are more spread out/ less descriminative).



(a) *Activations for the last layer with D65 illuminant.*

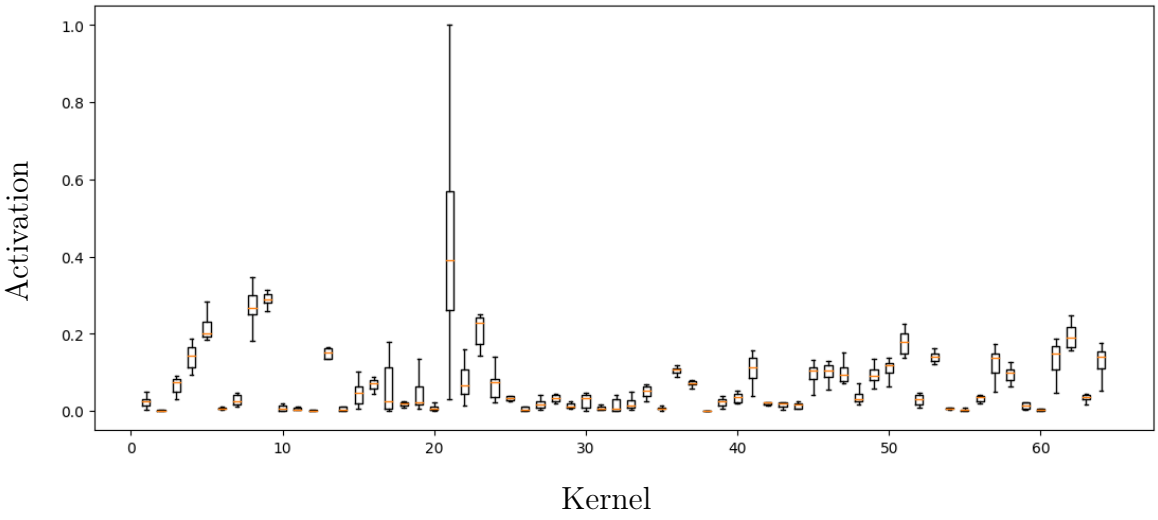


(b) *Activations for the last layer with A illuminant.*

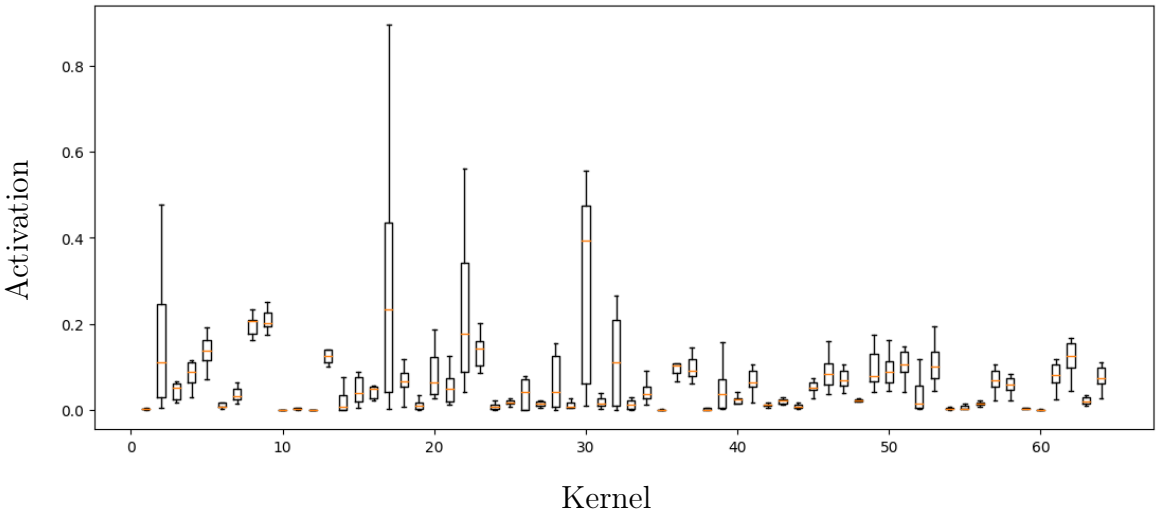
Figure 3.28: *Normalized activations for all images in the Manchester dataset for both D65 and A illuminant for the HRNET model.*

3.5.5 Test Conclusion

These results support the theory that local models try to generate a range of possible illuminants around the original illuminant 'apparent' in the original image.



(a) *Activations for the 29/30 block with D65 illuminant.*

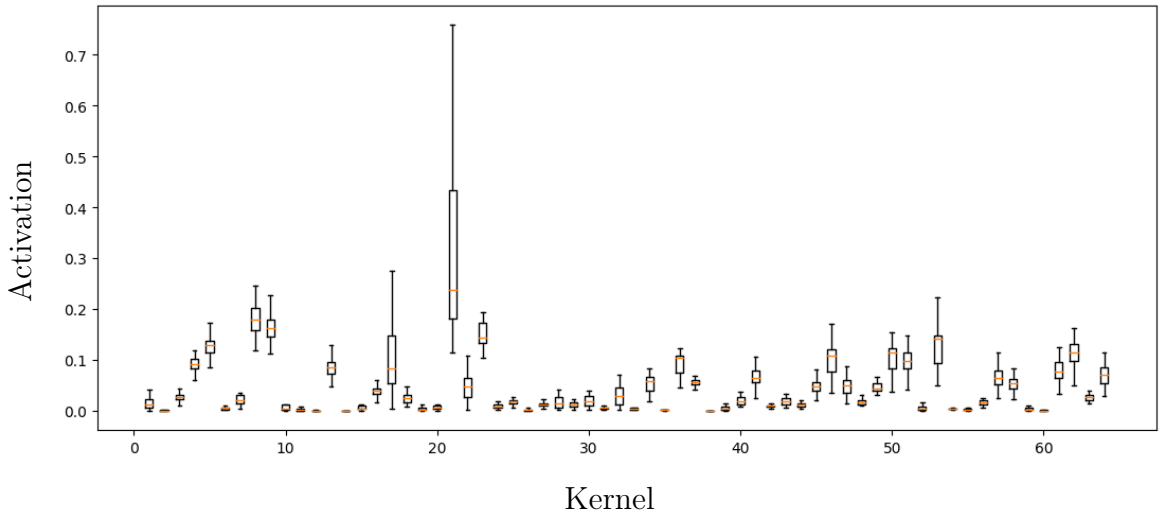


(b) *Activations for the 29/30 block with A illuminant.*

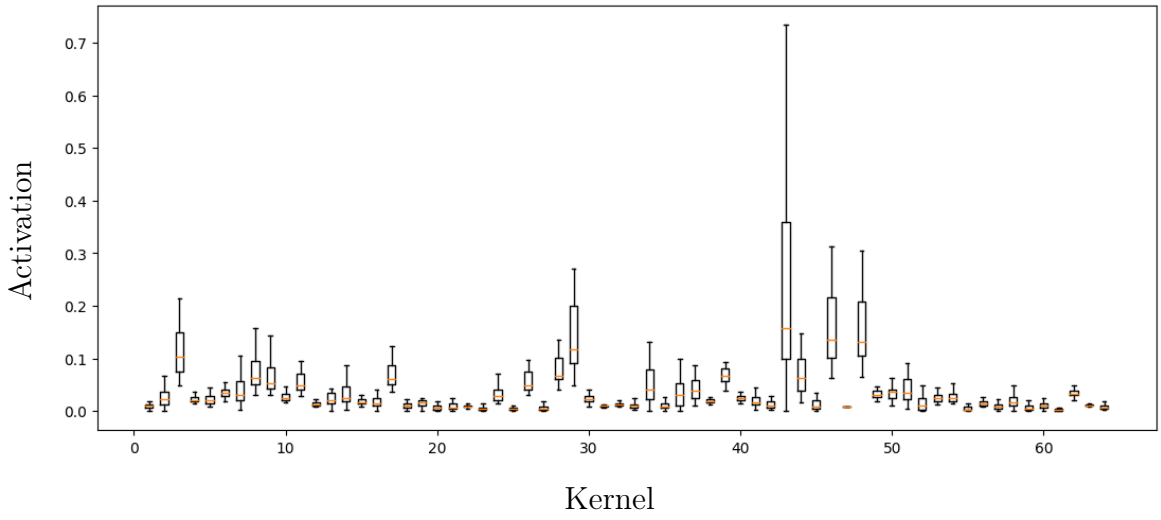
Figure 3.29: *Normalized activations for all images in the Manchester dataset for both D65 and A illuminant for the HSCNN+ model.*

Then tries to 'pick' a certain illuminant out of possible generated ones around the original one in the image based on the training the model had.

There are 2 important things to note here. *The first* is that we cannot say which method the model uses to generate these possible illuminants. It could be as



(a) *Activations after the 29/30 block for the HSCNN+ model.*

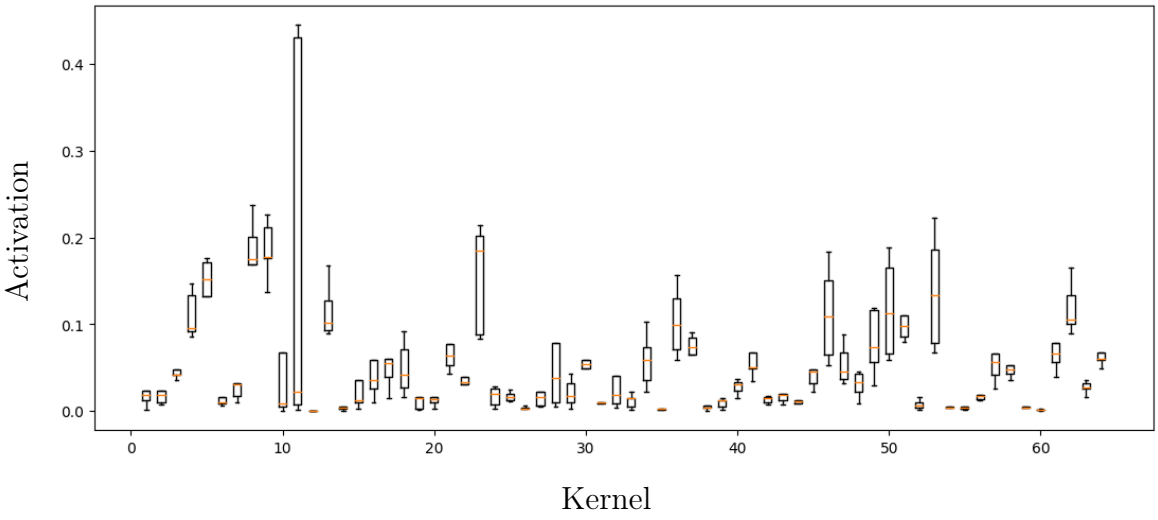


(b) *Activations for the last layer of the HRNET model.*

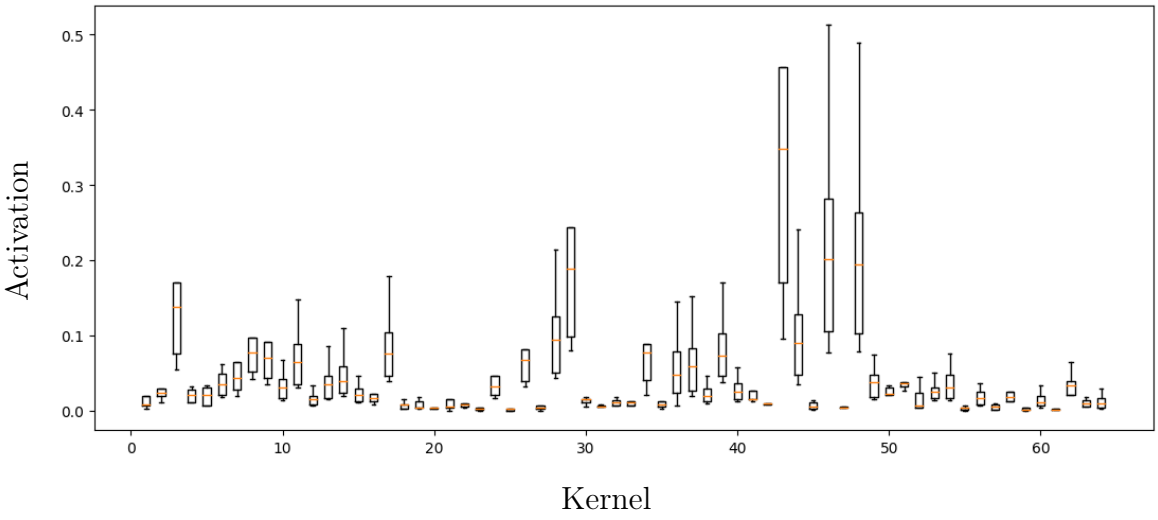
Figure 3.30: *Normalized activations for 2 local models on their last layers across the outdoor NTIRE images showing even more discrepancy between activated kernels.*

simple as a normal distribution around the apparent illuminant, or it could be a complex learned distribution. It is more likely to be the later since it is a trained CNN that learns its own function through training.

