Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Erling Fridtjof Olweus

Deep Neural Network Architectures for Detection and Segmentation of Solar Farms in Satellite Imagery

Master's thesis in Computer Science
Supervisor: Ole Jakob Mengshoel
July 2023

NTNU
Norwegian University of
Science and Technology

Erling Fridtjof Olweus

Deep Neural Network Architectures for Detection and Segmentation of Solar Farms in Satellite Imagery

Master's thesis in Computer Science
Supervisor: Ole Jakob Mengshoel
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

In response to the urgent global call for renewable energy alternatives, as dictated by the Paris Agreement, there is an increasing need to accurately map and monitor the growth of solar farms around the world. A critical component of achieving net zero emissions by 2050 is a clear understanding and accurate tracking of photovoltaic solar energy capacities globally. However, the undocumented nature of many of these installations presents a serious obstacle.

To address this challenge, this thesis develops Solis-seg, a Deep Neural Network designed to detect and segment solar farms in satellite imagery. The Solis-seg model pushes the boundaries of current capabilities in photovoltaic detection, attaining a mean Intersection over Union (IoU) score of 96.26% on a dataset covering approximately 30,000 solar farms in Europe. As demonstrated in comparative experiments as reported and discussed in this work, this performance surpasses any previous result on a continental-spanning dataset reported in the literature.

As part of this work, we apply Neural Architecture Search (NAS) to the problem of segmenting solar farms in satellite imagery, thus unveiling significant insights and potential avenues for future exploration. In doing so, it assesses the practicality of NAS in an important sustainability context. Furthermore, this thesis offers a critical reassessment of the widely endorsed method of utilizing transfer learning from classification tasks for semantic segmentation.

Therefore, this research is a meaningful contribution to the field of satellite imagery analysis, encouraging experimentation with advanced techniques in the rapidly developing domain of machine learning for earth observation. By underscoring the escalating importance of renewable energy resources and offering an efficient and scalable solution to track global progress towards sustainable energy goals, this thesis aligns with the broader goal of facilitating the global energy transition towards sustainable sources.

Preface

This master's thesis, penned in the spring of 2023, is submitted to the Norwegian University of Science and Technology by its author, Erling Fridtjof Olweus, a candidate for a Master's degree in Computer Science.

I wish to extend my sincerest gratitude to my academic advisor, Professor Ole Jakob Mengshoel, for his invaluable assistance throughout the development of this manuscript. His involvement, which extended beyond enriching academic discussions and insightful article suggestions, encompassed constructive feedback that greatly enhanced both the research methodology and the overall quality of the thesis.

Equally, my appreciation goes to the Enernite team for providing such an exciting problem and excellent support. Particularly I have to thank Mikkel Stårvik, for his indispensable support in understanding and integrating the Solis Oculus project into my research. Parts of this thesis would have been neigh impossible if not for his pivotal contribution.

Moreover, I'd like to express my heartfelt thanks to my friends whose presence significantly enriched my time at NTNU, making it an unparalleled experience. My recognition also extends to Start NTNU, the exceptional student organization that introduced me to Enernite, revolutionizing my perspective toward career prospects as a soon-to-be graduate.

Finally, my profound appreciation goes to my family. Their relentless emotional backing, coupled with their emphasis on the importance of an academic qualification, has ingrained in me the tenacity needed to chase my aspirations. Their support has been instrumental in this journey, and this thesis stands as an embodiment of their continuous encouragement.

Erling Fridtjof Olweus
Trondheim, 8th July 2023

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Rationale Behind the Study

With the Paris agreement of 2015, a vast majority of nations globally have committed to reach net zero emissions by 2050. Achieving this monumental goal requires a large-scale transition from fossil fuels towards renewable energy alternatives like solar and wind power. Currently, fossil fuels are responsible for nearly 80% of the global energy consumption and emit over 14 gigatonnes of CO2, as reported by the International Energy Agency [1]. The imperative shift towards green energy sources such as wind, hydro, and solar is fundamental to meet the Paris agreement's climate objectives within the prescribed timeline. Non-compliance with these objectives could lead to catastrophic impacts on human civilization.

Recent geopolitical events, such as Russia's invasion of Ukraine and the subsequent shutdown of the Nord Stream II pipeline in response to economic sanctions, have underscored the significance of energy self-reliance and spurred the growth of green energy in European countries [2].

A startup called Enernite[1], established at the NTNU School of Entrepreneurship, is contributing to this shift by shortening the time required to identify ideal locations for green energy projects. They facilitate quick site evaluation by generating and utilizing global datasets on variables that determine the suitability of a location to host a power plant. One of Enernite's offerings, Solis Oculus (Solis), employs an ML model to identify existing Photo Voltaic (PV) energy facilities globally on remote sensing imagery.

Understanding the location and configuration of PV plants globally assists in:

1. Evaluating current worldwide and regional solar capacities
2. Estimating worldwide and regional investments in renewables
3. Identifying characteristics that contribute to an ideal PV facility site
4. Understanding the types of lands that are being replaced for solar energy plant constructions [3]
5. Monitoring worldwide and regional development of solar energy over time

[1]https://www.enernite.com/

However, Enernite is not the pioneer in creating such a dataset. Several research papers have published solar farm datasets for multiple countries, collected using machine learning algorithms on satellite images [3–7]. These studies will be analyzed in more detail in section 2.6. Even though these papers contribute significantly to the cause, none, to our understanding, have endeavored to employ Neural Architecture Search (refer to section 2.7) to fine-tune a neural network architecture aimed at identifying solar farms on satellite imagery.

1.2 Study Objectives

Despite the demonstrated prowess of Neural Architecture Search (NAS) in surpassing human-designed architectures in image classification competition datasets [8], its application in the field of solar farm identification on satellite imagery remains uncharted territory. Furthermore, while NAS has seen extensive use in well-established benchmarks, its practical application for novel datasets is still under-researched [9]. Recognizing these gaps, our study embarks on a multifaceted mission. Building upon previous research [5, 10], which investigates different architecture performances but does not explore NAS-derived solutions, we aim to harness NAS optimization for the real-world task of semantic segmentation of solar farms and assess its broader performance beyond established benchmarks. In the process, we critically re-evaluate the strategy employed by Enernite's current leading model, which is based on transfer learning from classifying solar farms to segmenting them. Our research, therefore, aims to make significant contributions to both the task of solar farm segmentation and the wider application of NAS.

Consequently, we formulate the following central goal for our research:

Research Goal: To advance the state-of-the-art in detecting and segmenting solar farms in satellite imagery by examining the practical utility of NAS for architecture optimization and critically reassessing the applicability of transfer learning from classification to segmentation tasks.

In alignment with this overarching goal, we define the following study objectives:

SO1 **Evaluating the effectiveness of Transfer Learning:** Determine whether using the backbone of a model trained on a classification task can deliver performance equivalent to a model entirely trained on segmentation.

SO2 **Investigating the Robustness of NAS in Satellite Image Segmentation:**

- **SO2.1** Understand the influence of various factors on NAS including the impact of dataset sizes, different data subsets, and special features of satellite imagery
- **SO2.2** Evaluate the robustness of the relative ranking produced by NAS and the risk of overfitting to a specific dataset during architecture search

SO3 **Assessing Computational Trade-offs in NAS Application:** Assess the computational cost implications of employing NAS for a practical application on a novel dataset and determine the potential benefits when contrasted with the usage of a pre-existing off-the-shelf architecture.

1.3 Research Method

This study aims to address the research objectives outlined in section 1.2 by conducting a series of Neural Architecture Searches on a comprehensive dataset encompassing all solar farms in Europe known to Enernite as of 01.03.2023. The performance of the resulting models will be benchmarked against other high-performing segmentation models and baseline models. An in-depth account of the experiments conducted and their results can be found in Chapter 4. These experiments have been strategically designed leveraging the insights gleaned from an extensive literature study pertaining to Artificial Neural Networks (ANN), Computer Vision (CV), Neural Architecture Search, and Geographic Informatic Systems (GIS), by Olweus, E [11] as well as existing literature.

1.4 Thesis Structure

The subsequent chapters of this thesis are organized as follows:

- **Chapter 2: Background Theory and Related Work** Delves into the core theories and concepts essential for understanding the research undertaken in this thesis, while also highlighting significant scholarly works in the fields that inform our study.
- **Chapter 3: Methodology** Articulates the methodologies and techniques employed in our experiments as detailed in Chapter 4, explaining the rationale behind these selected methods.
- **Chapter 4: Experiments and Results** Provides an exhaustive description of the experiments executed in pursuit of the research objectives listed in Section 1.2, accompanied by an analysis and commentary on the results obtained.
- **Chapter 5: Discussion** Expands on the implications, interpretations, and limitations of our findings, further deliberating the outcomes from the conducted experiments and suggesting potential avenues for future research that could extend and build upon this thesis.
- **Chapter 6: Conclusion** Summarizes the study's key findings and contributions.

Chapter 2

Background Theory and Related Work

This chapter draws heavily from and extends the literary review by Olweus, E [11]. It delves into the theoretical foundations and relevant previous studies that underpin the practical work detailed in chapters 3 and 4. The insights derived from this theoretical grounding will be further elaborated and discussed in chapter 5, providing the reader with a comprehensive understanding of the broader context within which our research is positioned.

2.1 Artificial Neural Networks

Artificial Neural Networks, machine learning algorithms designed to mimic human brain neuron behavior, constitute an integral part of advanced computational studies. These networks comprise numerous interconnected processing units or "neurons", systematically arranged into layers, each executing a distinct computation on the input data. A standard and elementary form of a neural network is the feedforward neural network, which features an input layer, one or more hidden layers, and an output layer[12].

Raw input data is received by the input layer and subsequently processed through the hidden layers. Every hidden layer executes a non-linear transformation on the data using a set of weights that are refined during training. The output layer generates the final predictions or outcomes of the neural network. Modern networks incorporate varying types of hidden layers, with the convolutional layer being a prominent example [13].

Activation functions are another significant component of a neural network. They define the output of a neuron based on its input [12]. Popular activation functions comprise the sigmoid function, the rectified linear unit (ReLU)[14], and the hyperbolic tangent (tanh) function [12].

Beyond layers and activation functions, a neural network employs a loss function to evaluate the discrepancy between the predicted and actual outputs. The neural network computes the gradient of the loss function, which aids in updating its weights during training. The objective is to minimize loss and enhance the model's accuracy. This process, commonly termed backpropagation, involves error propagation starting from the output layer and moving backward through the hidden layers. At each layer, the error informs the calculation of the loss gradient with respect to that layer's weights. These gradients subsequently update the weights, aiming to reduce the loss. This procedure persists until the gradients have been determined for all weights within the network [15].

The backpropagation algorithm offers an efficient methodology to compute gradients in a neural network, forming the backbone of several neural network training algorithms. It empowers the network to learn from the data and enhances its performance iteratively [15].

A noteworthy feature of neural networks is their ability to perform well on data unseen during training. Achieving this characteristic requires extensive example data, assisting the network to learn generalized weights rather than those applicable solely to specific training examples.

When a network significantly outperforms on training data compared to similar, unseen data, the model is deemed "overfitted" to the training data [12].

Artificial Neural Networks have demonstrated high performance across a multitude of tasks and disciplines, particularly in computer vision tasks such as image classification [13], object detection [16], and semantic segmentation [17].

2.1.1 Transfer Learning

Transfer learning is a potent technique in machine learning where the knowledge from a pre-trained model, specialized in a particular task, is used as a starting point to develop a new model. This process, when applied to artificial neural networks (ANNs), involves using the weights from the previous model as the initial weights for the new one, effectively transferring what the original model has learned to the new one.

The portion of the pre-trained model that is used as a basis for the new model is often referred to as the 'backbone'. This backbone typically comes from general networks that have shown strong performance across multiple tasks. A more task-specific component, known as the 'head', is then added to the end of the backbone to tailor the new model to its specific task [18].

The rationale behind transfer learning is to harness the lower-level features learned by the initial model as the base for the new model. The new model is then fine-tuned on the novel task using the fresh data. This technique can significantly conserve time and computational resources since training a model from scratch on an expansive dataset can be labor-intensive. Moreover, transfer learning has the potential to enhance the performance of the secondary model, as the pre-trained weights offer a robust base for learning the new task [19].

As elaborated in section 2.6, transfer learning is a prevalent strategy, particularly when repurposing the learning from a classification model for a segmentation task.

2.2 Image Classification

As previously discussed, image classification stands as a significant domain of research involving Artificial Neural Networks (ANNs), particularly in recent years. Several prominent annual competitions, such as the ImageNet Large Scale Visual Recognition Challenge[1], fuel this field. These competitions have incited substantial scientific advancements, including the popularization of Convolutional Neural Networks (CNNs) via AlexNet [13] in 2012 and residual blocks coupled with shortcut connections via ResNet in 2015 [20].

Convolutional Neural Networks constitute a class of NNs that incorporate convolutional kernel layers in addition to the standard fully connected layers. These networks enable the learning of specific image features, rather than comprehending the image as a whole. Generally, a fully connected layer is appended to the network's end to generate an outcome based on the collective feature extractors in the network.

Until 2015, a prevalent challenge in neural networks was the diminishing returns in learning when increasing the layers beyond a certain threshold. Post this threshold, deepening the networks paradoxically yielded deteriorating results, observable not just during validation but training as well. The decline in training performance indicated a problem with more complex NNs not attributable to overfitting. ResNet [20] addressed this issue via a network with **residual blocks** and shortcut connections between them. In a residual block, the input data goes through a function F consisting of normal neural network operators like convolutional filters, activation functions, and pooling, but there is an additional "shortcut" path where the input bypasses F and is added to the block output directly. The crucial point here is that function F is now trying to learn the 'residual' or the difference between the input and the desired output, rather than trying to learn the output directly. These blocks make ResNet-inspired architectures generally easier to optimize than networks not using them, especially in very deep networks. This innovation led to a surge in accuracy on visual tasks with networks as deep as 1000 layers. Additionally, it was discovered that the residual architecture was general, making it applicable to a plethora of tasks. Beyond triumphing in several ILSVRC categories, ResNet also secured the top position in the COCO challenge[2] that year

2.2.1 Performance Metrics for Image Classification

Precision and recall are commonly used measures for evaluating the performance of classification models, each providing distinct perspectives on a model's effectiveness.

[1]`https://www.image-net.org/`

[2]`https://cocodataset.org/#home`

In this context, four categories of prediction outcomes from the model are defined:

- *True Positives (TP)*: Instances where the model correctly identifies a positive case.
- *False Positives (FP)*: Instances where the model incorrectly identifies a negative case as positive.
- *True Negatives (TN)*: Instances where the model correctly identifies a negative case.
- *False Negatives (FN)*: Instances where the model incorrectly identifies a positive case as negative.

Recall calculates the proportion of actual positive cases that the model correctly identifies. It is computed as the ratio of True Positives to the sum of True Positives and False Negatives (Equation 2.1). Essentially, it assesses the model's capability to correctly identify all positive cases.

On the other hand, precision determines the proportion of predicted positive cases that are indeed correct. It is computed as the ratio of True Positives to the sum of True Positives and False Positives (Equation 2.2). This metric evaluates the model's accuracy in its positive predictions.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

For a dataset comprised of images, some with and others without solar farms, recall represents the proportion of actual solar farm images that are correctly labeled, whereas precision illustrates the fraction of the images labeled by the model that does in fact contain solar farms.

F1 score, another popular metric, combines both precision and recall. It calculates the harmonic mean of the two metrics, as illustrated in equation 2.3.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.3}$$

2.3 Semantic Segmentation

Semantic Segmentation, in contrast to image classification, involves categorizing the contents of an image at the pixel level. Rather than merely assigning labels to an entire image based on its overall content, this approach assigns labels to each individual pixel in accordance with the specific object or class that the pixel represents. As such, semantic segmentation allows for granular analysis of an image's composition. In the context of our work with Solis Oculus, for instance, the objective is to identify those pixels in an image that are part of a solar farm.

Semantic segmentation has a wide range of applications, including autonomous driving, medical diagnostics, and land cover analysis, to name a few [17]. A typical semantic segmentation dataset comprises images and corresponding masks, which depict pixel-level labels. An example of this structure from the Solis dataset is illustrated in figure 2.1. The output of a semantic segmentation algorithm mirrors the structure of these masks, allowing us to compute evaluation metrics such as Intersection over Union, detailed further in section 2.3.2.

The field of Semantic Segmentation, along with Neural Architecture Search (see section 2.7), constitutes the primary focus of the research presented in this work.

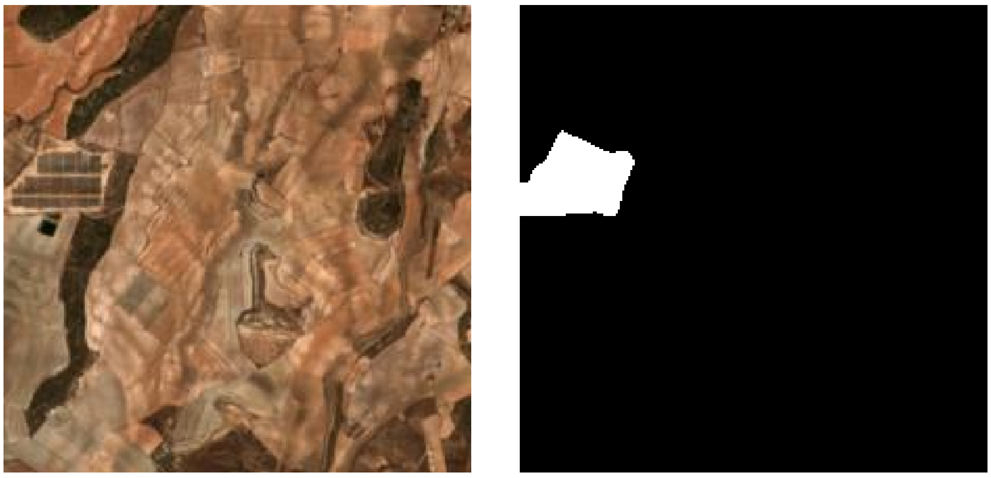

(a) Image from the Solis dataset **(b)** Mask from the Solis dataset.

Figure 2.1: Examples from the Solis dataset of the parts of a semantic segmentation dataset. White pixels in the mask represent the solar farm class.

2.3.1 Advancements in Semantic Segmentation

Semantic segmentation is an area where CNNs have exhibited substantial success, highlighted by the victory of the Fully Convolutional Network (FCN) [21] in the COCO segmentation task in 2014. This achievement was credited to replacing the fully connected layers at the end of popular networks like AlexNet, VGG, and GoogLeNet with convolutional layers. This modification led to significant speed increases during both forward and backward passes in training [21]. The method employs upsampling techniques to restore the output feature map of the image to its original size for pixel-by-pixel predictions.

U-Net

U-Net further improved this process in 2017 by incorporating the output before each subsampling stage as input during the upsampling phase. This enhancement aids in more accurately mapping recognized features back to the original image size [22]. As per [23], U-Net is particularly effective for semantic segmentation on remote sensing imagery in a lot of cases due to its superior performance with less training data in comparison to other algorithms. This can be an advantage if the original dataset is very small. Hou et al. [6] and Kruitwagen et al. [3] both use a U-Net for semantic segmentation of solar farms.

Dilated Convolutions

Dilated convolutions, also referred to as "atrous" convolutions, are a variant of convolutional neural network (CNN) layers that utilize dilated kernels to enlarge the receptive field of a layer without augmenting the number of parameters [24]. Traditional CNNs determine the receptive field of a layer based on its filter size and stride. However, dilated convolutions employ filters with gaps or "dilations," the size of which is decided by the dilation rate, enabling the filters to cover a larger input area without augmenting the number of parameters or computational complexity. This characteristic is particularly beneficial for semantic segmentation, where maintaining spatial resolution while increasing receptive field to capture long-range dependencies in data is crucial [17].

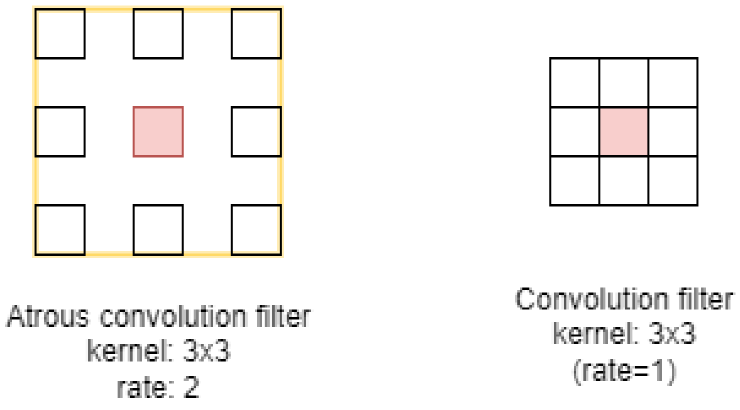


Figure 2.2: Left: A dilated/atrous convolution filter. Right: A normal convolution filter

DeepLab

DeepLab [24] is a state-of-the-art semantic image segmentation model, known for its effectiveness in tasks requiring understanding at the pixel level. The core idea behind DeepLab is to apply dilated convolutions to enlarge the field of view in convolutional neural networks, thereby capturing more contextual information without losing resolution. Additionally the head

An integral part of the DeepLab architecture is the Atrous Spatial Pyramid Pooling (ASPP) module [24]. This module applies several parallel dilated convolutions with different dilation rates to the input feature map. By doing so, it effectively captures the context of objects at different scales. The outputs are then concatenated and passed through a 1x1 convolution for dimension reduction before being fed into the next layers or being upsampled for pixel-wise prediction. This way it enhances the model's capability to handle objects of different sizes and maintain precise boundaries.

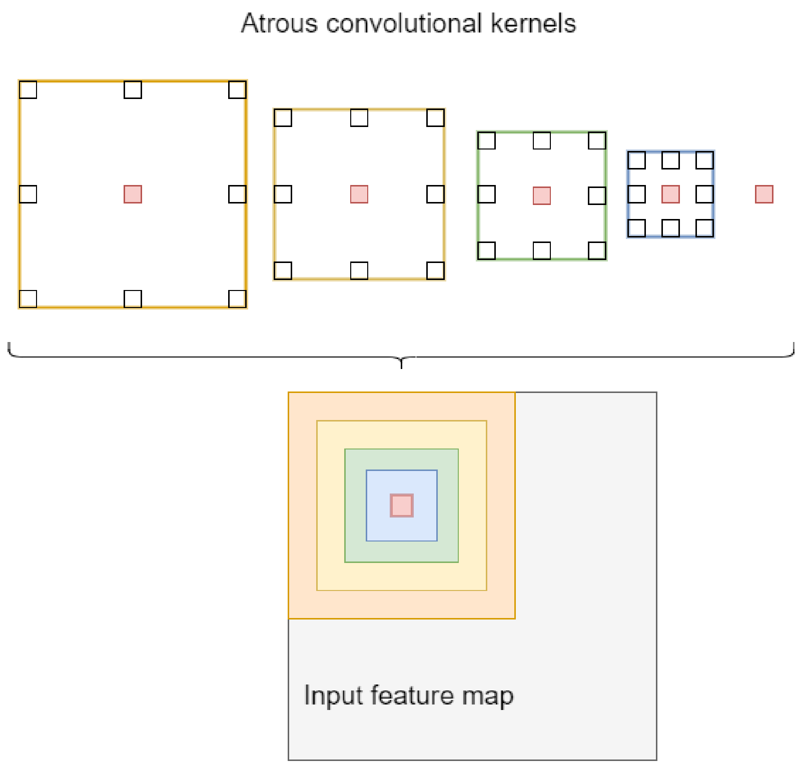


Figure 2.3: Figure of an ASPP module

Segment Anything Model

A very recent, yet very impressive, contribution to the semantic segmentation field is Meta's Segment Anything (SA) project and the corresponding SA Model (SAM) [25]. The project contains the largest segmentation dataset in the world as of April 2023 boasting more than 11 million images with 1 billion corresponding masks. SAM is a 'zero-shot' model that can easily be adapted to any task, and the demo available at `https://segment-anything.com/demo` provides some very convincing demonstrations, some of which can be seen in Figure 2.4.

An additional project by Wu and Osco [26] provides open software[3] to use SAM for segmenting geospatial data.

