Neural Architecture Search & Explainable Deep Learning for Multispectral Imaging

Andrew Lee
Samuel Garske

June 27, 2021

Abstract

Saliency explanations have become extremely popular for evaluating and visualising the predictions of black-box deep learning models. However, research into saliency methods has been primarily centred around 3-channel RGB images and convolutional neural networks designed for state-of-the-art performance on the challenging 1000-class ImageNet dataset. Accordingly, the application of saliency methods lacks nuance when applied directly to multispectral satellite images with the features captured by additional channels remaining largely ignored.

This paper proposes a novel approach of first using neural architecture search to obtain a network that is both effective and efficient for multispectral imaging using the EuroSAT dataset. The common Vanilla Gradient, Guided Backpropagation, Integrated Gradients and DeepLIFT saliency methods were then implemented to visualise the feature differences between multispectral and RGB data. Integrated Gradients^2 was also used to improve salient feature accentuation for low spatial resolution satellite imagery.

This work established a deep learning CNN architecture that is over 13 times more efficient in floating point operations per second (FLOPs) and uses over 4 times fewer parameters compared to the most conservative EfficientNet baseline model while matching the performance in terms of accuracy. The compounding scaling approach of the EfficientNet is applied to obtain a similar family of models, coined as EfficientNet-Liter, where the largest models are over 17 times more FLOPs efficient and use 9 times fewer parameters. The saliency methods revealed that multispectral data provides unique features for each class in comparison to RGB data in the same model, but model architecture also impacts the salient features extracted.

Contents

1 Introduction

1.1 Motivation for Research

Computer vision has been one of the fastest growing fields within deep learning for the last decade [1, 2, 3]. The rapid increases in computational power and available imaging data have fuelled interest amongst academics, open source communities and industrial leaders [4]. In particular, well-maintained deep learning libraries and cutting edge research into new architectures and algorithms have propelled the specialty of image classification forward astronomically [5]. Image classification is considered a valuable backbone within computer vision, especially for more nuanced topics like objection detection and semantic segmentation where developments in classification architectures have been conduits for similar success in other vision-related tasks [6]. However, at the same time, breakthroughs in the methods and technology used to acquire image data has also prompted vast improvement in data quality and quantity. Advanced sensors able to collect significantly more detailed information than standard red-green-blue (RGB) cameras are also growing in popularity. One of the most prominent examples of these technologies are multispectral sensors, which collect multispectral images.

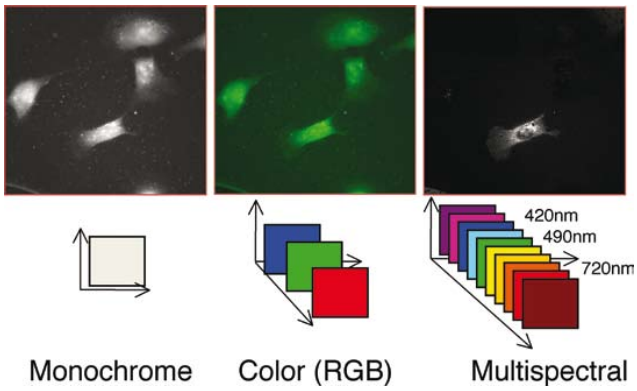

Figure 1: A comparison between monochrome (single band), RGB (three band), and multispectral (multi-band) imagery. The top row of pictures are depicting of the image type itself, while the bottom row indicates the relative band numbers and depth of information provided [7].

Multispectral images capture the reflected energy within the visible light, infra-red and thermal spectral wavelength bands [8]. While standard RGB images contain only three visible-light bands of spectral data, multispectral images generally refer to those containing four bands or more (visual representation provided in figure 1 in the appendix below). These additional bands act as additional features which can improve the separability between different image classes. This increase in information provided by multispectral image data has led to many applications in laboratory research (biochemistry, food processing, medical research, etc.), but even more prominently in remote sensing from aerial or space-based platforms (mineralogy, environmental monitoring, precision agriculture, etc.) [9]. It is an increasingly important source of data for remote-sensing applications, especially with the imminent rise of the space industry and the use of satellite image analysis.

The increased number of bands in multispectral data offers unique opportunities for deep learning models. Intuitively, the weights of the deep learning model should take advantage of potentially beneficial features that improve the discriminative power. This can be observed empirically in previous work, which obtained greater predictive accuracies on multispectral data when compared to a mirrored RGB dataset [10]. However, the authors and the wider machine learning community typically adapt and apply a variety of state-of-the-art CNN architectures directly to multispectral classification. Despite improvements over existing benchmarks, most commonly due to improvements innate to the architecture components, these architectures are hand-crafted to perform on an entirely unrelated RGB task with little consideration for practical efficiency. More efficient models are becoming an increasingly important aspect of deep learning implementations, especially for multispectral satellite image classification due to more affordable and quickly-deployable micro-platforms such as CubeSats [11, 12]. This provides the motivation for smaller, faster and more accurate models for the multispectral satellite image classification problem, one such topic being Neural Architecture Search (NAS).

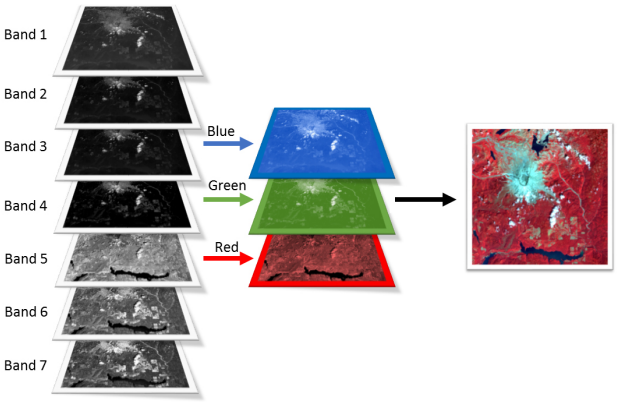

Figure 2: An example of a multispectral satellite image with multiple bands. The image bands are aggregated to provide a false-colour composite RGB image for a human-interpretable visual representation [13].

However, one of the largest limitations of deep learning architectures lies in their complexity and inherent nature to act as a "black box", providing limited or no insights into a models decision making process [14]. Despite classification accuracies surpassing previous state-of-the-art benchmarks for a variety of machine learning problems, many industries have been hesitant to adopt deep learning, preferring much less complex approaches for understanding and evaluating the robustness [15]. Being able to explain why networks make certain decisions helps one learn and understand new concepts, but also builds trust and con-

fidence in a model while ensuring a high degree of safety and ethics [15]. Additionally, obtaining insights from deep learning architectures specifically designed for multispectral data analysis could also provide perspectives not found in pretrained ImageNet architectures. Having interpretable deep learning models is critical for remote sensing applications, for which multispectral image analysis plays a major part in [16].

This work involves a study of explainable deep learning for multispectral satellite image classification. This work will consist of designing and training lightweight CNN architectures optimised for a multispectral satellite imaging dataset using modern architecture optimisation. Saliency methods will then be adapted and applied to extract high-level multispectral features captured by deep learning architectures as a means of explainability. Further analysis will be conducted to investigate the additional value attained by working with multispectral data, especially in comparison to standard RGB imaging.

1.2 Aims

The aims of this research are:

- To design and train an efficient deep learning architecture for multispectral satellite image classification using Neural Architecture Search (NAS).
- To investigate and evaluate various saliency methods for improved transparency and explainability of the network.
- To evaluate the differences in the network saliency maps between RGB and Multispectral data.

2 Related Work

2.1 Deep Learning for Multispectral Satellite Image Classification

Multispectral satellite images have been studied in computer vision for decades, particularly for various remote-sensing applications which employ even the most recent deep learning methodologies. Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs), Stacked Denoising Autoencoders (SDAE), and a semi-supervised ensemble framework called DeepSat were all applied to the SAT-4 and SAT-6 datasets [17]. The SAT-4 and SAT-6 datasets are sets of 4 band multispectral satellite images with 4 and 6 output classes, with 500,000 and 405,000 images, respectively. The DeepSat framework consisted of a feature extraction phase followed by a DBN based classifier (figure 3) and obtained a classification accuracy of 97.95% for SAT-4 and an accuracy of 93.92% for SAT-6 whilst making note of improved speed, fewer layers and neurons via the DBN. Following this, various implementations of the popular AlexNet and VGGNet network architectures were also applied, achieving classification accuracies as high as 99.98% for both multispectral datasets [18].

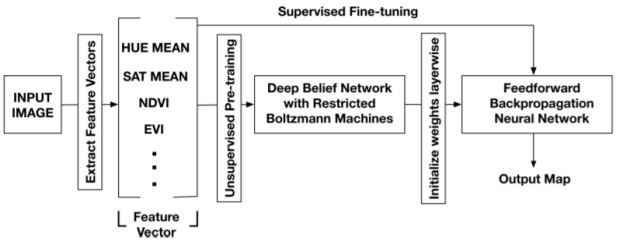

Figure 3: The DeepSAT architecture developed for classifying the SAT-4 and SAT-6 datasets [17].

As satellites containing multispectral cameras have become more prominent, the quantity and complexity of publicly available multispectral datasets have grown. In 2017, an 8-band dataset with over 1 million images and 63 categories was also made available, which is called Functional Map of the World (fMOW Christie et al. [19]). This dataset also contains images of the same location from different time periods, locations and sun positions/angles, introducing a temporal aspect for the classification and object detection task. The authors implemented various CNNs including VGG-16, ResNet-50 and DenseNet, along with recurrent neural networks like Long Short-Term Memory models (LSTMs) and were able to obtain an f1-score of 73.4%, which is a strong result considering the sheer increase in dataset complexity in comparison to SAT-4 and SAT-6. Further implementations of ensembles of NNs that learnt from the feature maps of CNNs (figure 4), combined with transfer learning and data augmentation obtained an improved f1-score of 79.7% on fMOW [20]. It was also shown that reducing the number of classes to 15 allowed deep learning models to improve their classification accuracy to as high as 95%. Other larger multispectral satellite datasets include BigEarthNet [21], with 590,326 multi-labelled images across 12 bands, and SEN12MS, a 541,986 multispectral and multi-label dataset.

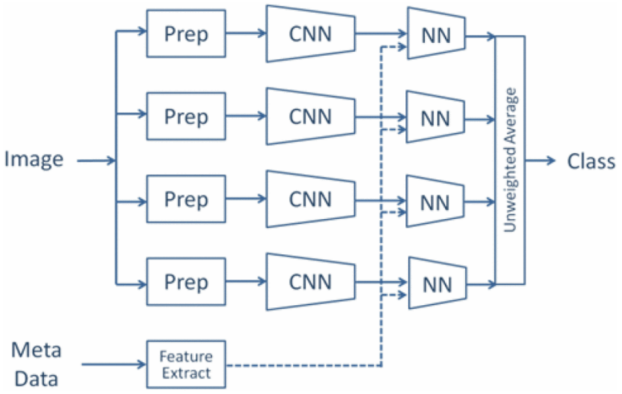

Figure 4: Ensemble model for object detection and labelling using the FMOW multispectral satellite dataset [20].

Helber et al. [22] released a 13-band, 27000 multispectral satellite image dataset with 10 classes; EuroSat. The authors applied the GoogLeNet and ResNet50 CNN architectures to the EuroSat dataset,

with the final ResNet50 model providing a classification benchmark accuracy of 98.57%. A very recent extension of this work on the EuroSat dataset applied the modern EfficientNet architectures in a semi-supervised classification approach [10]. EfficientNet refers to a family of models created by applying "compound scaling" to a strong baseline network (figure 5), scaling the width, depth and resolution of the network uniformly to achieve improved accuracy and time-efficiency [23]. This approach was extremely successful, achieving a strong performance while significantly reducing the data-requirements and processing power required for the model, whereby the authors obtained an accuracy of 95.86% while only using 50 labelled images (5 labelled images per class).