However, figuring out the exact method here is not in the scope of this work.



(a) *Activations after the 29/30 block for the HSCNN+ model.*



(b) *Activations for the last layer of the HRNET model.*

Figure 3.31: *Normalized activations for 2 local models on their last layers across the indoor NTIRE images showing more random activated kernels.*

This is because as we discussed before, our job is to break down the model into as many explainable equivalents as possible (break it down into class B models, see the model classes discussed in the literature review). Although we do not know the exact machinations of this particular block of the model's pipeline, we understand what it does and in this case we can even produce an explainable equivalent for

it. For example, we can use a simple normal distribution to generate possible illuminants, then train a simple network to find the best one based on examples.

The second is that we do not claim with complete evidence that the model works that way. This is because although we introduced 3 supporting evidences to the theory, none of these evidences is conclusive enough to fully claim it. We only say that these observations strongly suggest that theory. Further work could be done to fully establish a conclusive proof. It still can be that these observations are behaviours of the network to do something else rather than what we suspect it to do, but it is probably not a coincidence since it shows in a highly consistent pattern, so it probably means something. It is just that it could be something other than what we think.

Either way, these observations and the theory they suggest provide a very possible way for the network to achieve part of its goal (illuminant estimation) so even though it might not capture the full depth of the model, it could serve as a way to apply an explainable model, or provide an equivalent for the existing models using the same ideas.

Now that we investigated the illuminant part of the reconstruction, we turn to investigate other features that could help the models perform the up-sampling of the spectra. There are many parameters that could be included in this process. In the next section, we try to investigate some of the most important ones and see their effect on the models.

3.6 Parameters Sensitivity

In this section, we try to test multiple parameters and see if the models react to them, and if so, how do they perceive them. The goal here is to get as much clarity about the model's processing workflow as possible. We try to find how sensitive these models are for variations of multiple parameters that are suspected of affecting the reconstruction decision using a perturbation based method.

3.6.1 The Parameters

These parameters are chosen to be hue, saturation, frequency, and noise:

Hue is one of the most important parameters since it has a close connection to the illuminant which we already noted that the models are highly sensitive to. Especially that they seem to generate variations of it towards the end layers. Hue is a main contributor to the illuminant characteristics so it would be interesting to know about the networks' reaction to it and we would expect it to be similar to that of the illuminant.

Saturation is also interesting here since it also has a contribution to the illuminant and the general RGB value of both the target and the observed area pixels. Also, looking at the Activation Maximizers discussed before, we can see a clear difference in saturation between the beginning layers (high saturation) and the end layers (low saturation) which leads us to suspect it could be a parameter the models use in their workflow.

Frequency is suspected to have an effect on the model since it can be responsible for features defining texture and materials. Detecting frequency changes might suggest that the models use it to define different materials and their characteristics and eventually their spectra. However, frequency of original images is heavily reduced in convolutional networks naturally with the convolution process, so the results will be affected by that.

Noise is an important parameter since it exists in all natural images and in multiple amounts. High performing models are able to deal with a noisy images and multiple types of noise. As we discussed before, the NTIRE dataset in previous issues used to have both clean and real life tracks. However, since models had similar high performance on the clean track, they authors of NTIRE decided to remove it and only keep the noisy real world track since it is more realistic and proves a challenge to some models. This also suggests the importance of processing noise in such models. Here we introduce a typical white Gaussian noise with multiple variances.

3.6.2 Concept

We introduced the concept of hooks and feature extractors previously. In this test, we only use hooks since we simply need the natural activations of the network at multiple locations. Starting with any parameter we need to test (for example hue), we generate a variety of images each with a different hue for each image in the NTIRE test set. So, each image's hue is split into 50 hues distributed uniformly around the hue circle. ending with a dataset 50 times larger than the original which makes the computation very time demanding. Of course, this was done after converting the image to HSV space. After the variations were done, the image was turned back to RGB color space.

Now, the model is run on this large dataset while the hooks are monitoring specific layers. We cannot monitor all layers since that would take a very long time, but we monitor the same layers we used feature extractors on in the Activation Maximizers tests to make the tests comparable.

We get the average activations of the monitored layers and their kernels for each image's variations. Then, we calculate the variance of these activations across the variations of each image, then we average that variance across all images in

the dataset. That would give us how much these activations changed in response to change in the varied parameter. In other words, how sensitive that part of the network is to change in this parameter.

For saturation, it was varied by also first converting the RGB images to HSV. Then, we distribute the saturation value (0 to 255) uniformly across 50 variation images, and turn it back to RGB.

For frequency, they were varied by first applying a Fast Fourier Transform (FFT) to transform the image to the frequency domain, then a frequency shift was applied (both positive and negative from the original) to also make 50 variations (25 positive and 25 negative shift). Finally, the image is converted back using the inverse FFT. The frequency shift's absolute value was calculated by:

$$FrequencyShift = \pm i * factor$$

Where i is the variation number (1 to 25 for positive and 1 to 25 for negative) and $factor$ is a scaling parameter that was chosen to be 0.5 to get perceivable frequency shifts.

For noise, we used white Gaussian noise with a fixed zero mean and a varied standard deviation (SD). The SD was calculated by also multiplying the variation number i by a factor. The factor was chosen to be (1/16) or (0.0625). This was chosen to have realistic noise. Larger values would have too much noise where the image becomes unrealistic. Figure 3.32 shows examples of these generated variations for an example image.

Similar to Activation Maximizers, each model had different layers where it was suitable to monitor, some had 4 monitored layers and some had 3 with the same reasoning used before.

3.6.3 Results

Again, all models were tested and had similar results so we show here the most significantly varied ones that are representative of the rest. These are the 4 models shown in figures 3.33 for HRNET and 3.34 for MST++. While figure 3.35 and 3.36 show the same for HSCNN+ and MultiScale models respectively. These activations are all normalized relative to the maximum value of *All* their values (for all parameters). This is important because it allows the comparison between these activations across multiple parameters in a meaningfull way. Also, for HRNET and MST++, 3 layers are observed and each with 30 kernels. This is because the most important layers at the beginning, middle, and end of these networks have only 32 kernels (unlike the 64 or more in the other models).

For HRNET (figure 3.33), 3 layers are observed. Hue has the strongest variance in its activations, showing a strong sensitivity for it for the network. Also, it

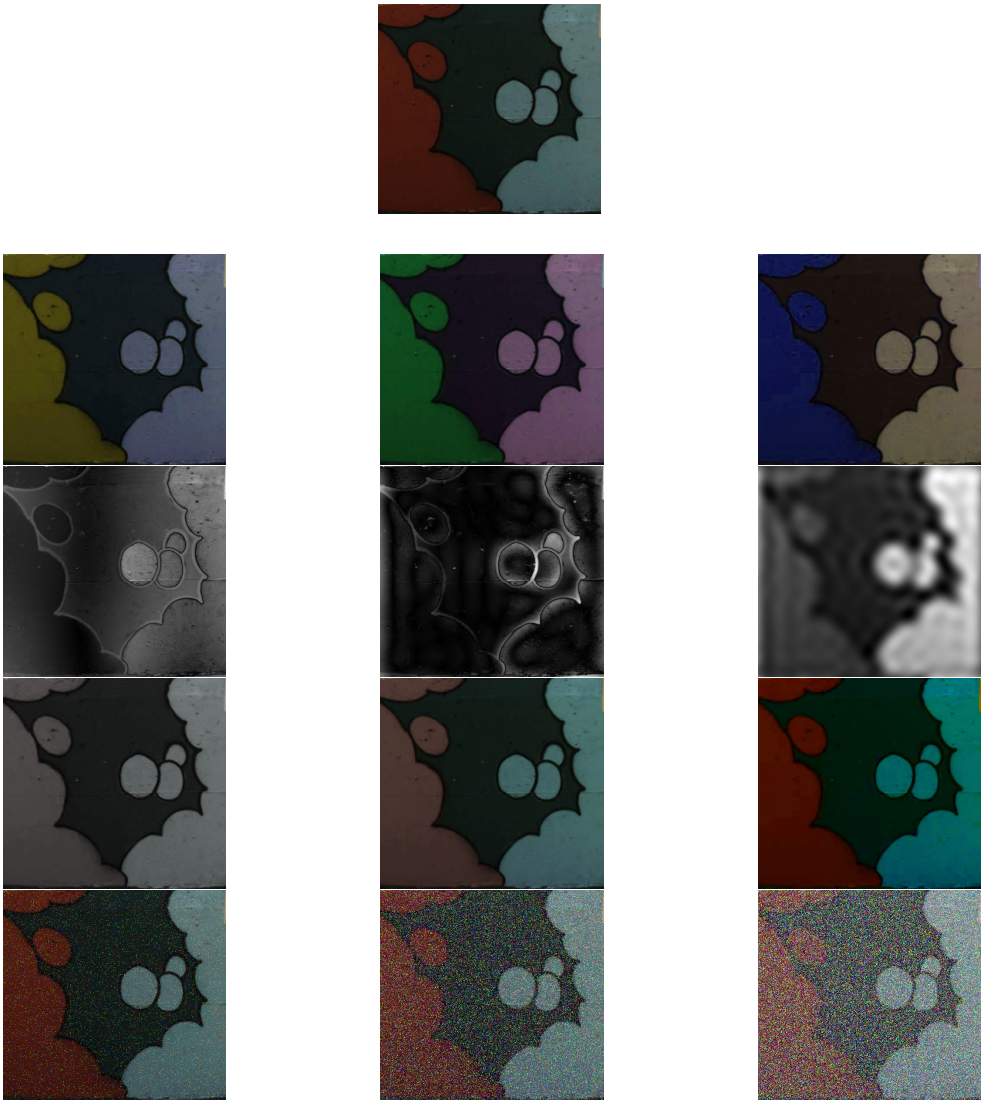
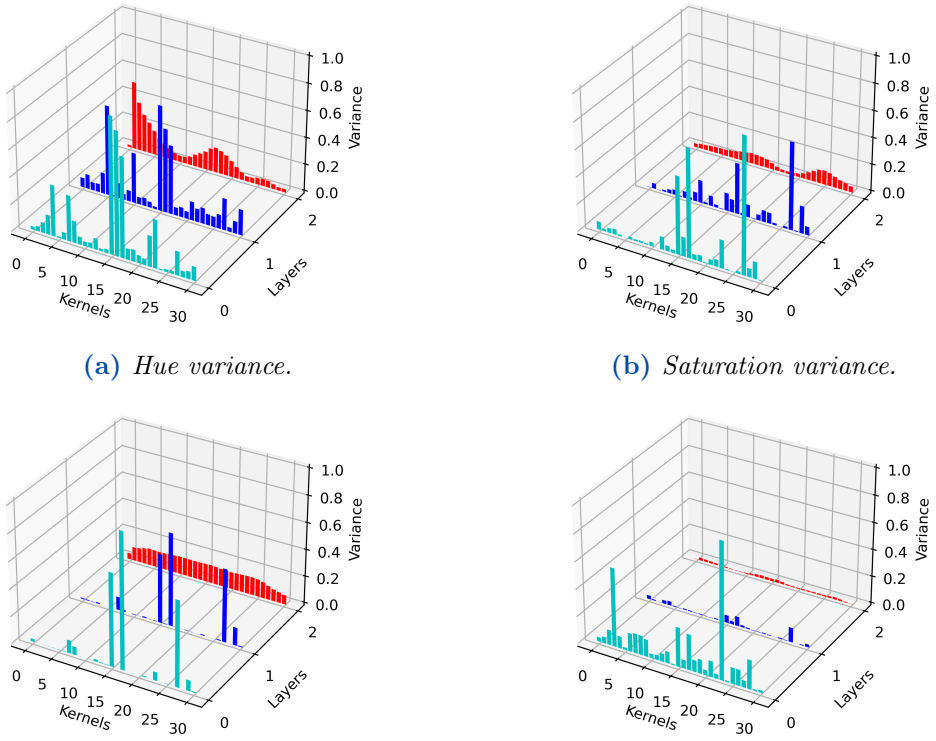


Figure 3.32: *Examples of the generated variations for an example original image (top) with variations of hue, frequency, saturation, and noise (respectively from top to bottom).*

shows specific areas being activated strongly in the last layer (layer 3) with 2 main smoothed peaks. Hue variation intensity seems to be quite consistent throughout the network. This is unlike saturation which shows variance in the first, mid layers, but not towards the end, suggesting it is processed at the beginning. That agrees with what the Activation Maximizers showed especially for first layers. Similar behaviour is observed for Frequency. As for noise, it seems to be only processed at the very beginning of the network. This is also to be expected since CNNs usually



