University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

# Contrastive Learning for Enhanced Feature Extraction in Hyperspectral Imagery

**Andras Bodrogai**

March 22, 2024

# Abstract

Vision transformers (ViTs) demonstrated outstanding performance in various computer vision downstream tasks, but their application to hyperspectral image (HSI) data analysis lags behind the RGB domain-based image processing. This dissertation investigates using ViTs for HSI data processing and semantic segmentation. We adopt several ViT architectures, including the ViT-B and a custom Spatial-Spectral ViT-B backbone, and experiment with how different self-supervised pre-training methods, specifically contrastive-learning-based methods, perform in the hyperspectral domain over Auto-Encoder-based counterparts. Despite promising characteristics, such as the strong feature correlation that was shown in the attention maps and strong feature retention shown during evaluation in the self-supervised DINOv2, our results indicate that the tested ViTs significantly underperform compared to both previously reported and state-of-the-art HSI segmentation methods presented in other papers. We analyze potential causes, and propose further work, highlighting potential work to be made.

# Education Use Consent

# Contents

# 1 Introduction

Initially, satellites and their application mainly focused on data collection, opening up space mission applications in a data-driven manner. Humans initially manually processed data, such as with the Hubble Space Telescope, mainly used to collect data from the vast distances of our universe and gain new knowledge (NASA 1990). However, applications that are similar to Hubble are infrequent nowadays and tailored towards everyday tasks, such as communications and monitoring, later generating vast amounts of data, as mentioned in this article Republic (2023), requiring systems capable of processing such information more efficiently. This transition from data collection to data processing applications resulted in the evolution of imaging sensors, moving from multispectral imaging, consisting of 5-12 bands, to hyperspectral imaging.

## 1.1 Motivation

Hyperspectral image (HSI) data comprises up to hundreds of numbers of spectral bands from the low twenties, covering a vast range of the electromagnetic spectrum, which provides more information on materials reflecting light into the camera sensor. This information is crucial for remote sensing as through this broad range of spectral bands, a considerable number of properties of objects can possess based on their spectral signature (Specim 2021). Because of the high information density of Hyperspectral images, they are widely used in monitoring the environment, such as for carbon-dioxide emission monitoring, which is mainly detected in Short-Wave Infrared (SWIR) region (Varchand et al. 2022). This information is lacking from traditionally red, green, and blue (RGB) images, such that, as mentioned above, HSI data provides a considerable advantage over RGB in the application of remote sensing. These characteristic of HSI data pushed researchers to explore how deep learning networks in the HSI domain differ from networks operating in the RGB domain. The difference is very fundamental. Hyperspectral optimised networks have additional extraction capabilities to take advantage of the high spectral resolution of HSI data, allowing networks optimised for this domain to map information across spectral bands (Liu et al. 2024).

## 1.2 Research Question

Most of the research above focuses on proposing new network architectures; however, it still relies on old pre-training training methods. We asked *'Would the emerging contrastive learning methods, such as DINOv2 (Oquab et al. 2024) improve on the already existing methods in the Hyperspectral domain?'* This research focuses on how modern approaches to pre-training change feature extraction performance using contrastive learning.

## 1.3 Structure

In this dissertation, we will first discuss the training methods and architectures used for the comparison and the research behind elevating those methods into the range of the state-of-the-art.

# 2 Background

Satellite applications are becoming increasingly common in earth observation missions for environmental monitoring, disaster management, and even urban planning. Satellites often provide unmatched value, providing a large-scale, consistent perspective, often inaccessible with any other observation device. Their ability for quick location revisit allows them to perform extensive periodic monitoring, identifying problems quickly (Voigt et al. 2016), such as floods and fires. Satellites are also equipped with much more advanced sensors, such as Hyper-Spectral cameras and synthetic aperture radars (SAR), allowing them to see information that would otherwise be invisible, hence making them much more capable for remote sensing applications (Pause et al. 2019). Ultimately, satellites are the ultimate eye in the sky, giving an overview of global events and allowing us to understand our surroundings better.