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 5: EfficientNetB0 Architecture - the baseline model for the EfficientNet architecture, which is easily scaled up accordingly using compound scaling [23].

2.2 Neural Architecture Search

The choice of architecture has always been an important consideration for deep learning, with early innovation being driven by hand-crafted architectures (e.g. AlexNet, VGGNet, Inception, ResNet) that improved upon each other with new components and features [24, 25, 26, 27, 28]. However, designing networks by hand requires a substantial amount of domain knowledge and trial and error, rendering the process inefficient, unscalable and inaccessible when confronted with the computation costs and training time required to evaluate candidate models.

Recently, the deep learning network architecture design landscape has shifted in favour of Neural Architecture Search (NAS), a procedure which automates the model design and evaluation process over a specified search space by requiring much less expertise and manual labour from the human perspective. NAS has even produced architectures which have improved upon the performance of existing hand-crafted architectures across a variety of different deep learning tasks [29, 30, 31, 32, 23, 33].

NAS is typically portrayed as having 3 essential, overarching constituents (and shown in figure 7) [34]:

1. Search Space.
2. Search Strategy.
3. Performance Evaluation.

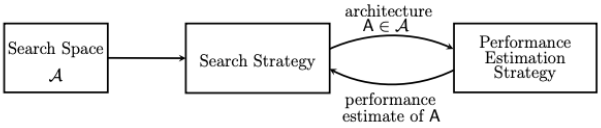

Figure 6: Neural Architecture Search (NAS) Process - Recursive operation to find optimal deep network architectureb[34].

The search space governs the different architectures that should be considered in an architecture search and introduces an element of domain knowledge and human input when specifying the necessary building blocks (i.e. the number of layers, operations of each layer, kernel size). Despite this, recent hand-crafted innovations like the skip connections of ResNets, inverted residual blocks of MobileNets and components of the Transformer have all been successfully incorporated into NAS search spaces [26, 35, 36, 32, 37, 38]. In recent works, search spaces have been used to make decisions on the macro-architecture level, choosing units known as cells or blocks which are templates reflective of repeating patterns shown to be effective in hand-designed networks e.g. (Inverted Residual block, Squeeze-and-Excitation block) [39]. With fewer choices to make, cell-based searches innately have greater efficiency compared to global search spaces which operate with individual components at the micro-architecture level.

However, Tan et al. [32] and Wu et al. [40] both choose to operate at the layer level citing the slow, complicated structures within cell structures and need for diversity between layers in their search for computational efficiency. Since the proposed research objective maintains intentions of optimising the efficiency of the network alongside its size and efficiency, it would be prudent to consider working with micro-architecture as the benefits and proven nature of cell-based approaches cannot be ignored. A beneficial compromise worth emulating involves maintaining a cell-based structure while varying parameters like the number of layers and filter sizes to factor in layer-level response. Zoph et al. [30] empirically shows that such an approach also aids a cell-based network in being transferable from CIFAR-10 to ImageNet and is accompanied by recent NAS studies like that of EfficientNetV2, which keeps the original EfficientNet architecture as a backbone and considers layer-wise choices like operations and kernel sizes [33].

Search strategies articulate the manner in which chosen search spaces are explored. Multi-trial methods involve training and evaluating architectures independently within a candidate set and the most basic multi-trial methods are standard hyperparameter optimisation techniques like grid and random searches. Other more complex multi-trial methods introduce an “oracle” to help guide the searching. Reinforcement learning is an example of this where typically the controller agent learns to maximise reward in an environment where each action sequence defines an architecture

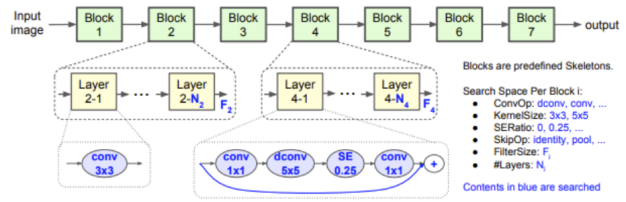

Figure 7: Factorized hierarchical NAS search space used for MnasNet - breakdown of the search space into a series of blocks that are modular and layer-wise components that can be changed and repeated [32].

with the trained architectures validation accuracy being the reward [29].

Evolutionary algorithms are also employed to choose the optimal architecture from a surviving pool of candidates after random mutation and recombination of an initially chosen architecture population [41, 42, 43]. Bayesian optimisation and progressive methods which create surrogate models that predict architecture performance have also been proposed to speed up multi-trial methods [44, 45]. While computationally inefficient, multi-trial methods are easily parallelised and have been the method of choice for many pieces of innovative deep learning literature including the works of MobileNetV3 and EfficientNet [36, 23].

On the contrary, single-trial or "one-shot" approaches which only train a single network throughout the entire search process have also been employed in NAS and are known to be much more efficient [28]. Weight sharing methods like Efficient Neural Architecture Search (ENAS) are examples of single-trial approaches, in particular ENAS constructs an over-parameterised network (supernet) covering the search space and leverages reinforcement learning to discover the optimal architecture subgraph [46]. Differentiable methods are also popular amongst single-trial approaches, Differentiable Architecture Search (DARTS) relaxes the search space to be continuous and formulates a differentiable loss function for both structural and model parameters which is then used to jointly optimise both the network weights and the architecture via gradient descent [47].

Despite search efficiencies, work done by Yu et al. [48] finds that weight sharing strategies like that of ENAS make it hard to locate the best candidate with the shuffling of the architecture ranking during searching and alongside other one-shot methods like DARTS, empirically perform no better than random search in terms of accuracy. Additionally, one-shot NAS tends to suffer from multi-model forgetting due to the overwriting of shared parameters that degrade the performance of previous models and requires special loss functions to be introduced to dampen this effect [49]. Since robustness and suitability are the priorities for the eventual multi-spectral analysis, a multi-trial method is more aligned with this objective and just like the authors of EfficientNetV2 [33], a limited search space can be imposed through the aforementioned cell-based approach with varying layer-level features to make use of the now feasible random search.

Estimating and observing the performance of an architecture is often a task and search strategy dependent concern. For example, in methods like ENAS that use reinforcement learning, the controller agent is trained with policy gradient to maximise the reward of subgraph architecture accuracy on the validation set amongst actionable samples [46]. In multi-trial methods it is common to see the early termination of training where the validation accuracy of a premature model serves as a performance estimate and an approximation of the final performance [28]. In some cases, the performance estimate is part of the strategy itself such as in progressive searches where the surrogate model used to predict architecture accuracy [44].

Going beyond accuracy, authors of FBNet Wu et al. [40] propose a hardware efficient architecture via a differentiable NAS which incorporates latency into the differentiable loss function. In another situation, the baseline EfficientNet-B0 architecture was searched by Tan and Le [23] with a multi-objective NAS which optimised both accuracy and floating point operations (FLOPs) at the same time. Altering this in EfficientNetV2, the authors add the parameter size of the model as another element in the multi-objective trade-off. Certainly, making use of performance evaluation that takes into account metrics like FLOPs alongside accuracy will greatly assist the research goal of achieving an effective and efficient architecture for multispectral imaging.

Although not particularly common, NAS has been used before in multispectral imaging problems to obtain optimal neural network architectures. Zhang et al. [50] introduces a customised, differentiable NAS known as NAS-HRIS and uses the searched network for semantic segmentation where it outperforms other preset architectures like SegNet, U-Net and Deeplab. The use of evolutionary NAS has also been proposed by NASA as means of obtaining an optimal set of autoencoders to learn the important features of multispectral satellite imagery for input reconstruction [51]. Moreover, a greedy and progressive architecture search (GPAS) has also been proposed for the task of remote sensing image scene classification, the search aims to reduce the search space of a differentiable NAS with path-level pruning [52]. Without much related research into the field, there exists an exciting opportunity to apply NAS and achieve not only an optimal architecture that is innately designed for working with images with 13 channels as opposed to RGB networks biased to ImageNet but also one that could potentially be much smaller and efficient for the task.

However, even with more efficient models there continues to be a lack of research into understanding why these deep learning models are so effective for classifying multispectral satellite imagery. There is an especially large gap in understanding the difference between a model's interpretation of multispectral data and the traditional RGB data. Hence, this provides the motivation for investigating the visual interpretation of the efficient model via explainable deep learning methods.

2.3 Explainable Deep Learning: Visualisation/Saliency Methods

Given the inherent black-box nature of deep learning models, it can be difficult to truly understand what information is being extracted and used within a deep networks decision-making process [14]. Explainable deep learning is the science of understanding the why deep learning models make certain decisions, predictions and classifications [15]. Explainable deep learning methods branch into 3 core methods:

- Visualisation - methods that visualise the characteristics that the model extracts relative to the input.
- Model Distillation - the implementation of an additional, already explainable model that mimics the decision-making and the input-output of the deep neural network.
- Intrinsic - models that have in-built explainability. An example is attention mechanisms which produce weights for each input, highlighting the relative contribution of each input throughout the network to the models final decision.

Given that the aim is to investigate the differences in the characteristic behaviour of a model when classifying multispectral and RGB satellite data, the primary focus is on visualisation methods to identify relevant features in the image scene. These methods are also commonly referred to as saliency methods, due to their ability to identify the most salient (or relevant) features that the networks see.

Since the invention of CNNs and their applications to Computer Vision problems, visualisation/saliency methods have been the de-facto tool of choice for the explainability of deep learning based imaging models [53]. These methods have been used extensively for RGB images, and are designed to reveal the relationship between the inputs and outputs of a model. This is relevant in imaging where important features can be extracted and visualised on the original image (such as key objects or pixels) [54].

There are a multitude of visualisation methods, which can generally be categorised into two categories; backpropagation and perturbation [15]. Backpropagation methods involve the comparison of the backwards passed gradients of the model to the sample/image inputs, while perturbation methods involve removing inputs/pixels and visualising the change in the model prediction (highlighting input feature importance). This research will focus on methods from the former. While perturbations are effective for network visualisation, they generally have a higher associated time and computational cost [15].

One of the first backpropagation saliency methods used for visualisation within CNNs is the vanilla gradient [54]. This approach only keeps the gradient of the loss function for a specified output class (setting the other classes' gradients to zero) and propagating the gradient back to the input pixels. The magnitude of the gradient of the loss with respect to the input can be directly interpreted as a measure for the most sensitive pixels behind the prediction of the targeted class, the larger the magnitude the more relevant the pixel. These gradients can be applied to the image (e.g. boolean mask, direct multiplication) to create "saliency maps", which illuminate the visually salient areas of the image for the trained model [55]. This can be seen in figure 8, which highlights the outlines of the various objects identified by a CNN when classifying the image.

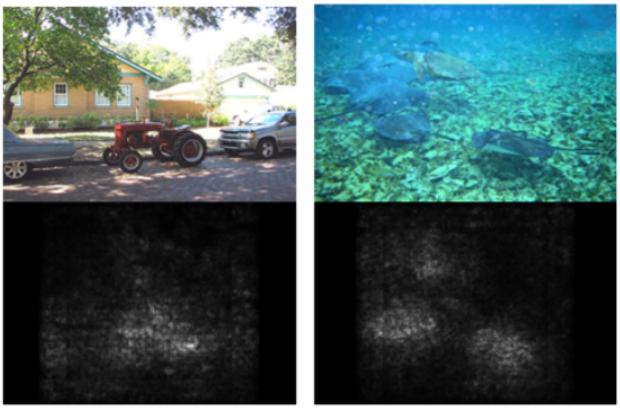

Figure 8: Vanilla Gradient saliency maps of a tractor (left) and sting rays (right) [54].