(a) *Hue variance.*

(b) *Saturation variance.*

(c) *Frequency variance.*

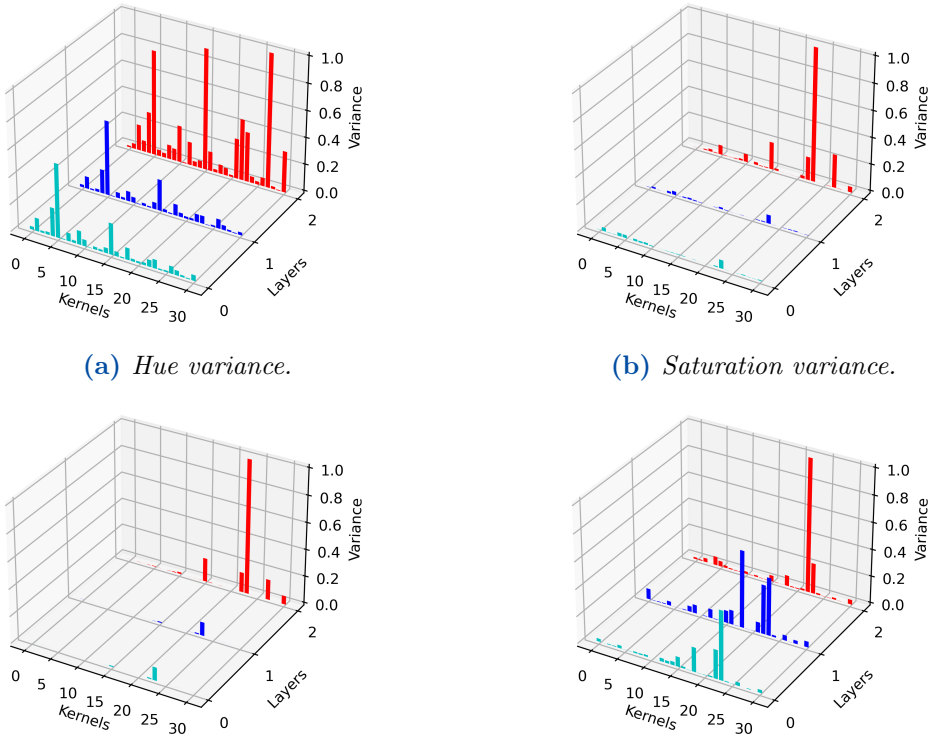
(d) *Noise variance.*

Figure 3.33: *Activation variance for HRNET throughout the network.*

process noise at the beginning.

For HSCNN+ (figure 3.35), 4 layers are monitored. Hue also behaves the same way but with less smooth behaviour and stronger intensity. Saturation is the same as in HRNET. Frequency is not strongly activated however in this network. Noise has a similar behaviour and is mostly processed at the beginning, while saturation processing is spread around the network.

For the MultiScale model, Hue and saturation have a very similar variance and it shows maximum variance for their activation in the second part of the network but not at the very end. Since this particular network is a U-Net, this corresponds to the middle of the upward side of the U shape. Noise is again only processed at the beginning, but frequency is also mostly processed at the same layer as that of saturation and hue (number 2 in the figures) which points that this area of the network seems to be the most important. It is worth noting that this does not confirm with the illuminant theory we concluded in the Activation Maximizers tests since this model is not local and that theory was only for local models (local saliency).



(a) *Hue variance.*

(b) *Saturation variance.*

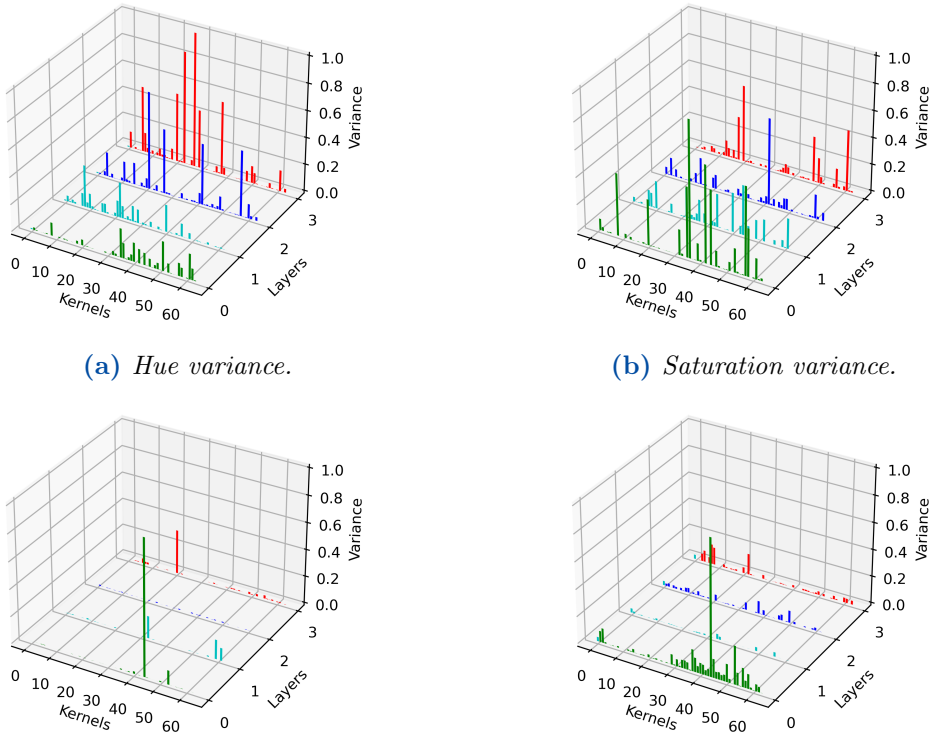
(c) *Frequency variance.*

(d) *Noise variance.*

Figure 3.34: *Activation variance for MST++ throughout the network.*

For MST++ (also not a local model), Hue has strong variance throughout the network and especially at the end, saturation, noise, and frequency are mostly processed at the end. This test among other tests like Activation Maximizers and saliency maps show that MST++ (and MST) are very different from other networks and their behavior is non confirming with the theories of this study. This does not affect the aim of this study since MST and MST++ are advanced and complicated models that add a lot of changes to more typical hyperspectral reconstruction models. Mainly, they do not depend on only the minimum requirement and fundamental features we are trying to find in this study.

As for the rest of the models (HDNET, HINET, and MST) we did not include their plots to save space, but each of them is similar to one of the models displayed in the figures. We can already see that HSCNN+ and HRNET have a very similar behaviour in the figures. Also, HINET and HDNET share that same behaviour. As for MST, it is similar to its successor MST++. MultiScale is the only one with its unique behaviour which makes sense since it is the only one with the simple U-Net architecture.



(a) *Hue variance.*

(b) *Saturation variance.*

(c) *Frequency variance.*

(d) *Noise variance.*

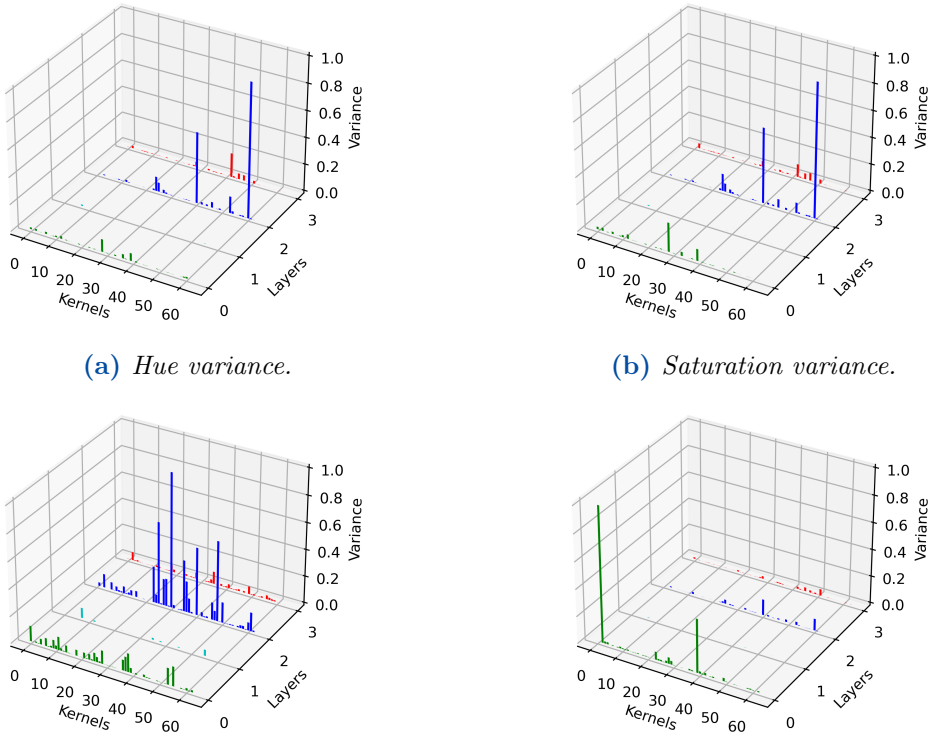
Figure 3.35: *Activation variance for HSCNN+ throughout the network.*

3.6.4 Test Conclusion

This test shows that the models have high similarity in treating some parameters (like noise and frequency) but differ in treating other parameters especially the ones related to the illuminant (hue and saturation).

Most models treat noise and frequency at the beginning (except for MultiScale, MST, and MST++). All local models and some semi-local ones (HSCNN+, HINET, HRNET, HDNET) have high sensitivity towards hue and saturation and have scarcity in the activation array for these parameters (across kernels) towards the end layers. This agrees with the results seen in the Activation Maximizers tests. MST and MST++ have their distinct behaviour which is also sensitive to hue, saturation, and noise. Finally, MultiScale seems to also have its own distinct behaviour that have most of its variance towards the mid part of the upward U path (middle of the decoder).

It is important to point here that this test (like Activation Maximizers) has the limitations of being specific to certain kernels and does not fully encompass



(a) *Hue variance.*

(b) *Saturation variance.*

(c) *Frequency variance.*

(d) *Noise variance.*

Figure 3.36: *Activation variance for the MultiScale model.*

the whole network. It is similar to peeking inside the black box but only having limited view on certain areas within it. So, the results these tests produce should be treated as pointers and suggestions but not conclusive evidence. However, their consistency and coherence with other tests gives them more credibility as valid insights on to the network's general workflow.

3.7 Illuminant and Color Tests

Now we extend the investigation of illuminants to test how the models behave under specific illuminants. This test focuses on specific sample colors and tries to check if these models behave differently for a color and how much the illuminant affects that. To test specific colors, we needed a color checker, and we needed real hyperspectral and RGB images of that color checker as well as all the accompanying white correction required.

3.7.1 Concept

To do that, we used images taken as part of the color science lab course at the University of Eastern Finland (UEF). These images include taking both RGB and hyperspectral images of an X-Rite 24 patch color checker under a lighting box that provided both D65 and A lighting. Starting with RGB, both photos (D65 and A) were taken from the same Nikon D800 28 mm camera where the angle from the light to the checker and from the checker to the camera is 45 degrees and the distance between the camera and the checker is 60 cm, while the distance between the light and the checker is 65 cm.

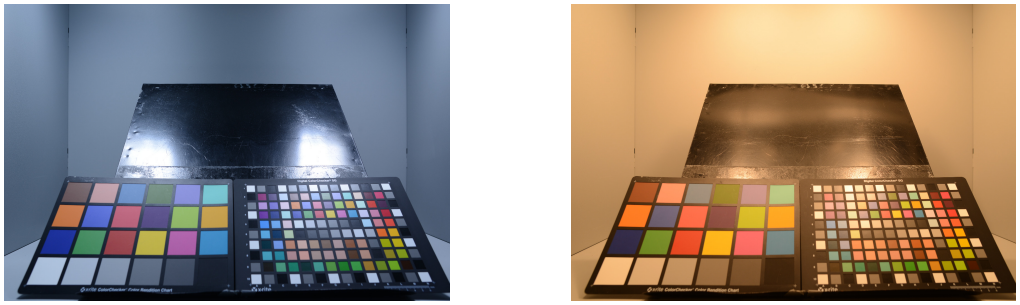
It is worth mentioning that the images were taken as RAW and the importing process of the images was done while ensuring the images are kept without any automatic white balancing or any sort of interfering from the camera or the software. To get the colors of the 24 color checker, we make a rectangle inside each patch and take the average color of that rectangle. The images are shown in figure 3.37. Although the images also contain a 72 color checker as well, it is ignored in this analysis.

For the hyperspectral images, they were taken with a Specim IQ V10E hyperspectral camera (same camera as with the NTIRE dataset). The number of bands taken by the camera was 204 which was much higher than the required 31 bands of the ground truth. For that, they were resampled to get 31 bands in the same range and manner as the NTIRE dataset ground truth bands were acquired. In fact, we used the same interpolation codes published by NTIRE for this particular task. Then, the 8 colors in the middle left side of the 24 color checker were used since they conveyed popular colors and they were not affected by the specular reflection apparent in the top of the color checker. In each color patch, a spatial box was drawn on both the RGB and the hyperspectral images with the same dimensions, making an RGB input and a hyperspectral ground truth pair available for testing with the models.