[3]`https://samgeo.gishub.org/`

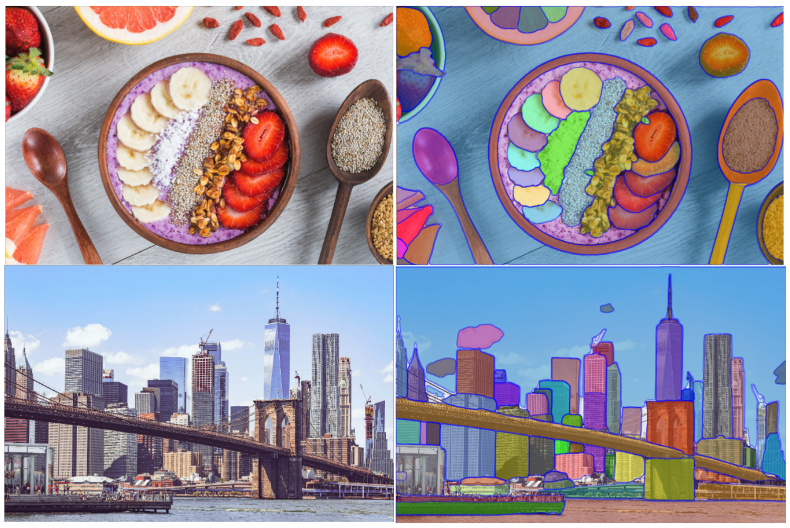


Figure 2.4: Left: Example images from the SAM demo. Right: SAM's segmentation predictions

2.3.2 Performance Metrics for Semantic Segmentation

Despite semantic segmentation essentially being pixel-wise classification, using pixel-wise accuracy as a performance metric may not always yield accurate insights. As Thoma [27] indicates, one major issue arises when parts of an image predominantly exhibit the same class. For instance, in our case of solar farm detection, our interest lies in the binary separation of pixels into two classes: solar farm and non-solar farm. However, even in images containing solar farms, the vast majority of pixels are classified as non-solar farm. This imbalance can potentially inflate the perceived accuracy of a model that simply predicts every pixel as not being a part of a solar farm, thereby yielding an artificially high accuracy score.

The *Intersection over Union* (IoU) seen in equation 2.4, also known as the Jaccard Index, and the *Dice score* or *F1-score* seen in equation 2.3 are among the most widely used metrics for semantic segmentation in 2D images that help to address this shortcoming. IoU is also visually represented in Figure 2.5.

The F1-score, still calculated by equation 2.3, is particularly suitable for binary classification tasks according to Thoma et al. [27].

$$IoU = \frac{TP}{TP \cup FP \cup FN} \tag{2.4}$$

2.4 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of computational models inspired by biological evolution's principles and mechanisms. These stochastic optimization

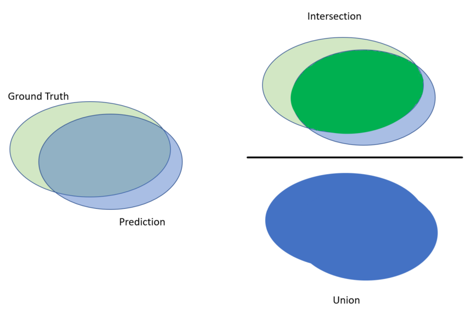


Figure 2.5: Visual representation of Intersection over Union

algorithms function on the idea of survival of the fittest. They operate by representing possible solutions as genotypes, evaluating their fitness with a function reflective of the problem at hand, and then applying evolutionary operators such as selection, crossover, and mutation. The fittest individuals are selected for reproduction to produce offspring of the next generation. The algorithms then evaluate the new generation, and this process repeats until a satisfactory solution is reached, or until a given termination criterion is met [28].

EAs find extensive application in solving complex optimization problems, particularly those with no direct analytical solutions or large solution spaces that are computationally expensive to explore exhaustively. One notable application area is Neural Architecture Search (NAS), where EAs are used to discover optimized neural network architectures with little human intervention [29].

The basic process of an EA involves three primary components: solution representation, fitness evaluation, and evolutionary operators [28].

Solution Representation: The core of an EA is the encoded representation of potential solutions, often termed as 'individuals' or 'chromosomes.' These chromosomes embody the 'genotype', the encoded solution to the problem under consideration.

Fitness Evaluation: To estimate the quality or effectiveness of a given solution, a fitness function is employed. This function assesses each solution's 'fitness' in the population, reflecting the problem's particular objectives and constraints.

Evolutionary Operators: Evolutionary operators are responsible for generating new solutions from the existing population. Typical operators include selection, crossover, and mutation. The selection operator chooses the fittest solutions to be the parents of the next generation. The crossover operator combines the selected parent solutions to produce offspring. The mutation operator introduces small, random changes into a solution to maintain diversity in the population and avoid premature convergence.

2.5 Computer Vision Challenges in Remote Sensing Imagery

Applying computer vision techniques to remote sensing imagery presents a unique set of challenges. One of the primary issues stems from the varying characteristics of satellite images depending on their sources. These images often span vast geographical areas, representing substantial chunks of countries, which means they also comprise volumes of data in the magnitude of gigabytes. The sheer size of these images makes them computationally demanding to process.

The resolution of the satellite cameras also presents a particular dilemma – a tradeoff between higher resolution and more frequent image capture. High-resolution images, while facilitating more detailed analysis, come with their own set of complications. They are not only expensive to acquire but also captured infrequently due to the increased data demands and limited satellite resources. On the other hand, lower-resolution images can be captured more often but may not provide the level of detail necessary for some use cases.

The annotation or labeling process for these images is another laborious task, particularly when dealing with objects not present in public databases. This process demands significant time investment and specific knowledge about the objects' locations. Moreover, many objects in these images share similar visual characteristics as shown in figure 2.6, making differentiation challenging for both humans and algorithms.

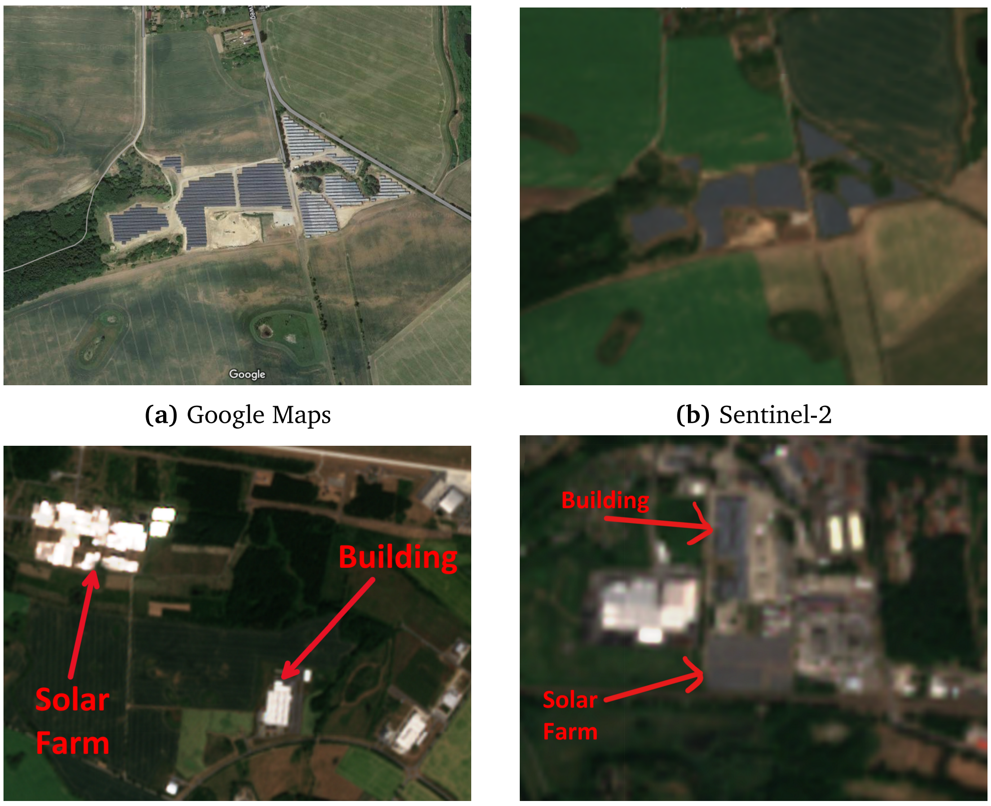
2.5.1 Sources of Satellite Imagery

Satellite imagery comes from a myriad of sources, each varying in their unique traits and levels of accessibility. One of the most notable sources is the European Space Agency's Sentinel project[4]. This project comprises numerous missions, each catering to different aspects of Earth observation. Among these, the Sentinel-1 and Sentinel-2 missions provide invaluable data for terrestrial visual tasks.

Sentinel-1 utilizes a C-band synthetic aperture radar (SAR) for image capture. This technology allows it to take clear images regardless of weather conditions or cloud cover, providing reliable imaging data under all circumstances. On the other hand, Sentinel-2, launched in 2015, focuses on tracking changes on the Earth's surface. It uses a multispectral camera, which captures images across 13 spectral bands.

The resolutions provided by Sentinel-1 and Sentinel-2 are $5m^2$ and $10m^2$ per pixel, respectively. While this level of resolution could pose constraints for certain tasks, for large structures such as grid-connected photovoltaic (PV) plants, it tends to be sufficient. As pointed out by Kruitwagen et al. [3], these facilities are often larger than 10 000 m^2, which makes them distinguishable even at these resolutions. That said, there can still be challenges in discerning similar-looking structures, such as rice paddies, greenhouses, parking lots, and lakes from solar

[4]https://sentinels.copernicus.eu/web/sentinel/home



(a) Google Maps **(b)** Sentinel-2

(c) Solar farm reflecting sunlight **(d)** Solar farm and building

Figure 2.6: Images highlighting the challenge of discerning solar farms in Sentinel-2 imagery

farms. Sentinel-2's multispectral camera can partially mitigate this issue by utilizing the unique spectral profile solar farms possess [3, 4, 6].

There are alternative sources, such as Planet[5], offering higher-resolution imagery. However, the high cost associated with these images often makes them unsuitable for large-scale projects requiring global coverage.

2.5.2 Data Related Challenges

Deep learning models, especially convolutional neural networks (CNNs), heavily rely on ample quantities of accurately annotated data for effective training. While automatic labeling methods can expedite this process, manual annotation remains the gold standard for ensuring label accuracy, albeit at the cost of substantial time and effort.

Data collection at a large scale, such as imaging an entire country or continent, presents further challenges. The exhaustive review of each image for quality is impractical in such a vast dataset. The Solis project operates with 224x224 pixel

[5]`https://www.planet.com/`

chips, which are subsets of larger Sentinel-2 tiles. Given that each pixel represents a 10x10 meter area, each chip covers approximately 5 square kilometers. Considering the area of Europe is around 10.18 million km^2, approximately 2 million such chips would be needed to comprehensively cover it.

Cloud cover in Sentinel images adds another layer of complexity, despite attempts to mitigate its effects through techniques like mosaicing or cloud cover filtering. Services such as Sentinel Hub provide an option to specify a maximum permissible cloud coverage, yet this percentage is calculated for tiles spanning 12,000 km^2 [6]. As a result, it is entirely feasible for a 224x224 chip derived from such a larger tile to be completely shrouded by clouds, even if the parent tile meets the cloud cover threshold.

2.6 Previous Work on Solar Farm Detection

Several studies have explored the detection of solar panels on satellite imagery, utilizing both ANNs and other methods. For instance, a random forest model was employed by Plakman et al. [4] to detect solar panels, and this model was trained and evaluated using a publicly accessible dataset from the Netherlands. Hou et al. developed SolarNet, a system that integrates the merits of Expectation-Maximization Attention Networks and a U-Net architecture, to uncover new photovoltaic (PV) systems in China [6]. Meanwhile, in Brazil, a study used high-performing segmentation models with different pre-trained backbones [5].

A group from Stanford has identified and compiled large-scale solar platforms and rooftop solar installations in the US into the publicly accessible DeepSolar database [7][7]. Astraea Earth trained a Deep Convolutional Neural Network in the US and utilized it to identify new solar farms in China [10].

One particularly significant contribution is the paper by Kruitwagen et al. in 2019 [3]. Along with the paper, they released a global dataset of solar energy facilities, which expanded the existing asset-level data by an impressive 432%. This work represents the most substantial single contribution to this field to date, measured purely by the number of previously unknown facilities discovered and added to public datasets. Focusing on PV platforms larger than 10 000 m^2, they achieved a precision of 98.6%, a recall of 90%, and an Intersection over Union (IoU) of 90% for the segmentation task on their test set. They employed a U-Net-based CNN model and used two sources of remote sensing imagery to achieve these results. Importantly, they leveraged the non-visible bands of Sentinel-2, demonstrating their significant role in the model's solar panel recognition.

2.6.1 Established Methodology for Detecting Solar Farms

In the realm of discovering solar farms on remote sensing imagery, a specific set of processes emerges as a common factor across various studies [3, 6, 7, 10].

[6]According to `https://docs.sentinel-hub.com/api/latest/data/sentinel-2-l2a/`

[7]`https://deepsolar.web.app/`

These works typically employ a complex, sophisticated pipeline for both training the model and deploying it in real-world scenarios. Although there are slight variations, the core steps remain largely consistent across these studies. Figure 2.7 provides an illustration of this process.

The process commences with the identification and labeling of known solar farms on satellite images as georeferenced polygons, often using a GIS tool such as QGIS[8]. These images then go through a series of preprocessing operations, including cloud removal, image standardization, and chipping or subdividing the images into smaller segments that can be efficiently processed by the network. These chips[9] become our dataset.

The subsequent phase involves training a classification model using a dataset of chips, with and without solar farms. Once trained, a segmentation head (see 2.1.1) is attached to the model and this amalgamated network is further fine-tuned for segmentation tasks. Approaches differ in whether they completely freeze the weights of the backbone, or allow the weights to be modified in the training of the segmentation model. It's noteworthy that slight modifications are usually introduced to the backbone to preserve spatial information during its application for segmentation tasks [17].

As per the findings presented by Hou et al. the success of this approach is largely credited to the activation mapping for the classification model, which resembles a dense prediction or segmentation [6]. This assertion is intuitively plausible as the model would necessitate learning the unique features of solar farms to correctly predict their presence in an image. A notable advantage of this strategy is the time efficiency it offers compared to training an entirely new network from scratch.

Lastly, the trained models are deployed over extensive geographical areas (represented by the globe in Figure 2.7). The images of these areas undergo the same preprocessing steps, without prior manual identification and labeling of solar farms. Following this process, the models' findings are manually inspected, and confirmed solar farms are added to our dataset. Now we can repeat the cycle with the new dataset.

2.6.2 The Solis Oculus Project

The Solis Oculus (Solis) project, launched by Enernite, aspires to construct a comprehensive database of all solar farms worldwide, with the motivations for this endeavor explained in 1.1. We've adopted the pipeline illustrated in figure 2.7 to develop two proficient models.

Firstly, for the classification task, we employ the ResNet50 model, with a binary classifier implemented as the head. Secondly, for the semantic segmentation task,

[8]`https://qgis.org/en/site/`

[9]The chipped images derive their ground truth from the labeled polygons. If any segment of the image overlaps with a part of the polygon, it is labeled as a "solar farm". For classification purposes, any chip encompassing a portion of a solar farm is labeled as "solar farm".

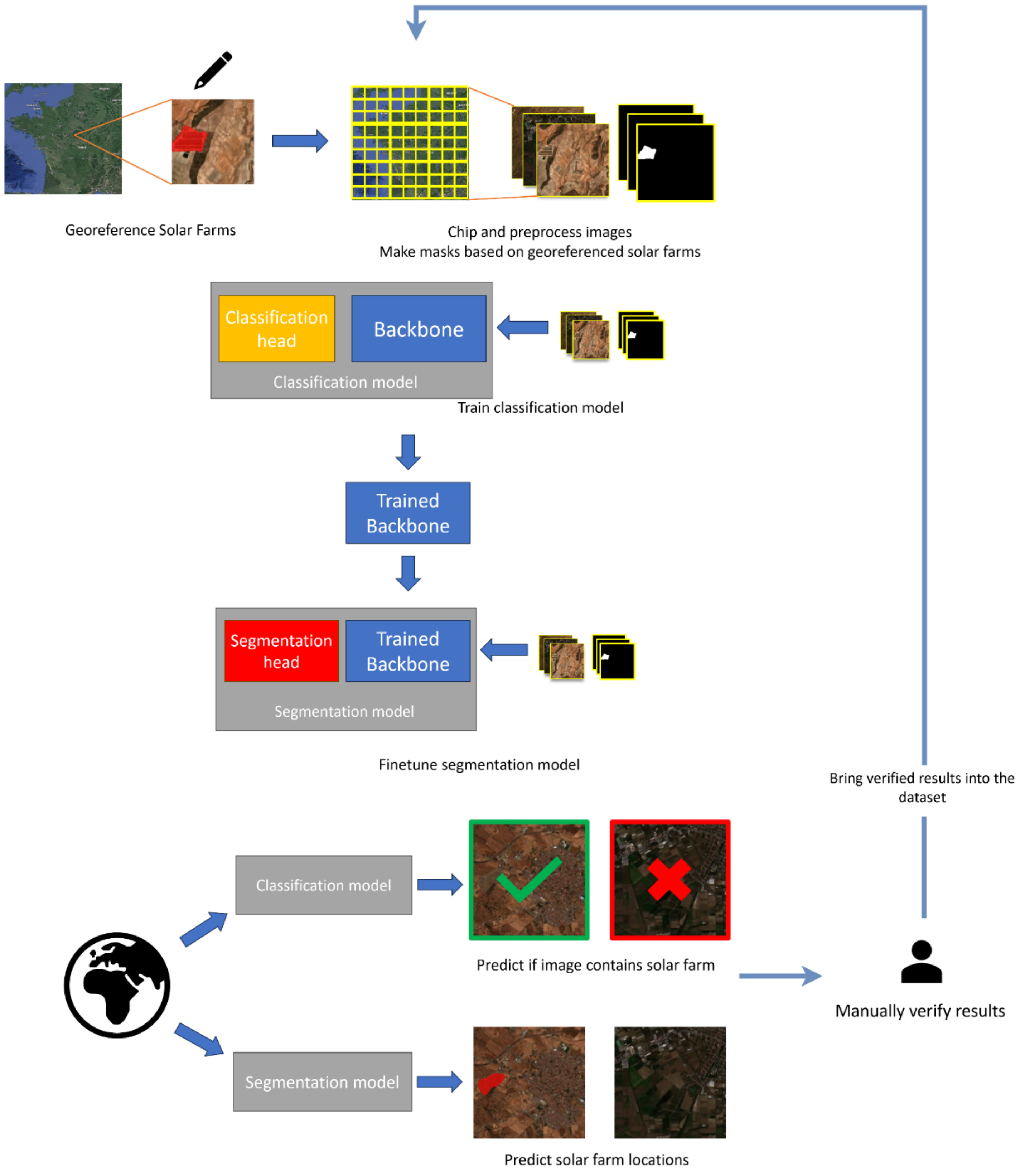

Georeference Solar Farms
Chip and preprocess images
Make masks based on georeferenced solar farms
Classification head
Backbone
Classification model
Train classification model
Trained Backbone
Segmentation head
Trained Backbone
Segmentation model
Finetune segmentation model
Bring verified results into the dataset
Classification model
Predict if image contains solar farm
Manually verify results
Segmentation model
Predict solar farm locations

Figure 2.7: A typical machine learning pipeline for discovering new solar farms

we amend the trained ResNet by substituting the strided convolutions with dilated convolutions, a common practice for using classification backbones in segmentation [24], and attach DeepLabV3 as a segmentation head. During the training process, the ResNet weights are fine-tuned for segmentation.

This implementation of transfer learning significantly reduces the time it takes for the segmentation model to converge. In the context of this study, we will refer to this ResNet50-DeepLab composite pre-trained on classification as the *Solis-transfer* model. This model serves as the most important baseline for comparing the architectures produced via neural architecture search.

2.6.3 Potential Enhancements for Solar Farm Detection Techniques

As we strive to improve the accuracy and efficiency of solar farm detection methods, several promising avenues appear ripe for exploration. Among the most impactful of these, amplifying the volume of training data takes precedence. Astraea Earth reports that augmenting training data significantly outperformed other measures such as hyperparameter tuning or swapping CNNs in terms of improving network performance [10].

Incorporating more spectral bands from Sentinel imagery into the mix presents another intriguing possibility. A study by Plakman et al. [4] suggests that Synthetic Aperture Radar (SAR) products from Sentinel could enhance detection techniques. However, the utilization of this data can be challenging if leveraging transfer learning, given that publicly available pre-trained networks typically used as backbones, like ResNet, primarily are trained on RGB images. Direct addition of new input channels might inadvertently degrade the performance of feature extractors, a phenomenon thoroughly explained by Zhang et al. [23]. Nonetheless, Zhang et al. offer a potential workaround with their proposed MSNet architecture, which separately encodes RGB and other bands and later combines them during decoding. This is not an issue if the backbone is also trained with the same spectral data.

On the other hand, one could also consider deploying a hybrid model to conserve computational and financial resources during model application over vast territories like countries. This could involve a two-step process, where an initial model (which could be something less computationally expensive, such as a Random Forest algorithm), eliminates areas unlikely to feature photovoltaic (PV) platforms on low-resolution imagery, leaving the more computationally demanding model to handle only the remaining potentially relevant locations with higher resolution images.

Lastly, and significantly for the focus of this thesis, we have the prospect of refining the neural network's architecture. This proposition is especially pertinent given the proven effectiveness provided by additional spectral bands' and the fact that most popular ANNs are designed with RGB images in mind. Designing effective neural networks, however, is a complex task demanding substantial expertise. Therefore, in the subsequent section, we turn our attention to an approach that automates the design of neural network architectures.

2.7 Introduction to Neural Architecture Search

Neural Architecture Search (NAS) constitutes a specialized research domain focused on the automated creation of optimal Neural Network configurations for specific tasks. Demonstrating its capability to outperform conventional networks on diverse tasks such as image classification and semantic segmentation, NAS is swiftly gaining traction in the AI community [8].

According to White et al. [9] NAS is a process that can be mathematically expressed as below in equation 2.5. The goal of NAS is to minimize the validation loss by finding the optimal weight $w^*(a)$ for each possible input a within a certain time limit t.

$$\min_{a \in A} L_{\text{val}}(w^*(a), a) \quad \text{s.t.} \quad w^*(a) = \arg\min_{w} L_{\text{train}}(w, a) \tag{2.5}$$

In this equation, L_{val} represents the validation loss and L_{train} represents the training loss. $w^*(a)$ denotes the optimal weight corresponding to an architecture a, and A represents the set of all possible architectures.

According to Elsken et al., NAS can be dissected into three primary dimensions: the defined search space, the performance evaluation technique applied to assess the networks, and the search strategy adopted to explore the architectures in the search space. [8].

The **Search Space** sets the bounds for the possible network configurations that a NAS algorithm can generate. The size of these spaces can vary significantly, ranging from a few thousand potential architectures to exceptionally vast, sometimes approaching infinite.

The **Performance Evaluation Strategy (PES)** is the metric used to measure the effectiveness of a neural network within the search space.

The **Search Strategy** is the algorithmic process used to navigate through the search space in order to optimize the neural network as per the PES.

While the components mentioned above form the foundation of NAS, the particular focus of this thesis necessitates a more in-depth exploration of two pivotal developments in the field: Auto-DeepLab [30] and, by extension, Differentiable Architecture Search (DARTS) [31] - the foundation upon which Auto-DeepLab is built.

DARTS as a gradient-based method, marks a significant milestone in architecture search by pioneering an approach that simplifies the management of extensive search spaces and complex network configurations [31]. Meanwhile, where most NAS strategies have primarily been investigated in the context of classification tasks [9], Auto-DeepLab expands upon DARTS' principles, tailoring this methodology for segmentation tasks, thus amplifying the scope and applicability of NAS [30].

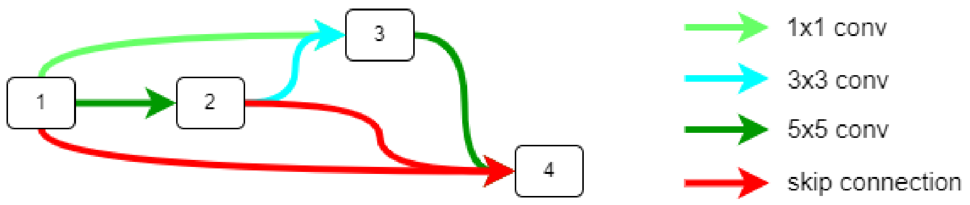


Figure 2.8: Illustration of a Directed Acyclic Graph (DAG) where each edge signifies a distinct operation. This figure represents merely one example; in reality, the characteristics of the edges are determined through an optimization process.

In the ultimate subsections, we delve into a comprehensive analysis of DARTS (Section 2.7.4) and Auto-DeepLab (Section 2.7.5). Through these in-depth explorations, we aim to shed light on the core principles, features, and potential benefits offered by these methods, setting the stage for their application in the experiments conducted in this thesis.