Other backpropagation saliency methods adopt a similar approach of passing gradient values back to the input with slight nuances. DeconvNets have an almost identical setup to the vanilla gradient methods with the main difference being in the application of and indicator function to the upstream gradients in the backpropagation phase [56]. Guided backpropagation combines both ideas, applying both the traditional ReLU indicator along with the DeconvNet indicator in the backpropagation step [57]. Research has also been undertaken to optimise input to maximise the layer activations resulting in the visualisation of hierarchical representations [58].

More methods have emerged such as Class Activation Maps (CAM) [59], Gradient Class Activation Maps (GradCAM) and Guided Gradient Class Activation Maps (Guided GradCAM) [60]. CAM uses Global Average Pooling (GAP) to calculate the importance of each feature as represented in the final convolution layer for a given class, and GradCAM generalises this approach by using the activation gradients to generate saliency maps for any layer. Guided GradCAM combines Guided Backpropagation and GradCAM to present even more refined saliency maps that further emphasise the most important features to the model while dampening the prominence of the less important ones. Examples of GradCAM and Guided GradCAM are shown in figure 9. Other more recent methods commonly used for RGB saliency include DeepLIFT [61] and Integrated Gradients [62].

CAM and GradCAM have been used to help evaluate a model's feature detection when detecting solar power plants using multispectral satellite imagery from the Landsat 8 dataset [63]. The authors compared

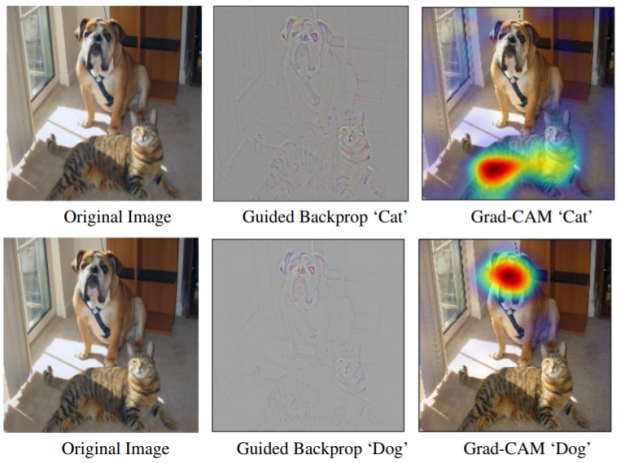

Figure 9: Heatmaps comparing GradCAM and Guided GradCAM [60]

.

different combinations of models and CAM/GradCAM methods, finding that both methods were effective in highlighting the proposed models' self-identified multispectral features. A detailed review of multiple backpropagation-based saliency methods for multispectral satellite images was conducted, which applied a 121-layer DenseNet model to the 19 channel SEN12MS dataset [64]. This work applied vanilla gradient, integrated gradient, guided-backpropagation, GradCAM, guided GradCAM and DeepLIFT (as well as the Lime and Occlusion perturbation methods). Overall, it was concluded that GradCAM was the most interpretable, scalable, reliable and computationally efficient saliency method for analysing the models interpretation of the multispectral satellite data.

The recent application of EfficientNet to the Multispectral EuroSat dataset applied guided backpropagation to the RGB imagery to generate saliency maps [10]. This work showed a comparison between the guided backpropagation saliency maps of the model after using 50 training samples versus 3000 samples, which showed that their models identify more interpretable features as the training samples increased. However, there is no investigation into the differences between RGB and multispectral imagery. Therefore, while applications of saliency methods to multispectral data are beginning to appear, it is clear that there is still a knowledge gap in the field when it comes to understanding the interpretable difference between deep learning models that use RGB or Multispectral data, especially when it comes to satellite imagery.

3 Method

3.1 Dataset: EuroSAT

All experiments will be performed on the EuroSat dataset [65, 22]. EuroSat comprises of 27,000 multispectral satellite images containing 13 bands/channels and 10 classes (.tiff format), with a corresponding dataset of identical standard RGB/JPEG images for a comparative study. The details of each band for the EuroSAT multispectral data is shown in table 1 below.

A sample of the RGB images for each class can be seen below in figure 10. The data has already been geometrically corrected, so no additional formatting was applied.

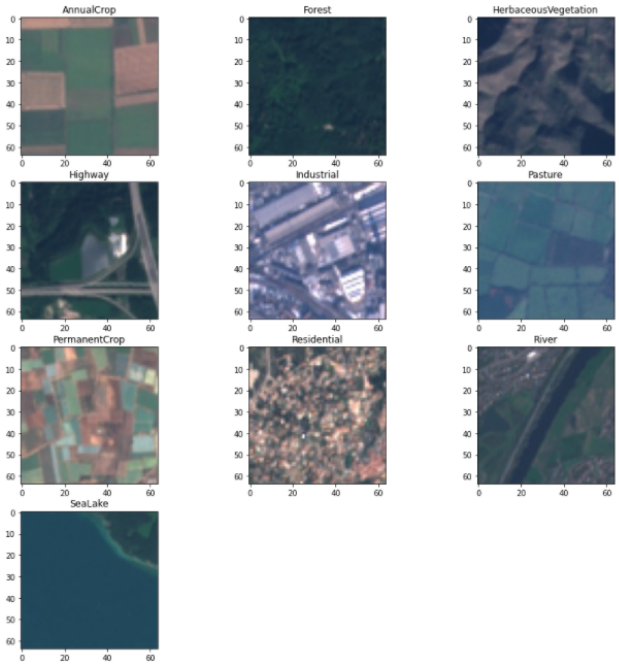

Figure 10: Sample images for each of the 10 classes in the EuroSAT dataset.

3.2 Architecture Optimisation: Neural Architecture Search

Earlier the benefits of using NAS was discussed along with the particular searching methodologies that were best suited for a mulitspectral classification problem. The reasoning for considering NAS remained grounded in the fact that the fixed state-of-the-art architectures featured in related work tended to be hand-crafted or searched for on an entirely unrelated imaging dataset with a differing number of channels (usually RGB ImageNet) and divergent distribution of images. It was speculated that the additional channels introduced by multispectral imaging would serve as additional discriminative features which could allow the network to be more efficient and just as accurate. These objectives consciously shaped the decisions undertaken in the pursuit of architecture optimisation.

3.2.1 Search Space

A well-designed search space is crucial for both the outcome and efficiency of the search. The search space limits the possible architectures that can be explored, but whilst a larger search space considers a greater number of candidates and could be argued as being more robust, the time complexity of the search increases exponentially with each additional option considered. Previously, trade-offs were discussed between performing

Band/Channel	Central Wavelength (nm)	Bandwidth (nm)	Spatial Resolution (m)
Band 1 – Coastal aerosol	443	21	60
Band 2 – Blue	490	66	10
Band 3 – Green	560	36	10
Band 4 – Red	665	31	10
Band 5 – Vegetation red edge	705	15	20
Band 6 – Vegetation red edge	740	15	20
Band 7 – Vegetation red edge	783	20	20
Band 8 – NIR	842	106	10
Band 8A – Narrow NIR	865	21	20
Band 9 – Water vapour	945	20	60
Band 10 – SWIR – Cirrus	1375	31	60
Band 11 – SWIR	1610	91	20
Band 12 – SWIR	2190	175	20

Table 1: EuroSAT band specifications derived directly from Sentinal 2A [22, 66, 67].

a costly global-search of individual micro-architecture components and the more preferred and efficient cell-based approach of only searching for blocks mimicking the motifs found in hand-crafted design. Whilst the cell-based approach was deemed essential for efficiency reasons, especially with the limited time and computational budget, related work emphasises the necessity of considering layer-wise structures and their diversity when designing architectures aimed at optimising for efficiency [40, 32].

As the proposed search objectives articulated a desire to obtain faster and smaller networks, layer-wise considerations were incorporated into the search space. Inspired by the factorised hierarchical search space introduced by Tan et al. [32] for MnasNet and used in NAS of EfficientNet [23] and EfficientNetV2 [33], a similar approach of retaining block-wise structure and searching for layer-wise operations of these blocks was to be adopted.

To concretise the proposed NAS, inspiration was drawn from the layout of MnasNet [32] in featuring a "sub search space" for each block. However, the components (e.g. operations, kernel size) were tailored along with their choices. The EfficientNet [23] uses the mobile inverted residual bottleneck (MBConv) inspired by MobileNetV2 [35] with a squeeze-and-excitation optimisation [68] and EfficientNetV2 [33] adds the Fused-MBConv proposed by Gupta and Akin [69] to the list of block choices. In another rendition, the squeeze-and-excitation optimisation were removed from the MBConv blocks of the proposed search space in an attempt to emulate a more hardware efficient variant known as EfficientNet-Lite [70]. Fused-MBConv was ignored as it added more FLOPs and parameters to the network.

Where Tan and Le [33] in EfficientNetV2 does not see a need to search for output channel sizes, this was a critical consideration earlier in effectively adapting to the extra input channels of multispectral input. The convention established in MnasNet of gradually increasing output channels across blocks to decrease the input resolutions was also followed [32]. For other block considerations including the number of layers, kernel sizes, expansion ratios and strides, the specified search spaces outlined across the MnasNet, EfficientNet and EfficientNetV2 papers were used as guides.

Overall, for the sub search space of each block was choosen from the following:

- Convolution Operations: {MBConv}
- Output Channel Size: $\{16, 24, 40, 64, 80, 112, 160, 192, 256, 320\}$
- Kernel Size: $\{ 3 \times 3 , 5 \times 5 \}$
- Stride: $\{ 1, 2 \}$
- Expansion Ratio: { 1, 4, 6 }
- Number of Layers: { 1, 2, 3, 4 }

With the choices defined, calculations can be performed to evaluate the size of the proposed search space after setting the number of blocks to 6, which is the choice observed in all variants of the EfficientNet models. There are a total of 10 output channel sizes that are in consideration and since the MnasNet constraint of employing increasing output channels has been applied, there are a total of $\binom{10}{6} = 210$ combinations for the output channels of the 6 blocks. Now ignoring the output channel size, each block has a total of $2 \times 2 \times 3 \times 4 = 48$ combinations of the other parameters. If 6 combinations are chosen from these 48 configurations and paired with the 210 output channels then the total search space size is $210 \times \binom{48}{6} = 2.5 \times 10^9$. The search space is several orders smaller than the 10^{13} reported for both MnasNet and EfficientNet and likely to be smaller than the $\binom{24 \times |L|}{6}$ of EfficientNetV2 where $|L|$ is the cardinality of the choice of layers which is omitted from the paper but appears to contain values at least up to 15 which is the number of layers in the last block of their searched EfficientNetV2-S architecture. For $|L| \geq 5$, the total search space size of EfficientNetV2 will be greater than the proposed search space.

3.2.2 Search Strategy

Choosing a search strategy to explore the determined search space is another significant decision in the NAS,

the strategy needs to align with outlined objectives and be feasible to compute with constrained resources. Earlier, this paper broke down and compared notable multi-trial to one-shot methods and established the computational efficiencies of one-shot methods. However, it was also learnt that one-shot methods are plagued by multi-model forgetting due to overwriting of parameters and have difficulty selecting the optimal candidates, making it less ideal for the focus on locating a robust structure that works effectively for imaging data with additional channels. Amongst the remaining multi-trial methods, reinforcement learning with an RNN controller is used in both MnasNet[32] and EfficientNet [23] whilst EfficientNetV2 [33] argues that the search space is sufficiently small and advocates for a random search of 1000 models with early terminating training of 10 epochs. It was shown mathematically that the proposed search space was most likely smaller than that of EfficientNetV2 and explored on a dataset that is many orders of magnitudes smaller than ImageNet in size. With more favourable conditions than EfficientNetV2 and random search being much easier to implement and parallelise, it was decided that this project would also make use of a random search.