The importance of using a color checker here is not only to test specific colors, but also to eliminate any other features that the model uses and focus only on the colors themselves. This is unlike the previous section which tested full real life images with multiple features. That gives a different perspective not only on the colors but on the illuminants. Is illuminant estimation dependent on the shapes of the image, or on its general color of the observed area?

3.7.2 Results

The results are shown in figure 3.38 where the average SAM error is shown for some of the models. All other models and error metrics have similar results, so the ones



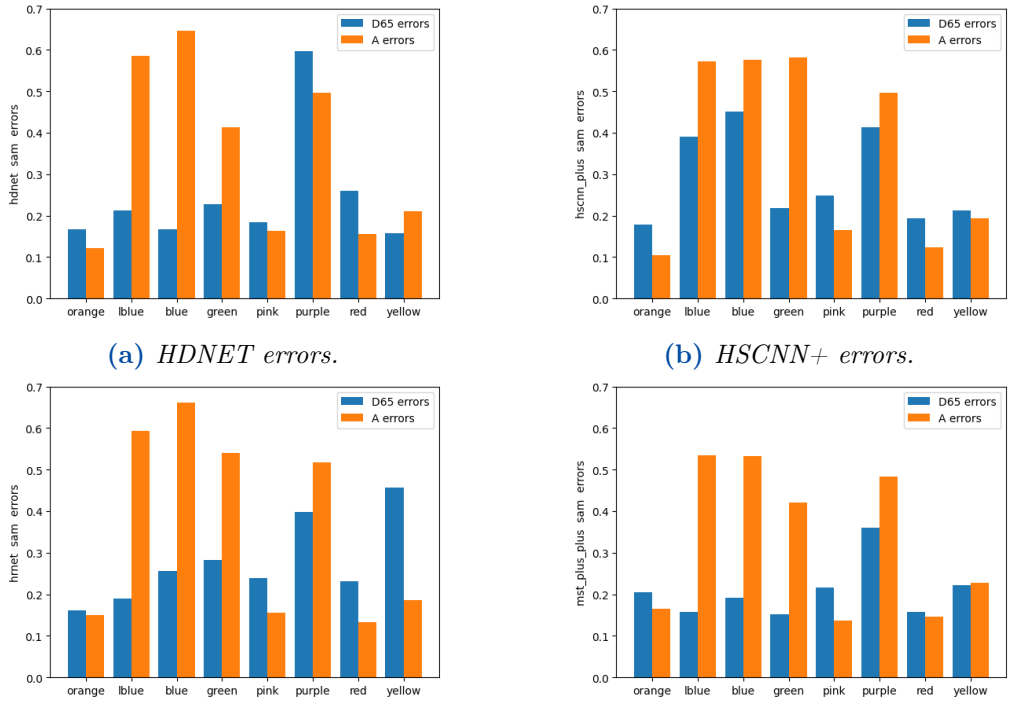
(a) *D65.* (b) *A.*

Figure 3.37: *RGB images taken for the X-Rite color checker under both D65 and A lights. Same setup was used to take hyperspectral images of the same scene.*

shown are representative of all models. The first clear result is that for most colors and most models, the A illuminant error is far higher than that of D65. This was not the case in the previous section which had real life images. That suggests that these models depend heavily on shapes and image features in the observed area to detect the illuminant. So, when they do not have that information anymore in this test, they failed to distinguish the A illuminant and receded to the most common illuminant in the dataset they were trained at which is much closer to D65. This is because most NTIRE images are shot in an outdoor sun-lit setting. This also highlights the importance of having a larger variety of images in the dataset to include more illuminants and settings to avoid dataset bias.

Now, looking at the colors, it can be seen that the colors most affected by this are the ones that change hue dramatically under A light from D65 which are blue, light blue, and green. However, colors that do not exhibit such a dramatic change in hue under A light are not as affected like orange and yellow. In fact, for those colors the D65 error is higher in most cases.

Although this test suggests interesting ideas, it is very limited with its confidence level since the data tested is very scarce. More investigation is encouraged to actually gain more confidence in this analysis.



(a) *HDNET errors.*

(b) *HSCNN+ errors.*

(c) *HRNET errors.*

(d) *MST++ errors.*

Figure 3.38: *Mean SAM errors for some of the models on specific colors under both D65 and A illuminants (lblue is light blue).*

4 Discussion

Many tests were done on multiple aspects of reconstruction models. We started with investigative tests that provided clues on the models' general points of failure with error maps, then we went deeper to find about the models' range and spatial features with saliency maps. Box tests provided more certainty and detail on which parts of these spatial areas and features the model uses. Activation Maximizers provided insight on the inner workings of illuminant estimation within the model. Parameters sensitivity tests allowed us to quantify how much and where important parameters are being processed in each network. Finally, illuminant and color tests provided details on what kind of information illuminant correction needs and performance for each color and illuminant.

In this section, we aim to compile all these results, and list the main outcomes these tests concluded, but from a different perspective. Then, we try to relate them to each other to draw a big picture image of how these models work. We also try to evaluate each of these results in terms of their validity, importance, and generalization to most models.

4.1 Trust, and Points of Failure

One of the main targets of this thesis (like most explainability works), is to provide some level of trust (or distrust) to a certain black box model. Unfortunately, we cannot say that our results provide full trust with a model. In fact, they pointed out multiple points of failure that could contribute to not trusting the model under certain circumstances.

First, we showed (through error maps and activation maximizers) that there are inherent problems with RGB images that prevent reconstruction models from performing as expected. These include images that have no clear setup or context information (especially indoor images) as well as saturated images (or parts of the image). Although this might not be because of a problem with the models, it is still a problem with the setup that users have to take into account when using the models.

Also, the narrow saliency and observation area of local saliency models especially makes them prone to error when the target pixel does not have similar pixels in close proximity around it. This includes edge pixels or very small objects in the image. This was shown by error maps, saliency maps (both random and edge ones), and box tests. The area local models look at when reconstructing a pixel is no more than 50 x 50 pixels and averages on 45 x 45.

These models proved they can reconstruct spectra from this small area, so they must depend on features in that area. However, as shown by the tests, these features are not always present for all pixels. This creates a point of uncertainty or distrust by users (when can a user safely use the models and when they cannot). We tried to find all these features and we did find some of them (through activation maximizers, illuminant tests, and parameters sensitivity tests) but our list is not comprehensive and cannot exhaust all possible features the models used. This is why we cannot draw a fine line between where the users can trust the model and where they cannot. All we can do is give recommendations to users where the models (especially local ones) are expected to fail (points of failure) mentioned earlier.

Also, the problem with the setup (indoor images) could be due to the dataset having most of its images taken in an outdoor sunny day setup. This could be avoided by taking more images in the dataset that include more setups and more illuminants. Then, we can verify if the performance limitation in these setups is due to the actual model, or simply the dataset it uses.

4.2 Illuminant Estimation

One of the main tasks of reconstruction is to estimate and correct the illuminant (color correction). Without it, the model cannot deal with metamersim and materials with different colors than their original due to lighting conditions.

Activation maximizers provided a lot of insight on how that is done within the local networks. We showed that these networks seem generate possible illuminations around the original illumination in an illuminant feature space. Then, the network 'chooses' one of the these illuminants as the most likely true illuminant of the scene. This was shown in figures 3.23 through 3.29. As we mentioned, this theory is well founded since it has 3 different tests suggesting the same conclusion.

To summarize, these tests are: the activation maximizers generated images showing variations of the original image color in the end layers for local models. Then, the natural activations of these end layers kernels which showed clear and consistent scarcity and only one or 2 chosen kernels. Finally, proving that these chosen kernels were dependent on the illuminant since they only changed by changing the illuminant and stayed consistent to that change.

However, none of these tests is conclusive enough prove that this is how the models work. This is because as we mentioned, these results could be the outcome of a hidden underlying mechanism that only correlates with the illuminant variations but not explicitly driven by it. That said, these results provide a possible way for the illuminant to be corrected. Even if this way was not the actual method used in the networks, it could still be used to build an equivalent explainable model that does the same thing.

4.3 Different Models

It is important to note how different reconstruction models are from each other. In the saliency maps section, we classified them based on their saliency (observed area) size into local, partially local, and global models. However, even models from the same type vary significantly. For example, we mentioned how the HDNET model is considered global, but it has a very random and relatively small global saliency that is very different from other global models (MST and MST++). We also see a clear difference in architecture parameters used by each model (from the parameters sensitivity tests).

What is important to note here is that most of our conclusions and theories only apply to local models that include HSCNN+ and HRNET. They also sometimes apply to the partially local models (HINET and MultiScale). For example, applying Activation Maximizers to global models did not provide any conclusive or identifiable results. No illuminant variations were noticed, and no specific kernels being activated towards the end of the network. This suggests that these models either work in a different way or -because of their globality- are harder to detect and understand.

However, one of the main targets of this work is to identify how a reconstruction CNN works on a fundamental level. That means we need to find the minimum amount of information required to do a reconstruction. This is the information inside the local saliency area observed by local models. Global information used by global models only increase the accuracy but are not absolutely necessary to perform reconstruction. Also, most models as we have seen do not have complete global saliency (only MST and MST++ do) and *All* models have a far higher saliency on the local area than on the rest of the image.

All that does not necessarily support the importance of that area (to the reconstruction task in general). However, it does support the *dependency* of these models on that area (how these models perform reconstruction). This is an important distinction because what we are trying to find is not how the model should work (optimally), but how the model actually works.

4.4 Big Picture

Now we try to relate these tests together to paint a big picture of how the models (at least parts of the models) *are suspected to work*. As mentioned earlier, we consider reconstruction models to be of Class D, which means that they are black boxes, and that we do not have any expectations for how they might be doing their task. This is the part where we try to break down the Class D models into more explainable Class B models, since class B models have a clear workflow even though they can still be vague in terms of their exact parameters. As discussed, this series of class B models is not sequential in reality and is very mixed, but it can act as an *equivalent* to how the real model works. This helps with providing an equivalent explainable model to the black box CNN. Again, this is only for local models.

Figure 4.1 shows the main parts of that big picture. Clearly it is not very detailed since it is made to be as general as possible to encompass the general workflow of local models. The following is a possible equivalent workflow and *not necessarily* how the models work. At first, each target pixel is taken along with a small local area around it as was shown.

At first layers, different parts of the network detect and process features like edges, foreground, background, noise, high frequencies, and strong colors and sometimes highly saturated colors (box number 1 in diagram 4.1). This can be seen in the results of activation maximizers and parameters sensitivity tests.

Going to middle layers, the networks start generating the illuminant variations around each pixel while still processing other parameters that help with the illuminant generation, feature extraction, and illuminant choice (box number 2 in the diagram). Going towards end layers, the generated illuminants are mostly complete. Now, we do not know how the generation of multiple illuminants is made exactly. It is part of the learning the model does. However, we can think of its equivalent as any illuminant distribution around the illuminant of each pixel. It could be a normal distribution, or something more customized (table and illuminant feature space shown in the diagram).

Now, if we combine that with the knowledge that the observation area is small for local models, we can understand this better. The illuminant choice is made by having each part of the network activated by a particular illuminant. So, each pixel in the observed area will activate a range of these parts (a range of illuminants) around its original pixel color. There will be a clear overlap in these activations, which would act as a voting mechanism to activate the most likely illuminant in the feature space. This is shown in the process from box number 2 to 3 in figure 4.1.

In the figure, the table beneath box number 2 takes each of the observed pixels

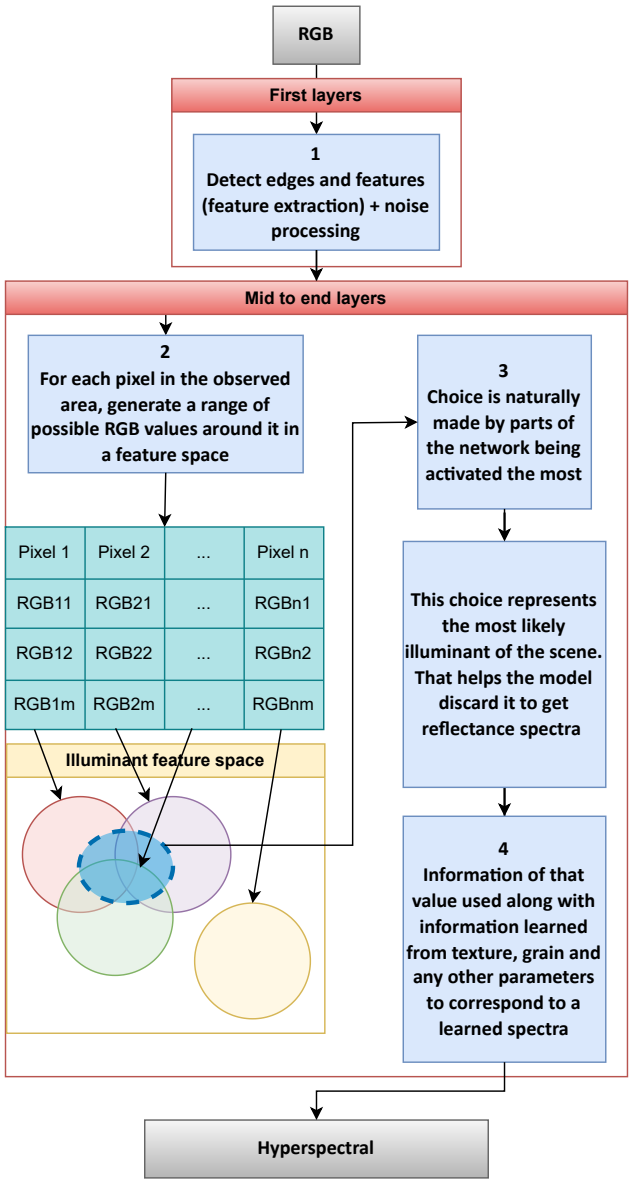


Figure 4.1: *Diagram explaining the suggested equivalent to local reconstruction models.*

and produces a range of possible values around it in the illuminant features space. We note here that the table has these values as RGB, but in reality they could be in any space depending on the network. These generated variations will create a range in the illuminant feature space (represented by circles in the diagram). Each

pixel will make a contribution to that activation which would ultimately activate the most overlapped parts of the space (blue ellipse in the diagram).