2.7.1 Search Space

The NAS search space embodies a set of parameters that outline potential neural network architectures. These parameters can span the number and nature of layers, connection types between layers, and the number of neurons per layer. Essentially, the search space specifies the potential neural network architectures that the search strategy algorithm can traverse [8].

One prevalent category within search spaces is **cell-based search spaces** [9]. Here, potential neural network architectures are outlined by a series of "cells" - modular units that combine to form more extensive neural networks [32]. Each cell symbolizes a specific configuration of layers and connections and can be combined and reiterated with other cells to manifest a broad spectrum of potential network architectures [31, 33].

The cell-based search space is generally characterized by a set of parameters specifying the potential configurations for each cell, including the number and types of layers and the possible connections between different layers. This structure is frequently illustrated as a Directed Acyclic Graph (DAG) [33, 34] where the edges are layers/operations as in figure 2.8.

The allure of using a cell-based search space in NAS lies in its capacity to facilitate the representation of relatively complex architectures in a compact form, especially if the cells recur in a predefined macro architecture like in NAS-Bench-201 [32] as illustrated in figure 2.9. This results in a compact yet diverse search space, simplifying the quest for the optimal architecture for a particular task while enabling the discovery of innovative architectures that might have been overlooked with traditional hand-designed methods[9].

Cell-based search spaces, much like their counterparts, are typically defined by a single level of topology, primarily focusing on optimizing one architectural dimension, namely, the recurrent cell. This stands in contrast to **hierarchical search**

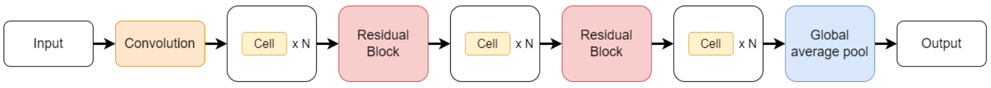


Figure 2.9: Depiction of the fixed macro architecture specific to NAS-Bench-201. The quantity of cells, denoted by *N*, along with the characteristics of the cells themselves, are variables subject to change

spaces that expand the optimization process beyond a single axis. Such spaces typically integrate macro-architectural parameters into the optimization process, thereby adding a layer of complexity [9]. In section 2.7.5, we will delve deeper into the intricacies of hierarchical search spaces, taking Auto-DeepLab [30] as a primary case study.

2.7.2 Performance Evaluation in NAS

While Neural Architecture Search (NAS) is known to enhance the efficacy of Deep Neural Networks (DNNs) across a range of tasks, it also presents certain challenges. A particularly important challenge stems from the extensive computational resources required to evaluate the performance of architectures within the search space during the search [8, 9]. While the straightforward method involves training every network to convergence on the task and employing loss or task-specific metrics for performance measurement, this can be extremely computationally intensive and time-consuming given the sheer size of some search spaces. A significant body of research thus aims at refining the efficiency of the evaluation phase of the search, proposing several innovative strategies for achieving this [35–37].

Low Fidelity Estimates

Low fidelity estimates comprise one category of solutions, aiming to predict final network performance based on a "reduced" training process. Experiments in this domain have reduced various aspects of the process, including training the entire network for fewer epochs, diminishing the number of filters in the layers, or reducing the resolution of training set images, among others. Zhou et al. [38] take a comprehensive approach with their EcoNAS, reducing multiple aspects simultaneously. While these methods significantly curtail the computational cost of evaluation, they introduce biases. Initial observations suggest that the relative ranking of architectures can substantially fluctuate if the disparity between reduced and full training is considerable [8].

Learning Curve Extrapolation

Another tactic involves the estimation of the architecture's future performance based on the learning curve after a fixed number of epochs. This technique is known as **learning curve extrapolation** [8].

Weight Inheritance

Weight inheritance offers a method to expedite the initialization of an architecture's weights by leveraging the data from a previously trained network. This process employs a technique known as network morphisms. These mathematical constructs enable modifications to a neural network's architecture, such as layer addition, removal, or duplication, while preserving the learned function, thereby eliminating the need to retrain every network from scratch. This practice significantly accelerates evaluations by allowing a network's capacity to grow incrementally without compromising the existing high-performance level [39].

One-Shot Models

One-shot models, in contrast to weight inheritance, allow many networks to share the same weights. They achieve this by training a large supernet (sometimes called hypernet) and subsequently applying the supernet's weights to multiple architectures that are sub-graphs of the supernet. This approach creates an inherent copulation between the PES and the Search Strategy (see section 2.7.3) as the method simultaneously learns the model weights and architecture [9]. The main advantage of one-shot methods is that they drastically reduce the training time and computational resources required as they train one large model continuously rather than retraining every new architecture it explores from scratch. However, one notable disadvantage of this approach is the requirement for all of the supernet's weights to be in memory, setting a strict limit on the size of the networks. As a result, one-shot models are often employed with cell-based approaches, where the same cell is used multiple times successively with a fixed macro architecture [8]. Efforts are being made to address the memory issue where DCNAS by Zhang et al. [40] is a good example. We further discuss the search aspect of one-shot models in section 2.7.3

Zero Cost Proxies

Zero-cost proxies, a relatively new development in NAS evaluation, offer rapid evaluations based on easy-to-calculate metrics such as a single mini-batch of training or the number of parameters in the network, thereby adding virtually no cost to the search [35]. According to Mellor et al. [41], these proxies can evaluate an architecture in just a few seconds. Yet, as White et al. [36] note, while they are extremely fast, they can be unreliable predictors in isolation. However, they can be combined with other predictors to enhance overall performance [37]. For example, the OMNI method presented by White et al. [37] integrates a learning curve method (SoTL-E [42]), a zero-cost method (Jacobian covariance [35]), and a model-based method (SemiNAS [43] or NGBoost [44]), and generally outperforms any single method alone.

Tabular NAS Benchmarks

Tabular NAS Benchmarks, a groundbreaking development in the field of neural architecture search (NAS), denote an exhaustive procedure of training and appraising all architectures within a defined search space using one or multiple datasets. They were introduced to address the reproducibility concerns prevalent in the NAS research community [45]. Ying and Klein et al. [45] took the initiative to publish NAS-Bench-101, which cataloged the training and validation results for approximately 423k architectures within a large cell-based search space across several widely-used datasets, including CIFAR datasets [46] and ImageNet. The intent was to completely remove the evaluation cost associated with executing a NAS within these parameters, hence fostering more research into NAS and significantly reducing the requirement for computational resources. This, in turn, simplified the comparison of different search algorithms within the search space encompassed by the table. In the ensuing years, numerous papers have been released featuring tables presenting evaluation data for alternate search spaces and for various datasets and ML tasks [32, 47, 48].

By nature, NAS research is extremely sensitive to minor changes in random seeds, hyperparameters, and training pipelines, and Tabular NAS Benchmarks have revolutionized its reproducibility [49].

Model-based Predictors and Surrogate NAS Benchmarks

Model-based predictors, which employ a machine learning model to map an architecture to a fitness score, have been demonstrated to be particularly efficient according to White et al. [37]. These predictors come to the fore in the creation of Surrogate NAS Benchmarks, a solution proposed to overcome the size limitations of Tabular Benchmarks.

While Tabular Benchmarks are often extensive, typically incorporating between 6k-423k architectures, they pale in comparison to the search spaces usually contemplated in NAS literature, such as DARTS [31], which can include as many as 10^{18} or more architectures depending on the choice of parameters. This disparity could result in findings on benchmarks not being transferrable to a more expansive search space [49].

Surrogate NAS Benchmarks address this issue by evaluating fewer architectures within larger search spaces, then training a regression model to map an architecture to a fitness score, typically a validation score on some task [49]. According to Siems et al., surrogate benchmarks can offer more realistic loss scores than single tabular entry benchmarks[10], as the surrogate model ameliorates the noise typically found in benchmark tables due to the stochasticity of mini-batch training of Neural networks. This method has been used to create the NAS-Bench-301, a surrogate benchmark boasting a search space with 10^{21} potential architectures [49].

[10]Benchmark tables trained with only one random seed. It is common however to use 3 random seeds like in NAS-Bench-101 [45]

2.7.3 Prominent NAS Search Strategies

The roots of Neural Architecture Search (NAS) can be traced back to 1989, when an evolutionary algorithm was first applied by Miller et al. [50] to optimize neural network architectures. Since that seminal work, an array of diverse algorithms has been introduced to enhance the efficiency and robustness of neural architecture generation.

As we survey the landscape of NAS in 2023, search strategies predominantly fall into two main categories: One-Shot methods and black-box methods. It's noteworthy that these categories are not mutually exclusive, and a particular strategy may not fall squarely into either category or may straddle both [9].

Black-box methods have been notable for their frequent use in the field. Conceptually straightforward, these strategies, including Bayesian optimization, evolutionary algorithms, and reinforcement learning, have been widely adopted [51]. However, one downside of these techniques is their significant computational cost, with some studies reporting the use of thousands of GPU days for their experiments [8, 9].

In contrast, one-shot methods have gained traction due to their considerable efficiency. These methods manage to generate promising results within a far shorter time span - typically a few GPU days, and in some cases, as reported by Dong et al. [52], even within a span of just a couple of hours.

In the ensuing discussion, we will delve into some of the most prevalent strategies in current NAS research. Our exposition will heavily draw upon comprehensive surveys by prominent researchers in the field, notably, Elsken et al. [8] and White et al. [9].

Random Search

Random Search is a basic yet invaluable algorithm that indiscriminately generates architectures, returning the best one encountered within a specific time limit. Its primary utility in NAS research stems from its unguided nature, making it an ideal benchmark for other algorithms. While it may not consistently deliver the most robust performance, its value lies in its unpredictability. Both Zoph et al. [53] and Ochoa et al. [33] state that Random Search sets a significant challenge, as many cutting-edge strategies find it hard to outperform.

Reinforcement Learning

Reinforcement Learning is another commonly employed technique in NAS. This approach, initiated by Zoph et al. [54], treats the NAS problem as a Reinforcement Learning task and has contributed significantly to the mainstreaming of NAS research [8]. In this setup, the search space is regarded as the action space of the RL agent. The actions taken by the agent, or the architectures it generates, are then evaluated [8].

Evolutionary Computation

Early in the history of Evolutionary NAS (ENAS), the approach was heavily reliant on neuroevolution, a technique that employs evolutionary algorithms to optimize both the architecture and weights of the networks [55]. However, with the evolution of Deep Neural Networks, gradient-descent methods have proven more effective for weight optimization [8], leading contemporary ENAS research to focus on using evolutionary algorithms primarily for architectural optimization. This approach entails initializing a population within a predetermined search space, with each individual serving as a coded solution or architecture. The fitness of the initial population, indicative of the expected performance of the architectures, is then evaluated. This is followed by the evolutionary process, where architectures are selected, crossed over, and mutated to create novel solutions until a predefined stopping criterion is met [51].

Local Search

Local Search is an approach that has recently gained significant popularity. Multiple experiments on NAS-Benchmarks have demonstrated that Local Search can deliver state-of-the-art performance, particularly when the search space has minimal noise [33, 56, 57]. It encompasses three primary components:

1. A search space $\boldsymbol{S}$ where a valid solution/architecture $\boldsymbol{s}$ belongs, i.e., $\boldsymbol{s \epsilon S}$
2. A neighborhood function $\boldsymbol{N}$ such that $\boldsymbol{N(s)}$ denotes the set of neighbors for a specific solution
3. A fitness/PES function (see section 2.7.2) $\boldsymbol{f(s)}$ to assess the capability of an architecture

To illustrate, the paper [33] utilizes a straightforward encoding with a Directed Acyclic Graph (DAG) representing the search space of NAS-Bench-201 [32]. The search space is a cell-based structure with six nodes each having five different options, where $N(s)$ are the set of solutions such that the hamming distance from s is equal to one. This is the algorithm in broad terms:

Algorithm 1 Local Search(s)

```
Find neighbors $N(s)$
for $n$ in $N(s)$ do
    if $f(n) < f(s)$ then
        Local Search(n)
    end if
end for
return $s$
```

This algorithm starts by randomly picking a solution within the search space and then recursively applies Local Search until it cannot find a better neighbor. When the algorithm returns, it attempts to escape from what could potentially be a local optimum by introducing small perturbations and then reruns the Local Search algorithm. Please note that other implementations evaluate all neighbors before taking the recursive step [33] or employ other strategies to escape from local minima.

Gradient-based NAS Algorithms

A key focus of this thesis is gradient-based NAS algorithms and especially in the context of one-shot models. These algorithms leverage the power of gradient-based optimization to explore the search space, often augmenting the traditional loss function to allow for this integration [8, 30, 31]. A significant advantage of many gradient-based methods is their ability to simultaneously tune a model's architecture and weights, which potentially mitigates the risk of falling into local minima due to the premature exclusion of beneficial architectural components that may not exhibit promising performance during the nascent stages of model training. However, it is important to note that not all gradient-based NAS algorithms follow this simultaneous optimization approach. Section 2.7.4 will delve deeper into Differentiable Architecture Search (DARTS) which is a popular strategy that exemplifies many aspects of gradient-based NAS algorithms [31].

2.7.4 DARTS: Differentiable ArchiTecture Search

The Differentiable Architecture Search (DARTS) paradigm, proposed by Liu et al. in 2019, presents a novel approach to the automation of network architecture search [31]. DARTS utilizes a unique combination of a cell-based search space and a gradient-based one-shot model, facilitating efficient exploration and evaluation of architectures. The search space in this context is realized as a Directed Acyclic Graph (DAG) where each edge of which can perform one out of 8 potential operations as shown in figure 2.10.

The DARTS Search Space

A cell, in the DARTS methodology, is essentially a DAG constructed from B blocks. Here, the 'ith' block within the 'lth' cell is defined using a 5-tuple (I_1, I_2, O_1, O_2, C), where:

- I_i^l encompasses the potential input tensors for the 'lth' block. This set includes the output from the preceding cell H^{l-1}, the output from the cell two steps prior H^{i-2}, and the output from earlier blocks within the current cell $H_1^l, ..., H_i^l$.
- C signifies the approach used for integrating the outputs from the two branches to generate the output tensor of this block. In this model, the only available choice for the set of possible combination operators 'C' is element-wise addition.

- H_i^l corresponds to the blocks' output tensors. The output tensor for the cell is derived by concatenating these block output tensors $H_1^l, ..., H_B^l$ sequentially.
- O denotes the possible layers applicable to the corresponding input tensor. The operator pool consist of:
    - 3 x 3 Depthwise-separable convolution
    - 5 x 5 Depthwise-separable convolution
    - 3×3 Atrous convolutions with rate 2
    - 5×5 Atrous convolutions with rate 2
    - 3 x 3 Average pooling
    - 3 x 3 Max pooling
    - Skip connection
    - Null (zero) operation

  They apply a ReLU activation function before and batch normalization after every convolutional operator and separable convolutions are applied twice [31]

Despite the search space being derived from a discrete set of operations, DARTS introduces continuity by allowing the softmax over all possible operations to replace the categorical selection of a particular operation, as indicated by equation 2.6

$$o^{-(i,j)}(x) = \sum_{o \in O} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})} o(x) \tag{2.6}$$

The mixing weights for operations between a pair of nodes (i, j) are characterized by a vector $\alpha^{(i,j)}$, the dimension of which equals |O|. The architecture search process in DARTS is thus translated into the learning of continuous variables $\alpha = \alpha(i, j)$.

Post the search process, a discrete architecture is retrieved by substituting each mixed operation $o^{-(i,j)}$ with the operation of highest probability, expressed as $o^{(i,j)} = argmax_{o \epsilon O} \alpha_o^{(i,j)}$. In this context, α serves as the encoding of the architecture, its cardinality (length) being equivalent to the number of operators O.

At the conclusion of the search, a discrete architecture can be derived by replacing each mixed operation $\bar{o}^{(i,j)}$ with the most probable operation, providing a link back to the original discrete search space [31]. This process results in a final architecture that can be efficiently re-trained from scratch.

Architecture Search with DARTS

The DARTS search process integrates the learning of the operations α and the network weights w in a bi-level optimization framework, which can be formulated as:

$$\min_{\alpha \in \mathcal{A}} \mathcal{L}_{val}(w^*(\alpha), \alpha) \tag{2.7}$$

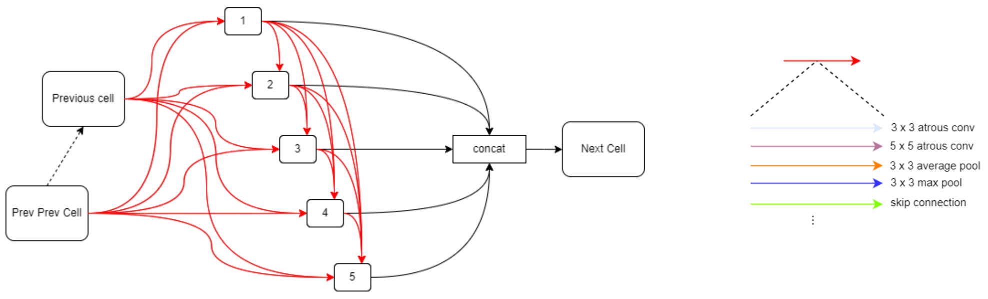


Figure 2.10: The DARTS search cell. Each red line represents the operations that can be assigned to the edge.

$$\text{such that } w^*(\alpha) = \arg\min_{w} \mathcal{L}_{train}(w, \alpha) \tag{2.8}$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the training and validation loss, respectively. The operator weights α are updated using the validation set while the weights of the network w are updated using the training set.

The gradient-based one-shot model employed by DARTS enables efficient architecture search. Like other NAS methods in the one-shot category, model weights do not need to be trained from scratch for every new architecture but are drawn from a supernet encompassing all possible architectures in the search space. The architectural parameters of this supernet can then be efficiently optimized using gradient-based methods. As evaluating the gradient for α is very expensive due to the inner optimization, the algorithm employs the following scheme to approximate $w^*(\alpha)$ in a single training step:

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha) \approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha) \tag{2.9}$$

The algorithm proceeds as follows:

Algorithm 2 DARTS Algorithm

```
Initialize weights $w$ and architecture parameters $\alpha$
repeat
  Update weights $w$ by optimizing $\mathcal{L}_{train}(w, \alpha)$ with respect to $w$
  Update architecture parameters $\alpha$ by optimizing $\mathcal{L}_{val}(w, \alpha)$ with respect to $\alpha$
until convergence or stop criterion is reached
Derive the final architecture by selecting the operation with the highest weight for each edge in the DAG.
```

This approach dramatically reduces the computational cost of the architecture search process, making it feasible to perform NAS on tasks with larger datasets and more complex architectures.

Results from DARTS

In the original DARTS paper [31], Liu et al. provided empirical evidence of DARTS's efficiency and effectiveness. They conducted experiments on CIFAR-10, Penn Treebank (PTB) [58], and ImageNet datasets, comparing the performance of the architectures found by DARTS with other state-of-the-art models. The architectures obtained using DARTS achieved competitive performance, often exceeding the performance of human-designed architectures.

Compared to other NAS methods, DARTS stands out regarding the computational cost. While conventional NAS algorithms often require thousands of GPU days to complete the architecture search [29, 59] DARTS achieves similar performance levels with only a fraction of the computational resources. This methodology has inspired a whole suite of follow-up work and made one-shot methods one of the most popular families of NAS search strategies [9].

2.7.5 Auto-DeepLab

Auto-DeepLab (ADL) is a tailored application of differentiable Neural Architecture Search (NAS) designed to effectively generate architectures for semantic segmentation tasks with the DeepLab framework [30]. Proposed by Liu et al., it significantly extends the cell-based search space of DARTS [31] by introducing a hierarchical spatial resolution component into the architecture search process [60]. As a hallmark of the DeepLab family, the architecture search ends with an Atrous Spatial Pyramid Pooling (ASPP) module [24], though it only uses 3 branches instead of 5 [30]. Given the prominence of ADL in the experiments detailed in Chapter 4, understanding its structure and operation is crucial for this thesis.

The Search Space

The search space of ADL is characterized by two major components:

- Micro-level: Inner cell-based search space, inherited from DARTS
- Macro-level: Outer spatial search space, unique to ADL

For the micro-level, ADL employs the same gradient-based search methodology as DARTS to select suitable operators within the Directed Acyclic Graph (DAG) of a cell. The operator pool is the same as for DARTS. Refer to section 2.7.4 for a thorough description of the cell-based architecture search.

At the macro level, ADL consists of $\mathcal{L}$ cells, also called layers, that can have four different levels of depth $s \in S$ where $S = 4, 8, 16, 32$ depending on how much we downsample the original image (See Figure 2.11). It incorporates the spatial resolution of cell inputs into the optimization process. Each cell with an output tensor of spatial resolution s can receive an input tensor of resolution s, $2s$, or $s/2$. The selection is controlled by a parameter β, which smoothly interpolates the discrete choices, akin to the role of α in the micro search space. Consequently,

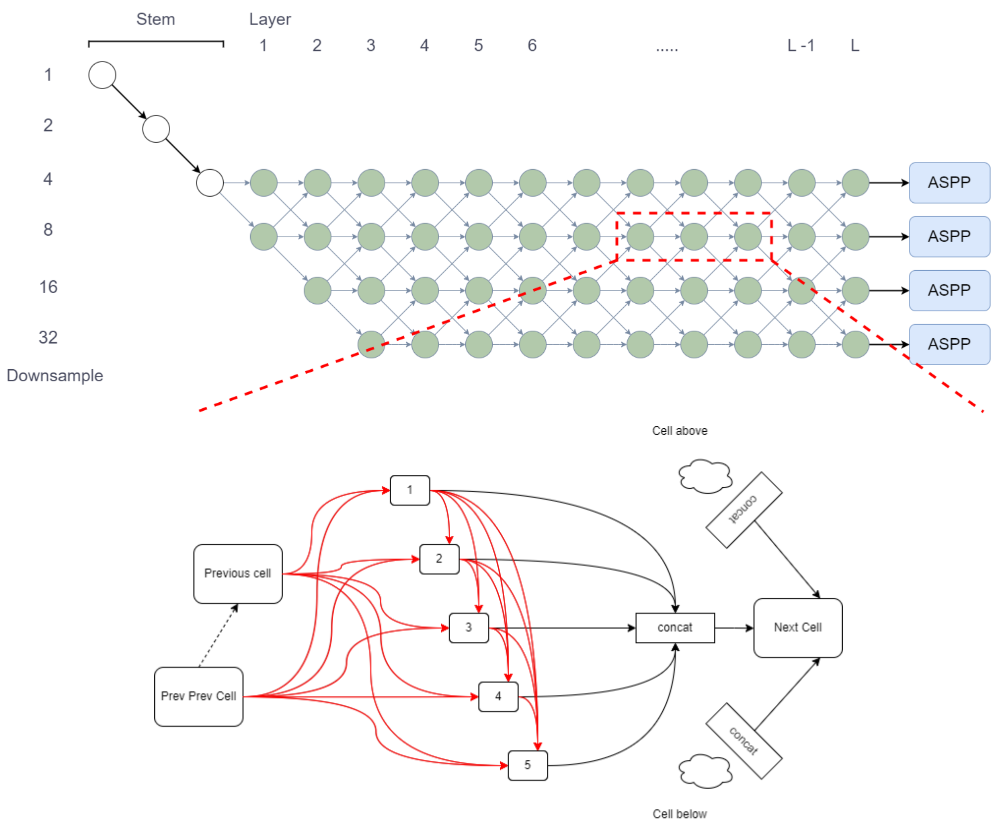


Figure 2.11: A visual representation of Auto-DeepLab's macro and cell level search space. This figure is inspired by a similar one made by Liu et al. [30]

both parameters are utilized in a unified continuous relaxation scheme as shown in Equation 2.10:

$$\begin{aligned} {}^{s}H_l =& \beta^l_{s/2s\rightarrow s}\text{Cell}({}^{s/2}H_{l-1}, {}^{s}H_{l-2}; \alpha) \\ &+ \beta^l_{s\rightarrow s}\text{Cell}({}^{s}H_{l-1}, {}^{s}H_{l-2}; \alpha) \\ &+ \beta^l_{2s\rightarrow s}\text{Cell}({}^{2s}H_{l-1}, {}^{s}H_{l-2}; \alpha) \end{aligned} \tag{2.10}$$

The number of filters per layer is given by $\mathcal{B}x\mathcal{F}xs/4$ where the filter multiplier $\mathcal{F}$ is a constant set at the beginning of the search and retraining. Liu et al. use $\mathcal{F} = 8$ in the search process.