Similarly, a random search was also performed on approximately 1000 architectures but unlike EfficientNetV2, which did not search for output channels at all, this search valued finding an optimal arrangements of output channels for the blocks, so random selections were balanced uniformly among the 210 combinations of output channels. Additionally, EfficientNetV2 also applies a constraint in selecting only architectures to explore that are of comparable size to the EfficientNet-B4, one of the larger sized models from the EfficientNet paper. However, this research did not share the same objective of competing with large state-of-the-art models so the constraint was altered to select models that are smaller or of a similar size to the EfficientNet-Lite0 model, the most conservative and lightweight variant.

Specifically, the following was done:

1. Compute the $210 \times \binom{48}{6}$ block combinations
2. Separate the block combinations into 210 groups based on their output channel combinations
3. Within each group sample 5 architectures with both less than or equal to $500M$ FLOPs and less than or equal to $5.5M$ parameters (for reference EfficientNet-Lite0 has $480M$ FLOPs and $4.6M$ parameters)
4. Train each model for 10 epochs and terminate
5. Evaluate performance

3.2.3 Performance Evaluation

The choice of performance evaluation is dependent on the task and the search strategy used. This research valued the efficiency of the model and this sentiment is echoed by the EfficientNet authors [23, 33] who optimise their NAS for both FLOPs and parameter efficiency. They do this through a customised weighted product between model accuracy and FLOPs for the optimisation goal with chosen hyperparameters to control the trade-off between accuracy and FLOPs. Since a similar goal was shared, it was deemed appropriate to leverage their expertise in the construction of such a performance metric and adapt it to the problem. The problem can be written out explicitly as an optimisation problem for each model m:

$$\begin{aligned} \underset{m}{\text{maximise}} \quad & ACC(m) \times \left[\frac{FLOPs(m)}{T}\right]^w \\ \text{subject to} \quad & FLOPs(m) \leq 5.5M \end{aligned}$$

Here, T refers to the target FLOPs of their baseline model and w the hyperparameter chosen to control the trade-off between FLOPs and accuracy. In the EfficientNet paper, the target FLOPs T is provided as $400M$ which is a reasonable value to use in the optimisation seeing as it is lower than even the smallest value found in their most lightweight EfficientNet-Lite0. The value of w provided in the paper is -0.07, this is a value taken directly from the MnasNet paper [32] which had originally optimised w for an optimisation equation which combined accuracy with latency. Seeing as the units are completely different between FLOPs and latency added onto the fact that FLOPs is hardware agnostic whilst latency is not, the authors must not have seen a need to recompute a new w for their equation. Furthermore, the authors of EfficientNetV2 also use $w = -0.07$ in their studies despite now optimising for parameter size, so the intuitive reasoning behind the decisions to keep w the same must lie in the fact that w still provides effective balancing of trade-offs in all 3 settings. Seeing as the choice of setting that is identical to that of EfficientNet, the predetermined value of $w = -0.07$ was also used.

3.2.4 Compound Scaling

Whilst a small, efficient model with good accuracy may have been achieved with the previous methods, scaling up the model up to see if any improvements in accuracy are worth the increase in size should be explored. The authors of EfficientNet, Tan and Le [23], propose a compound scaling technique that balances width, depth and height during the scaling of ConvNet and achieves greater accuracy and efficiency relative to comparable models of similar size. To achieve this, the authors introduce a compound scaling coefficient ϕ and apply a depth multiplier (α^ϕ), width multiplier (β^ϕ) and resolution multiplier (γ^ϕ) to the network to scale it up. The coefficient ϕ controls the change in resources whilst the constants, α, β, γ, control the distribution of the new amount of resources to depth, width and resolution respectively. To scale, ϕ is similarly set to 1 and a grid search is performed over the following ranges:

- α: from 1.0 to 2.0 with increments of 0.05
- β: from 1.0 to 1.5 with increments of 0.05
- γ: from 1.0 to 1.5 with increments of 0.05

The grid search is constrained on searching for values in which $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, according to the authors this filtering ensures that FLOPs will only increase approximately by 2^ϕ. The different ranges of β and γ compared to α have been chosen with quadratic relationships in the constraint in mind. The tolerance is not specified by the authors so a sensible value of $\epsilon = 0.01$ was chosen. Each valid combination is then used to scale the architecture which is trained and then ranked by best validation accuracy. Using the searched values for α, β, γ, the values of ϕ can now set. Although these values are not explicitly provided in the paper or accompanying codebase they can be inferred from models implemented in the code to be $\phi = 0, 0.5, 1.0, 1.5, 2.0$ for the family of EfficientNet-Lite models [70]. These values were then used to scale the the baseline model.

3.3 Model Training

With the scaled networks, the EuroSat multispectral dataset can be used to train the and evaluate their performance. From the original dataset of size 27000, a holdout 20% of testing data (5400) was chosen and from the remaining 21600 samples a training set of 17280 samples and validation set of 4320 was created. The partitioning of the testing data is done via a 5-fold stratified cross-validation so that repeated trials could be performed for reliability. For each fold, the training and validation sets are fixed with a single stratified split so that each candidate model trains and validates on the same data.

Architectures were trained for a total of 300 epochs using the following training configurations:

- Batch Size: 64
- Optimiser: Adam
- Initial learning rate: 0.001
- Dropout: 0.2
- Loss function: Cross Entropy Loss

To keep things consistent, no form of multispectral image processing was allowed. Any transfer learning by pretraining on an external dataset first or via porting of weights was also prohibited, everything was trained from scratch. The weights for the architecture are chosen from its best performing epoch on the validation data and then its performance on the testing dataset is used to evaluate the performance for the architecture.

3.4 Saliency Implementations

The open-source Python library *Captum* is used for all visualisation method implementations [71]. Captum is built on-top of PyTorch, which makes it directly compatible with all PyTorch models. The term "attributions" refers to the saliency of a feature/pixel for each channel. The methods implemented and their descriptions are detailed below.

3.4.1 Vanilla Gradient/Saliency

The first saliency method implemented is the Vanilla Gradient. Vanilla gradient provides a strong baseline saliency map as one of the initial saliency methods for evaluating the model's self-identified features of importance. As stated earlier, the vanilla gradient calculates the simple gradient of the output with respect to the input [54]. For any input image ($\mathbf{x}_i$), the objective is to calculate a score that indicates each pixels' relative importance ($\mathbf{S}_c$) for a target class prediction (c). For deep CNNs, the score function for a given image can be closely approximated as a linear representation (equation 1

$$\boldsymbol{S_c}(\boldsymbol{x_i}) \approx \boldsymbol{w}^T x_i + \boldsymbol{b} \tag{1}$$

$$\frac{\partial \boldsymbol{S_c}}{\partial \boldsymbol{x_i}} = \boldsymbol{w}^T \tag{2}$$

$$\boldsymbol{s_i^{vg}} = max_k(\frac{\partial \boldsymbol{S_c}}{\partial \boldsymbol{x_i}}) \tag{3}$$

The weights (w) are then calculated during the backwards propagation of the model, which is the derivative of the score with respect to the input image (equation 2). These weights provide the gradients and the relative levels of importance for each pixel for a given class (c) and channel (k). The maximum weight across each channel is taken as the final saliency score for each pixel ($\boldsymbol{s}_i^{vg}$), as seen in equation 3.

3.4.2 Guided Backpropagation

Guided Backpropagation is the second method used as it provides sharper saliency maps with more defined feature [57]. Similar to Vanilla Gradient, Guided Backpropagation also calculates the gradient of the output relative to the input, however it "zeroes" the irrelevant features during backpropagation. In more technical terms, any negative activations from the current layer ($\boldsymbol{f}_i^l$) and any negative gradients from the following layer ($\boldsymbol{R}_i^{l+1}$) are set to zero.

$$\boldsymbol{A}_i = \begin{cases} 1 & \text{if } \boldsymbol{f}_i^l > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$\boldsymbol{G}_i = \begin{cases} 1 & \text{if } \boldsymbol{R}_i^{l+1} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\boldsymbol{R}_i^l = \boldsymbol{A}_i \cdot \boldsymbol{G}_i \cdot \boldsymbol{R}_i^{l+1} \tag{6}$$

$$\boldsymbol{s}_i^{gbp} = max_k(\frac{\partial \boldsymbol{R_N}}{\partial \boldsymbol{x_i}}) \tag{7}$$

$\boldsymbol{A}_i$ and $\boldsymbol{G}_i$ are the binary activations and gradient masks (respectively) that indicate if the gradient for the current pixel is kept or zeroed. i represents the specific training samples/image being visualised. $\boldsymbol{s}_i^{gbp}$ is the Guided Backpropagation saliency score matrix.

3.4.3 Integrated Gradients

The integrated gradients method requires the satisfaction of two characteristics (also known as "axioms") [62]:

- Sensitivity - that a non-negative attribution/saliency score is given to every pixel which varies between a baseline and an input with differing predictions (i.e. pixels that contribute to a change in prediction should maintain a positive attribution/saliency score).
- Implementation Invariance - That for two functionally equivalent networks (i.e. two models provide the same output for a given input), the attributions should be the same.

These two characteristics are upheld by the integrated gradients equation (equation 8). $\mathbf{x}_i$ is the input image, $\mathbf{x}'_i$ is the baseline (commonly a black image), and α is the weighting of the input-baseline difference. $\mathbf{s}_i^{ig}$ is the Integrated Gradient score matrix.

$$\mathbf{s}_i^{ig} = max_k((\mathbf{x}_i - \mathbf{x}'_i) + \int_{\alpha=0}^{1} \frac{\partial F(x'_i \cdot \alpha(\mathbf{x}_i - \mathbf{x}'_i))}{\partial \mathbf{x}_i} \partial\alpha) \tag{8}$$

3.4.4 DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) highlights the difference between the output and a reference output (Δt), in terms of the difference between the input and a reference input (Δx) [61].

$$\Delta t = t - t_0 \tag{9}$$

$$\Delta x = x - x_0 \tag{10}$$

$$\sum_{i=1}^{n} C_{\Delta x_i, \Delta t} = \Delta t \tag{11}$$

DeepLIFT then applies a series of rules to assign attributions to each pixel/feature (or "contributions" as referenced in the paper). These rules are detailed in the original paper by Shrikumar, Greenside, and Kundaje [61], and include the:

- Linear Rule - used to assign the sign of the attribution based on the input and output values.
- Rescale Rule - used to address saturation issues and threshold issues.
- RevealCancel Rule - used to prevent the loss of salient features during aggregation.

For the purpose of this research, the reference/baseline values are set to 0 (i.e. the baseline is a black image where all pixel values are set to 0). However, the authors have noted that the reference/baseline can be critical in the interpretability of DeepLIFT, and that a zeroed background is not always effective [61].

3.4.5 Integrated Gradients^2

While the above saliency methods provide effective ways of identifying important features within the scene, they are not as directly interpretable given the low spatial resolution of the images in comparison to the standard ImageNet data. The relative pixel size makes it difficult to distinguish some of the salient features within the satellite images. Therefore, a simple modification of the IntegratedGradients function is applied, Integrated Gradients^2, is suggested to focus on the more salient features while dampening the saliency of smaller and less interpretable objects (equation 12):

$$\mathbf{s}_i^{ig2} = \mathbf{s}_i^{ig^2} \tag{12}$$

3.5 Multispectral vs. RGB Models

The Vanilla Gradients and Guided Backpropagation are more simplistic and limited, and DeepLIFT requires additional analysis to find an appropriate baseline for effective saliency mapping for each class. Integrated Gradients is relatively simple while still reducing effects like saturation and information loss, and Integrated Gradients^2 focuses on only the most salient features. Hence, to compare the multispectral image saliency to the RGB imagery, the Integrated Gradients^2 method will be used to compare a series of RGB models to the best multispectral model. These models are their purposes are detailed below:

1. Best Model: Multispectral - The best model/architecture obtained via NAS from using the multispectral image data.
2. Best Model: RGB - The best model/architecture obtained via NAS from using the multispectral image data, however the input channels are reduced to 3 and the model is trained on the RGB EuroSAT data.
3. RGB: NAS - A model derived from conducting the Neural Architecture Search using the RGB data instead of the multispectral data. This will help establish the validity of the multispectral data identifying unique features.
4. RGB Benchmark - The EfficientNet-Lite0 architecture is trained on the RGB EuroSAT data to help establish validity, as it helps to determine if the multispectral features are still unique.
5. Multispectral Benchmark - The EfficientNet-Lite0 architecture is trained on the multispectral EuroSAT data to serve as a comparison to the searched multispectral model.