We consider this chosen illuminant to be the illuminant of the scene (the local scene in the observed area). Finding it is equivalent to the network applying color correction to that scene. This helps the model discard the illuminant to find true reflectance spectra. We suspect that this part is only affected by the illuminant since we varied other parameters but they all had little change in the activation of the end layers except for the illuminant (and contributing parameters like hue and saturation). This was done in both activation maximizers and parameters sensitivity tests.

This means that by the end layers, other parameters are already processed like frequency, grain, noise, and patters. They could also contribute to the generation and choosing of the illuminant.

4.5 What Was Achieved

We discussed before that the model needs to perform 2 main tasks to achieve reconstruction. These are illuminant correction and up-sampling. Our results show a general possible description of how illuminant estimation is done. As for the up-sampling part, we unfortunately did not find any specific mechanism the models use to do it. It is still hidden in the black box.

However, once the illuminant problem is out, up-sampling can theoretically be done by learning relationships between color corrected colors and their corresponding spectra. We have an expectation for how it could work out in a neural network. That makes the problem of up-sampling a class C model. So, using the results of this work, we reduced the Class D reconstruction model into a combination of class B (illuminant correction) and a Class C (up-sampling). This falls short of what we aspired to achieve, but it still sheds a lot of light into the original obscurity of reconstruction models.

4.6 Conclusion

We conducted a series of tests and explainability methods that aim to break down RGB to hyperspectral reconstruction CNN models into explainable equivalents and analyze how they work and the parameters they depend on in their workflow. We established the spatial area used by the networks and classified the networks to local, partially-local, and global models based on their observed spatial area. We established the model's dependency on features within that limited area of 45 x 45 around the target pixel. We established points of failure for these models

like saturation, vague setting, and thin object edges and small objects within the image.

We presented a theory on how the models perform illuminant estimation and correction supported by various tests and proofs discussed in the work. We also analysed the model's sensitivity and processing to various parameters and where in the model those parameters are processed. We identified many differences and similarities on how each type of models work and connected that to their architecture and class. We identified color and illuminant sensitivity in each model and for various colors. Finally, we introduced a general workflow for local models that describes how illuminant estimation could be done in an equivalent explainable model.

Furthermore, the XAI methods implemented and used throughout this work were customized and modified to be used specifically for these models and can also be used for any hyperspectral reconstruction and even similar models with similar architecture like spatial super-resolution models. This opens the door to explain CNN models in many fields that had no explainability methods before.

4.7 Future Work

We mentioned where our work has shortcomings in each test. This is where we believe future work can be done to improve certainty and fill the gaps of these shortcomings. Activation maximizers and parameters sensitivity tests have limited generalization (to all layers of the network) which has the consequence of having no complete conclusive evidence for the theory we presented on illuminant estimation workflow in the models. Also, more parameters can be studied and their contributions connected directly to the workflow of the models.

The Activation Maximizers tested in this thesis are applied to some kernels in some layers, but more work can be done to optimize the objective using matrix factorization as we discussed. This can lead to more informative and reliable results. Also, more layers and models can be tested with more regularization parameters which can yield even more insight into the models.

Most importantly, the up-sampling workflow used in these models is still unclear and needs further studying. Many insights from this work can be tracked further to establish their connection to the models. For example, the shape-dependent saliency seen in some models like HINET and MST++ (single image analysis section), which needs investigating on how the models track these shapes and how they incorporate them in the workflow. Also, many ideas presented by Activation Maximizers can be studied further like background segmentation.

Also, the color and illuminant test needs to be conducted on a larger scale to establish its validity since the test done here was done on limited number of images.

All that is towards the goal of ultimately establishing a line of trust for the users so they know which models to use and how to use them. Finally, more parameters can be studied besides the ones analysed in the parameters sensitivity test and more color spaces can be employed to extract more important parameters.

A Appendix

Here we show some more results for the tests that we could not show due to lack of space in the main body of the report. We also show a test that we chose not to include in the main body due to its redundancy, but it acts as a supporting evidence.

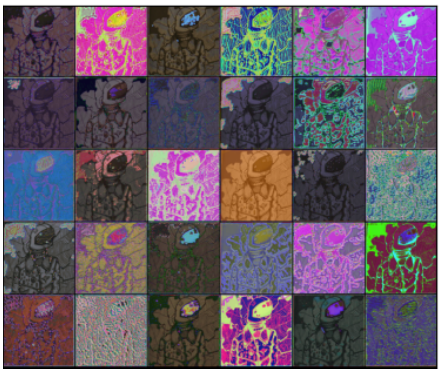
A.1 Activation Maximizers for Global Models

Figure A.1 shows the same Activation Maximizers method applied to the 30 kernels on 3 stages of the MST++ model. The architecture of this model consists of 3 encoder decoder modules connected to each other making the main stream of the model. We use the output of each of these modules to visualize here since these resemble the division of the network into 3 thirds. Only 30 kernels exist in these layers of the MST++ model so we show all these 30 kernels.

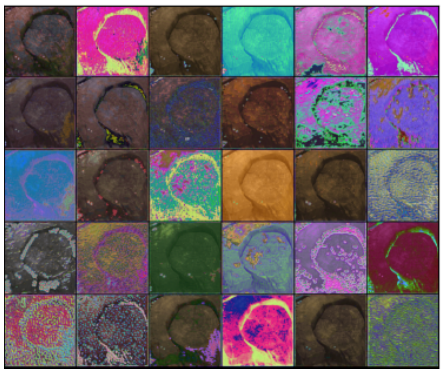
As we can see, the behaviour that was seen in local models like HSCNN+ and HRNET shown before cannot be seen here in this global model. All stages seem to have various kinds of shapes and colors. We could not find a specific pattern in these models like we did for local models. The colors of the last stage does not seem to have colors of an illuminant (not a natural illuminant at least). They instead seem to portray highly saturated colors and strong not very different from those in the first stage. Clearly, more work needs to be done on these models to understand how their behaviour.

A.2 More Error Maps

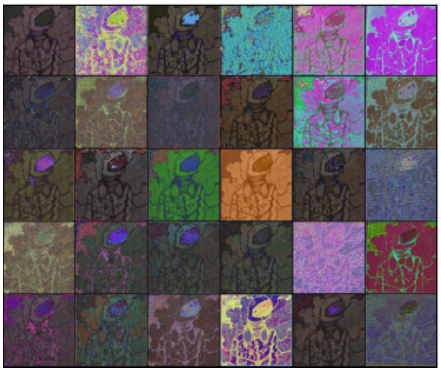
Since error maps are examined individually, we show here a few more examples to affirm the reader of our conclusions. Figure A.2 shows these maps for other images than previously shown and can lead to the same conclusions. We can also see in



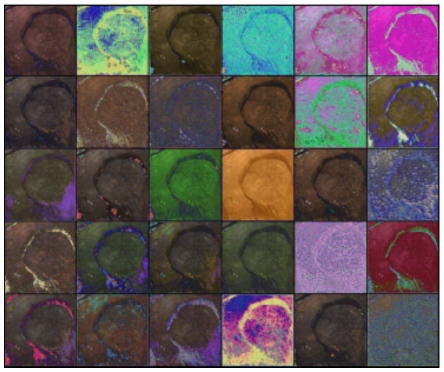
(a) *After first module example 1.*



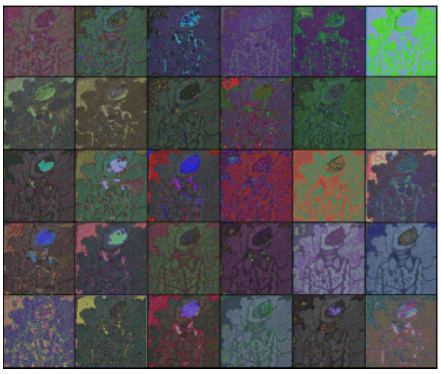
(b) *After first module example 2.*



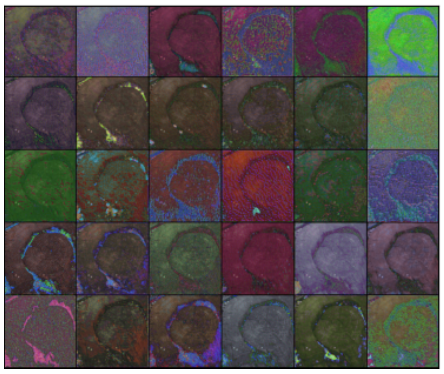
(c) *After second module example 1.*



(d) *After second module example 2.*



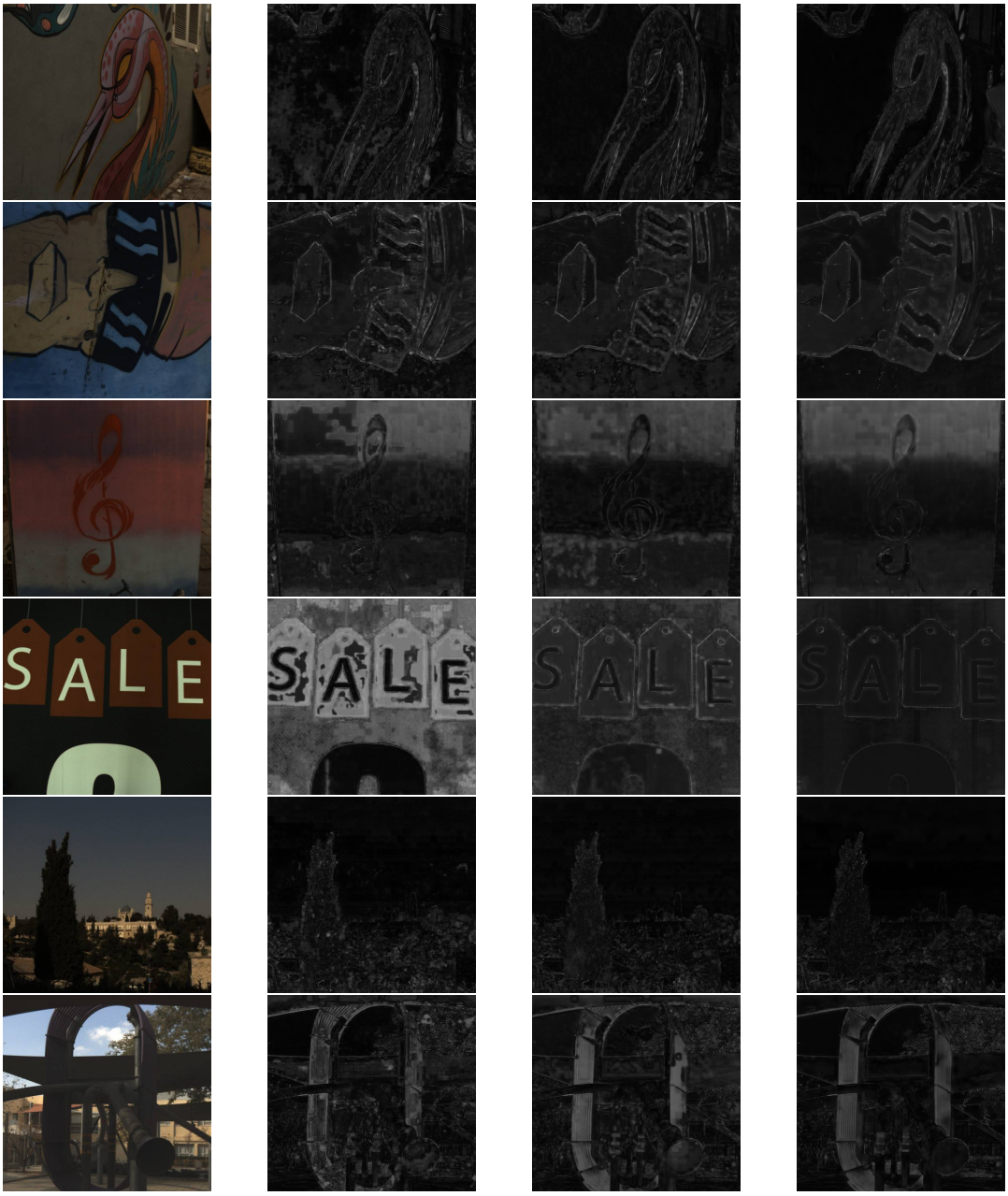
(e) *After third module example 1.*



(f) *After third module example 2.*

Figure A.1: *Activation Maximizers output of 2 examples on 30 kernels for the MST++ model at 3 stages, each stage after an encoder-decoder module in the network.*

the last 2 photos that when the sky is not overcast (blue sky in this case), the models can easily reconstruct it.