Architecture Search with Auto-DeepLab

To avoid overfitting the architecture to the training data, the original paper divides the training data into two separate datasets, *trainA* and *trainB*. *trainA* is employed to optimize the weights, while *trainB* is used for fine tuning the architecture parameters α and scaling parameters β. Furthermore, they recommend training the weights for 20 epochs before initiating the optimization of α and β. The overarching steps of the Auto-DeepLab algorithm are encapsulated as follows:

Algorithm 3 Auto-DeepLab Algorithm

```
Initialize weights $w$, architecture parameters $\alpha$, and scaling parameters $\beta$
repeat
    Update weights $w$ by minimizing $\mathcal{L}_{trainA}(w, \alpha, \beta)$
    Update architecture parameters $\alpha$ and scaling parameters $\beta$ by minimizing $\mathcal{L}_{trainB}(w, \alpha, \beta)$
until convergence or a predetermined stop criterion is met
Derive the final architecture by selecting the operation with the highest weight for each edge in the DAG. And derive the final macro architecture using the Viterbi algorithm.
```

Retraining Auto-DeepLab

The architecture exploration phase with Auto-DeepLab is a proxy task, given that it generally utilizes fewer convolutional filters per layer during this search process compared to the subsequent retraining of the derived architecture [60]. Once the search concludes the final architecture is decoded. The cell is decoded the same way as described in section 2.7.4 and the macro architecture is decoded using the Viterbi algorithm. the model parameters are reset and the process of retraining begins anew. Liu et al. differentiate between Auto-DeepLab-S, M, or L in the retraining phase depending on the increase of $\mathcal{F}$ to 20, 32, or 48 respectively.

Chapter 3

Methodology

This chapter meticulously delineates the methodology and the suite of technological assets that underpin the experiments presented in this thesis. It aims to shed light on the rationale behind specific technological decisions, grounded in the context of contemporary leading-edge research, the available resources of Enernite, and learnings drawn from both our prior research endeavors and challenges identified within the broader scholarly community.

3.1 Selecting NAS Methodology

Determining the appropriate Neural Architecture Search (NAS) methodology, as discussed in section 2.7, hinges on several factors. For our purposes, four criteria emerged as critical: the computational expense associated with the search, the task specificity, the documented performance of the algorithm, and the availability of source code or libraries for implementing the chosen method.

Our analysis led us towards one-shot models, primarily due to their pronounced computational efficiency [9]. Within the spectrum of one-shot methodologies outlined in the comprehensive NAS surveys by White et al. [9] and Elsken et al. [60], Auto-DeepLab (ADL) appeared as the optimal choice. Its specialization for semantic segmentation, coupled with our prior successful experiences with DeepLab models contribute to the validation of this selection.

Although AutoDeepLab was introduced in 2019, various works have since emerged to expand upon it, with alterations to the search space or specific tailoring for tasks such as real-time video segmentation [18, 61, 62]. Among these, DCNAS by Zhang et al. [40] is the sole methodology that directly enhances the performance on inference measured in mIoU of ADL as shown in table 3.1. Regrettably, the lack of public access to the DCNAS code impedes its experimental usage. It also has almost double the search time, which would make it challenging to use with our dataset

Given these limitations and considering that inference latency is not a significant metric for our work, we opted for the highest-performing model akin to ADL, as of February 2023, which happens to be the original. The selection of this model was also influenced by the availability of source code, simplifying its integration into our experimentation process.

Table 3.1: Comparison of one-shot NAS methods specializing in segmentation on the Cityscapes test set

Architecture	**GPU Days (search)**	**mIoU**
Auto-DeepLab [30]	**3**	**82.1**
DCNAS [40]	**5.6**	**84.3**
GAS [61]	6.7	73.5
SqueezeNAS [18]	14.6	75.54
FasterSeg [62]	2	71.5

3.2 Training Environment Description

3.2.1 Hardware Limitations

The majority of the experiments, which will be detailed in chapter 4, were conducted using hardware from the NTNU IDUN High-Performance Computing Cluster [63]. This included either an NVIDIA A100 GPU equipped with 40/80GB memory or an NVIDIA V100 GPU with 32GB memory. An NVIDIA RTX 3090 GPU[1] was also used for some tests.

3.2.2 Data Selection

The empirical foundation of this project is based on a proprietary dataset of Enernite, encompassing solar farms situated across Europe. This expansive dataset, consisting of 224x224 pixel chips from 12-band Sentinel-2 level-2A (l2a) images[2], contains more than 200,000 images with about a 50/50 split between positives (containing solar farms) and negatives. All the positives additionally have masks. A couple of thousand are manually drawn and the rest are sourced from previous Solis deployments, OpenStreetMap[3], or other sources with free available masks for solar farms. Note that while Sentinel-2 captures 13 bands as mentioned in section 2.5.1, band B10 is excluded from l2a as it is used to monitor the atmosphere rather than the ground. Some examples of Solis images can be seen in figure 3.1

Given the resource-intensive nature of architecture search and practical considerations concerning time, representative subsets of this dataset are employed during the architecture search process. Comprehensive details about dataset sizes

[1]https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/

[2]https://sentinels.copernicus.eu/web/sentinel/sentinel-data-access/sentinel-products/sentinel-2-data-products/collection-1-level-2a

[3]https://wiki.openstreetmap.org/wiki/Tag:generator:source%3Dsolar

are expanded upon in chapter 4. It is worth noting that while the dataset is proprietary and cannot be shared in its entirety due to business considerations, the framework presented is agnostic to the dataset and could potentially be employed with a similar dataset. One such dataset is that collated and shared by Kruitwagen et al.[4].

As highlighted by Elsken et al. [8], the scale of disparity between the sampled and full dataset size can influence the relative ranking of architectures. This presents a potential concern, given that our final objective is optimizing the validation score on the larger dataset, not the subset. Nonetheless, the two tasks are indisputably closely related, and we believe that a random selection of images from a wide geographic coverage incorporating diverse geographical features will mitigate potential biases.

Furthermore, it is worth noting the successful model performances achieved on relatively smaller datasets (1000-2000 images) as reported by Hou et al. [6] and Plakman et al. [4]. While this doesn't directly speak to the relative ranking among models, it suggests that good results can be obtained even with smaller datasets, which may result in lesser impact from a reduced training set. This observation is particularly pertinent for SolarNet [6], considering China's diverse landscape.

To further diversify the training process, both during the search and retraining phases, we implement data augmentation. Specifically, images are subjected to horizontal and vertical flips with a 50% probability each before being fed into the model within the training loop. This data augmentation strategy makes for a robust and varied training dataset, enhancing the model's generalization capabilities even with smaller dataset sizes.

3.3 AutoDeepLab Search Implementation Details

This study closely aligns with the methodologies outlined in AutoDeepLab concerning the search space, strategy, and performance evaluation [30]. Notably, the original model of the paper is written in TensorFlow[5], while Enernite's Solis Oculus is developed using PyTorch[6]. Consequently, our implementation is based on the open-source PyTorch AutoDeepLab repository maintained by GitHub user NoamRosenberg[7].

We have integrated the Solis data loaders and pre-processing into this repository, further optimizing the training pipeline with enhancements to memory usage, code readability, checkpointing, and model monitoring. The code is available at `https://github.com/eolweus/autodeeplab`.

[4]`https://zenodo.org/record/5005868`
[5]`https://www.tensorflow.org/`
[6]`https://pytorch.org/`
[7]`https://github.com/NoamRosenberg/autodeeplab`

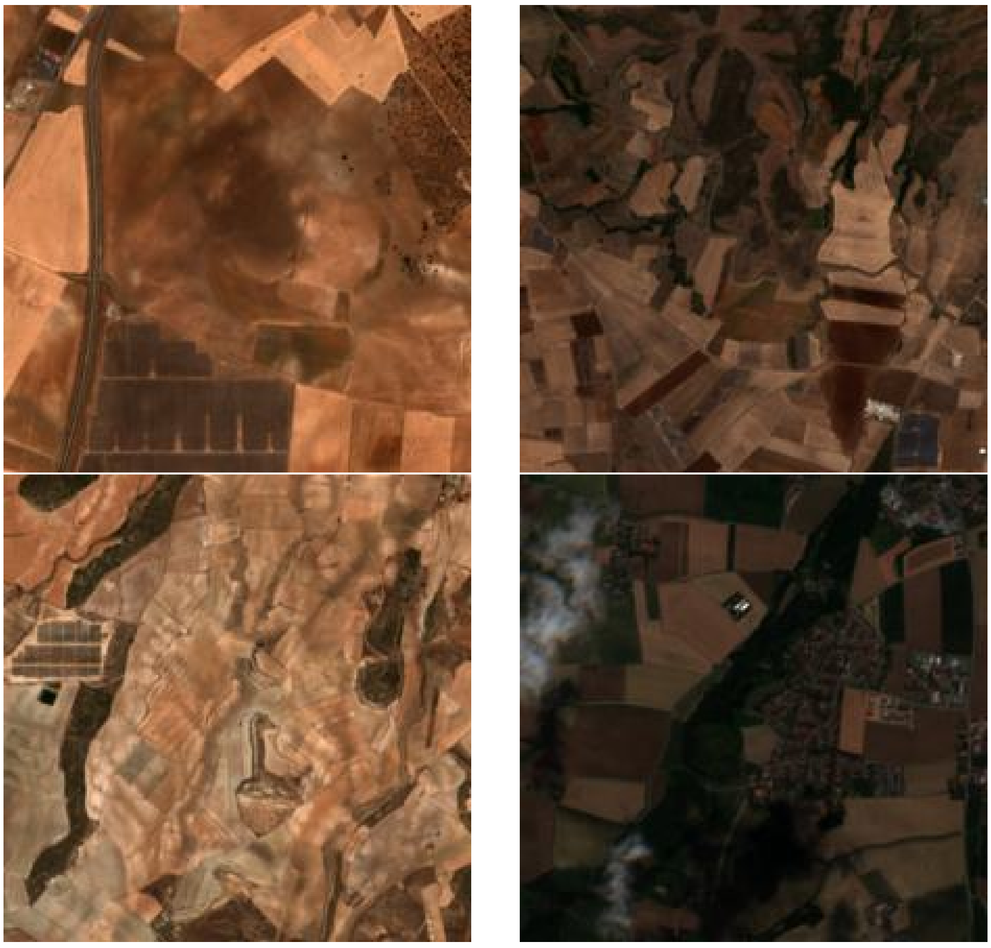


Figure 3.1: Some examples of images from the Solis dataset

3.3.1 Parameter Selection Details

Our experimental setup closely follows the methodology employed by Liu et al. [30], based on their thorough testing. We adopted their parameter settings, including:

- Layers $\mathcal{L} = 12$
- Blocks per cell $\mathcal{B} = 5$
- Filter multiplier $\mathcal{F} = 8$
- Epoch count during the search phase = 40
- Number of epochs allocated for weights w optimization before initiating α, β optimization = 20
- Utilization of ASPP modules, with a modification to use 3 branches instead of the usual 5 during the search phase
- Hyperparameters pertaining to the optimizers of w, α, β

Nevertheless, given our access to A100 GPUs via the IDUN cluster [63], we employed batch sizes of 22 or 12, dictated by the available memory capacity (80 or 40 GB). The Solis dataset, which is purposed for both classification and segmentation, maintains a standard image resolution of $224x224$—a common format for classification input [18].

The Solis dataset subsets, as detailed in Section 3.2.2, were used during the architecture search. These subsets were divided into two equally sized training sets, *trainA* and *trainB*, along with a validation set comprising 20% of each training set's size, following the procedure in Section 2.7.5.

The retraining process for the discovered architectures remained consistent across all experiments. We applied the Auto-DeepLab-M configuration ($\mathcal{F} = 32$, see Section 2.7.5) and used the entire Solis dataset with an 80/20 train-test split, incorporating 12 out of the 13 available spectral bands.

The only exception was the best-performing model identified via NAS, termed 10k (refer to Section 4.3), which was retrained with a filter multiplier of $\mathcal{F} = 48$, representing the Auto-DeepLab-L configuration. To avert confusion, we refer to this model as **10k-L**.

For retraining, we opted for a 100-epoch duration as prior experimentation with the Solis-transfer model indicated that the model performance plateaued after this point and was more prone to overfitting. Training for extended periods would not only increase the likelihood of overfitting but also limit our capacity to conduct diverse parameter experiments. This epoch count aligns with the approach adopted by Liu et al. [30], further validating our choice.

3.4 Comparing NAS Results

In this research, we embark on a journey to critically evaluate and compare the outcomes of Neural Architecture Search (NAS) experiments. Our focus will be on the differences in the architectures generated using the Auto-DeepLab method, particularly on how well the search performs or the same task with varying input data and thus also how these variations to input impact the search. Additionally, we aim to investigate the trade-offs of using NAS versus an off-the-shelf model like ResNet.

To unravel the intricacies of the generated architectures and facilitate a comprehensive comparison, we will adopt a two-pronged approach encompassing both hierarchical and cell-level analysis. The hierarchical perspective will compare the macro-architectural organization of the networks, whereas the cell-level scrutiny will focus on the finer cell-level structure.

In our quest to answer these questions, we will probe into various dimensions of the architectures, including:

- **The Performance During Search**: Can we see a difference as to how well Auto-DeepLab performs during search depending on the input data?
- **The Chosen Operations:** Is there a pattern in the operators chosen during the different searches?
- **Macro Architecture:** Does the down and up-sampling differ significantly with various run configurations?

Ultimately, through this examination, we aim to cultivate a better understanding of the architectural nuances that emerge from NAS experimentation. This understanding will, in turn, inform our perspective on the robustness and flexibility of NAS, and most of all how consistent the results of NAS are across slight variations of parameters and inputs.

3.5 Performance Evaluation of the Final Models

The assessment of the final models' performance is more straightforward than evaluating the outcomes of the architecture search. For this comparison, we train the chosen models on the complete Solis dataset, adhering to an 80/20 train-test split. We then compare their performance based on F-score, mIoU, and the rate of convergence. Additionally, we juxtapose these models with several benchmarks:

- The existing Solis Oculus model (Solis-transfer)
- The new Solis Oculus model, Solis-seg (see 4.2)
- The best-performing model identified by Liu et al. [30] during their Cityscapes search. We refer to this model as **ADL-cs**
- A model randomly generated with the help of ChatGPT[8] within our search space. We refer to this model as **chatgpt**. A detailed description of how it was generated can be found in Appendix C.

It is crucial to note that our primary performance metric is the validation set mIoU (mean intersection over union, section 2.3.2). Owing to its widespread adoption and intuitiveness, it is an effective tool for evaluation. However, an excep-

[8]https://openai.com/blog/chatgpt

tion is made when contrasting with the Solis-transfer model, for which we employ the F1-score, given that mIoU was not captured during its training.

As a final evaluation, we will deploy the best-performing model on an area it has not been trained on: the state of New York in the US, and see if it is able to detect any new solar farms. This will also provide valuable insight into how the model performs on a continent it has not been trained on.

Chapter 4

Experiments and Results

This chapter forms the core of our research, presenting the comprehensive suite of experiments conducted to address our research goal and study objectives (section 1.2). Each experiment is designed to directly interrogate a facet of the research goal. Our experimental design was driven by the intention of not only refining our understanding of the architecture search process, but also to evaluate the performance of the resulting architectures and, ultimately, to unearth a model that outperforms the Solis-transfer model.

This chapter contains figures of some of the cells discovered through NAS. All the cells can be found in Appendix B.

4.1 Experimental Plan

Our experimental design directly corresponds to our study objectives, with each experiment intended to provide insights that contribute towards our overarching research goal and the study objectives. The experiments are structured as follows:

- **Experiment 1: Evaluating the Effectiveness of Transfer Learning (SO1 and SO3)** This experiment targets our first study objective, seeking to evaluate whether a model trained purely for semantic segmentation will outperform a model with a backbone trained on classification tasks. This evaluation also provides an additional benchmark against which the performance of NAS-derived architectures can be compared.
- **Experiment 2: Assessing the Impact of Dataset Size on NAS (SO2.1)** Recognizing the constraints of computational resources and the necessity to work with subsets of our dataset, this experiment seeks to understand how varying dataset sizes influence the NAS process.
- **Experiment 3: Robustness of Relative Rankings in NAS (SO2.2)** Conducting searches on a reduced dataset could potentially distort the relative ranking of architectures determined by the search algorithm. This experiment aims to evaluate this potential distortion by comparing high-ranked architectures with those presumed to be inferior on the subset using

the full dataset.

- **Experiment 4: Influence of Spectral Bands on NAS (SO2.1)**
  While it is desirable that NAS methods generalize well across different datasets and tasks, it isn't guaranteed. As the Solis project currently benefits from the usage of spectral bands, we were interested in evaluating how the incorporation of this additional data might influence the search process.
- **Experiment 5: Overfitting Risk with Auto-DeepLab (SO2.2)**
  In line with Liu et al.'s recommendation [30] we use a split training set to reduce the risk of overfitting to the training dataset. With this experiment we seek to measure the extent of overfitting risk by deliberately deviating from this split-training approach, allowing us to assess the repercussions of not adopting such preventative measures in NAS.
- **Experiment 6: Sensitivity of NAS to Different Data Subsets (SO2.1)**
  This experiment probes the robustness of our searches to changes in the data subset used, providing further insights into the generalization capabilities of NAS.
- **Experiment 7: Comparative Evaluation against a Broadly Applicable State-of-the-art Model (SO3)**
  This experiment juxtaposes our best model against SAM (see section 2.3.1), a state-of-the-art segmentation model designed with a broad focus, aiming to excel across a wide array of tasks. This comparison allows us to better understand the trade-offs between specifically optimized models as opposed to the novel approach of designing models for broad, generalized performance.
- **Deploying the Best Model to Find New Solar Farms**
  As a final experiment, we planned to test the ability of our best model to augment our existing Solis dataset by deploying it in a real-world context for the discovery of new solar farms.

The subsequent sections provide a detailed description of each experiment and present their respective results.

4.2 Experiment 1: Evaluating the Effectiveness of Transfer Learning (SO1)

As outlined in our study objectives, the purpose of this experiment is to evaluate the effectiveness of transfer learning, particularly as employed by the current Solis model, Solis-transfer, discussed in Section 2.6.2. Our intention is to investigate if the prevalent approach of transfer learning from classification tasks remains the optimal strategy or if training directly on segmentation tasks from the outset can produce improved outcomes. To facilitate this analysis, we implemented a variant of the Solis model, Solis-seg, trained exclusively on segmentation.

Contrary to our expectations, not only did the Solis-seg model exhibit a marked performance improvement compared to Solis-transfer by increasing the best F1-

score from 0.89 to 0.9621, it even ascended to the position of the highest-performing model. With an impressive final validation mIoU of 0.9629, it surpassed all the models obtained through our NAS experiments, emerging as the only model breaching the 0.96 threshold.

Table 4.1 provides a summary of the top five models, ranked based on the mIoU scores achieved during the retraining phase. It underscores the dominance of Solis-seg in this experiment.

Name	**mIoU**	**F1-score**
Solis-seg	0.9629	0.9621
10k-L	0.9593	0.9582
ADL-cs	0.9586	0.9575
10k	0.9567	0.9555
chatgpt	0.9565	0.9552
Solis-transfer	N/A	0.89

Table 4.1: Top 5 models ranked by validation mIoU achieved during retraining

4.3 Experiment 2: Assessing the Impact of Dataset Size on NAS (SO2.1)

One of the constraints that arose from the extensive computational demands of NAS was the necessity to reduce the number of images employed during the search process. This constraint led us to an essential research question: How would conducting a search on a subset of the data influence the architecture outcome? Although running the algorithm on both a subset and the full dataset would be the optimal way to investigate this, practical constraints of time and resources rendered this approach impossible.

Consequently, we decided to experiment with data subsets of varying sizes, specifically 2,000, 5,000, 10,000, and 20,000 images, comparing the search processes and the resulting architectures. We subsequently refer to these searches and the resulting architectures as 2k, 5k, 10k, and 20k respectively. We hypothesized that these subsets would serve as satisfactory proxies for the full dataset, given the problem's inherent similarity across different scales. Our anticipation was that the architectures discovered with various dataset sizes would demonstrate consistency in both their structural configuration and performance. The numbers specified denote the combined count of training and testing images. Thus, the experiments conducted with 10,000 images, for instance, had training sets of 8,000 images (meaning *trainA* and *trainB* had 8000 each) and a test set comprising 2,000 images. We chose this configuration because both the architectural parameters and the weights are then trained and tested on a total of 10,000 images respectively. Detailed specifications of the dataset sizes used are available in appendix A.

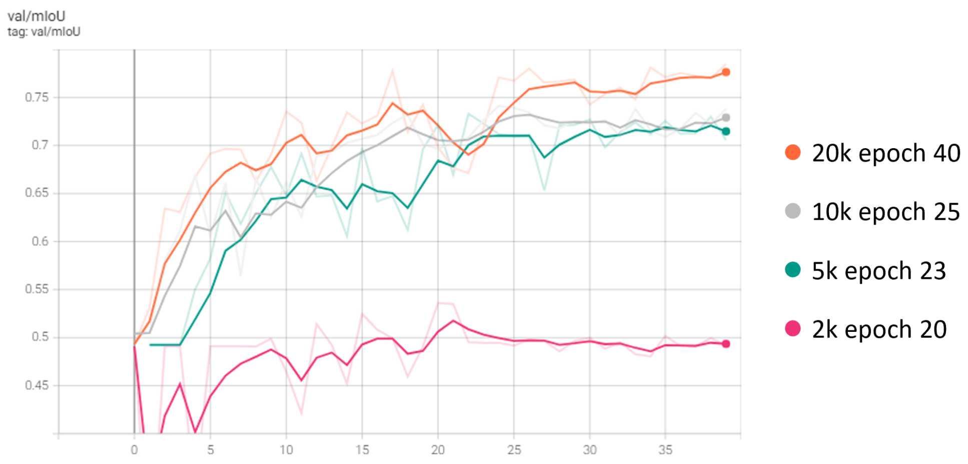


Figure 4.1: The mIoU on the validation sets for the different dataset sizes during search

4.3.1 Search Results

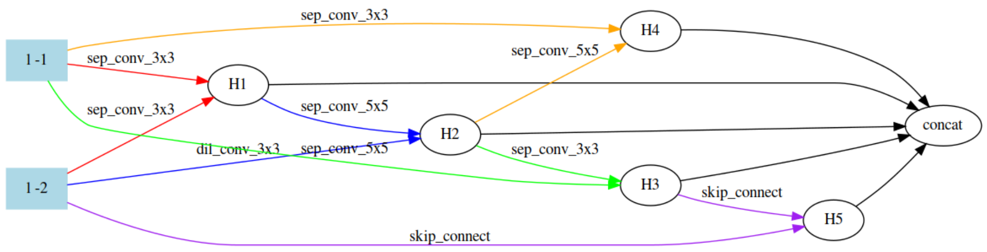
As anticipated, there is a direct correlation between the size of the dataset and the resultant validation mIoU. While the performance difference during the search across the three largest datasets is marginal, using only 2000 images presented a broader range of results. Specifically, the results of the searches conducted with this small dataset appear to exhibit more sensitivity to randomness and possibly data selection during the search phase as we will see in section 4.7.

Interestingly, almost all the searches reached their peak performance shortly after the commencement of architectural parameter optimization, which began following the initial 20 epochs of training. This pattern led us to further investigate by comparing some of the top-performing architectures with the final ones discovered during the search, as detailed in Section 4.4.

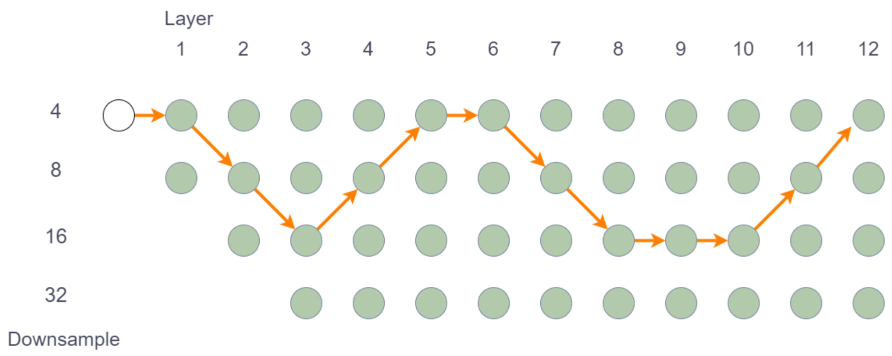
Figure 4.2 showcases a selection of architectures identified during various search processes. It's important to clarify that *l-1* and *l-2* represent the outputs from the immediate preceding cell and the one prior to it, respectively. For a more detailed interpretation of this configuration, readers are advised to refer to Section 2.7.4. Upon examination of Figures 4.2 and B, one can observe a marked variance in cell structures, despite them yielding similar results on the extensive dataset. This observation contradicts our initial presumption that the optimally identified architectures in the search processes would exhibit structural similarities.