4 Results

4.1 System Specifications

Experiments were performed on an Intel Xeon Gold 6240R with 96 cores which had 2 NVIDIA RTX 2080Ti as the GPUs. The CUDA version installed on the system was 10.1 and the driver version was 418.181.07. The operating system of choice was Linux (Debian 10 Buster). Code was written in Python and all requirements were installed via a virtual environment of

Python 3.7. Tasks that required deep learning leveraged the PyTorch framework, we used the version 1.5.0. Torchvision and pytorch-image-models [72] provided the preset architectures and building blocks for network design. FLOPs and total number of parameters of different models was measured using the thop package [73].

4.2 Searched Architecture

In the NAS, approximately 1000 varying architectures were trained for 10 epochs and the Top-1 accuracy on the held out testing set for these early terminated models was used as a proxy for their final performance. A scatter plot of the accuracy of each candidate architecture and their FLOPs can be seen in figure 11. It is clear that certain configurations were more effective than others when it came to working with 13-channel multispectral data with accuracy scores that varied from low 50s to low 90s. Also observable in figure 11, is the general trend that increasing the FLOPs of the model results in a higher range of accuracies. The distribution of FLOPs was however more centered towards the lower 0-200M FLOPs as a opposed to the higher 300-500M FLOPs.

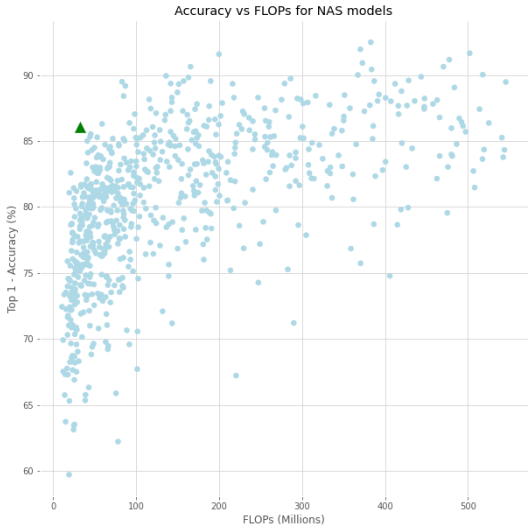

Figure 11: Accuracy vs FLOPs comparison for the searched NAS models. The green triangle represents the chosen model. We can see that a variety of architectures across a wide range of FLOPs and accuracies were considered in the NAS

Candidate architectures were ranked after using the performance evaluation detailed in 3.2.3, where a customised weighted product was used to control the trade-off between model accuracy and FLOPs. The best performing candidate was chosen under the evaluation to be the baseline model moving forward. The performance and FLOPs relative to peers is highlighted with green triangle in figure 11. Based off the position of the green triangle in the top-left corner, it is clear that the evaluation has effectively attempted to maximise both accuracy whilst also minimising FLOPs. It can also be inferred that the metric mostly likely is slightly more biased towards minimising FLOPs than optimising accuracy in choosing the current candidate over ones which have accuracies closer to 90% but more than double the FLOPs ($\approx$ 100M).

The chosen candidate in shown in table 2. It was predetermined in the search that all blocks would be mobile inverted residual bottleneck to emulate EfficientNet-Lite [70]. It can see that the architecture did not choose to repeat many of its layers within each block with only the last inverted residual block having 4 layers. Also observed are some anticipated general trends such as the decreasing resolution and increasing number of channels. It also appears that the earlier blocks tended to use a 3x3 kernel whilst the later blocks preferred the 5x5 kernel. The choice of stride and expansion ratios were all varied throughout.

Stage	Operator	Resolution	Channels	Layers	Stride
1	Conv3x3	64×64	24	1	2
2	MBConv1, k3x3	32×32	24	1	1
3	MBConv6, k3x3	16×16	40	1	2
4	MBConv1, k5x5	16×16	80	1	1
5	MBConv4, k5x5	8×8	112	1	2
6	MBConv6, k5x5	4×4	160	1	2
7	MBConv1, k5x5	2×2	192	4	1
8	Conv1x1 & Pooling & FC	2×2	1280	1	1

Table 2: The chosen architecture from the Neural Architecture Search. The convolution stem and end were included in the table for completeness despite not being part of the search.

4.3 Compound Scaling

With the model obtained from the NAS the EfficientNet [23] compound scaling procedure outlined in 4.3 was applied. The constrained grid search with compound scaling coefficient $\phi = 1$ returned the following optimal hyperparameters seen in table 3. As ϕ is increased to increase the amount of resources available to the network, the constants determine the allocation of these resources in the form of multipliers (α^ϕ, β^ϕ and γ^ϕ). It can be seen from the results that the compound scaling approach scales resolution the most, followed by depth and then width. Although the chosen set of constant values $(1.10, 1, 15, 1.20)$ was identical to that of EfficientNet, their values choose to scale depth the most, followed by resolution and then width.

Hyperparameter	Value
Depth constant (α)	1.15
Width constant (β)	1.10
Resolution constant (γ)	1.20

Table 3: The optimal hyperparameters from the compound scaling grid search. These values are similar to the values found by the EfficientNet paper [23] but are not exactly the same.

Next the baseline network is scaled with the values

$\phi = 0, 0.5, 1.0, 1.5, 2.0$ matching that of the smaller EfficientNet-Lite [70] family. For the ease of comparison, this scaled up family of the searched baseline will known as EfficientNet-Liter from now on. As expected, from table 4, as ϕ increases so too do the FLOPs and the number of parameters in the model increases. The largest model in EfficientNet-Liter with $\phi = 2.0$ has over 5 times the number of FLOPs and over twice the number of parameters compared to the baseline model ($\phi = 0$).

Model Name	FLOPs (M)	Parameters (M)	α^ϕ	β^ϕ	Input Resolution
EfficientNet-Liter0	32.52	0.85	1.0	1.0	64x64
EfficientNet-Liter1	77.11	1.46	1.1	1.1	70x70
EfficientNet-Liter2	101.14	1.57	1.2	1.1	77x77
EfficientNet-Liter3	140.94	1.77	1.3	1.2	84x84
EfficientNet-Liter4	170.29	1.92	1.4	1.3	92x92

Table 4: Summary table of FLOPs, number of parameters, multipliers and input resolution for different values of ϕ. As ϕ increases, so too does the number of FLOPs and parameters

The size and efficiency of EfficientNet-Liter can be compared to other families of models, in particular, ones that also use compound scaling, EfficientNet and EfficientNet-Lite. However, these models have been constructed for ImageNet classification which works with RGB images and conducts multi-class classification on 1000 classes. In order to establish a fairer comparison, the input channels of the architectures is changed to 13 and the classifier head is reset to one that predicts for 10 classes. This type of simplistic adaptation is the typical approach used by other works that perform multispectral classification so comparisons between these modified networks and the searched and scaled models are indeed meaningful.

When examining the FLOPs comparison between models in table 5 and figure 12, it can immediately be seen that our models are markedly more efficient in operation usage. The baseline model is over 13 times better in FLOPs than both the EfficientNet-Lite and vanilla EfficientNet baselines. This trend continues through the family of models, although it is acknowledged that only the set of values are matching but the allocation between each constant and value are not the same, with the models with $\phi = 2.0$ being over 15 times more efficient than EfficientNet-Lite and 17 times for EfficientNet.

Model Family Index	EfficientNet-Liter	EfficientNet-Lite	EfficientNet
0	32.52	433.57	434.21
1	77.11	660.75	748.83
2	101.14	933.02	1068.59
3	140.94	1476.56	1673.84
4	170.30	2674.36	3020.45

Table 5: Table of FLOPs (Millions) comparison across other compound scaling models. As ϕ increases, the ratio of FLOPs between EfficientNet vs EfficientNet-Liter and EfficientNet-Lite vs EfficientNet-Liter continue grow.

A similar story can be observed in the parameters comparison, each EfficientNet-Liter model uses far fewer total parameters than their direct competitors

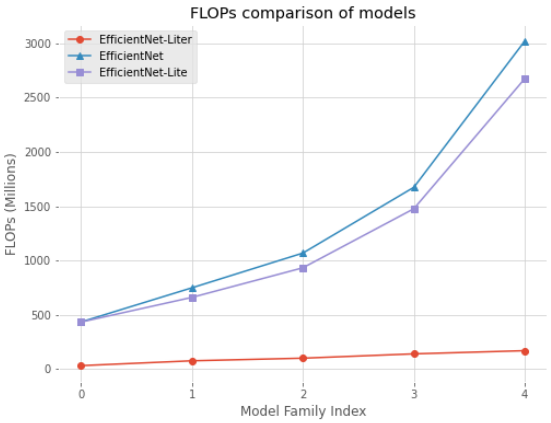

Figure 12: Graph illustrating the FLOPs comparison between EfficientNet-Liter and the EfficientNet and EfficientNet-Lite families. The increasing FLOP discounts offered by EfficientNet-Liter are evident here

from EfficientNet and EfficientNet-Lite. In figure 13 and table 6, it can be seen that the baseline model uses over 3 times fewer parameters than EfficientNet-Lite and 4 times fewer than vanilla EfficientNet. This becomes more pronounced as when considering the larger model variants. When $\phi = 2$ with the EfficientNet-Liter model is 6 times more efficient in parameter usage compared to EfficientNet-Lite and 9 times for EfficientNet.

Model Family Index	EfficientNet-Liter	EfficientNet-Lite	EfficientNet
0	0.85	3.37	4.01
1	1.46	4.14	6.52
2	1.57	4.81	7.70
3	1.77	6.92	10.70
4	1.92	11.73	17.55

Table 6: Table of parameter (Millions) comparison across other compound scaling models. It appears that the ratio of total parameter usage between EfficientNet vs EfficientNet-Liter and EfficientNet-Lite and EfficientNet-Liter also increases as larger models are compared.

4.4 Performance Evaluation

The performance of models is done on held out testing datasets. As outlined in 3.3 each testing set has a sample size of $n = 5400$ which was 20% of the original dataset and there are 5 completely distinct testing sets in total. The accuracies recorded for this section are all reported as averages over the 5 repeats. In each repeat, the training and validation sets are controlled so that each model learnt from and validate against the same data. Additionally, the training configurations (i.e. optimiser, batch size) were the same for each model. All models were trained from scratch (no transfer learning allowed) and no multispectral image processing was performed.

According to figure 14, scaled up models all performed better than the baseline but the trend upwards was not strictly increasing. The largest model ($\phi = 2$)

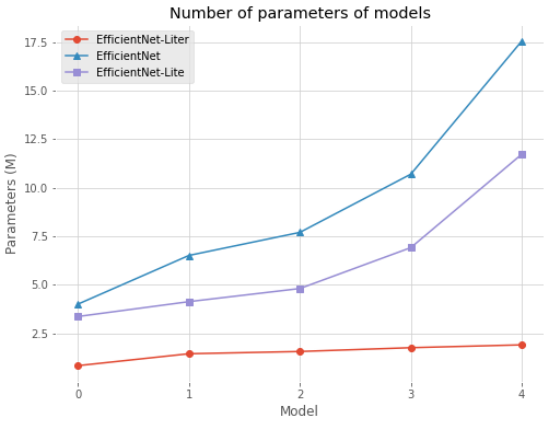

Figure 13: Graph showcasing the parameter comparison between EfficientNet-Liter and the EfficientNet and EfficientNet-Lite families. The increasing parameter usage discounts offered by EfficientNet-Liter family are illuminated in the chart.

failed to improved upon its predecessor ($\phi = 1.5$) as did ($\phi = 1$) which received a result approximately equal to ($\phi = 0.5$). From table 7 the maximum performance gain from using a larger model ($\phi = 1.5$) in the family over the baseline was around 1.4% whilst the minimum was around 0.6%.