(a) *Original.* (b) *HSCNN+.* (c) *HDNET.* (d) *MST++.*

Figure A.2: *More example error maps for images from NTIRE for some of the models showcasing a variety of scenes with various characteristics.*

A.3 Thin Edge Test

This test is not well established and was made redundant, but it points to the edge problem in the models. It was actually my first clue into the edge problem and local saliency in the models before saliency maps and box tests. It is a very simple test that intends to find the impact of having an edge in the image on the model's performance and how thin that edge should be to start causing performance problems to the model. The test does the following:

For an original image, take a thin spatial line (we choose it to be vertical but it could be in any orientation). Take the values of that line in the image and assign it to another place in the image. Then, assign the values replaced to the place of the original line (switch 2 thin lines with each other). That creates 2 artificial edges for each image like shown in the top row of figure A.3 where an original image has these 2 thin lines. Then, the width of that line is increased gradually from 1 pixel to 20 pixels. The increase is shown (in 4 images out of 20) in the top row of figure A.3. Then, the error map (here it is SAM error) is monitored for each image with each line for each model. We can see that the very thin line at the beginning has high error (white line in the error maps), but when the width increase to a certain point, the error recovers (becomes black).

This is a very simple and naive approach and was not quantified and substantiated well, but it was my first clue to the models having problems with edges and their possible local saliency.

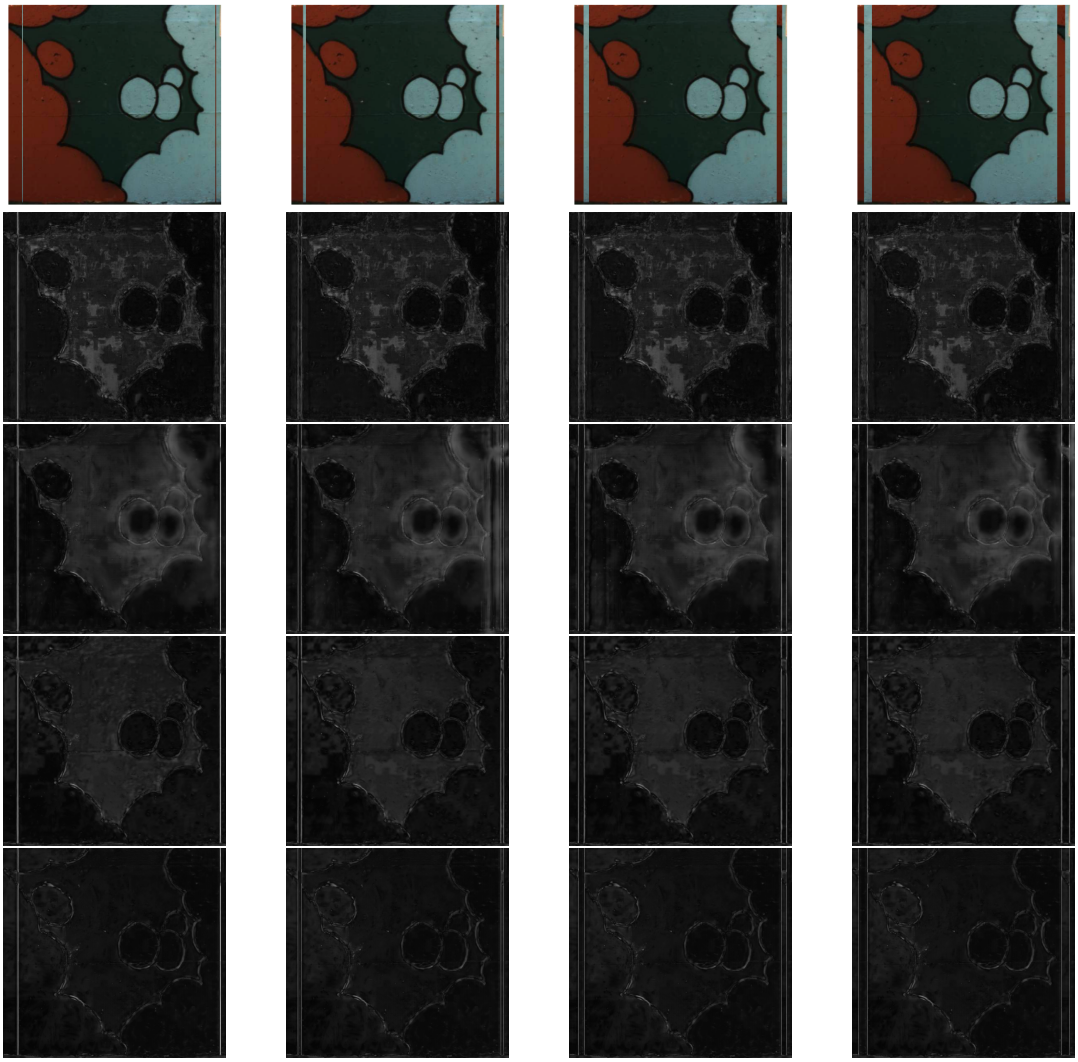


Figure A.3: *Thin edge test results showing the original image input to the model and then the SAM error map for HSCNN+, HRNET, HDNET, and MST++ from top to bottom respectively.*

Bibliography

Aeschbacher, J., Wu, J., and Timofte, R. (2017). In defense of shallow learned spectral reconstruction from rgb images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 471–479. (cited on page 6)

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433. (cited on page 17)

Arad, B. and Ben-Shahar, O. (2016). Sparse recovery of hyperspectral signal from natural rgb images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 19–34. Springer International Publishing. (cited on pages 7, 12, and 52)

Arad, B., Timofte, R., Yahel, R., Morag, N., Bernat, A., Cai, Y., Lin, J., Lin, Z., Wang, H., Zhang, Y., et al. (2022). Ntire 2022 spectral recovery challenge and data set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 863–881. (cited on pages 3, 11, 41, 44, 45, and 132)

Arrieta, A. B., Rodríguez, N. D., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115. (cited on page 26)

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140. (cited on pages 16, 23, 32, 33, and 132)

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831. (cited on pages 16 and 24)

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549. (cited on page 37)

Becker, S., Ackermann, M., Lapuschkin, S., Müller, K.-R., and Samek, W. (2018). Interpreting and explaining deep neural networks for classification of audio signals. *arXiv preprint arXiv:1807.03418*. (cited on page 19)

Brendel, W. and Bethge, M. (2019). Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*. (cited on page 16)

Buhrmester, V., Münch, D., and Arens, M. (2021). Analysis of explainers of black box deep neural networks for computer vision: A survey. *Machine Learning and Knowledge Extraction*, 3(4):966–989. (cited on page 23)

Byun, S.-Y. and Lee, W. (2022). Recipro-cam: Gradient-free reciprocal class activation map. *arXiv preprint arXiv:2209.14074*. (cited on page 26)

Cai, Y., Hu, X., Wang, H., Zhang, Y., Pfister, H., and Wei, D. (2021). Learning to generate realistic noisy images via pixel-level noise-aware adversarial training. *Advances in Neural Information Processing Systems*, 34:3259–3270. (cited on page 7)

Cai, Y., Lin, J., Hu, X., Wang, H., Yuan, X., Zhang, Y., Timofte, R., and Van Gool, L. (2022a). Mask-guided spectral-wise transformer for efficient hyperspectral image reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17502–17511. (cited on pages 5, 6, 11, and 131)

Cai, Y., Lin, J., Lin, Z., Wang, H., Zhang, Y., Pfister, H., Timofte, R., and Van Gool, L. (2022b). Mst++: Multi-stage spectral-wise transformer for efficient spectral reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 745–755. (cited on pages 6, 7, 10, 12, 42, 62, and 131)

Calin, M. A., Parasca, S. V., Savastru, D., and Manea, D. (2014). Hyperspectral imaging in the medical field: Present and future. *Applied Spectroscopy Reviews*, 49(6):435–447. (cited on page 1)

Cao, X., Yue, T., Lin, X., Lin, S., Yuan, X., Dai, Q., Carin, L., and Brady, D. J. (2016). Computational snapshot multispectral cameras: Toward dynamic capture of the spectral world. *IEEE Signal Processing Magazine*, 33(5):95–108. (cited on page 5)

Carter, S., Armstrong, Z., Schubert, L., Johnson, I., and Olah, C. (2019). Activation atlas. *Distill*, 4(3):e15. (cited on pages 21, 23, 24, 38, 73, 74, and 134)

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (2015a). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730. (cited on page 15)

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (2015b). *Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission*, page 1721–1730. (cited on page 16)

Chakrabarti, A. and Zickler, T. (2011). Statistics of real-world hyperspectral images. In *CVPR 2011*, pages 193–200. IEEE. (cited on pages 2 and 6)

Chen, L., Chen, J., Hajimirsadeghi, H., and Mori, G. (2020). Adapting grad-cam for embedding networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2794–2803. (cited on page 23)

Chen, L., Lu, X., Zhang, J., Chu, X., and Chen, C. (2021). Hinet: Half instance normalization network for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 182–192. (cited on pages 13, 14, 42, and 131)

Christophe, E., Léger, D., and Mailhes, C. (2005). Quality criteria benchmark for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43(9):2103–2114. (cited on page 1)

Cucci, C., Delaney, J. K., and Picollo, M. (2016). Reflectance hyperspectral imaging for investigation of works of art: old master paintings and illuminated manuscripts. *Accounts of chemical research*, 49(10):2070–2079. (cited on page 1)

Dale, L. M., Thewis, A., Boudry, C., Rotar, I., Dardenne, P., Baeten, V., and Pierna, J. A. F. (2013). Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review. *Applied Spectroscopy Reviews*, 48(2):142–159. (cited on pages 1 and 18)

Deng, Z., Cai, Y., Chen, L., Gong, Z., Bao, Q., Yao, X., Fang, D., Yang, W., Zhang, S., and Ma, L. (2022). Rformer: Transformer-based generative adversarial network for real fundus image restoration on a new clinical benchmark. *IEEE Journal of Biomedical and Health Informatics*, 26(9):4645–4655. (cited on page 7)

Dong, C., Loy, C., He, K., and Tang, X. (2014a). Image super-resolution using deep convolutional networks. arxiv e-prints. *arXiv preprint arXiv:1501.00092*, 2. (cited on page 20)

Dong, C., Loy, C. C., He, K., and Tang, X. (2014b). Learning a deep convolutional network for image super-resolution. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*, pages 184–199. Springer. (cited on page 20)

Došilović, F. K., Brčić, M., and Hlupić, N. (2018). Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE. (cited on pages 17 and 26)

Du, H., Tong, X., Cao, X., and Lin, S. (2009). A prism-based system for multispectral video acquisition. In *2009 IEEE 12th International Conference on Computer Vision*, pages 175–182. IEEE. (cited on page 5)

Eberle, O., Büttner, J., Kräutli, F., Müller, K.-R., Valleriani, M., and Montavon, G. (2020). Building and interpreting deep similarity models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1. (cited on pages 15 and 16)

Egmont-Petersen, M., de Ridder, D., and Handels, H. (2002). Image processing with neural networks—a review. *Pattern recognition*, 35(10):2279–2301. (cited on page 19)

Elmasry, G., Kamruzzaman, M., Sun, D.-W., and Allen, P. (2012). Principles and applications of hyperspectral imaging in quality evaluation of agro-food products: a review. *Critical reviews in food science and nutrition*, 52(11):999–1023. (cited on pages 1 and 18)

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1. (cited on page 35)

Fei, B. (2019). Hyperspectral imaging in medical applications. In *Data Handling in Science and Technology*, volume 32, pages 523–565. Elsevier. (cited on page 1)

Foster, D. H., Amano, K., Nascimento, S. M., and Foster, M. J. (2007). The frequency of metamerism in natural scenes. *Optics and Photonics News*, 18(12):47–47. (cited on pages 45, 80, and 85)

Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., and Li, B. (2020). Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. *arXiv preprint arXiv:2008.02312*. (cited on page 26)

Galliani, S., Lanaras, C., Marmanis, D., Baltsavias, E., and Schindler, K. (2017). Learned spectral super-resolution. *arXiv preprint arXiv:1703.09470*. (cited on pages 7 and 52)