4.3.2 Retraining Results

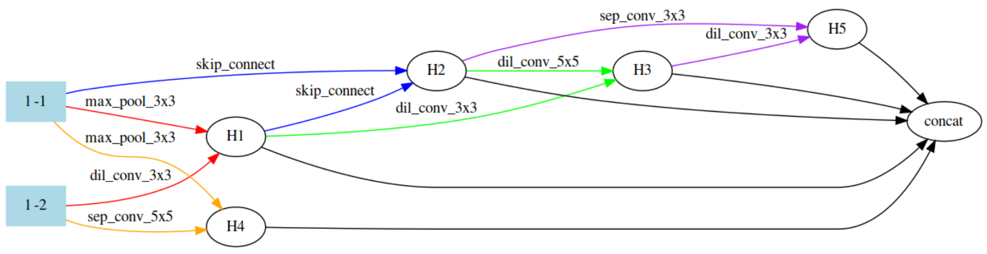
As might be anticipated, models trained with smaller datasets manifested lower performance on unobserved data during the search phase. However, as presented in Table 4.2, this performance discrepancy does not translate directly to the performance of the architecture when implemented on the full dataset. This di-



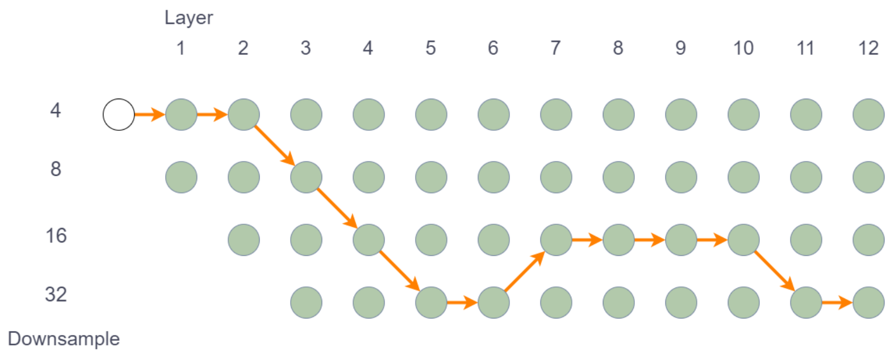
(a) 20k cell



(b) 20k Macro Path



(c) 10k cell



(d) 10k Macro Path

Figure 4.2: Some of the architectures produced by searching different dataset sizes

vergence occurs because, during the search, the models are ranked relative to the data they are trained on, as opposed to any absolute scale.

In a fascinating turn of events, the model trained on the most extensive dataset demonstrated the lowest performance. Our results didn't indicate a discernible correlation between the size of the subset and the final dataset performance, with all scores in close proximity. Furthermore, none of the models displayed significant overfitting, with the training mIoUs marginally exceeding those on validation. This outcome could be attributed to the element of randomness due to the restricted number of searches conducted. Conversely, it could signify that the dataset is quite homogeneous, and even the smaller subsets accurately represent the larger dataset. As explored in section 2.5.1, working with satellite imagery presents its unique challenges. Thus, it's plausible that the 20k dataset, due to some unfortunate circumstances, contains flawed data that impedes the optimization process.

Name	val mIoU (search)	val mIoU (retrain)	train mIoU (retrain)
10k	0.741	0.9567	0.9653
2k	0.536	0.9563	0.9637
5k	0.733	0.9550	0.9630
20k	0.785	0.9531	0.9607

Table 4.2: mIoU Results for Different Dataset Sizes

4.4 Experiment 3: Robustness of Relative Rankings in NAS (SO2.2)

A primary concern when utilizing a subset of a dataset for the purpose of architecture search is the risk that this subset may not adequately represent the full dataset. This concern hinges on the potential alteration in the relative ranking of architectures when moving from the subset to the full dataset [8]. Although thoroughly assessing this mismatch in rankings would necessitate a comprehensive evaluation of all or a significant number of architectures on both the subset and full dataset, we can gauge the severity of this issue by contrasting some of the architectures evaluated during the searches. Consequently, we focus on comparing the top-performing results for three dataset sizes against the final models—or the second-best models in the case of 20k—derived from the same search.

4.4.1 Search Results

A key observation during training is that most architectures maintain relative similarity across different stages. For instance, the second-best model identified within the 20k subset (see Figure 4.3) exhibits strong resemblances with the top model (see Figure 4.2a).

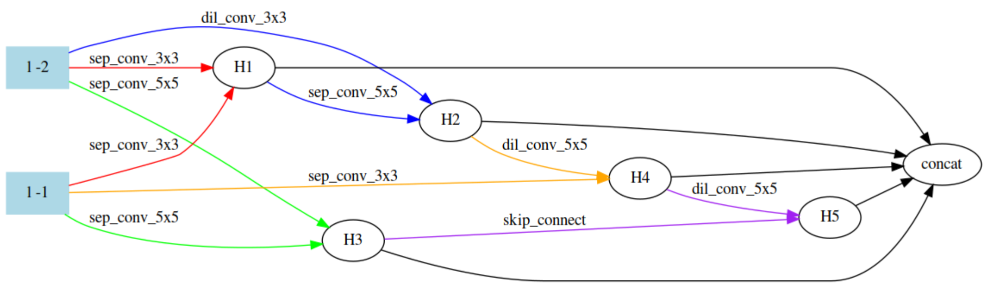


Figure 4.3: The second-best cell identified in the 20k subset

4.4.2 Retraining Results

As delineated in Table 4.3, in all instances we scrutinized, the top-performing model demonstrated at least marginal superiority over the final or second-best models. It is noteworthy that the size of the mIoU gap in the final results appears to have a loose correlation with the difference in mIoU during the search phase.

Model	mIoU (search)	mIoU (retrain)
20k best	0.7848	**0.953**
20k second best	0.781	0.9525
Difference	0.0038	0.0005
10k best	0.741	**0.9567**
10k final	0.7381	0.9553
Difference	0.0029	0.0014
5k best	0.733	**0.955**
5k final	0.7056	0.9394
Difference	0.0274	0.0156

Table 4.3: omparison of mIoU of the top-performing model and another model during search and retraining across three different dataset sizes

4.5 Experiment 4: Influence of Spectral Bands on NAS (SO2.1)

Historically, research initiatives [3, 4] as well as our hands-on experience with training the Solis model have suggested that leveraging the spectral bands available in Sentinel-2 imagery has the potential to significantly bolster the performance of visual tasks. Nevertheless, the influence of integrating additional spectral bands on the performance of NAS models, which are typically oriented around RGB images, remains inadequately explored.

To bridge this knowledge gap, we undertook a study to ascertain the potential influence of utilizing a wider spectrum of bands on the performance of the derived models. This involved comparing the performance of a model derived using the 12 bands available in the Sentinel-2 l2a imagery with a model trained solely on RGB bands. In both cases, a dataset comprising 10,000 images was utilized. A future investigation could examine the impact of different combinations of bands on the search, although such an approach was beyond the scope of this study. The insights from this exploration will help clarify the effect of spectral bands on the performance and utility of the resultant models.

4.5.1 Search Results

Predictably, the model trained with more data—comprising 12 bands versus 3—demonstrated superior performance on the validation set during the search. This outcome aligns with previous research underscoring the benefits of integrating the bands in the training process. It also intuitively supports the assumption that a greater volume of relevant data should yield superior results. The cells generated by the searches are quite distinct, as illustrated in Figure 4.5 and Figure 4.2c.

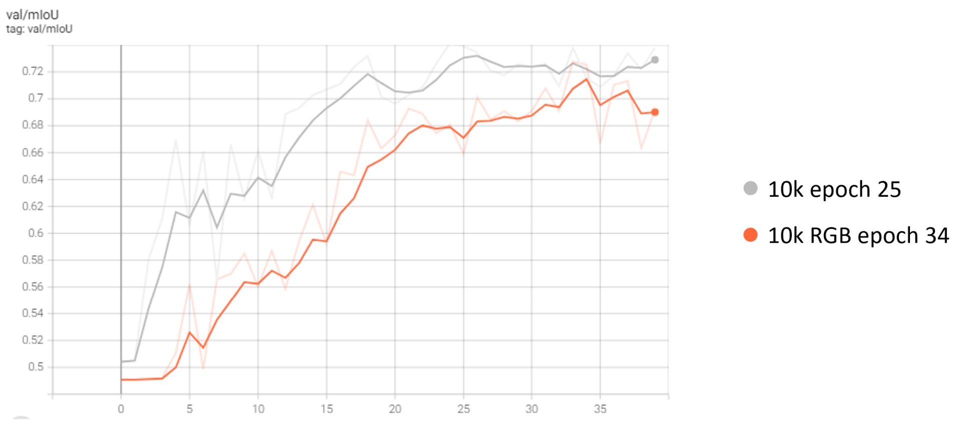


Figure 4.4: Comparative mIoU on the validation set for the search utilizing only RGB and 12 spectral bands respectively

4.5.2 Retraining Results

The retraining results, as depicted in Figure 4.6, provide compelling evidence that the architecture identified using all bands outperforms the one developed using only RGB. This outcome suggests that Auto-DeepLab is capable of effectively utilizing the additional data present in the extra bands to fine-tune the architecture to the dataset, further reinforcing the value of incorporating broader spectral information.

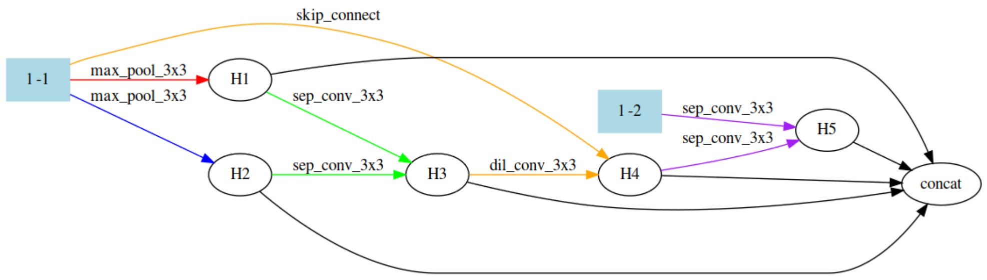


Figure 4.5: The cell resulting from the architecture search using 10k images and only the RGB bands of the images

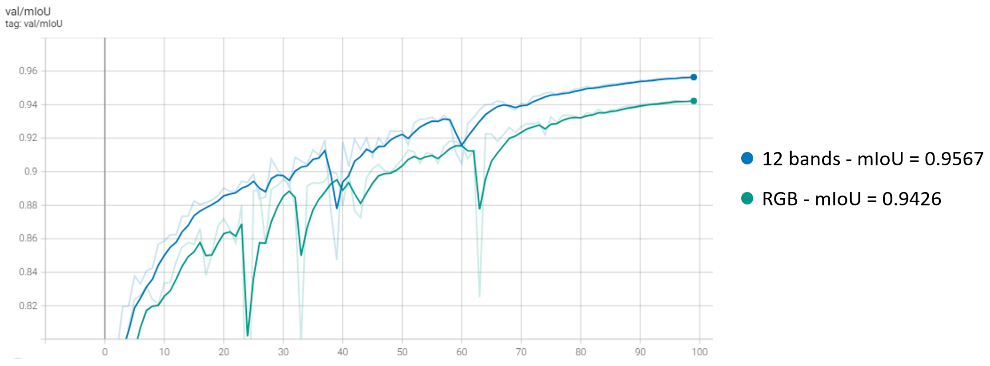


Figure 4.6: Comparative mIoU on the validation set for the architectures found searching with RGB and 12 spectral bands respectively

In summary, these results underscore the significant role that the integration of additional spectral bands, and likely other relevant data, can play in improving the performance of NAS models. This is particularly evident when models are trained with larger datasets that incorporate more diverse and richer spectral information. This finding could have profound implications for the development of future models, particularly those designed for tasks involving satellite imagery or other data sources that provide a broad range of spectral data.

4.6 Experiment 5: Overfitting Risk with Auto-DeepLab (SO2.2)

In the initial phase of our experimentation, an oversight led us to run the initial experiments using identical images to optimize both the weights and architectural parameters. Recognizing the potential for learning from this discrepancy, we chose to preserve these results to draw comparisons with the corrected implementation of the training set split. We envisaged this could provide meaningful insights into the hypothesized overfitting phenomenon associated with Auto-DeepLab [30]. The dataset sizes investigated in this experiment comprised 10k and 2k. The searches and subsequent models without an A/B training set split will henceforth be referred to as 10k-no-ab and 2k-no-ab.

4.6.1 Search Results

Although it's challenging to pinpoint the exact cause, the best model of 10k-no-ab (Figure 4.8) exhibited a heavy reliance on skip connections, a problem that is prevalent with DARTS [64].

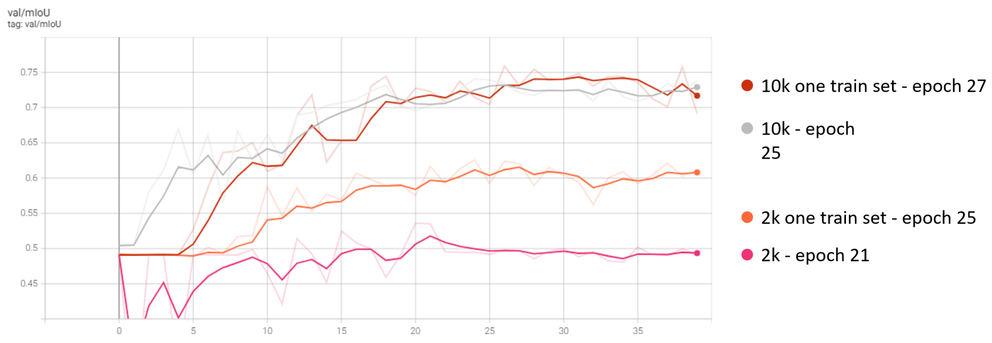


Figure 4.7: The mIoU on the validation sets for the various architectures

4.6.2 Retraining Results

Interestingly, as evidenced in Table 4.4, models trained on individual datasets performed better on the validation set during the search phase. However, these models displayed slightly lower performance during the retraining phase. Although these variations are minimal, they could potentially be ascribed to the inherent

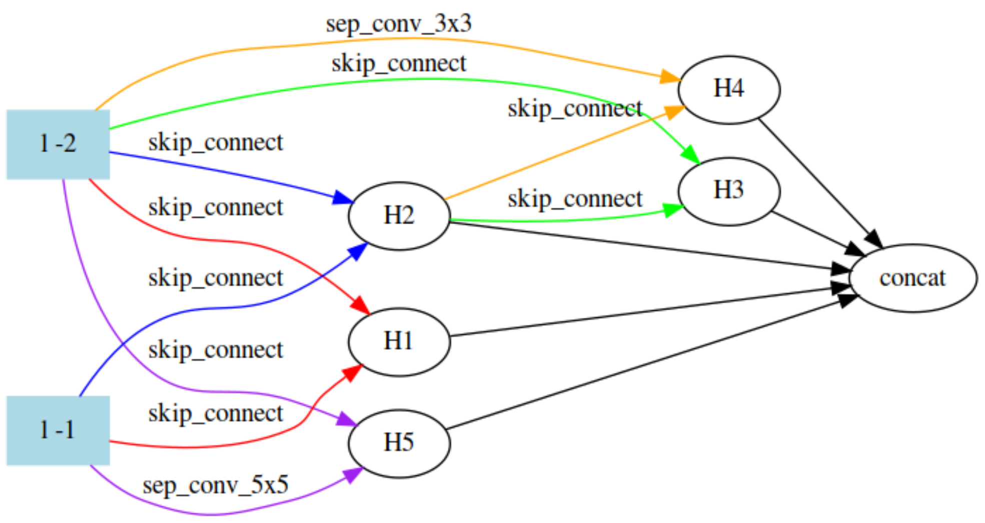


Figure 4.8: The cell discovered in 10k-no-ab

randomness in the search process, rather than indicative of a general trend. It's worth mentioning, however, that in instances with a smaller dataset where the alternative is to split it in two to circumvent this issue, it may be beneficial to evaluate the merits of utilizing the entire dataset during both optimization processes.

Model	mIoU (search)	mIoU (retrain)
10k	0.741	0.9567
10k-no-ab	0.759	0.9537
2k	0.536	0.9563
2k-no-ab	0.626	0.9547

Table 4.4: Comparison of mIoU for models found using one and two training datasets

4.7 Experiment 6: Sensitivity of NAS to Different Data Subsets (SO2.1)

For a more comprehensive understanding, we incorporated an additional subset for dataset sizes of 2k and 5k. This approach was taken to examine the influence of random subset selection on the performance of the discovered architectures when applied to the larger dataset. We selected the smallest dataset sizes of 2k and 5k under the assumption that these would be the most susceptible to potentially detrimental data that could undermine the search. The newly trained architectures, derived from differing subsets of data, are designated as 2k-dif and 5k-dif.

In terms of performance during the search, 2k-dif significantly outshone its counterpart, 2k. Conversely, the performance difference between 5k and 5k-dif was minimal and tipped in favor of 5k. However, the performance results upon retraining were strikingly similar for both pairings. These results, illustrated in Table 4.5, combined with the findings from Section 4.3, suggest that the data is relatively uniform. This implies the existence of numerous viable proxy subsets.

Model	mIoU (Search)	mIoU (Retraining)
5k	0.733	0.955
5k-dif	0.701	0.9559
2k	0.536	0.9563
2k-dif	0.6321	0.9548

Table 4.5: Comparative Assessment of Results from Searches using Diverse Data Subsets

4.8 Experiment 7: Comparative Evaluation against a Broadly Applicable State-of-the-art Model (SO3)

With the launch of Meta's remarkable Segment Anything Model (SAM) [25], we were intrigued to measure its performance against our best model, Solis-seg. Ideally, we would fine-tune SAM and compare the metrics with those of Solis-seg. However, as this exceeds the scope of our current study, we instead made use of the publicly accessible demo online[1].

For our comparison, we uploaded RGB images from the validation set, on which Solis-seg has not been trained, to SAM. We used the "segment everything" function to scrutinize the entire image for coherent structures. It's important to note that we have not provided SAM with any specifics about what it should identify, nor have we fed it any images for training. These results are purely zero-shot, with SAM, as its name suggests, striving to segment any identifiable unified structure in the image.

Three distinct outcomes emerge from this analysis. Notably, SAM's performance varies significantly across different images. In image A, where the solar farm is almost invisible to the naked eye, Solis-seg presumably gains an advantage through the use of spectral bands, as SAM fails to detect it entirely. In image B, SAM clearly distinguishes the solar farm from its surroundings, arguably drawing a more refined boundary than the ground truth. For image C, it not only identifies the solar farm but also segregates the various racks into individual partitions. However, these solar farms are relatively large, and many images depict smaller solar farms that blend into the environment and are very challenging to detect with the human eye. We suspect that a model trained solely on RGB might face increased difficulties with such images given Sentinel's resolution. While it might be

[1] `https://segment-anything.com/demo`

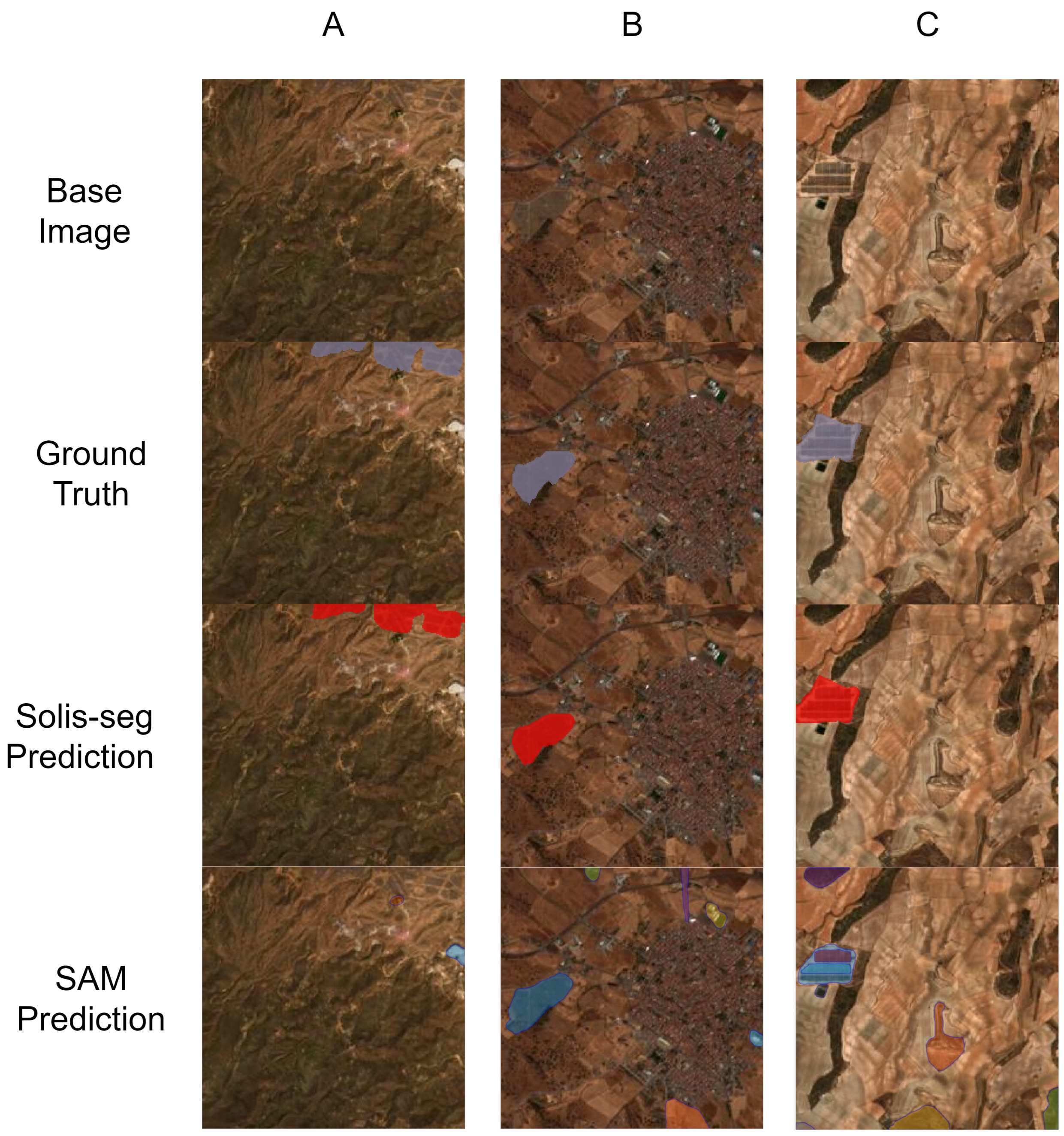


Figure 4.9: A side-by-side comparison of Solis-seg vs SAM

possible to fine-tune SAM with spectral bands, it is uncertain whether this would enhance its accuracy [23]. Therefore, despite the highly impressive performance on some of the images, this scenario might still favor a specialized model over a generalized zero-shot model in terms of discovering new facilities. An interesting approach for further work would be to combine SAM with a more specialized model to both optimize detection and obtain finer segmentations.

4.9 Experiment 8: Deploying the Best Model to Find New Solar Farms

In our concluding experiment, we deployed our best model, Solis-seg in a real-world scenario detecting new solar farms on satellite imagery. While not directly tied to any Study Objective, this experiment is an essential step in proving that our solution works on unseen data. We applied the model to Sentinel images of New York from 2022, where it identified 874 polygons, potentially indicating solar farm locations.

Since multiple polygons often constitute single facilities, we analyzed a random sample of 50 facilities. From this analysis, we estimated that each facility is composed of approximately 1.5 polygons on average, suggesting that we have identified around 583 potential solar farms. Notably, several of these do not feature in widely available datasets such as OpenStreetMap.

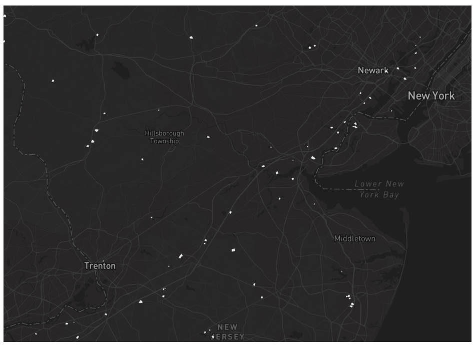
Figure 4.10 presents examples of detected solar farms and an overview of some of the detected facilities. While our sample review indicated minimal false positives, it should be noted that not all predictions may accurately represent solar farms, an issue we will revisit later.

Despite the model's effectiveness in detecting a significant number of solar farms, its performance was notably weaker on the New York imagery compared to the validation set's solar farms. A recurring pattern was the model only detecting partial solar farms in several images (figure 4.11a), and entirely overlooking some[2] (figure 4.11b). This discrepancy suggests the presence of unfamiliar elements or inherent differences in the data or landscape that the model has not previously encountered.