Model	Top 1 - Accuracy (%)
EfficientNet-Liter0	94.96
EfficientNet-Liter1	95.63
EfficientNet-Liter2	95.61
EfficientNet-Liter3	96.33
EfficientNet-Liter4	96.06

Table 7: Table showing the performance of the family of models on the held out testing data. In general scaling does benefit performance with all scaled up models being more accurate than baseline.

Results can also be compared to other well known architectures adapted to the multispectral problem by changing the input channels and resetting the classification head as described in 4.3. These architectures are also trained, validated and tested on exactly the same sets of data used for the models in the project. The training configurations is also exactly the same.

From figure 15 which compares model accuracy and FLOPs, it the aforementioned superior floating operation efficiency of EfficientNet-Liter models can be seen but this time the accuracy can be gauged alongside it. In general the performance of EfficientNet-Liter models measures up to that of the adapted architectures, with the EfficientNet-Liter best model only being surpassed by the best vanilla EfficientNet model. Even the baseline EfficientNet-Liter model surpasses or is in touching distance of the performance of the other models. In fact, every scaled up model of EfficientNet-Liter performed better than all the models in the EfficientNet and ResNet families at a fraction of the FLOPs.

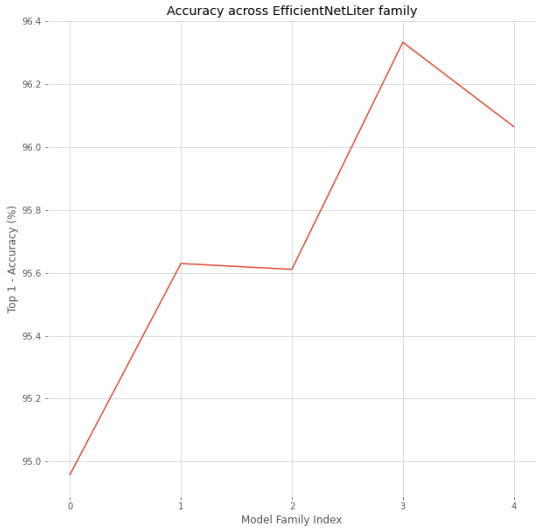

Figure 14: Graph of accuracy across the EfficientNet-Liter models. The general performance improvement due to scaling can be seen in the chart.

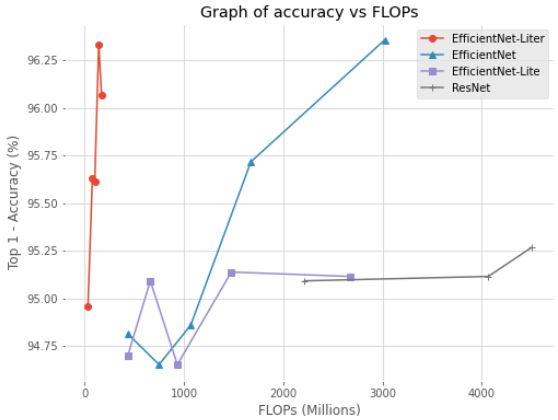

Figure 15: Accuracy vs FLOPs (millions) graph over different families of models. It is clear that the EfficientNet-Lite family is both more accurate and more efficient.

4.5 Multispectral Saliency Maps

The results of the implemented saliency methods can be seen in figure 16 below. The Vanilla Gradient and the Guided Backpropagation provide very similar results, highlighting similar features between each class. An example can be seen in the Forest class (first row), which have the same salient features in the same areas of the forest (and can be seen across all other classes as well.

The DeepLIFT and Integrated Gradients have different salient features when compared to the Vanilla Gradient and Backpropagation. DeepLIFT and Integrated Gradients seem to share some similar features, however they still differ noticeably from one-another. Using the Sea/Lakes class sample as an example, the DeepLIFT

method highlights a larger number of salient features within water-based pixels, while the Integrated Gradients method shows more of an importance on land-based features.

The suggested Integrated Gradients[2] approach as a simple method to accentuate the most salient features given the relatively lower image resolution has been effective. Compared to the original Integrated Gradients and the other saliency methods implemented, the Integrated Gradients[2] significantly reduces the relative "noise" of less salient features and focusing on the more salient ones. Using the Industrial sample, the Integrated Gradients[2] helps narrow down the salient features to a few specific buildings. The Highway samples shows the same effect, with the focus on features on the highway.

4.6 Multispectral vs RGB Saliency

As expected, the MSI EfficientNet-Liter identifies different salient features across all classes, in comparison to the RGB models (figure 17). The comparison between the multispectral and the RGB versions of EfficientNet-Liter show notiecably different features (i.e. the identical model architectures, aside from the input number of channels). This indicates that the data alone makes a notable difference the models' self-identified salient features.

An example of this can be seen when comparing the Sea/Lakes class, as the RGB version of EfficientNet-Liter completely focuses on water-based pixels, whereby the multispectral model also identifies land-based features rather than simple water pixels. Another example is the Pasture sample, where the RGB model has a much more evenly distributed saliency map in comparison to the multispectral being much more centralised. Similar observations can be made for all classes.

The RGB-NAS model highlights that the difference in saliency between the multispectral and RGB models are valid. There are many different features between the RGB-NAS, RGB-EfficientNet-Liter0 and the MSI-EfficientNet-Liter0 models, showing that the multispectral data provides different features that allow the model to classify the images. The RGB-EfficientNet-Lite0 also has different features to the other models. An overall observation is made that the multispectral data provides unique salient features to the RGB data across all models.

4.7 Multispectral NAS vs Multispectral

Whilst EfficientNet-Liter multispectral models are found via NAS, other typical multispectral applications seen in literature make use of fixed architectures by changing only the input channels. A multispectral EfficientNet-Lite0 with 13 input channels was trained for comparisons of their saliency maps. In figure 17, the saliency map for EfficientNet-Lite0 is shown on the right and like other aforementioned comparisons differs in the salient features detected to that of multispectral EfficientNet-Liter, second from the left. The most prominent difference is in the SeaLake classes where the EfficientNet-Lite0 model detects much fewer points of salience when compared to EfficientNet-Liter. On the contrary in the Highway image, EfficientNet-Lite0 detects many more salient features than EfficientNet-Liter. For the other classes, the quantity and location of the salient features detected by EfficientNet-Lite0 appear to revolve around similar locations in the scene compared to EfficientNet-Liter.

5 Discussion

The initial consideration of a random NAS (shown in 3.2.2) was centered around a smaller search space relative to Tan and Le [33], who used a random search to create EfficientNetV2 from similar MBConv building blocks. This works search was adjusted to be balanced across the 210 combinations of block output channels as it appeared more closely aligned with the objective of finding the right components, and by virtue the channel configurations, to work with the 13 channels of EuroSAT. Figure 11 shows that a wide variety of architectures with differing FLOPs and top-1 accuracies have been explored (whether by randomness or targeted sampling). This fulfils the basic expectations of an effective search, especially considering the limited computational power and timeframe.

The random NAS provided a suitable platform to work with, however it is important to note that this random search is not necessarily optimal. There are many aspects that could be easily scaled up or improved in a more comprehesive experiment including increasing the sample size (if using random search again), adding more choices for each parameter, adding other "blocks" to the search space other than MBConv (e.g. Squeeze-and-Excitation), having a different number of blocks (other than 6) and increasing the early termination of epoch. With greater resources a full grid search or other multi-trial methods such as reinforcement learning could be used to avoid random search. In particular, performing reinforcement learning with proximal policy optimization (PPO) could have been used to completely emulate the MnasNet and original EfficientNet papers [32, 23].

The performance evaluation used to choose the final candidate within the NAS is another result worth discussion. The use of a customized weighted product is leveraged to balance the trade-off between model accuracy and FLOPs. Even though this metric was likely hand-crafted to be used as a reward for the reinforcement learning with PPO method used in MnasNet [32], it can observed in figure 11 that the metric has still proven effective in providing this balance when used directly. In fact, it even prioritises floating operation efficiency over more expensive models with high performance which suits the project's use case perfectly well knowing that model scaling would occur later. However, to solidify the results future work could be done to investigate whether using the weighted product holds with a greater number of searched examples. Furthermore, investigating the use of the weighted product as a reinforcement learning PPO reward would also be of

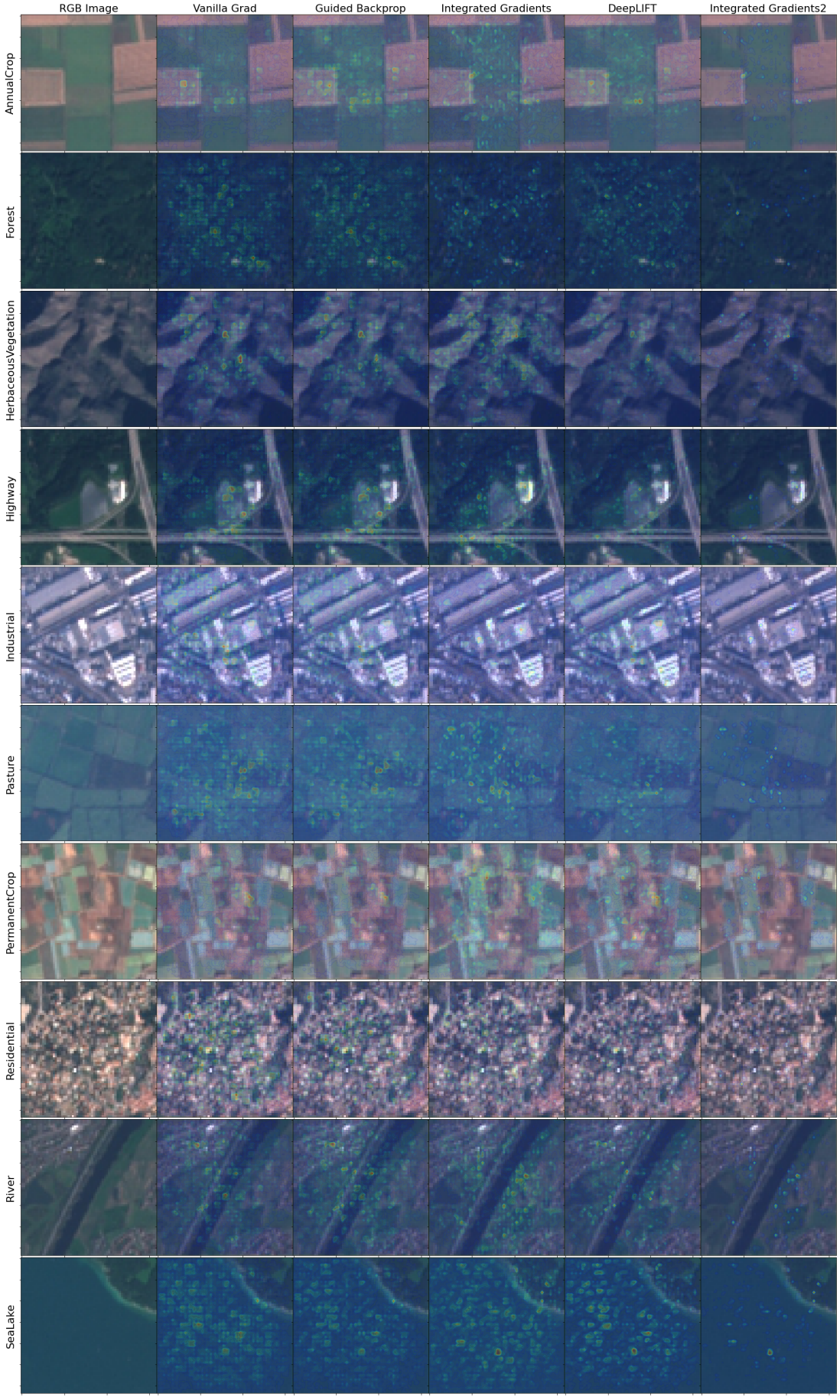

Figure 16: Saliency maps for method (horizontal) and each class (vertical) using the EfficientNet-Liter0 model, as trained on the EuroSAT multispectral data.