Goel, M., Whitmire, E., Mariakakis, A., Saponas, T. S., Joshi, N., Morris, D., Guenter, B., Gavriliu, M., Borriello, G., and Patel, S. N. (2015). Hypercam: hyperspectral imaging for ubiquitous computing applications. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 145–156. (cited on page 6)

Gupta, L. K., Koundal, D., and Mongia, S. (2023). Explainable methods for image-based deep learning: a review. *Archives of Computational Methods in Engineering*, 30(4):2651–2666. (cited on pages 15, 16, 25, 29, 131, and 132)

Hölldobler, S., Möhle, S., and Tigunova, A. (2017). Lessons learned from alphago. In *YSIP*, pages 92–101. (cited on page 17)

Horak, K. and Sablatnig, R. (2019). Deep learning concepts and datasets for image recognition: overview 2019. In *Eleventh international conference on digital image processing (ICDIP 2019)*, volume 11179, pages 484–491. SPIE. (cited on page 17)

Hou, Z., Qin, M., Sun, F., Ma, X., Yuan, K., Xu, Y., Chen, Y.-K., Jin, R., Xie, Y., and Kung, S.-Y. (2022). Chex: Channel exploration for cnn model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12287–12298. (cited on page 21)

Hu, X., Cai, Y., Lin, J., Wang, H., Yuan, X., Zhang, Y., Timofte, R., and Van Gool, L. (2022). Hdnet: High-resolution dual-domain learning for spectral compressive imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17542–17551. (cited on pages 13 and 42)

Hu, X., Cai, Y., Liu, Z., Wang, H., and Zhang, Y. (2021a). Multi-scale selective feedback network with dual loss for real image denoising. In *IJCAI*, pages 729–735. (cited on page 7)

Hu, X., Wang, H., Cai, Y., Zhao, X., and Zhang, Y. (2021b). Pyramid orthogonal attention network based on dual self-similarity for accurate mr image super-resolution. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE. (cited on page 7)

Huang, T.-H., Ferraro, F., Mostafazadeh, N., Misra, I., Agrawal, A., Devlin, J., Girshick, R., He, X., Kohli, P., Batra, D., et al. (2016). Visual storytelling. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1233–1239. (cited on page 17)

Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., and Baesens, B. (2011). An empirical evaluation of the comprehensibility of decision table, tree and rule

based predictive models. *Decision Support Systems*, 51(1):141–154. (cited on page 15)

Jia, Y., Zheng, Y., Gu, L., Subpa-Asa, A., Lam, A., Sato, Y., and Sato, I. (2017). From rgb to spectrum for natural scenes via manifold-based mapping. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713. (cited on page 7)

Johns, E., Mac Aodha, O., and Brostow, G. J. (2015). Becoming the expert-interactive multi-class machine teaching. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2616–2624. (cited on page 17)

Kang, X., Zhang, X., Li, S., Li, K., Li, J., and Benediktsson, J. A. (2017). Hyperspectral anomaly detection with attribute and edge-preserving filters. *IEEE Transactions on Geoscience and Remote Sensing*, 55(10):5600–5611. (cited on page 1)

Karpathy, A. (2014). What i learned from competing against a convnet on imagenet. *Andrej Karpathy Blog*, 5:1–15. (cited on page 17)

Kim, B. (2015). *Interactive and interpretable machine learning models for human machine collaboration*. PhD thesis, Massachusetts Institute of Technology. (cited on page 15)

Kim, B., Glassman, E., Johnson, B., and Shah, J. (2015). ibcm: Interactive bayesian case model empowering humans via intuitive interaction. (cited on page 15)

Kolekar, S., Gite, S., Pradhan, B., and Alamri, A. (2022). Explainable ai in scene understanding for autonomous vehicles in unstructured traffic environments on indian roads using the inception u-net model with grad-cam visualization. *Sensors*, 22(24):9677. (cited on page 23)

Krening, S., Harrison, B., Feigh, K. M., Isbell, C. L., Riedl, M., and Thomaz, A. (2016). Learning from explanations using sentiment and advice in rl. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1):44–55. (cited on page 15)

Kwaśniewska, A., Rumiński, J., and Rad, P. (2017). Deep features class activation map for thermal face detection and tracking. In *2017 10Th international conference on human system interactions (HSI)*, pages 41–47. IEEE. (cited on page 26)

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications*, 10:1096. (cited on page 16)

Lee, D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13. (cited on page 37)

Letzgus, S., Wagner, P., Lederer, J., Samek, W., Müller, K.-R., and Montavon, G. (2021). Toward explainable ai for regression models. *arXiv preprint arXiv:2112.11407*. (cited on pages 15, 16, 23, and 38)

Li, J., Wu, C., Song, R., Li, Y., and Liu, F. (2020). Adaptive weighted attention network with camera spectral sensitivity prior for spectral reconstruction from rgb images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 462–463. (cited on pages 3 and 10)

Lin, X., Liu, Y., Wu, J., and Dai, Q. (2014). Spatial-spectral encoded compressive hyperspectral imaging. *ACM Transactions on Graphics (TOG)*, 33(6):1–11. (cited on page 5)

Lipton, Z. C. (2016). The mythos of model interpretability (2016). *arXiv preprint arXiv:1606.03490*. (cited on pages 14, 18, and 19)

Llull, P., Liao, X., Yuan, X., Yang, J., Kittle, D., Carin, L., Sapiro, G., and Brady, D. J. (2013). Coded aperture compressive temporal imaging. *Optics express*, 21(9):10526–10545. (cited on page 5)

Lou, Y., Caruana, R., Gehrke, J., and Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. (cited on page 14)

Lu, G. and Fei, B. (2014). Medical hyperspectral imaging: a review. *Journal of biomedical optics*, 19(1):010901–010901. (cited on pages 1 and 18)

Lugmayr, A., Danelljan, M., and Timofte, R. (2020). Ntire 2020 challenge on real-world image super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 494–495. (cited on pages 11 and 45)

Mahapatra, D., Bozorgtabar, B., Hewavitharanage, S., and Garnavi, R. (2017). Image super resolution using generative adversarial networks and local saliency maps for retinal image analysis. In *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec*

City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20, pages 382–390. Springer International Publishing. (cited on page 23)

Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196. (cited on page 37)

Mahendran, A. and Vedaldi, A. (2016). Salient deconvolutional networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 120–135. Springer International Publishing. (cited on pages 23 and 31)

Maier, A., Syben, C., Lasser, T., and Riess, C. (2019). A gentle introduction to deep learning in medical image processing. *Zeitschrift für Medizinische Physik*, 29(2):86–101. (cited on page 17)

McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. (cited on page 15)

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*. (cited on page 38)

Meng, Z., Ma, J., and Yuan, X. (2020). End-to-end low cost compressive spectral imaging with spatial-spectral self-attention. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 187–204. Springer. (cited on page 5)

Millan, M. and Achard, C. (2020). Explaining regression based neural network model. *arXiv preprint arXiv:2004.06918*. (cited on pages 18, 19, 23, 24, 28, and 39)

Mnih, A. and Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. *Advances in neural information processing systems*, 20. (cited on page 37)

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern recognition*, 65:211–222. (cited on page 19)

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15. (cited on pages 19, 37, and 74)

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., and Boracchi, G. (2020). Augmented grad-cam: Heat-maps super resolution through augmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4067–4071. IEEE. (cited on page 26)

Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks. (cited on page 37)

Mücke, M., Sang, B., Heider, B., Honold, H.-P., Sornig, M., and Fischer, S. (2019). Enmap: hyperspectral imager (hsi) for earth observation: current status. In *International Conference on Space Optics—ICSO 2018*, volume 11180, pages 2228–2237. SPIE. (cited on pages 1 and 18)

Mundhenk, T. N., Chen, B. Y., and Friedland, G. (2019). Efficient saliency maps for explainable ai. *arXiv preprint arXiv:1911.11293*. (cited on pages 23, 31, and 34)

Naidu, R., Ghosh, A., Maurya, Y., Kundu, S. S., et al. (2020). Is-cam: Integrated score-cam for axiomatic-based explanations. *arXiv preprint arXiv:2010.03023*. (cited on page 26)

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. (2017). Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4467–4477. (cited on page 37)

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016a). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in neural information processing systems*, 29. (cited on page 37)

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436. (cited on page 37)

Nguyen, A., Yosinski, J., and Clune, J. (2016b). Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*. (cited on page 38)

Nguyen, A. M., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016c). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, pages 3387–3395. (cited on pages 15 and 23)

Oh, S. W., Brown, M. S., Pollefeys, M., and Kim, S. J. (2016). Do it yourself hyperspectral imaging with everyday digital cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2461–2469. (cited on page 6)

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*, 2(11):e7. (cited on pages 21, 35, 36, 37, 71, 73, 74, 132, and 134)

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*, 3(3):e10. (cited on pages 34, 35, 37, and 74)

Omeiza, D., Speakman, S., Cintas, C., and Weldermariam, K. (2019). Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *arXiv preprint arXiv:1908.01224*. (cited on page 26)

O'Shea, R. J., Horst, C., Manickavasagar, T., Hughes, D., Cusack, J., Tsoka, S., Cook, G., and Goh, V. (2022). Weakly supervised unet: an image classifier which learns to explain itself. *bioRxiv*, pages 2022–09. (cited on page 23)

Ozbulak, U. (2019). Pytorch cnn visualizations. `https://github.com/utkuozbulak/pytorch-cnn-visualizations`. (cited on pages 2, 28, 30, 33, 131, and 132)

Parmar, M., Lansel, S., and Wandell, B. A. (2008). Spatio-spectral reconstruction of the multispectral datacube using sparse recovery. In *2008 15th IEEE International Conference on Image Processing*, pages 473–476. IEEE. (cited on page 7)

Patro, B. N., Lunayach, M., Patel, S., and Namboodiri, V. P. (2019). U-cam: Visual explanation using uncertainty based class activation maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7444–7453. (cited on page 26)

Patro, R. N., Subudhi, S., Biswal, P. K., and Dell'acqua, F. (2021). A review of unsupervised band selection techniques: Land cover classification for hyperspectral earth observation data. *IEEE Geoscience and Remote Sensing Magazine*, 9(3):72–111. (cited on pages 1 and 18)

Picollo, M., Cucci, C., Casini, A., and Stefani, L. (2020). Hyper-spectral imaging technique in the cultural heritage field: New possible scenarios. *Sensors*, 20(10):2843. (cited on page 1)

Pinciroli Vago, N. O., Milani, F., Fraternali, P., and da Silva Torres, R. (2021). Comparing cam algorithms for the identification of salient image features in iconography artwork analysis. *Journal of Imaging*, 7(7):106. (cited on page 26)

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016a). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. (cited on pages 25, 26, and 131)

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016b). "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM. (cited on page 16)

Robles-Kelly, A. (2015). Single image spectral reconstruction for multimedia applications. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 251–260. (cited on page 7)

Robnik-Šikonja, M. and Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600. (cited on pages 19 and 20)

Robnik-Šikonja, M., Likas, A., Constantinopoulos, C., Kononenko, I., and Štrumbelj, E. (2011). Efficiently explaining decisions of probabilistic rbf classification networks. In *Adaptive and Natural Computing Algorithms: 10th International Conference, ICANNGA 2011, Ljubljana, Slovenia, April 14-16, 2011, Proceedings, Part I 10*, pages 169–179. Springer. (cited on page 19)

Salazar-Vazquez, J. and Mendez-Vazquez, A. (2020). A plug-and-play hyperspectral imaging sensor using low-cost equipment. *HardwareX*, 7:e00087. (cited on page 2)

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*. (cited on page 17)

Sattarzadeh, S., Sudhakar, M., Plataniotis, K. N., Jang, J., Jeong, Y., and Kim, H. (2021). Integrated grad-cam: Sensitivity-aware visual explanation of deep convolutional networks via integrated gradient-based scoring. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1775–1779. IEEE. (cited on page 26)

Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K. T., Müller, K.-R., and Montavon, G. (2021). Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page doi: 10.1109/TPAMI.2021.3115452. (cited on pages 15 and 16)

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626. (cited on pages 17, 23, 24, 26, 27, 28, 32, and 132)

Shi, Z., Chen, C., Xiong, Z., Liu, D., and Wu, F. (2018). Hscnn+: Advanced cnn-based hyperspectral recovery from rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 939–947. (cited on pages 7, 42, and 52)

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR. (cited on pages 23 and 32)

Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*. (cited on pages 32, 33, and 132)

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489. (cited on page 17)

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*. (cited on pages 23, 28, 30, 49, and 132)

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*. (cited on pages 15, 16, and 23)

Smilkov, D., Thorat, N., Kim, B., Viégas, F. B., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*. (cited on page 32)

Song, W., Dai, S., Huang, D., Song, J., and Antonio, L. (2021). Median-pooling grad-cam: An efficient inference level visual explanation for cnn networks in remote sensing image classification. In *MultiMedia Modeling: 27th International Conference, MMM 2021, Prague, Czech Republic, June 22–24, 2021, Proceedings, Part II 27*, pages 134–146. Springer. (cited on page 26)