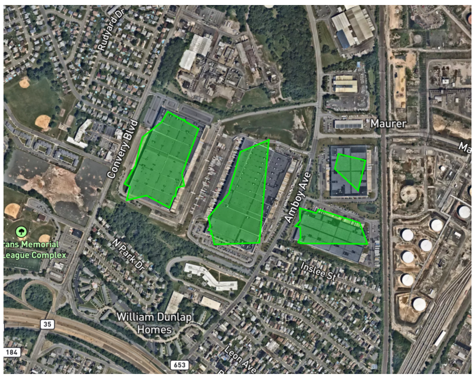
This highlights the critical role of diversity in training data for enhancing a model's ability to generalize. Although our model was trained using data from European solar farms, the landscapes, architectural styles, and environmental influences can vastly differ between regions and continents. These discrepancies can significantly impact the model's proficiency in accurately identifying solar farms in satellite images from different parts of the world, such as New York. This observation is congruent with the conclusions drawn by Layman et al. [10], further emphasizing the importance of data diversity for robust model performance.

Hence, to enhance the model's generalizability and adaptability to new continents and landscapes, it would likely be necessary to incorporate training data from

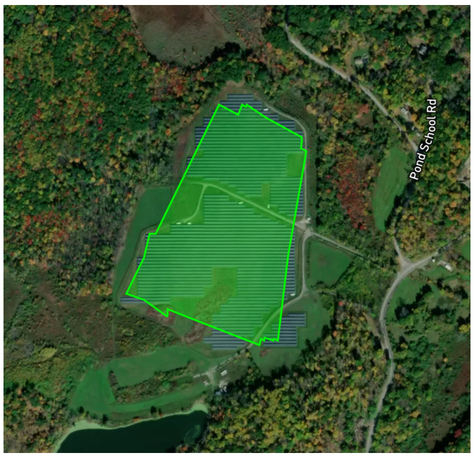
[2]We found some manually where they were close to a facility the model did detect



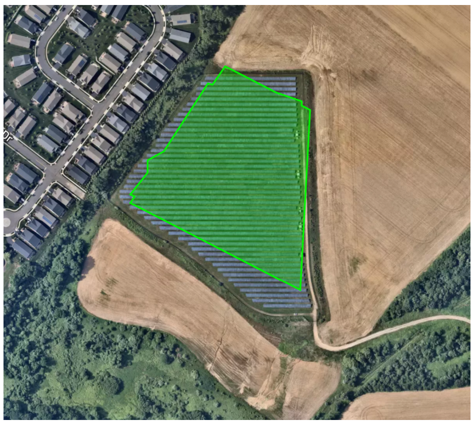
(a) An overview of an area in the state of New York where solar farms were detected



(b) Detection of a large rooftop solar installation

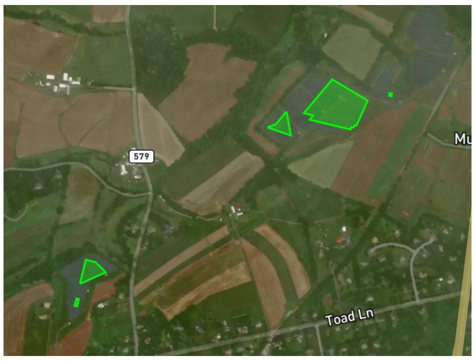


(c) Solar farm example 1

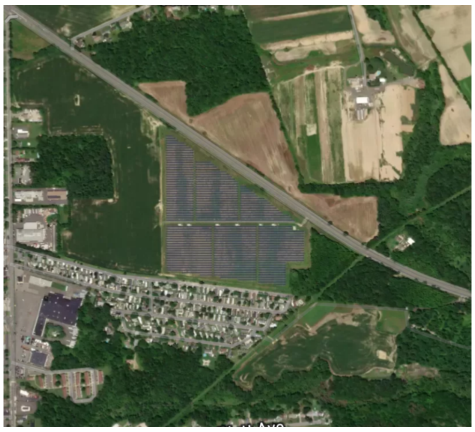


(d) Solar farm example 2

Figure 4.10: Examples of solar farms found by Solis-seg in New York State on higher resolution imagery from Maxar



(a) A solar farm the model only detected a very small part of



(b) A solar farm the model didn't detect

Figure 4.11: Examples of solar farms in New York Solis-seg struggled to detect shown on higher resolution imagery from Maxar

various geographic locations globally. This broader training scope would help the model adapt to diverse scenarios, improving its performance on unfamiliar landscapes and potentially uncovering new solar farms with higher precision.

Another significant insight from this experiment pertains to the trade-offs discussed in section 2.5. It often proves challenging to validate if a model's prediction in some Sentinel images actually represents a solar farm or a false positive, prompting us to resort to higher-resolution sources for manual verification. However, an accompanying issue is that higher-resolution images are often not as recent as the lower-resolution ones used for training. Figure 4.12 illustrates this discrepancy, where the high zoom level image is outdated and doesn't reflect the newly installed solar farm, while the image at a lower zoom level is recent enough to show it. Although this issue doesn't pose a problem in this particular instance (as the presence of a solar farm is clear), it becomes difficult to validate cases where the imagery isn't updated, and the solar installation is recent, as possibly shown in Figure 4.13. There are instances where detections are verifiable on lower-resolution imagery but are not present in any freely available high-resolution imagery.

It's important to note that although it would have been intriguing to compare the inference results of a greater number of models, the intricacies involved in creating a pipeline around a single architecture for such operations are quite substantial. As such, we consider ourselves fortunate that the model showing the best performance was the one most similar to the one we had previously developed. Implementing this experimental setup for the other models would have been a complex and time-consuming task. Nevertheless, exploring this direction remains a fascinating prospect for future work.

In conclusion, we enrich the publicly accessible body of solar farm datasets by sharing our dataset detailing solar farm locations detected by our Solis-seg

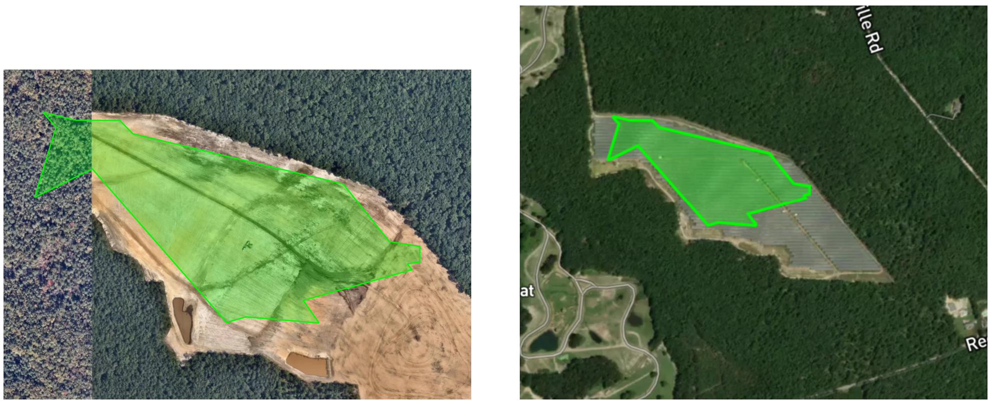
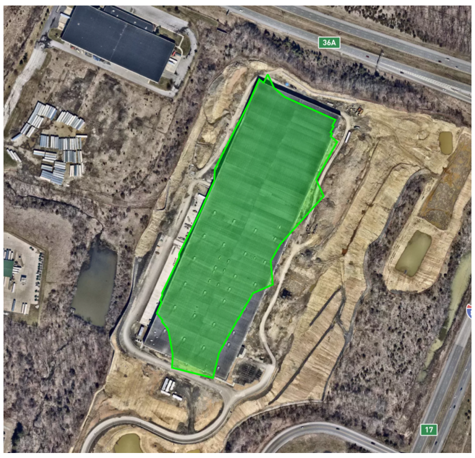
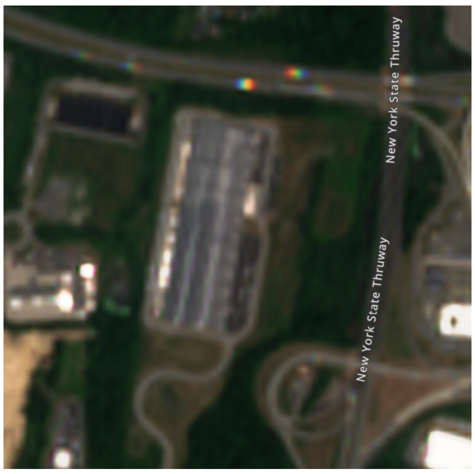


Figure 4.12: Demonstration of how more recent lower resolution satellite imagery can show new solar farms not present in older high-resolution imagery

model in New York. The information on these geometries can be found in the us_ny_preds.geojson file hosted on our GitHub repository at https://github.com/eolweus/autodeeplab. This contribution aids in expanding the existing repository of solar farm data available online, assisting further research and study in the field.



(a) The high-resolution image appears to indicate a false positive for this prediction



(b) Discerning the presence of a PV installation on the low-resolution image is a significant challenge

Figure 4.13: A comparison of the same area on high- and low-resolution images

Chapter 5

Discussion

The subsequent discussion delves into the various aspects of our study, providing an in-depth analysis of our findings and their implications. We begin by examining the prevailing trends in architectural design that emerged from our research. Next, we re-evaluate the efficacy of transfer learning, in the light of our findings (SO1).

Further, we assess the robustness of Neural Architecture Search in the context of satellite image segmentation (SO2), followed by a discussion on the computational trade-offs involved in the application of NAS (SO3). We then consider possible improvements to the NAS process we employed that could enhance its efficiency and performance for future research.

Advancing the discourse, we venture into an engaging comparison between NAS and massive generalized models, touching upon the trade-offs and the possible emergent synergies in forthcoming studies. As we approach the conclusion of this chapter, we recognize the limitations inherent in our study and propose avenues for further investigation.

5.1 Trends in Architectural Design

The distinct architectures generated through our methodology exhibited remarkable performance consistency on the complete dataset, regardless of their differences.

In examining the cell architectures, a few patterns emerged. Operation choices such as skip-connections and 3x3 separable depth-wise convolutions were predominantly favored. On the other hand, the two pooling operations were least preferred, possibly due to the low resolution of the images, combined with the ability to downsample within the macro architecture.

A distinct trend in operational preferences is also evident, with most models displaying a propensity towards a particular operation. An overwhelming majority of architectures analyzed incorporated at least four instances of a single operation, indicating a certain bias.

Macro architectures exhibited a high degree of variability. Nevertheless, akin to the findings of Liu et al. [30], we observed that paths tend to deepen in the

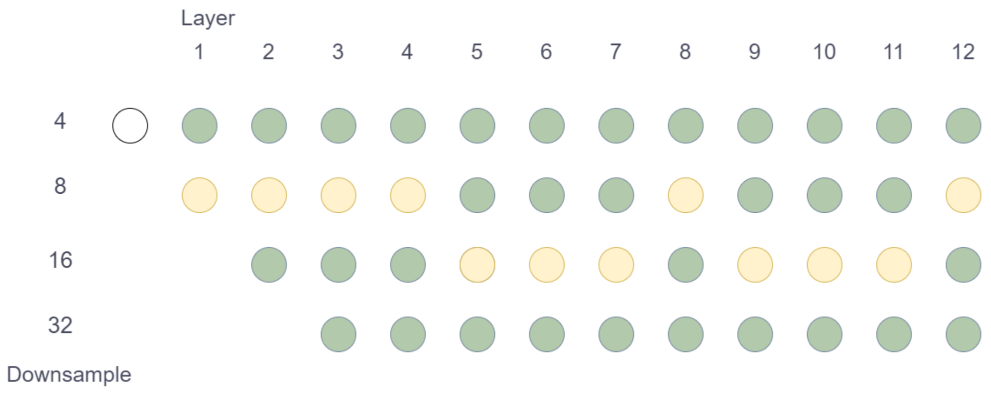


Figure 5.1: Average (and median) downsample level (rounded) for each layer across the discovered architectures

middle layers and remain shallow at the start. Multiple architectures followed a form of 'W' or inverted 'N' pattern, alternating between descending and ascending paths. In the concluding layer, most architectures culminate in either the lowest or highest level. The bulk of the layers are located within the two middle depths, as illustrated by the rounded average (and median) downsample level for each layer in Figure 5.1.

A noteworthy observation was the early peak performance achieved by most searches, soon after the commencement of architectural parameter optimization in epoch 21. The exact reason for this behavior is uncertain, but it may indicate potential areas of improvement for the current DARTS and Auto-DeepLab search strategies, as performance enhancement doesn't align with each search iteration. Alternatively, this could be a reflection of the complex search landscape or the presence of multiple deep local minima as suggested by Chen and Hsieh [65].

Intriguingly, despite the wide disparity in architectures, the performance on the full dataset was largely congruent. Yang et al. [66] propose that DARTS-based architectures may achieve superior performance by focusing more on adjusting macro architectural parameters, such as the number of cells (layers), rather than investing substantial efforts in refining the cell structure.

5.2 Re-evaluating the Efficacy of Transfer Learning (SO1)

As detailed in Sections 2.6.2 and 4.2, the Solis-seg and Solis-transfer models differ solely in their training methodology. Solis-seg is dedicated to the exclusive task of semantic segmentation of solar farms, whereas the ResNet component of Solis-transfer is initially trained to identify whether an image does or does not contain a solar farm (classification), and only thereafter it is trained for the task of segmentation.

Despite numerous trials with Solis-transfer, it has yet to surpass an F1 score of 0.89. In contrast, the single experiment conducted with Solis-seg yielded a significantly superior F1 score (0.962), clearly highlighting the effectiveness of task-specific training. The increase in performance is thus evidently attributable to the switch in training strategy, as no other alterations were made during the training process.

This surprising outcome suggests that the methods employed by the classification model to discern the presence of a solar farm differ considerably from the pixel-wise recognition performed during semantic segmentation. It posits the idea that the competencies required for these tasks might diverge to an extent that proficiency in one (classification) could potentially impede the ability to learn the other (segmentation).

It appears counterintuitive for a model's performance to be adversely affected by prior training in classification. However, this unexpected result underscores the intricacy of these tasks and emphasizes the critical role that appropriate selection of training strategies plays.

Moreover, this experiment highlights the notion that the benefits of transfer learning are not universally applicable, but are contingent upon various factors including the degree of similarity between the source and target tasks, and the specific nature of these tasks. Our study, for example, points towards instances where a model specifically trained for a particular task from inception can outperform one that capitalizes on transferred knowledge from an ostensibly related task.

In summarizing our findings, it's compelling to note that our best-performing model surpassed the IoU score of 0.9 obtained by Kruitwagen et al. [3]. While an apples-to-apples comparison isn't feasible due to their employment of a considerably larger and globally distributed dataset, our results hold significance given the markedly higher relative score attained on our dataset.

Future research endeavors could uncover further valuable insights by subjecting our model to the dataset employed by Kruitwagen et al. This approach would allow for the performance evaluation of our model in a more expansive and diverse setting. However, it's worth noting that developing a data pipeline, akin to the one employed by Kruitwagen et al., that synergizes their data with our trained model is likely a substantial undertaking due to the complex nature of these pipelines as discussed in section 2.6.1. This complexity is the primary reason we have not endeavored to attempt this in our current study.

5.3 The Robustness of NAS in Satellite Image Segmentation (SO2)

Our research offers insights into applying Neural Architecture Search (NAS) for semantic segmentation of solar farms on Sentinel-2 imagery. The uniformity of data quality across different dataset sizes and subsets resulted in little variation in performance among the various NAS-derived models. An exception to this was the model trained on the 20k dataset, which underperformed unexpectedly. The precise reasons for this remain unclear, although disruptive data elements or an unfortunate random seed choice may be possible causes.

The influence of spectral bands on architecture search emerged as a significant factor. As per our detailed analysis in Chapter 4, four separate trials were conducted with architectures identified using a 10,000-image dataset. As demonstrated in Figure 5.2, the model trained solely with RGB data underperformed compared to models that utilized additional spectral bands. Despite the need for further trials to conclusively attribute this performance discrepancy to spectral band usage, our findings hint at Auto-DeepLab's potential to leverage this extra information effectively. These results pave the way for intriguing prospects of employing NAS with more complex features, further discussed in section 5.6.

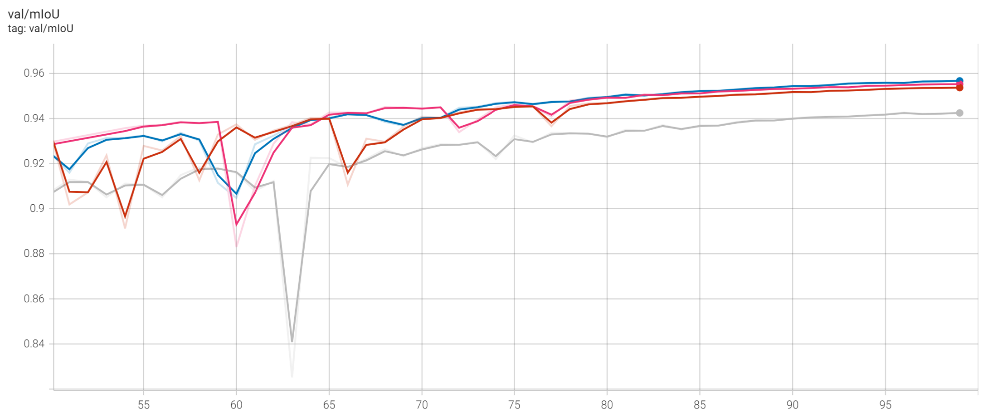


Figure 5.2: Comparison of the validation mIoU for the last 50 epochs of the models found with 10,000 images. The models are 10k, 10k-final, 10k-no-ab, and 10k-rgb. The gray line at the bottom is 10k-rgb

An intriguing finding from our experiments was that out of 14 NAS trials, only a single architecture managed to outperform any of the benchmarks, excluding Solis-transfer. This raises questions regarding the effectiveness and cost-benefit value of DARTS and Auto-DeepLab within this particular context, which will be further elaborated in section 5.4.

Surprisingly, the randomly sampled architecture produced by ChatGPT outperformed almost all of the architectures identified via the architecture search. While this might be an outlier event and additional random samples should be examined for validation, it raises questions about the consistency and effectiveness of the architecture search process in yielding superior architectures for certain use-cases even when it has demonstrated the capability to accurately assess comparative performance, as highlighted in Experiment 2.

Furthermore, it was observed that the performance of most models was closely aligned with that of the random model. This suggests that the search space may be densely populated with models that deliver comparable performance, thereby making it difficult to continually progress toward an optimal solution. This hypothesis is supported by the search graphs outlined in chapter 4, particularly by the observation of most searches reaching their peak early. This pervasive challenge is credited by Chen and Hsieh [65] to DARTS tendency to reach strong local minima in the search space.

Moreover, the top-performing NAS model, 10k-L, only slightly lagged behind the best-performing model, Solis-seg. This suggests that under appropriate conditions, NAS has the potential to generate architectures that approach or even match the state-of-the-art, even in specialized applications such as satellite imagery segmentation. The robustness and adaptability of NAS, despite the complexities and challenges, underscore its potential as a valuable tool in a researcher's arsenal.

5.4 Computational Trade-offs in NAS Application (SO3)

NAS is a demanding procedure, introducing substantial overhead to a machine learning pipeline. Not only does it necessitate training a model, it requires significant additional time and resources to discover the model architecture in the first place. This inherently prompts the question: When is the extra cost of performing NAS worthwhile?

The answer hinges on the anticipated performance increase and the significance of optimal performance for the application at hand. In the case of our study, the findings suggest that NAS might not represent the most effective option. As evidenced in Tables 5.1 and 5.2, the time required to produce a fully trained working model is at least doubled in most instances. Paired with the observation that an off-the-shelf model (ResNet50-DeepLab) yields the best results on the dataset after 14 NAS trials, underscores a potential doubt regarding the cost-effectiveness of NAS.

Reflecting on the top five models derived from our study, as shown in Table 5.3, three out of the five top performers are baseline models which we originally proposed for comparison. Interestingly, even a randomly suggested model outperformed all but one model discovered through NAS.

While the search outcomes might not seem particularly outstanding — failing to surpass a ResNet-based model, marginally exceeding a model found by searching on a different dataset, and the curious case of a random model outperforming all but one NAS architecture — it is important to recognize that the top model found through the search, 10k-L, does not lag significantly behind the best model, Solis-seg.

There are potential improvements to our NAS process that could potentially enhance the performance of the discovered models, as discussed in Section 5.5. Though even if we were to conduct additional trials and come across a model that outperforms Solis-seg, the total cost of the new model would exceed the cost we incurred by training the off-the-shelf model by magnitude for the sake of a slight increase in performance[1]

It's worth noting that all models outperformed Solis-transfer, implying that the

[1]Scoring a perfect 1 should be impossible due to data imperfections (refer to section 5.5)

DARTS search space is replete with viable model architectures. Additionally, given the low-resolution nature of the images in this study, this presents a relatively unconventional segmentation problem. Considering this, the obtained results speak to the robustness and versatility of the models derived from the DARTS search space.

The decision of whether or not to use NAS essentially hinges on the importance of incremental performance improvement and the available alternatives to increase the performance of the model. In our case, however, it might be more productive to allocate resources toward enhancing other aspects of the model, such as augmenting the quality and volume of data [10] or investigating the optimal combination of spectral bands.

Moreover, the high computational cost of NAS could potentially deter smaller entities or individual researchers who operate with more constrained resources. Without access to the IDUN cluster, this research project would have likely spanned well over a hundred continuous training days on Enernite's NVIDIA RTX-3090 GPU.

All these considerations should be factored in when deciding whether to employ NAS, further emphasizing the need for a case-by-case approach to the application of this technology.

Finally, it is also crucial to remember that NAS is a relatively nascent field. As with many emerging technologies, it will likely undergo considerable refinement and become more efficient and accessible in the coming years. Future advancements might mitigate many of the current limitations, enabling more widespread and accessible usage. As such, staying up to date on NAS development and its potential for evolving machine learning models will be critical to continuously evaluate its applications and benefits in the future.

Dataset size	**Search time (h)**
2k	20
5k	41
10k	62
20k	104

Table 5.1: Dataset Size and Search Time

Model	**training time (h)**
Solis-seg	46
NAS architectures	59

Table 5.2: Model Type and Search Time. The time for NAS architectures is an average. Note that the time varied quite a bit depending on the hardware used

Name	mIoU	F1-score
Solis-seg	0.9629	0.9621
10k-L	0.9593	0.9582
ADL-cs	0.9586	0.9575
chatgpt	0.9565	0.9552
2k	0.9563	0.9550
Solis-transfer	N/A	0.89

Table 5.3: The top 5 models ranked by validation mIoU obtained during retraining. The model 10k has been omitted here as it shares the same architecture as 10k-L. It would have been placed between ADL-cs and chatgpt, as shown in table 4.1.

5.5 Potential Improvements to the Neural Architecture Search Process

Although our most effective model did not emerge from NAS, there are numerous instances where NAS has superseded human-engineered models in performance [8, 9]. This implies that there might exist enhancements that could have been implemented to boost the effectiveness of our searches.

For example, allowing some models to undergo additional epochs during retraining could have led to performance improvements. Particularly, the 20k model does not appear to have fully converged even after 100 epochs as seen in figure 5.3. Extending the retraining process might have allowed this model to improve at least a bit.

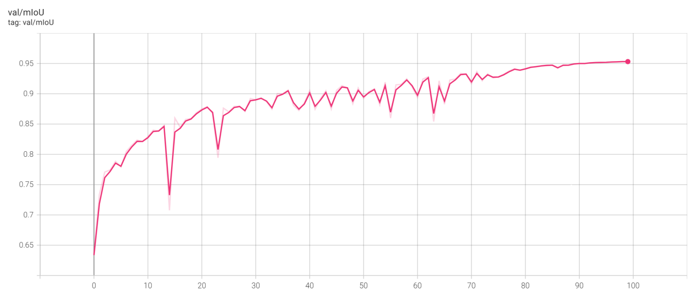


Figure 5.3: Graph showing 20k validation mIoU per epoch of retraining

We could have experimented with the parameters of Auto-DeepLab. Modifying the number of layers in the network could have led to different performance outcomes. Given the complex interplay of layers and operations in a neural network, such architectural adjustments may have unearthed more optimized solutions. Actually, some research suggests that tuning the hyperparameters of the DARTS search can yield better results than tuning the architectures themselves [66].