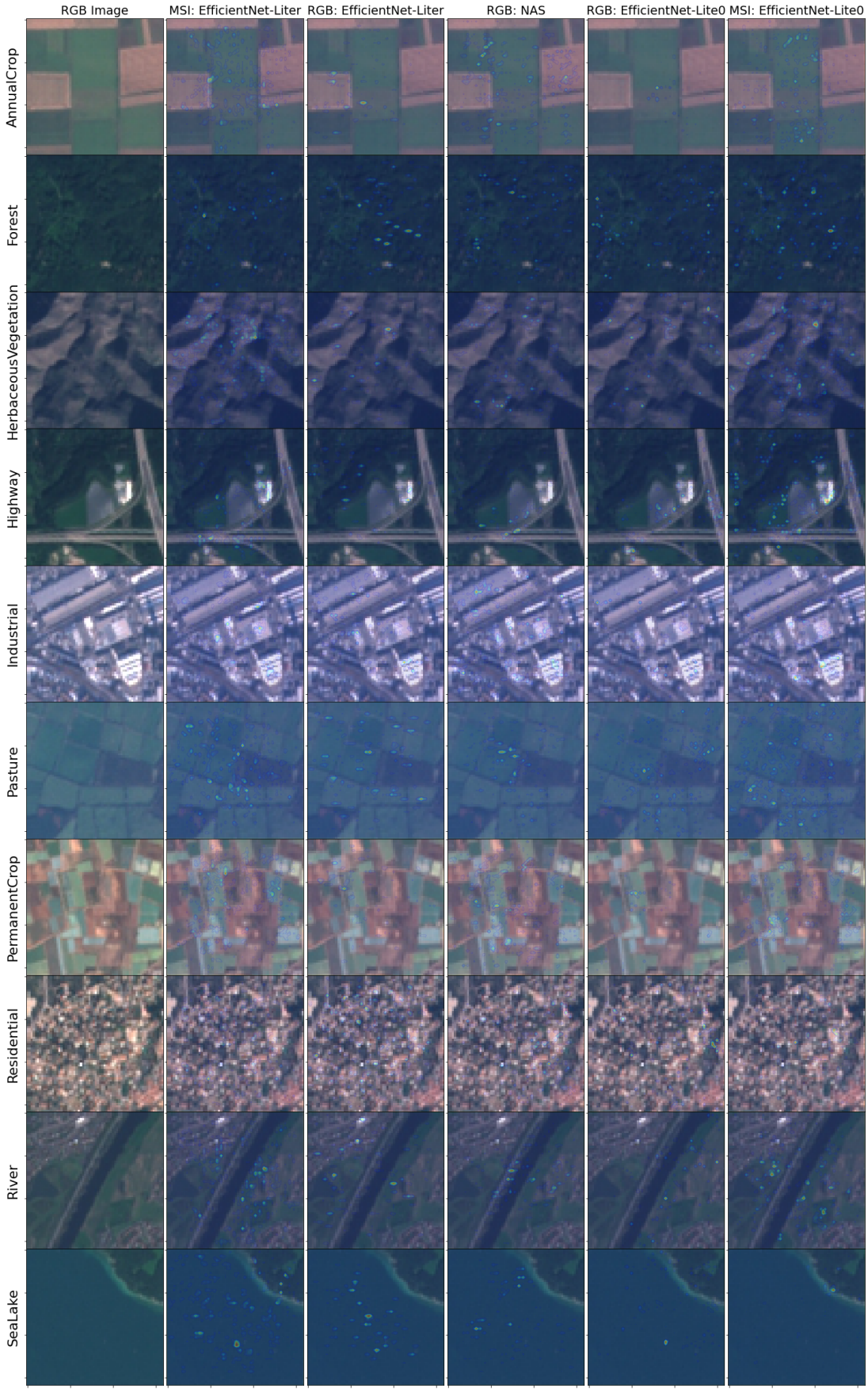

Figure 17: Saliency map comparison; raw RGB image (column 1), the best multispectral model EfficientNet-Liter0 (column 2), an adapted EfficientNet-Liter0 with only 3 channels for RGB (instead of the 13 for the multispectral version, column 3), fully NAS developed model purely for RGB (column 4), an RGB EfficientNet-Lite0 model (column 5) and an adapted MSI EfficientNet-Lite0 with 13 channels (column 6).

value to see if it eventuates in a better choice of candidate. Additionally, multiple candidates could also have been studied as well as the relationship between early termination performance and final performance.

The chosen model has been selected by the performance estimation as the candidate that best handles the compromise between accuracy and FLOPs. One of the obvious contributing factors to the efficiency is the number of repeated layers in each MBConv block as only the last block contains any repeated layers. Another contributing factor is the output channels chosen for each block, these values are on the lower end of the values available, avoiding larger choices like 256 and 320. For an MBConv block which makes use of depth-wise separable convolutions [74], if an input of size (H, W, M) is received and is transformed to an output of (H, W, N) with a kernel of size $K \times K$ where H, W are height and weight, M, N as input and output channels, then total number of multiplicative operations (T) is shown in Equation 13.

$$T = (H \times W \times M) \times (K^2 + N) \tag{13}$$

It is observed that a balance between the input volume and both the kernel size K and the number of output channels N needs to be struck to keep the number of operations low. This is exactly the relation that can observed in the chosen architecture which according to 2 starts off with large input volumes of $(64 \times 64 \times 24)$ but smaller 3×3 filters and output channels 24. When the last MBConv layer is reached there are now much smaller input volumes of $(2 \times 2 \times 192)$ which freely use large filter sizes of 5×5 and also larger output channels (192). An area of future research could be to investigate this observed relationship between the input volume and both the kernel sizes and output channels for efficient architectures, this information could prove insightful for those seeking to handcraft optimal architecture.

The compound scaling method is applied to the chosen architecture with a balanced set of height, width and resolution multipliers determined by via grid search. As expected the number of parameters and total FLOPs increases with larger values of ϕ. This can explained easily by examining each of the multipliers. The depth multiplier α^ϕ changes the number of repeating units in each block and as a result when it is increased the number of total operations and FLOPs follows. The width multiplier β^ϕ controls the number of channels at each layer, in the previous depth-wise separable convolution example, the number of input and output channels become $\beta^\phi M$ and $\beta^\phi N$ respectively giving us the total operations T^* shown in Equation 14 illustrating the quadratic relationship between FLOPs and the width multiplier β^ϕ.

$$T^* = (H \times W \times \beta^\phi M) \times (K^2 + \beta^\phi N) \tag{14}$$

The input resolution multiplier γ^ϕ is applied to the input resolution (H, W) giving us the total operations T' in Equation 15 which also forms a quadratic relationship.

$$T' = (\gamma^\phi H \times \gamma^\phi W \times M) \times (K^2 + N) \tag{15}$$

These relations between FLOPs and the multipliers served as the rationale for constraining the grid search to $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ in the EfficientNet paper [23]. Additional work could be done to whether 2 is a suitable value for the problem and whether the tolerance of the grid search should be loosened from being 0.01 within the target value of $\alpha \cdot \beta^2 \cdot \gamma^2$.

The scaled family of models achieves the performance of other commonly used architectures such as EfficientNet, EfficientNet-Lite and ResNet in terms of accuracy but with superior operational and parameter efficiency. From previous discussion, EfficientNet-Liter achieves this efficiency through its block constituents but the performance results empirically also support the architecture's representational power being on par with the other more complex and expensive choices for the multispectral task. The extra channels captured in multispectral are additional features for networks which EfficientNet-Liter inadvertently optimises for e.g. choosing optimal layer width (channels) for the right amount of feature granularity [75]. Comparing this to other models that have been optimised and trained for RGB ImageNet may explain the less efficient architectures that are more deep or wide than necessary. Whilst these results are derived from training from scratch, an investigation into whether these relationships hold when transfer learning is applied is of substantial interest however the budget did not exist to pretrain networks on ImageNet for fair comparisons to the other commonly used models which had publicly available weights. The reliability of the findings could also be improved via experimentation on other multispectral datasets, preferably harder datasets that have more classes. The application of EfficientNet-Liter as a backbone for other computer vision activities like object detection and semantic segmentation would also prove valuable for researchers looking for light weight and efficient options for time and memory sensitive tasks.

In terms of the saliency maps, the similarity between Vanilla Gradient and Guided Backpropagation is expected. This is because Guided Backpropagation is a simple modification of the Vanilla Gradient. The difference of Integrated Gradients and DeepLIFT compared to the previous methods can be explained by their increase in complexity, primarily to address issues such as saturation and salient feature loss during aggregation [62, 61]. The Integrated Gradients[2] reduces the less prominent salient features; which could still obscure information like clusters of these less salient features. Nonetheless, the results are as expected and simplify the interpretation of the saliency maps.

However, it is important to recognise that no direct evaluation between the different saliency methods has been implemented. While it is a challenging task, being able to quantitatively identify which visualisation methods are most effective for accentuating salient features establishes a more reliable study. Metrics such as Max-Sensitivity and Area Under Curve Most Relevant

First (AUC-MoRF) have been used to evaluate the reliability of saliency methods [64]. Max-Sensitivity measures the maximum change in the saliency map from small input perturbations, while AUC-MoRF evaluates how fast the contribution of the most relevant features decreases and pixels are perturbed.

The different features between the MSI EfficientNet-Liter and the RGB equivalent's saliency maps indicates that the additional channels of the multispectral image data directly impact the features extracted by the model. The different features can be explained by the different materials/objects present within each pixel, and that the multispectral data is allowing the model to identify salient features that simple RGB data cannot [8]. Taking the Sea/Lakes classification example, the salient features on the land section of the image can be explained by the model indirectly identifying recurring coastal features such as sand, coastal vegetation types and soil moisture/salinity levels inherent to land masses near large bodies of water [76, 77]. These potential domain-specific justifications can be investigated across all of the classes, which will be discussed further in the future work section.

When comparing the multispectral models it was noted that in general EfficientNet-Liter and EfficientNet-Lite0 appeared to detect some similar saliency points. For example in AnnualCrop, both maps tend to encircle the middle green patch whilst in HerbaceousVegetation points gather around the ridges in the upper middle section of the image. This is extremely advantageous for EfficientNet-Liter as it manages to uphold these features despite being several times smaller and more efficient. Furthermore, in certain instances EfficientNet-Liter detects important features that EfficientNet-Lite0 is unable to, such as in Sea/Lake where strong features in the sea are intuitively expected but noticeably absent from EfficientNet-Lite0. Further investigations into the EfficientNet-Lite0's predictions on Sea/Lake or synthetical copies of Sea/Lake like images could be investigated to see if the trend continues. In the Highway example, the EfficientNet-Lite0 detects additional features on the left side of the image that are not observed in EfficientNet-Liter and seemingly unrelated to highways. On the other hand, EfficientNet-Liter focuses mainly on the highway structure and the center of the water body which appear more relevant to highways. Further work could be done to determine whether the compact nature of EfficientNet-Lite0 has enabled it to also be more concise in its use of salient features, a notion suggested by Tan and Le [23].