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*. (cited on pages 4, 23, 30, 49, and 132)

Stiebel, T., Koppers, S., Seltsam, P., and Merhof, D. (2018). Reconstructing spectral images from rgb-images using a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 948–953. (cited on page 8)

Strumbelj, E. and Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18. (cited on page 16)

Stuart, M. B., McGonigle, A. J., Davies, M., Hobbs, M. J., Boone, N. A., Stanger, L. R., Zhu, C., Pering, T. D., and Willmott, J. R. (2021). Low-cost hyperspectral imaging with a smartphone. *Journal of Imaging*, 7(8):136. (cited on page 2)

Sundararajan, M., Taly, A., and Yan, Q. (2016). Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*. (cited on page 32)

Sundararajan, M., Taly, A., and Yan, Q. (2017a). Axiomatic attribution for deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR. (cited on pages 16, 32, 33, and 132)

Sundararajan, M., Taly, A., and Yan, Q. (2017b). Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR. (cited on page 16)

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. (cited on page 35)

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. (cited on page 37)

Takatani, T., Aoto, T., and Mukaigawa, Y. (2017). One-shot hyperspectral imaging using faced reflectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4039–4047. (cited on page 6)

Timofte, R., Gu, S., Wu, J., and Van Gool, L. (2018). Ntire 2018 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 852–863. (cited on pages 7 and 45)

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11). (cited on pages 15 and 38)

Van Nguyen, H., Banerjee, A., and Chellappa, R. (2010). Tracking via object reflectance using a hyperspectral video camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 44–51. IEEE. (cited on page 1)

Wagadarikar, A., John, R., Willett, R., and Brady, D. (2008). Single disperser design for coded aperture snapshot spectral imaging. *Applied optics*, 47(10):B44–B51. (cited on pages 2 and 5)

Wagadarikar, A. A., Pitsianis, N. P., Sun, X., and Brady, D. J. (2009). Video rate spectral imaging using a coded aperture snapshot spectral imager. *Optics express*, 17(8):6368–6388. (cited on page 5)

Wang, H., Naidu, R., Michael, J., and Kundu, S. S. (2020a). Ss-cam: Smoothed score-cam for sharper visual feature localization. *arXiv preprint arXiv:2006.14255*. (cited on page 26)

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., and Hu, X. (2020b). Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 24–25. (cited on page 26)

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR. (cited on page 15)

Wang, Z., Xie, X., Zhao, Q., and Shi, G. (2022). Filter clustering for compressing cnn model with better feature diversity. *IEEE Transactions on Circuits and Systems for Video Technology*. (cited on page 21)

Wiley, V. and Lucas, T. (2018). Computer vision and image processing: a paper review. *International Journal of Artificial Intelligence Research*, 2(1):29–36. (cited on page 19)

Xiong, Z., Shi, Z., Li, H., Wang, L., Liu, D., and Wu, F. (2017). Hscnn: Cnn-based hyperspectral image recovery from spectrally undersampled projections. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 518–525. (cited on pages 7, 8, and 131)

Yan, Y., Zhang, L., Li, J., Wei, W., and Zhang, Y. (2018). Accurate spectral super-resolution from single rgb image using multi-scale cnn. In *Pattern Recognition*

and Computer Vision: First Chinese Conference, PRCV 2018, Guangzhou, China, November 23-26, 2018, Proceedings, Part II 1, pages 206–217. Springer International Publishing. (cited on pages 8, 9, 42, and 131)

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer. (cited on pages 24, 29, 31, 49, and 132)

Zhang, J., Su, R., Fu, Q., Ren, W., Heide, F., and Nie, Y. (2022). A survey on computational spectral reconstruction methods from rgb to hyperspectral imaging. *Scientific reports*, 12(1):11905. (cited on page 2)

Zhang, L., Lang, Z., Wang, P., Wei, W., Liao, S., Shao, L., and Zhang, Y. (2020). Pixel-aware deep function-mixture network for spectral super-resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12821–12828. (cited on pages 8, 9, 10, and 131)

Zhang, Q.-s. and Zhu, S.-C. (2018). Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39. (cited on pages 17 and 22)

Zhao, Y., Po, L.-M., Yan, Q., Liu, W., and Lin, T. (2020). Hierarchical regression network for spectral reconstruction from rgb images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 422–423. (cited on pages 10, 11, 42, 52, and 131)

Zhou, B., Bau, D., Oliva, A., and Torralba, A. (2018). Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2131–2145. (cited on page 26)

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929. (cited on pages 26, 27, and 132)

Zitnick, C. L., Agrawal, A., Antol, S., Mitchell, M., Batra, D., and Parikh, D. (2016). Measuring machine intelligence through visual question answering. *AI Magazine*, 37(1):63–72. (cited on page 17)

BIBLIOGRAPHY

List of Figures

1.1 Difference between RGB, multi-spectral imaging, and hyperspectral imaging Ozbulak (2019). 2

2.1 Coded aperture snapshot spectral imaging (CASSI) Cai et al. (2022a). 6
2.2 Architectures of the baseline HSCNN (a), HSCNN-R (b), and HSCNN-D (c). Quoting from Xiong et al. (2017): The "C" with a rectangular block denotes a convolution, and the following "1" and "3" denote the kernel size (i.e., 1×1 and 3×3 respectively). The "R" represents a ReLU activation function. And the "C" with a circular block denotes a concatenation Xiong et al. (2017). 8
2.3 The U-Net architecture used in Yan et al. (2018). 9
2.4 Regular receptive fields (left) and adaptive receptive fields used in Zhang et al. (2020) (right). 10
2.5 HRNET Architecture with its 4-level hierarchy Zhao et al. (2020). . 11
2.6 (a) Global Multi-headed Self Attention (G-MSA). (b) Window MSA (W-MSA). (c) Spectral MSA (S-MSA) Cai et al. (2022b). 12
2.7 (a) Half Instance Normalization Block (HIN). (b) Residual Block (Res). Chen et al. (2021). 14
2.8 Trade-off between complexity and explainability in machine learning Gupta et al. (2023). 16
2.9 Different classes of models in terms of explainability need. 21
2.10 Breaking down a black box (Class C or D) into smaller task specific sub-nets (class A or B). The sequential case being unrealistic but equivalent representation of the more realistic overlapping nature of sub-nets doing specific tasks. 22
2.11 Perturbation based approaches: (a) Original image. (b) Perturbed dataset. (c) Attribution map based on the most sensitive parts of the image. Gupta et al. (2023) 25
2.12 LIME applied on Google Inception model showing the original image (left) and saliency maps for the top 3 classes (electric guitar, acoustic guitar, Labrador) respectively Ribeiro et al. (2016a). 26

2.13 CAM workflow and example Zhou et al. (2016). 27
2.14 Grad-CAM saliency maps examples (bottom) on top of original image (top) for classes (King Snake, Mastiff, Spider) left to right respectively Selvaraju et al. (2017); Ozbulak (2019). Most salient areas are red, then blue, then yellow, then green. 28
2.15 Gradient Based methods: (a) Forward pass. (b) Backward pass. Gupta et al. (2023) 29
2.16 (Top) Back-propagation examples, (bottom) Guided Back-propagation examples. Original images are shown in figure 2.14. Simonyan et al. (2013); Springenberg et al. (2014); Ozbulak (2019). 30
2.17 The concept of switches and unpooling in Deconvolution Zeiler and Fergus (2014). 31
2.18 (Top) LRP applied to layer 7 examples, (bottom) (Grad x input) examples. Original images and target classes are shown and mentioned in figure 2.14. Bach et al. (2015); Shrikumar et al. (2016); Ozbulak (2019). 33
2.19 Integrated Gradients example. Original images and target classes are shown and mentioned in figure 2.14. Sundararajan et al. (2017a); Ozbulak (2019). 33
2.20 Feature visualization for the 11th channel in the mixed4a layer of GoogleNet done by Olah et al. (2017). 36
2.21 Different possible objectives (top) and their respective feature visualization (bottom) Olah et al. (2017). 36

3.1 Camera function adopted by the NTIRE 2022 dataset and challenge Arad et al. (2022). 45
3.2 Example error maps for images from NTIRE for some of the models showcasing a variety of scenes with various characteristics. The bottom image shows a case of vague setup where errors are high. Some original images' brightness was adjusted for better visualization. 47
3.3 Examples of Guided back-propagation applied directly to HSCNN+. 50
3.4 Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 52
3.5 Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 54

3.6 Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency for randomly chosen pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. Full size images are displayed to show global saliency. 55
3.7 Detected edges using Canny edge filter to isolate them and generate their saliency maps. 57
3.8 Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 57
3.9 Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 58
3.10 Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency, in flat and 3D for edge pixels (full size images are displayed to show global saliency) averaged over all the NTIRE 2022 images. 59
3.11 Saliency maps generated using modified Guided Back-Propagation for tested models that showed local saliency, for image edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 60
3.12 Saliency maps generated using modified Guided Back-Propagation for tested models that showed partially local saliency, for image edge pixels (centered in the middle for convenience) averaged over all the NTIRE 2022 images. 60
3.13 Saliency maps generated using modified Guided Back-Propagation for tested models that showed global saliency, in flat and 3D for image edge pixels (full size images are displayed to show global saliency) averaged over all the NTIRE 2022 images. 61
3.14 Examples of saliency maps on a single pixel (x = 100, y = 200) for single images for HINET (top), HSCNN+ (mid), and MST++ (bottom). 63
3.15 Examples showing the images fed to the models in each one of the box tests and their varying size from small to large. 66
3.16 Results of test 1 for some of the models. 67
3.17 Results of test 2 for some of the models. 68
3.18 Results of test 3 for some of the models. 69

3.19 Difference between feature visualization Olah et al. (2017), Activation Atlases Carter et al. (2019), and the proposed method (Activation Maximizers). 73
3.20 Hooks applied to a model. The model keeps its original graph. . . . 75
3.21 Feature Extractors as sub-models each with a different graph. . . . 76
3.22 Original images for the examples used in figures 3.23 and 3.24. . . . 77
3.23 Activation Maximizers output of 2 examples on 64 kernels for the HSCNN+ model at 3 stages: input block, after block 15/30, and after block 29/30. 78
3.24 Activation Maximizers output of 2 examples on 64 kernels for the HRNET model at 3 stages: mainstream layers 2, 5 and 8 (last mainstream layer). 79
3.25 Generated visualizations of 4 examples on 64 kernels for the HSCNN+ model all after block 29/30. with the activation added as alpha channel. In each example, only 1 or 2 kernels are activated significantly. 82
3.26 Normalized activations for 2 local models on their last layers across the whole NTIRE dataset showing only particular kernels being strongly activated in each model. 83
3.27 Generated visualizations of 3 examples with 2 different illuminants (D65 and A) with the activations applied as the alpha channel from the last layer of the HRNET and the HSCNN+ models 84
3.28 Normalized activations for all images in the Manchester dataset for both D65 and A illuminant for the HRNET model. 86
3.29 Normalized activations for all images in the Manchester dataset for both D65 and A illuminant for the HSCNN+ model. 87
3.30 Normalized activations for 2 local models on their last layers across the outdoor NTIRE images showing even more discrepancy between activated kernels. 88
3.31 Normalized activations for 2 local models on their last layers across the indoor NTIRE images showing more random activated kernels. . 89
3.32 Examples of the generated variations for an example original image (top) with variations of hue, frequency, saturation, and noise (respectively from top to bottom). 93
3.33 Activation variance for HRNET throughout the network. 94
3.34 Activation variance for MST++ throughout the network. 95
3.35 Activation variance for HSCNN+ throughout the network. 96
3.36 Activation variance for the MultiScale model. 97
3.37 RGB images taken for the X-Rite color checker under both D65 and A lights. Same setup was used to take hyperspectral images of the same scene. 99

3.38 Mean SAM errors for some of the models on specific colors under both D65 and A illuminants (lblue is light blue). 100

4.1 Diagram explaining the suggested equivalent to local reconstruction models. 105

A.1 Activation Maximizers output of 2 examples on 30 kernels for the MST++ model at 3 stages, each stage after an encoder-decoder module in the network. 110
A.2 More example error maps for images from NTIRE for some of the models showcasing a variety of scenes with various characteristics. . 111
A.3 Thin edge test results showing the original image input to the model and then the SAM error map for HSCNN+, HRNET, HDNET, and MST++ from top to bottom respectively. 113

List of Tables

3.1 Table depicting the hyperspectral reconstruction models tested and used throughout this work ordered by their overall performance. . . 42
3.2 Table depicting the area of saliency (observed area) in pixels with a threshold of 0.1 (after 0 to 1 normalization) for all tested models on multiple types of target pixels. 56
3.3 Table depicting Spearman's correlation between saliency maps targeted at randomly chosen pixels and edge pixels and pixels falling on the image edge for all tested models. 58