Extending the duration of the search process could also have influenced the results. Despite Liu et al. [30] not reporting success with this approach in their

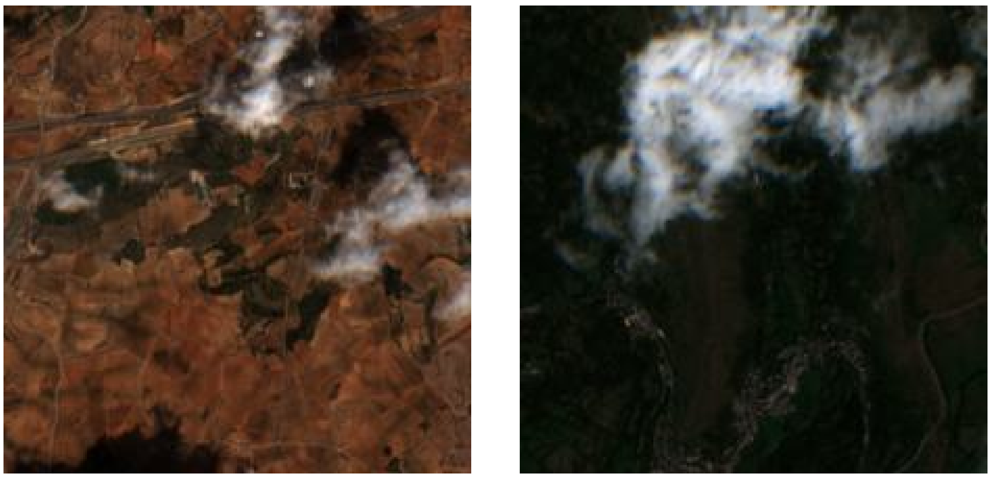


Figure 5.4: Some examples of images from the Solis dataset with clouds

study, it might have been worth exploring in our case.

Conducting a larger number of trials could also have been advantageous. A significant body of research points to the susceptibility of NAS to randomness [9, 32, 66]. Given the seemingly minor impact of the size or the composition of the data subset on the results, conducting a series of additional 2k experiments might have offered a more diverse set of architectural outcomes. These take significantly less time than the other searches as seen in table 5.1

Lastly, it is worth noting that achieving a perfect mIoU score with this dataset is not only unlikely but potentially concerning. This is due to the presence of images obscured by clouds (figure 5.4) and the fact that, given the sheer volume of over 200k images, it is challenging to guarantee a dataset devoid of any mislabeled images as shown in figure 5.5.

5.6 NAS versus Massive Generalized Models: Trade-offs and Emerging Synergy

The findings of this study, coupled with recent progress in diverse areas of artificial intelligence, engender an intriguing question about the future landscape of model development: Will highly specialized models discovered through Neural Architecture Search (NAS) dominate, or will we lean toward massive, versatile models that excel across a multitude of tasks within a domain, like GPT-4 [67] and SAM [25]?

NAS offers a mechanism to craft models optimized for particular tasks or datasets. This specialization, as our Experiment 4 suggested, can exploit additional image information like spectral bands, which are typically overlooked by broader models like SAM. This ability to tailor architectures to specific problems pushes performance boundaries, provides valuable insights into the nature of tasks at

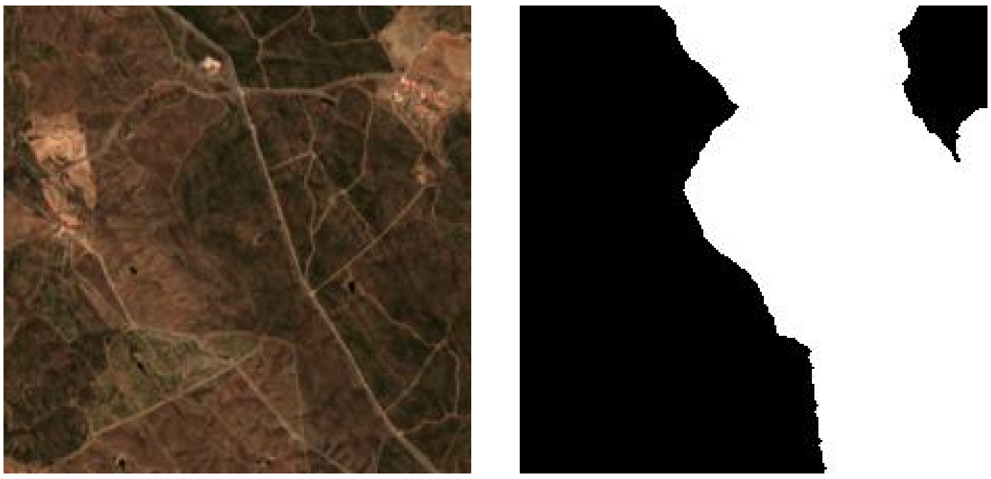


Figure 5.5: Despite the presence of a mask indicating otherwise, the provided image in the Solis dataset does not actually contain a solar farm.

hand, and can lead to efficient models adept at solving unique problems. However, the trade-off here includes a substantial computational cost, and the solutions proposed may lack generalizability across diverse tasks.

On the other hand, generalized models such as GPT-4 and SAM are built to perform well across a broad range of tasks within a specific domain. These models benefit from their scale and ability to leverage large amounts of diverse data, becoming proficient in multiple areas. They offer a more holistic approach and can handle a variety of tasks without needing task-specific customization. However, their vast size may restrict the peak task-specific performance that could be achievable by a NAS-generated model. Furthermore, their colossal size often translates to high resource requirements and substantial environmental impact and severely limits who can actually train these new networks. When trained however, many of these models become openly available and can be used for various tasks through APIs (GPT4) or through available source code and model weights (SAM).

The balance between specialized and generalized models will likely continue to shift as technological advances and computational resources evolve. Future research directions may explore hybrid strategies, blending the customization of NAS with the broad applicability of large-scale generalized models, or they may venture into entirely new paradigms as yet unforeseen.

The trade-offs between these two paradigms may even suggest a potential integration of the two. It is plausible that NAS could be the architect behind the design of future massive generalized models. As these super-models increase in size and complexity, the role of NAS in optimizing their structure may become increasingly valuable. While large models have proven to be proficient, NAS' ability to tailor architectures to specific problems could aid in refining these models, ensuring efficiency and improving performance.

The envisaged role of NAS in shaping future massive models necessitates its evolution. Presently, NAS is severely limited by the human-designed constraints within its search spaces. To truly architect these large-scale, versatile models, NAS must transcend these constraints [9]. If NAS evolves to autonomously generate not only novel architectures but completely new building blocks, it could potentially optimize these super-models more efficiently, driving their performance to new heights.

5.7 Limitations of this Study And Further Work

Throughout the course of this study, we have encountered and addressed numerous challenges. However, there are certain limitations that remained beyond our scope due to constraints of time, resources, and the nascent state of NAS methodologies. These limitations, in turn, also reveal compelling directions for future research.

While we have discussed several limitations in previous sections, such as limitations of our NAS implementation, there are a few notable ones that warrant further consideration:

- **Other NAS methodologies:** Our study utilized a specific NAS methodology, leaving other possibilities unexplored. Future research could examine the performance of different search methodologies or search spaces to provide a broader view of NAS applicability in satellite image segmentation.
- **Optimal Combination of Spectral Bands:** While we found that using all available spectral bands increased the performance, it remains uncertain whether all spectral bands are beneficial. Future work could investigate different combinations of spectral bands to find the optimal set for solar farm detection, in terms of both architecture search efficiency and overall model performance.
- **Random Baseline Performance:** Our study would have benefited from a larger set of randomly sampled models to determine whether the performance of our randomly sampled model was above average or a typical occurrence.
- **Consistent data selection:** While we made sure to select the same data for different models with the same dataset size, it would have been interesting to make it so that every dataset also included all the image present in smaller dataset sizes. This could have made it easier to evaluate the effect of additional data more thoroughly.
- **Enhancing data quality:** As deliberated in sections 2.5.2 and 5.5, dealing with satellite imagery presents its own set of challenges, and the quality of data in the Solis dataset is not exempt from imperfections. More time could have been invested in curating the dataset and eliminating detrimental data that may adversely impact the learning process of the models.

Despite these limitations, we believe that they present exciting opportunities for future research. Further areas of exploration could include:

- **Detailed Comparison to SAM:** A more in-depth comparison and analysis of SAM [25] could provide valuable insights into differentiating aspects of segmentation models.
- **Further Improvements to Solar Farm Recognition:** Additional enhancements to solar farm recognition, as suggested in section 2.6.3, warrant further investigation.
- **Exploration of NAS Parameters:** Modifying the architecture search parameters as suggested by Yang et al. [66] could yield different results and provide a more nuanced understanding of the NAS process.
- **Model Testing on Larger Datasets:** Our ResNet50-DeepLab model could be tested with larger datasets, such as the one curated by Kruitwagen et al [3], to further validate its performance.
- **Refinements to Auto-DeepLab:** While exploring different NAS methodologies presents one line of inquiry, there is also scope for augmenting our current approach. White et al. [9] highlight various pioneering works that rectify certain shortcomings inherent in the DARTS methodology. By association, many of these deficiencies are also evident in Auto-DeepLab. However, as per our knowledge, few of these enhancements have been implemented and tested with Auto-DeepLab. This provides an intriguing avenue for future research endeavors.
- **Geographic and Temporal Variations:** While we did include a test on data from a different continent, a more detailed analysis of the performance of our model in different parts of the world and across various seasons could be insightful. Future research could focus on the model's robustness in different geographic contexts.

Despite the limitations, we believe our study provides a solid foundation for future research on artificial intelligence and machine learning, especially in the context of solar farm detection. Our findings also offer critical insights into the role of transfer learning in segmentation tasks, opening doors to further exploration in this area.

Chapter 6

Conclusion

Throughout the course of our research, we have garnered several key insights with the potential to influence the future use of machine learning, including Neural Architecture Search (NAS), for semantic segmentation in satellite imagery and the detection of solar farms in general. This research contributes to our overall goals of advancing the state-of-the-art in detecting and segmenting solar farms in satellite imagery, examining the practical utility of Neural Architecture Search (NAS) for architecture optimization and critically reassessing the applicability of transfer learning from classification to segmentation tasks.

Let's now revisit the specific goals and study objectives set forth at the beginning of this research:

Research Goal: To advance the state-of-the-art in detecting and segmenting solar farms in satellite imagery by examining the practical utility of NAS for architecture optimization and critically reassessing the applicability of transfer learning from classification to segmentation tasks.

SO1 **Evaluating the effectiveness of Transfer Learning:** Determine whether using the backbone of a model trained on a classification task can deliver performance equivalent to a model entirely trained on segmentation.

SO2 **Investigating the Robustness of NAS in Satellite Image Segmentation:**

- **SO2.1** Understand the influence of various factors on NAS including the impact of dataset sizes, different data subsets, and special features of satellite imagery
- **SO2.2** Evaluate the robustness of the relative ranking produced by NAS and the risk of overfitting to a specific dataset during architecture search

SO3 **Assesing Computational Trade-offs in NAS Application:** Assess the computational cost implications of employing NAS for a practical application on a novel dataset and determine the potential benefits when contrasted with the usage of a pre-existing off-the-shelf architecture.

We now summarize our most crucial findings and their alignment with our study objectives:

Discovery of a Superior Model: Directly addressing our research goal, our investigation led to the development of Solis-seg, a deep learning model that outperforms Enernite's incumbent model, Solis-transfer, by a significant margin. Remarkably, Solis-seg has attained the highest validation mIoU on a major solar farm dataset with continental scale coverage known to us, outperforming solarnet [6] and Kruitwagen et al. [3] on their respective datasets as summarized in table 6.1.

Model	**Validation mIoU on Own Dataset**
Solis-seg (our novel model)	**0.9626**
Solarnet [6]	0.9421
Kruitwagen et al. [3]	0.9

Table 6.1: Performance Comparison of Models on Their Respective Datasets

Reevaluation of Transfer Learning: Corresponding to our first study objective (SO1), our research has reevaluated the efficacy of transfer learning from classification in semantic segmentation. In particular, contrary to previous findings [6], our results suggest that such transfer learning may not work so well. While transfer learning can economize on time, it may inadvertently substantially compromise the final performance when compared to training a model from scratch, as we did with Solis-seg.

Practical Application of NAS: Reflecting on our second study objective (SO2), our work serves as a real-world example of NAS application in semantic segmentation. While abundant NAS research focuses on classification, especially on ImageNet or the CIFAR datasets, there are few studies on semantic segmentation. There are even fewer cases of studies of NAS applied to complex problems such as solar farm detection and segmentation in satellite imagery. We aspire for our contribution to stimulate more research on practical applications.

Incorporating Additional Image Data: Furthering our exploration on the robustness of NAS (SO2.1), our research reveals that Auto-DeepLab, and possibly other NAS methodologies, can effectively utilize additional image data like spectral bands at an architectural level. This finding could spearhead the development of advanced, data-rich models, which could be particularly advantageous for tasks that derive substantial benefits from data not typically used in training broad, generalized models like SAM.

Balancing Efficiency and Resource Allocation: Addressing our third study objective (SO3), our research underscores the importance of considering time and resources when evaluating the pros and cons of NAS. For smaller businesses or individual researchers with limited resources, NAS may not be the most effective or efficient strategy for enhancing Artificial Neural Network (ANN) models. This is particularly applicable to Enernite's situation as a start-up with limited resources. Despite the potential of NAS, its utility needs to be balanced with practical constraints.

Introducing an Open Dataset of New York Solar Farms: A key contribution of our research is the provision of an openly accessible dataset comprising locations of solar farms identified by our Solis-seg model in New York. This adds to the pool of public solar farm datasets available online, thus enhancing the resources available for further research and analysis in this field. The geometrical data can be accessed via the file named us_ny_preds.geojson at our project's repository on GitHub, available at https://github.com/eolweus/autodeeplab.

In conclusion, our research findings provide valuable insights and contribute to the broader understanding and application of advanced methods in the field of satellite imagery segmentation, as well as the practical application of NAS on tasks where its potential remains largely untapped [9].

Bibliography

[1] IEA, *Electricity sector*, 2022. [Online]. Available: https://www.iea.org/reports/electricity-sector.

[2] IEA, *Electricity market report - july 2022*, 2022. [Online]. Available: https://www.iea.org/reports/electricity-market-report-july-2022.

[3] L. Kruitwagen, K. T. Story, J. Friedrich, L. Byers, S. Skillman and C. Hepburn, 'A global inventory of photovoltaic solar energy generating units,' *Nature*, pp. 604–610, Oct. 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03957-7.

[4] V. Plakman, J. Rosier and J. van Vliet, 'Solar park detection from publicly available satellite imagery,' *GIScience & Remote Sensing*, vol. 59, no. 1, pp. 461–480, 2022. DOI: 10.1080/15481603.2022.2036056. eprint: https://doi.org/10.1080/15481603.2022.2036056. [Online]. Available: https://doi.org/10.1080/15481603.2022.2036056.

[5] M. V. C. V. d. Costa, O. L. F. d. Carvalho, A. G. Orlandi, I. Hirata, A. O. d. Albuquerque, F. V. e. Silva, R. F. Guimarães, R. A. T. Gomes and O. A. d. C. Júnior, 'Remote sensing for monitoring photovoltaic solar plants in brazil using deep semantic segmentation,' *Energies*, vol. 14, no. 10, 2021, ISSN: 1996-1073. DOI: 10.3390/en14102960. [Online]. Available: https://www.mdpi.com/1996-1073/14/10/2960.

[6] X. Hou, B. Wang, W. Hu, L. Yin and H. Wu, *Solarnet: A deep learning framework to map solar power plants in china from satellite imagery*, 2019. DOI: 10.48550/ARXIV.1912.03685. [Online]. Available: https://arxiv.org/abs/1912.03685.

[7] J. Yu, Z. Wang, A. Majumdar and R. Rajagopal, 'Deepsolar: A machine learning framework to efficiently construct a solar deployment database in the united states,' *Joule*, vol. 2, pp. 2605–2617, 12 2018. DOI: https://doi.org/10.1016/j.joule.2018.11.021. [Online]. Available: https://doi.org/10.1016/j.joule.2018.11.021.

[8] T. Elsken, J. H. Metzen and F. Hutter, 'Neural architecture search: A survey,' 2018. DOI: 10.48550/ARXIV.1808.05377. [Online]. Available: https://arxiv.org/abs/1808.05377.

[9] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey and F. Hutter, *Neural architecture search: Insights from 1000 papers*, 2023. DOI: `10.48550/ARXIV.2301.08727`. [Online]. Available: `https://arxiv.org/abs/2301.08727`.

[10] C. Layman, *Using satellites to track solar farm growth*, 2019. [Online]. Available: `https://medium.com/astraeaearth/astraea-solar-farm-study-8d1b3ec28361`.

[11] E. Olweus, 'Network architecture search for detection of solar farms in satellite imagery,' NTNU, Tech. Rep., 2022.

[12] Y.-S. Park and S. Lek, 'Chapter 7 - artificial neural networks: Multilayer perceptron for ecological modeling,' in *Ecological Model Types*, ser. Developments in Environmental Modelling, S. E. Jørgensen, Ed., vol. 28, Elsevier, 2016, pp. 123–140. DOI: `https://doi.org/10.1016/B978-0-444-63623-2.00007-4`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780444636232000074`.

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks,' in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[14] V. Nair and G. Hinton, 'Rectified linear units improve restricted boltzmann machines vinod nair,' vol. 27, Jun. 2010, pp. 807–814.

[15] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' *Nature*, vol. 323, pp. 533–536, 1986.

[16] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: `1506.02640 [cs.CV]`.

[17] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez and J. Garcia-Rodriguez, *A review on deep learning techniques applied to semantic segmentation*, 2017. DOI: `10.48550/ARXIV.1704.06857`. [Online]. Available: `https://arxiv.org/abs/1704.06857`.

[18] A. Shaw, D. Hunter, F. Iandola and S. Sidhu, *Squeezenas: Fast neural architecture search for faster semantic segmentation*, 2019. arXiv: `1908.01748 [cs.CV]`.

[19] K. Weiss, T. M. Khoshgoftaar and D. Wang, 'A survey of transfer learning,' *Journal of Big Data*, vol. 3, 1 May 2016. DOI: `10.1186/s40537-016-0043-6`. [Online]. Available: `https://doi.org/10.1186/s40537-016-0043-6`.

[20] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: `10.48550/ARXIV.1512.03385`. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[21] J. Long, E. Shelhamer and T. Darrell, *Fully convolutional networks for semantic segmentation*, 2014. DOI: 10.48550/ARXIV.1411.4038. [Online]. Available: https://arxiv.org/abs/1411.4038.

[22] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. DOI: 10.48550/ARXIV.1505.04597. [Online]. Available: https://arxiv.org/abs/1505.04597.

[23] C. Tao, Y. Meng, J. Li, B. Yang, F. Hu, Y. Li, C. Cui and W. Zhang, 'Ms-net: Multispectral semantic segmentation network for remote sensing images,' *GIScience & Remote Sensing*, vol. 59, no. 1, pp. 1177–1198, 2022. DOI: 10.1080/15481603.2022.2101728. eprint: https://doi.org/10.1080/15481603.2022.2101728. [Online]. Available: https://doi.org/10.1080/15481603.2022.2101728.

[24] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, 2017. arXiv: 1606.00915 [cs.CV].

[25] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár and R. Girshick, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[26] Q. Wu and L. P. Osco, *samgeo: A Python package for segmenting geospatial data with the Segment Anything Model (SAM)*, version v0.8.0, May 2023. DOI: 10.5281/zenodo.7966658. [Online]. Available: https://doi.org/10.5281/zenodo.7966658.

[27] M. Thoma, *A survey of semantic segmentation*, 2016. DOI: 10.48550/ARXIV.1602.06541. [Online]. Available: https://arxiv.org/abs/1602.06541.

[28] A. Slowik and H. Kwasnicka, 'Evolutionary algorithms and their applications to engineering problems,' *Neural Computing and Applications*, vol. 32, no. 16, pp. 12 363–12 379, Aug. 2020.

[29] E. Real, A. Aggarwal, Y. Huang and Q. V. Le, *Regularized evolution for image classifier architecture search*, 2019. arXiv: 1802.01548 [cs.NE].

[30] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille and L. Fei-Fei, *Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation*, 2019. DOI: 10.48550/ARXIV.1901.02985. [Online]. Available: https://arxiv.org/abs/1901.02985.

[31] H. Liu, K. Simonyan and Y. Yang, *Darts: Differentiable architecture search*, 2018. DOI: 10.48550/ARXIV.1806.09055. [Online]. Available: https://arxiv.org/abs/1806.09055.

[32] X. Dong and Y. Yang, *Nas-bench-201: Extending the scope of reproducible neural architecture search*, 2020. DOI: 10.48550/ARXIV.2001.00326. [Online]. Available: https://arxiv.org/abs/2001.00326.

[33] G. Ochoa and N. Veerapen, 'Neural architecture search: A visual analysis,' in *Parallel Problem Solving from Nature – PPSN XVII*, G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa and T. Tušar, Eds., Cham: Springer International Publishing, 2022, pp. 603–615, ISBN: 978-3-031-14714-2. [Online]. Available: `https://link.springer.com/chapter/10.1007/978-3-031-14714-2_42`.

[34] Y. Shu, W. Wang and S. Cai, *Understanding architectures learnt by cell-based neural architecture search*, 2019. DOI: `10.48550/ARXIV.1909.09569`. [Online]. Available: `https://arxiv.org/abs/1909.09569`.

[35] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak and N. D. Lane, *Zero-cost proxies for lightweight nas*, 2021. DOI: `10.48550/ARXIV.2101.08134`. [Online]. Available: `https://arxiv.org/abs/2101.08134`.

[36] C. White, M. Khodak, R. Tu, S. Shah, S. Bubeck and D. Dey, 'A deeper look at zero-cost proxies for lightweight nas,' in *ICLR Blog Track*, https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/, 2022. [Online]. Available: `https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/`.

[37] C. White, A. Zela, B. Ru, Y. Liu and F. Hutter, *How powerful are performance predictors in neural architecture search?* 2021. DOI: `10.48550/ARXIV.2104.01177`. [Online]. Available: `https://arxiv.org/abs/2104.01177`.

[38] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang and W. Ouyang, *Econas: Finding proxies for economical neural architecture search*, 2020. DOI: `10.48550/ARXIV.2001.01233`. [Online]. Available: `https://arxiv.org/abs/2001.01233`.

[39] T. Wei, C. Wang, Y. Rui and C. W. Chen, *Network morphism*, 2016. DOI: `10.48550/ARXIV.1603.01670`. [Online]. Available: `https://arxiv.org/abs/1603.01670`.

[40] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang and W. Ren, *Dcnas: Densely connected neural architecture search for semantic image segmentation*, 2021. arXiv: `2003.11883 [cs.CV]`.

[41] J. Mellor, J. Turner, A. Storkey and E. J. Crowley, *Neural architecture search without training*, 2020. DOI: `10.48550/ARXIV.2006.04647`. [Online]. Available: `https://arxiv.org/abs/2006.04647`.

[42] B. Ru, C. Lyle, L. Schut, M. Fil, M. van der Wilk and Y. Gal, *Speedy performance estimation for neural architecture search*, 2020. DOI: `10.48550/ARXIV.2006.04492`. [Online]. Available: `https://arxiv.org/abs/2006.04492`.

[43] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen and T.-Y. Liu, *Semi-supervised neural architecture search*, 2020. DOI: `10.48550/ARXIV.2002.10389`. [Online]. Available: `https://arxiv.org/abs/2002.10389`.

[44] T. Duan, A. Avati, D. Y. Ding, K. K. Thai, S. Basu, A. Y. Ng and A. Schuler, *Ngboost: Natural gradient boosting for probabilistic prediction*, 2019. DOI: 10.48550/ARXIV.1910.03225. [Online]. Available: https://arxiv.org/abs/1910.03225.

[45] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy and F. Hutter, *Nas-bench-101: Towards reproducible neural architecture search*, 2019. DOI: 10.48550/ARXIV.1902.09635. [Online]. Available: https://arxiv.org/abs/1902.09635.

[46] A. Krizhevsky and G. Hinton, 'Learning multiple layers of features from tiny images,' *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[47] R. Tu, N. Roberts, M. Khodak, J. Shen, F. Sala and A. Talwalkar, *Nas-bench-360: Benchmarking neural architecture search on diverse tasks*, 2021. DOI: 10.48550/ARXIV.2110.05668. [Online]. Available: https://arxiv.org/abs/2110.05668.

[48] Y. Duan, X. Chen, H. Xu, Z. Chen, X. Liang, T. Zhang and Z. Li, *Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search*, 2021. DOI: 10.48550/ARXIV.2105.11871. [Online]. Available: https://arxiv.org/abs/2105.11871.

[49] A. Zela, J. Siems, L. Zimmer, J. Lukasik, M. Keuper and F. Hutter, *Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks*, 2020. DOI: 10.48550/ARXIV.2008.09777. [Online]. Available: https://arxiv.org/abs/2008.09777.