The difference in salient features between all of the models (both multispectral and all RGB variants) indicates that the model can also have a notable impact on the features identified. The RGB-NAS and RGB-EfficientNet-Lite0 models' differences in salient features compared to the EfficientNet-Liter0 models can be somewhat explained by the difference in architecture extracting different features. However, there is a sensitivity of saliency methods when comparing different models due to the inherent randomness in parameter initialisation [53]. A possible future step could involve averaging the saliency maps across multiple iterations of the same model.

Future work involves much more detailed and granular applications with higher resolution multispectral satellite data. Targeting more specific applications such as the classification of different forest types or species of crops should result in much more interpretable saliency results that can be better judged by domain experts. Another path for improving the interpretability of the saliency maps is to investigate methods that use the ground sample distance and object size to account for varying image and pixel resolution. This will help visualise whether the model is identifying interpretable objects or not (rather than individual pixels that are hard to interpet). Having larger and more complex datasets with significantly more classes, like BigEarthNet and SEN12MS, will utilise the increased bands of multispectral data much more effectively for classification and provide a much more noticeable improvement over the RGB classification results.

Hyperspectral data also provides a significant opportunity for future research. Hyperspectral image data can have hundreds of bands, providing even more information than multispectral imagery [8]. Hyperspectral data is even more complex, and improving model efficiency and explainability is an even larger challenge in comparison. However, it is important to recognise that large, labelled datasets are not always easy to obtain or create. Investigating transfer learning and meta-learning for multispectral satellite image datasets, and improving overall generalisation are promising paths for multispectral satellite image classification.

6 Conclusion

This work explored the field of multispectral satellite image data using the EuroSAT dataset, and the approaches of machine learning practitioners when processing the additional channels/bands. Inspired by the recent success of the EfficientNet architecture, NAS was applied alongside compound scaling to develop a more efficient architecture for processing multispectral satellite imagery. This yielded a family of compact model architectures; EfficientNet-Liter. The EfficientNet-Liter family of models are just as accurate as existing multispectral models and several orders of magnitude more efficient for both computational costs and total number of parameters.

To better understand the types of features extracted by multispectral images for each class, Vanilla Gradient, Guided Backpropagation, Integrated Gradients, DeepLIFT, and Integrated Gradients^2 were implemented. Integrated Gradients^2 improved the visual inspection of the lower spatial resolution EuroSAT dataset by accentuating only the most salient features (in comparison to ImageNet or other common datasets, where salient features are less dispersed and much higher resolution). Finally, the multispectral saliency features were compared to the features from various RGB models. As expected, the multispectral satellite imagery provided unique insights and identified features that

are not always obtainable by using RGB imagery. More work involving more complex classes, datasets and generalisable methodologies will validate these findings, and provide better and more interpretable saliency maps.

References

[1] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". In: *Neural computation* 29.9 (2017), pp. 2352–2449.

[2] Tong He et al. "Bag of tricks for image classification with convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 558–567.

[3] Athanasios Voulodimos et al. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018).

[4] Yann LeCun. "1.1 deep learning hardware: Past, present, and future". In: *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2019, pp. 12–19.

[5] Giang Nguyen et al. "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey". In: *Artificial Intelligence Review* 52.1 (2019), pp. 77–124.

[6] Wei Wang et al. "Development of convolutional neural network and its application in image classification: a survey". In: *Optical Engineering* 58.4 (2019), p. 040901.

[7] James R Mansfield and Richard M Levenson. "Paving a new path: multispectral imaging in pathology". In: *Infocus Magazine* 14 (2009), p. 4.

[8] Kumar Navulur. *Multispectral image analysis using the object-oriented paradigm*. CRC press, 2006.

[9] José Manuel Amigo. "Hyperspectral and multispectral imaging: Setting the scene". In: *Data Handling in Science and Technology*. Vol. 32. Elsevier, 2020, pp. 3–16.

[10] Pablo Gómez and Gabriele Meoni. "MSMatch: Semi-Supervised Multispectral Scene Classification with Few Labels". In: *arXiv preprint arXiv:2103.10368* (2021).

[11] Abhas Maskey and Mengu Cho. "CubeSatNet: Ultralight Convolutional Neural Network designed for on-orbit binary image classification on a 1U CubeSat". In: *Engineering Applications of Artificial Intelligence* 96 (2020), p. 103952.

[12] Jakub Nalepa et al. "Towards On-Board Hyperspectral Satellite Image Segmentation: Understanding Robustness of Deep Learning through Simulating Acquisition Conditions". In: *Remote Sensing* 13.8 (2021), p. 1532.

[13] Sara Hanna. *Natural and False Color Composites*. 2019.

[14] Ravid Shwartz-Ziv and Naftali Tishby. "Opening the black box of deep neural networks via information". In: *arXiv preprint arXiv:1703.00810* (2017).

[15] Ning Xie et al. "Explainable deep learning: A field guide for the uninitiated". In: *arXiv preprint arXiv:2004.14545* (2020).

[16] John E Ball, Derek T Anderson, and Chee Seng Chan Sr. "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community". In: *Journal of Applied Remote Sensing* 11.4 (2017), p. 042609.

[17] Saikat Basu et al. "Deepsat: a learning framework for satellite imagery". In: *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*. 2015, pp. 1–10.

[18] M Papadomanolaki et al. "Benchmarking deep learning frameworks for the classification of very high resolution satellite multispectral data." In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 3.7 (2016).

[19] Gordon Christie et al. "Functional map of the world". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6172–6180.

[20] Mark Pritt and Gary Chern. "Satellite image classification with deep learning". In: *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE. 2017, pp. 1–7.

[21] Gencer Sumbul et al. "Bigearthnet: A large-scale benchmark archive for remote sensing image understanding". In: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2019, pp. 5901–5904.

[22] Patrick Helber et al. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019).

[23] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[25] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[26] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[28] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. “A survey on neural architecture search”. In: *arXiv preprint arXiv:1905.01392* (2019).

[29] Barret Zoph and Quoc V Le. “Neural architecture search with reinforcement learning”. In: *arXiv preprint arXiv:1611.01578* (2016).

[30] Barret Zoph et al. “Learning transferable architectures for scalable image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.

[31] Esteban Real et al. “Regularized evolution for image classifier architecture search”. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4780–4789.

[32] Mingxing Tan et al. “Mnasnet: Platform-aware neural architecture search for mobile”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[33] Mingxing Tan and Quoc V Le. “Efficientnetv2: Smaller models and faster training”. In: *arXiv preprint arXiv:2104.00298* (2021).

[34] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. “Neural architecture search: A survey.” In: *J. Mach. Learn. Res.* 20.55 (2019), pp. 1–21.

[35] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[36] Andrew Howard et al. “Searching for mobilenetv3”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1314–1324.

[37] Yukang Chen et al. “Detnas: Neural architecture search on object detection”. In: *arXiv preprint arXiv:1903.10979* 1.2 (2019), pp. 4–1.

[38] Changlin Li et al. “Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search”. In: *arXiv preprint arXiv:2103.12424* (2021).

[39] Pengzhen Ren et al. “A comprehensive survey of neural architecture search: Challenges and solutions”. In: *arXiv preprint arXiv:2006.02903* (2020).

[40] Bichen Wu et al. “Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10734–10742.

[41] Esteban Real et al. “Large-scale evolution of image classifiers”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2902–2911.

[42] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. “A genetic programming approach to designing convolutional neural network architectures”. In: *Proceedings of the genetic and evolutionary computation conference*. 2017, pp. 497–504.

[43] Hanxiao Liu et al. “Hierarchical representations for efficient architecture search”. In: *arXiv preprint arXiv:1711.00436* (2017).

[44] Chenxi Liu et al. “Progressive neural architecture search”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 19–34.

[45] Colin White, Willie Neiswanger, and Yash Savani. “Bananas: Bayesian optimization with neural architectures for neural architecture search”. In: *arXiv preprint arXiv:1910.11858* (2019).

[46] Hieu Pham et al. “Efficient neural architecture search via parameters sharing”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4095–4104.

[47] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “Darts: Differentiable architecture search”. In: *arXiv preprint arXiv:1806.09055* (2018).

[48] Kaicheng Yu et al. “Evaluating the search phase of neural architecture search”. In: *arXiv preprint arXiv:1902.08142* (2019).

[49] Yassine Benyahia et al. “Overcoming multi-model forgetting”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 594–603.

[50] Mingwei Zhang et al. “NAS-HRIS: Automatic design and architecture search of neural network for semantic segmentation in remote sensing images”. In: *Sensors* 20.18 (2020), p. 5292.

[51] Roberto Campbell et al. “Feature Learning for Multispectral Satellite Imagery Classification using Neural Architecture Search”. In: *Earth and Space Science Open Archive ESSOAr* (2019).

[52] Cheng Peng et al. “Efficient Convolutional Neural Architecture Search for Remote Sensing Image Scene Classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* (2020).

[53] Julius Adebayo et al. “Sanity checks for saliency maps”. In: *arXiv preprint arXiv:1810.03292* (2018).

[54] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[55] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. “Understanding deep networks via extremal perturbations and smooth masks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2950–2958.

[56] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[57] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[58] Dumitru Erhan et al. "Visualizing higher-layer features of a deep network". In: *University of Montreal* 1341.3 (2009), p. 1.

[59] Bolei Zhou et al. "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.

[60] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[61] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3145–3153.

[62] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328.

[63] Nevrez Imamoglu et al. "Solar Power Plant Detection on Multi-Spectral Satellite Imagery using Weakly-Supervised CNN with Feedback Features and m-PCNN Fusion". In: *arXiv preprint arXiv:1704.06410* (2017).

[64] Ioannis Kakogeorgiou and Konstantinos Karantzalos. "Evaluating Explainable Artificial Intelligence Methods for Multi-label Deep Learning Classification Tasks in Remote Sensing". In: *arXiv preprint arXiv:2104.01375* (2021).

[65] Patrick Helber et al. "Introducing EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification". In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 204–207.

[66] The European Space Agency. *Spectral bands for the SENTINEL-2 sensors (S2A S2B)*. 2015.

[67] Ferran Gascon et al. "Copernicus Sentinel-2A calibration and products validation status". In: *Remote Sensing* 9.6 (2017), p. 584.

[68] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[69] Suyog Gupta and Berkin Akin. "Accelerator-aware neural network design using automl". In: *arXiv preprint arXiv:2003.02838* (2020).

[70] R Liu. "Higher accuracy on vision models with EfficientNet-Lite". In: *TensorFlow Blog.[online] Available at: https://blog. tensorflow. org/2020/03/higher-accuracy-on-visionmodels-with-efficientnet-lite. html [Accessed 30 Apr. 2020]* (2020).

[71] Narine Kokhlikyan et al. "Captum: A unified and generic model interpretability library for pytorch". In: *arXiv preprint arXiv:2009.07896* (2020).

[72] Ross Wightman. *PyTorch Image Models*. `https://github.com/rwightman/pytorch-image-models`. 2019. DOI: 10.5281/zenodo.4414861.

[73] Ligeng Zhu. *PyTorch-OpCounter*. `https://github.com/Lyken17/pytorch-OpCounter`. 2018.

[74] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[75] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[76] GUO Yan et al. "Integrating remote sensing and proximal sensors for the detection of soil moisture and salinity variability in coastal areas". In: *Journal of Integrative Agriculture* 12.4 (2013), pp. 723–731.

[77] Leila Hassan-Esfahani et al. "Assessment of surface soil moisture using high-resolution multi-spectral imagery and artificial neural networks". In: *Remote Sensing* 7.3 (2015), pp. 2627–2646.