[50] G. F. Miller, P. M. Todd and S. U. Hegde, 'Designing neural networks using genetic algorithms,' in *International Conference on Genetic Algorithms*, 1989.

[51] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen and K. C. Tan, 'A survey on evolutionary neural architecture search,' *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021. DOI: 10.1109/tnnls.2021.3100554. [Online]. Available: https://doi.org/10.1109%5C%2Ftnnls.2021.3100554.

[52] X. Dong and Y. Yang, *Searching for a robust neural architecture in four gpu hours*, 2019. arXiv: 1910.04465 [cs.CV].

[53] L. Li and A. Talwalkar, 'Random search and reproducibility for neural architecture search,' 2019. DOI: 10.48550/ARXIV.1902.07638. [Online]. Available: https://arxiv.org/abs/1902.07638.

[54] B. Zoph and Q. V. Le, *Neural architecture search with reinforcement learning*, 2016. DOI: 10.48550/ARXIV.1611.01578. [Online]. Available: https://arxiv.org/abs/1611.01578.

[55] D. Floreano, P. Dürr and C. Mattiussi, 'Neuroevolution: From architectures to learning,' *Evolutionary Intelligence*, vol. 1, pp. 47–62, Mar. 2008. DOI: 10.1007/s12065-007-0002-4. [Online]. Available: https://doi.org/10.1007/s12065-007-0002-4.

[56] C. White, S. Nolen and Y. Savani, 'Exploring the loss landscape in neural architecture search,' 2020. DOI: 10.48550/ARXIV.2005.02960. [Online]. Available: https://arxiv.org/abs/2005.02960.

[57] T. D. Ottelander, A. Dushatskiy, M. Virgolin and P. A. N. Bosman, *Local search is a remarkably strong baseline for neural architecture search*, 2020. DOI: 10.48550/ARXIV.2004.08996. [Online]. Available: https://arxiv.org/abs/2004.08996.

[58] M. P. Marcus, B. Santorini, M. A. Marcinkiewicz and A. Taylor, *Treebank-3*, English, Web Download, Text, LDC Catalog No.: LDC99T42, Philadelphia, 1999. DOI: 10.35111/gq1x-j780. [Online]. Available: https://doi.org/10.35111/gq1x-j780.

[59] B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, *Learning transferable architectures for scalable image recognition*, 2018. arXiv: 1707.07012 [cs.CV].

[60] T. Elsken, A. Zela, J. H. Metzen, B. Staffler, T. Brox, A. Valada and F. Hutter, *Neural architecture search for dense prediction tasks in computer vision*, 2022. DOI: 10.48550/ARXIV.2202.07242. [Online]. Available: https://arxiv.org/abs/2202.07242.

[61] P. Lin, P. Sun, G. Cheng, S. Xie, X. Li and J. Shi, *Graph-guided architecture search for real-time semantic segmentation*, 2020. arXiv: 1909.06793 [cs.CV].

[62] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li and Z. Wang, *Fasterseg: Searching for faster real-time semantic segmentation*, 2020. arXiv: 1912.10917 [cs.CV].

[63] M. Själander, M. Jahre, G. Tufte and N. Reissmann, *EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure*, 2019. arXiv: 1912.05848 [cs.DC].

[64] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei and J. Yan, *Darts-: Robustly stepping out of performance collapse without indicators*, 2021. arXiv: 2009.01027 [cs.LG].

[65] X. Chen and C.-J. Hsieh, *Stabilizing differentiable architecture search via perturbation-based regularization*, 2021. arXiv: 2002.05283 [cs.LG].

[66] A. Yang, P. M. Esperança and F. M. Carlucci, *Nas evaluation is frustratingly hard*, 2020. arXiv: 1912.12522 [cs.LG].

[67] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

Appendix A

Dataset sizes

For our experiments, we used precisely 2040, 5000, 10010, and 20000 images, maintaining an 80/20 split between training and testing datasets. The seemingly unconventional number of images, particularly for the 2040 and 10010 datasets, arises from our initial attempt to ensure that both the subsets (training and testing) were divisible by the batch size, thereby achieving exact batches.

In the early stages of our work, we varied the batch sizes frequently due to ongoing experimentation and fluctuations in hardware availability. For these reasons, we moved away from this divisibility requirement for the 5000 and 20000 image datasets.

However, we decided to keep the original "unusual" dataset sizes of 2040 and 10010 consistent throughout all our experiments. This was to ensure that, when using a given random seed, the set of images randomly selected would remain identical, thus preserving experimental consistency and reproducibility.

Appendix B

Architecture Cells

We provide figures of the cells we discovered through NAS.

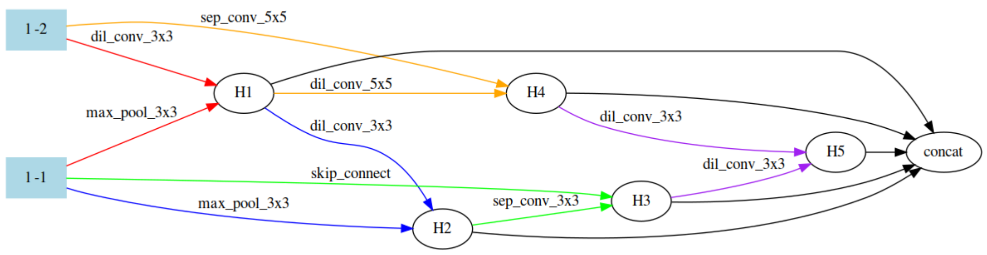


Figure B.1: 2k

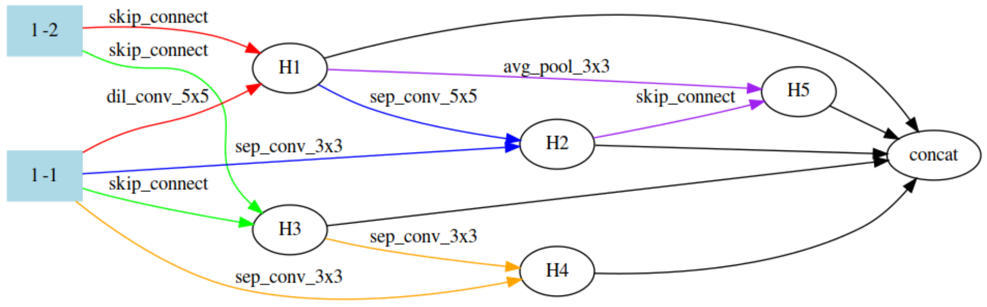


Figure B.2: 2k-dif

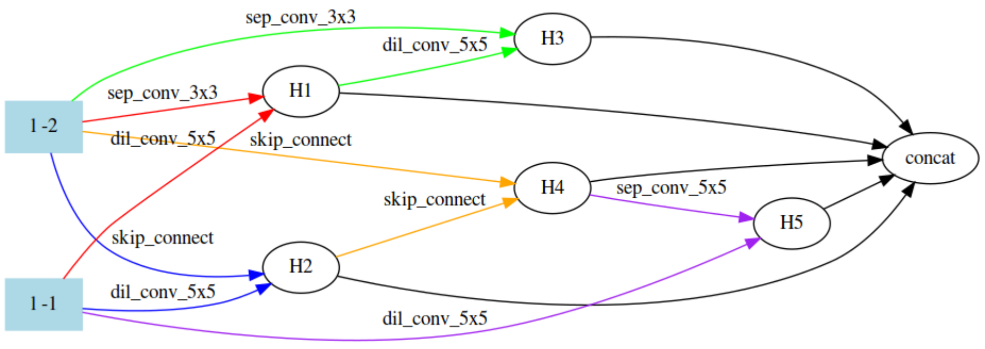


Figure B.3: 2k-no-ab

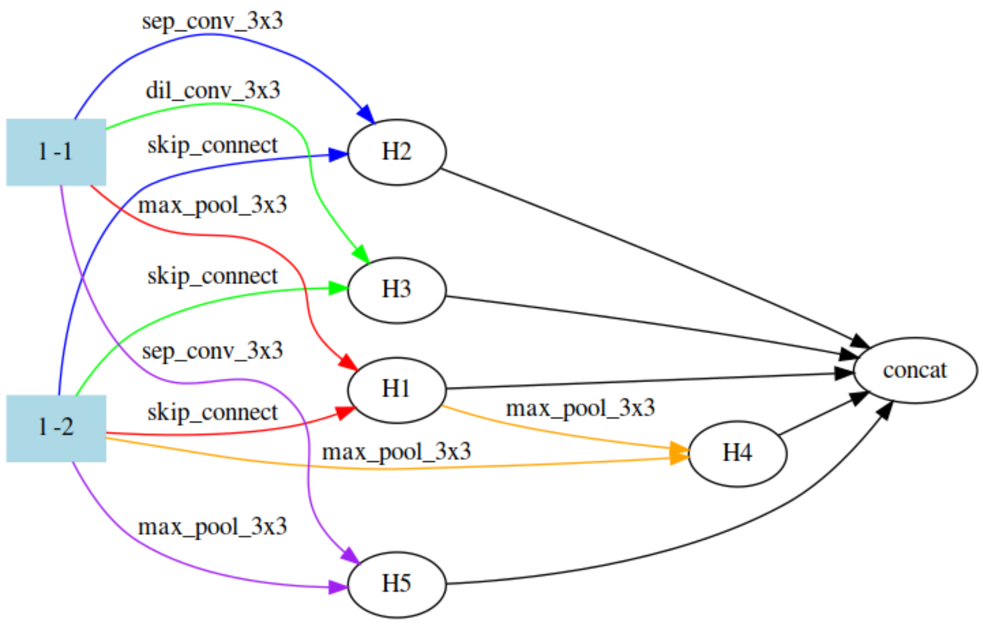


Figure B.4: 5k

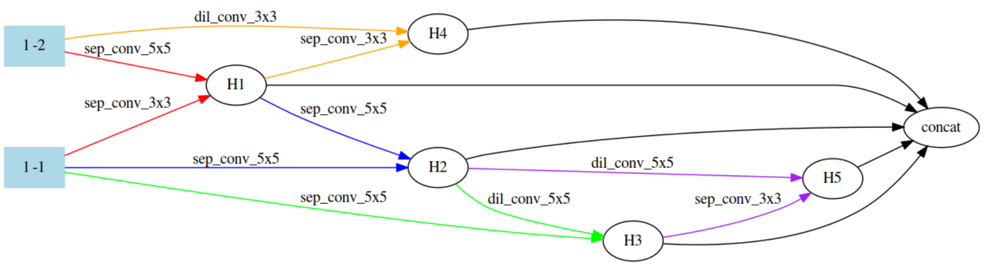


Figure B.5: 5k-dif

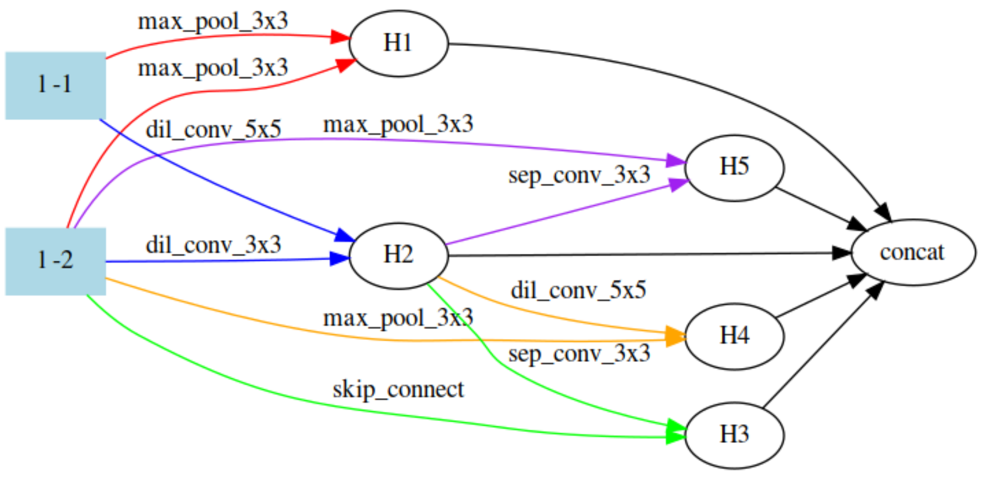


Figure B.6: 5k-final

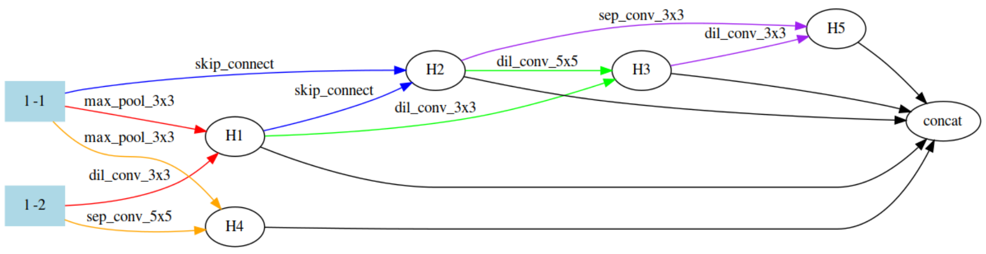


Figure B.7: 10k

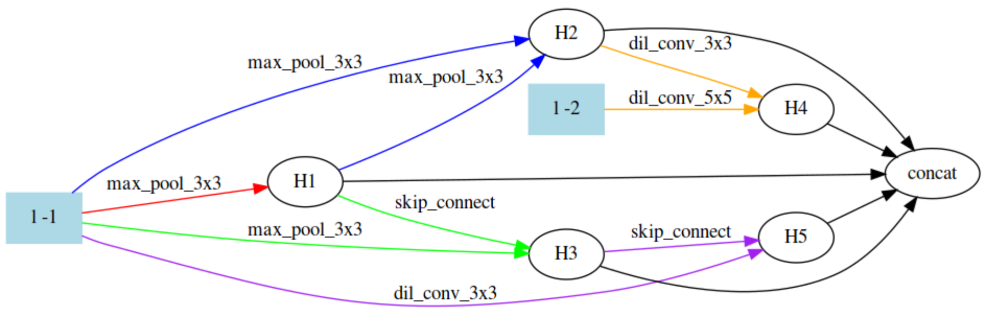


Figure B.8: 10k-final

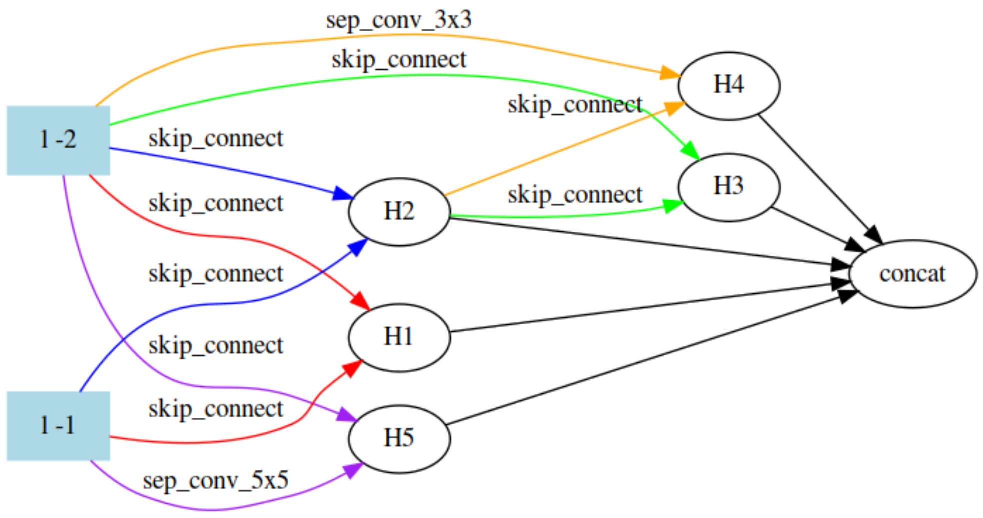


Figure B.9: 10k-no-ab

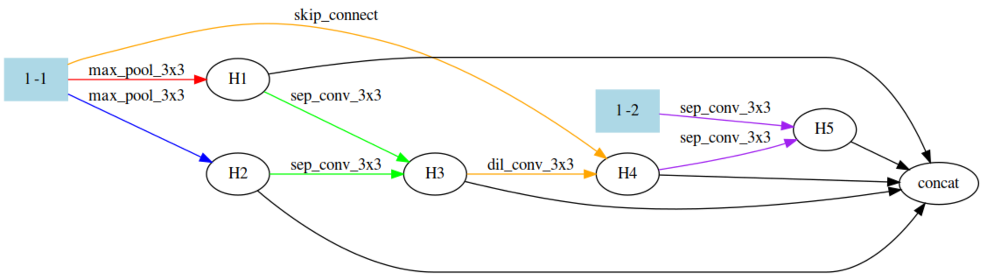


Figure B.10: 10k-RGB

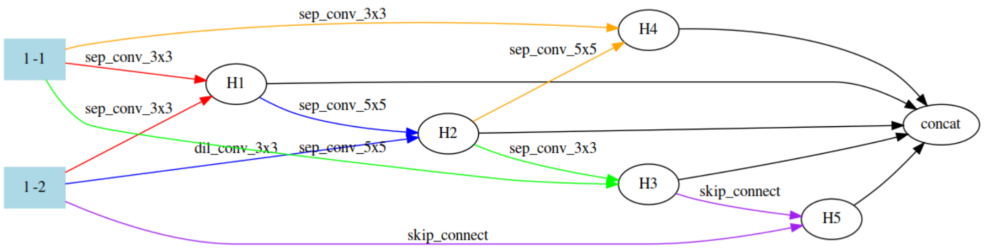


Figure B.11: 20k

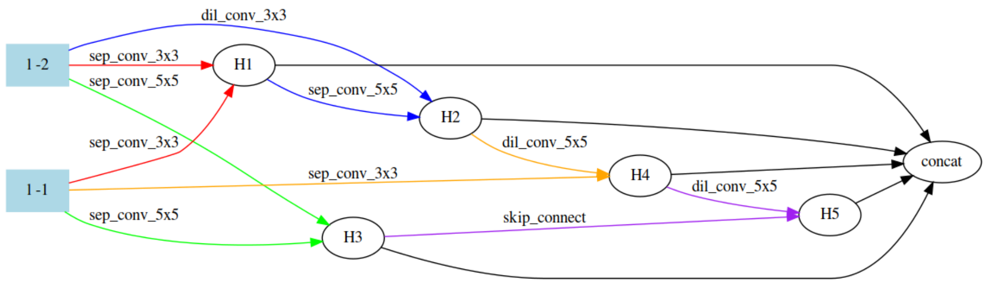


Figure B.12: 20k-second-best

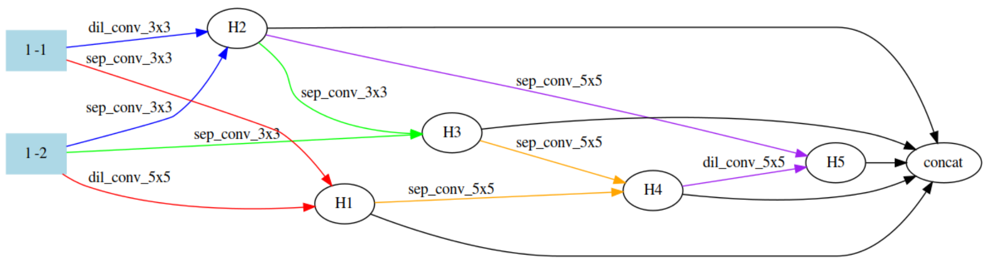


Figure B.13: The best cell discovered in the original Auto-DeepLab paper when searching on the Cityscapes dataset

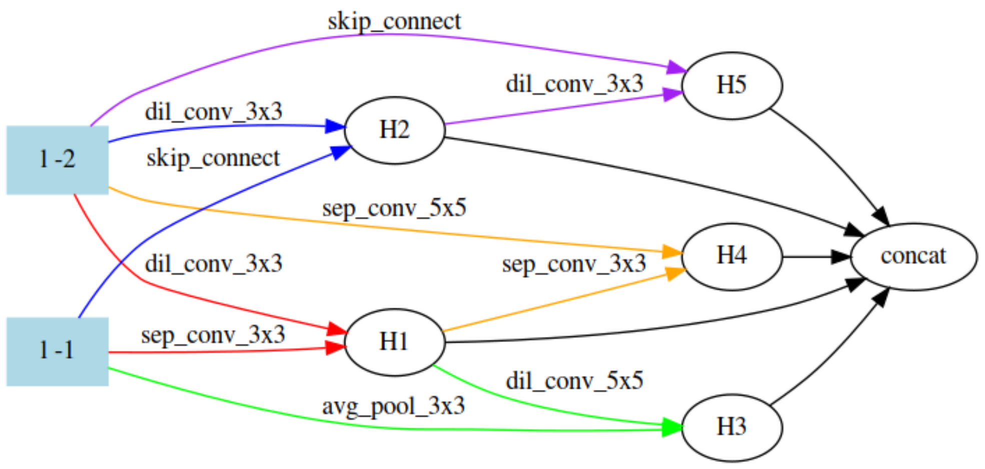


Figure B.14: The random cell created by ChatGPT

Appendix C

Generation of the Random Architecture

This chapter briefly outlines the way we randomly generated an architecture with ChatGPT.

C.1 The Architecture Encoding

The encoding for an Auto-DeepLab consists of two arrays. The first array details the depth level for each layer in the architecture (the macro level). For example, the Array: [0 1 2 3 2 2 1 0 0 1 2 3] is the encoding for the macro architecture in figure C.1

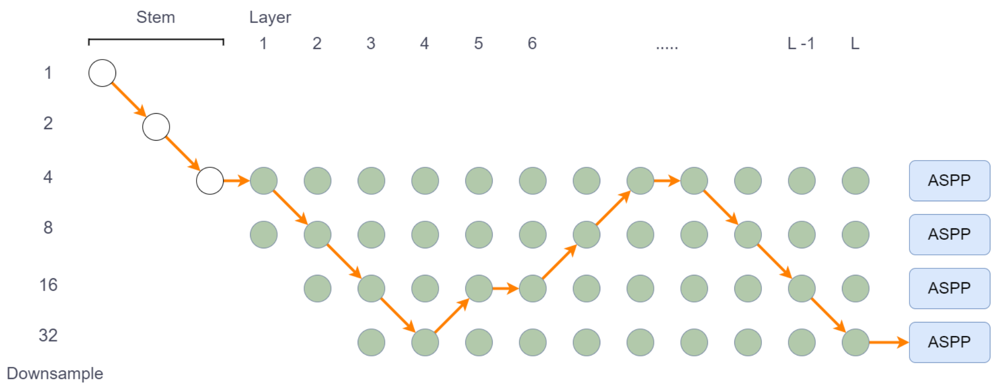


Figure C.1: Example of a Macro Architecture

The cell is encoded by an array of 10 pairs where each pair consists of an edge and an operation. This is the array for the 10k cell:

[[1 1] [0 6] [3 3] [4 3] [8 7] [7 6] [10 1] [9 5] [18 6] [17 4]]

C.2 Conversation with ChatGPT

Our objective was to establish a baseline using a random architecture, and for this task, we decided to utilize ChatGPT. As the subsequent discussion in this appendix will demonstrate, the chatbot was not aware of the implications of its suggestions, thereby ensuring that it didn't introduce any significant biases, thus serving our purposes effectively.

The following is a copy of the prompt that lead to the random cell: **quote start** give me two random arrays that look kind of like this:

[0 0 0 1 2 1 2 2 3 3 2 1]

[[0 7] [1 4] [2 4] [3 6] [5 4] [8 4] [11 5] [13 5] [17 5] [19 7]]

for the first array, the length should be 12, and the numbers have to be in range 0-3, also, the difference between subsequent numbers cannot be larger than 1.

for the second array, the right number in each instance is between 0-7

the left side is between 0 and 19 **quote end**

We had to repeat the prompt ChatGPT a few more times with prompts like: "*the first array is not valid*" or "*make new arrays*" until it generated something that was almost valid:

[0, 0, 1, 1, 2, 2, 1, 1, 0, 1, 2, 3]

[[0, 6], [1, 4], [2, 6], [3, 3], [4, 7], [6, 2], [9, 5], [11, 4], [14, 3], [17, 6]]

The second array has restrictions on what edges can exist simultaneously due to the maximum number of edges into a node being two. Thus we had to make a manual alteration by swapping edge 4 in the encoding to edge 7. The final cell array was:

[[0, 6], [1, 4], [2, 6], [3, 3], [6, 2], [7, 7], [9, 5], [11, 4], [14, 3], [17, 6]]

As we do not have any detailed knowledge of how this minor alteration impacts the architecture's performance, we consider it a randomly generated model for all practical purposes even if it was slightly modified by a human.