## 2.1 Hyper-Spectral Imaging

As previously mentioned, Hyper-Spectral imaging allows us to collect data that is simply impossible to see by the naked eye or for which traditional sensors are incapable of collecting. Compared to the widely used red, green, and blue images, hyper-spectral images consist of hundreds of electromagnetic channels ranging from infrared to ultraviolet wavelengths. The usefulness of data lies in the unique spectral signature of different materials due to their chemical composition, which ultimately decides whether a wavelength will be absorbed or reflected, resulting in the above mentioned unique signature (Hecker et al. 2011).

This data can also be used to understand the health of plants through their spectral signature. Plants reflect near-infrared (NIR), as can be seen in figure 2.1 and 2.2, and their health becomes assessable through it; for this purpose, vegetation indices are used such as the normalised difference vegetation index (NDVI). NDVI data is not just useful for health monitoring but is also helpful in determining groundwater levels through water stress levels (Walsh et al.) as there is shown to be a direct correlation between the health of plants and groundwater content (Shao et al. 2008).

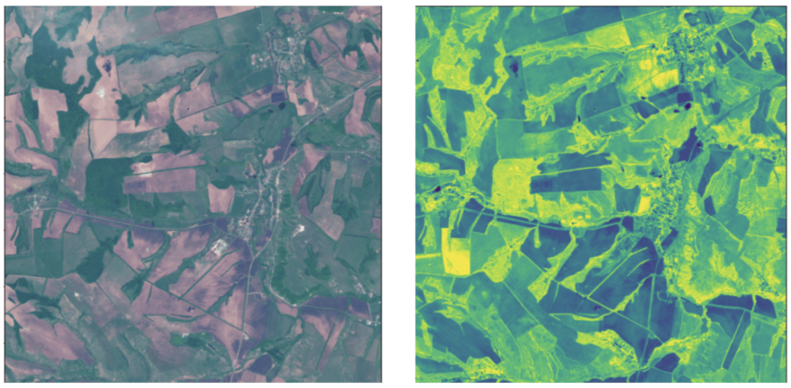

***Figure 2.1:** RGB image (left), Near-infrared (NIR) image (right) (Illarionova et al. 2021).*

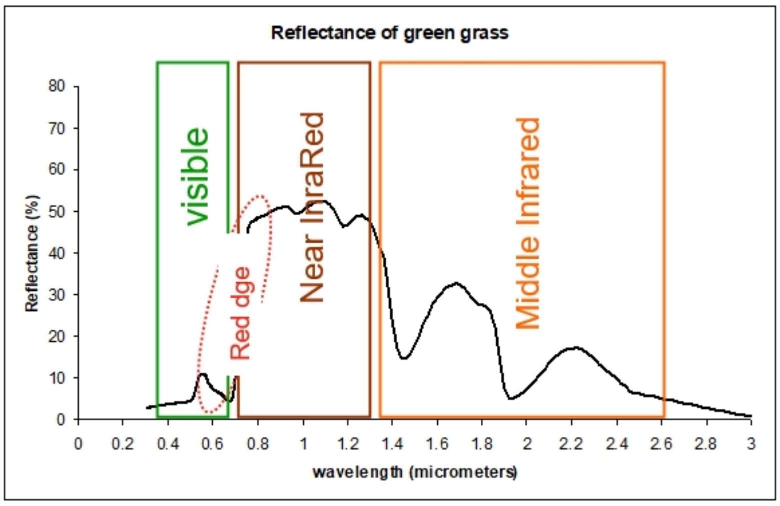

***Figure 2.2:*** *The spectral reflectance of plants in different spectral bands (EUMeTrain 2010).*

This uniqueness and versatility in the signature of materials later can be used in the analysis of satellite-collected remote sensing data and is especially useful for deep networks to learn on due to the drastic increase in information depth, providing guide rails for deep networks that is lost by the reduction of the uniqueness of the object being detected in remote sensing applications, as an example forests come in all shapes and sizes, but so are other biomes which makes them hard to classify, but due to the extra depth provided through HSI data, these features become much easier to identify.

## 2.2 Deep neural networks

Deep neural networks (DNN) are statistical models with many parameters, hence the name deep. The idea behind DNNs is that any function can be approximated well if there is enough input data and enough learning parameters in the DNN; this also means that they excel at generalisation compared to traditional machine learning (ML) models (DataCamp 2023), which results in better results on complex tasks. DNNs are built up of so-called layers which describe their steps of operation, such as shown below:

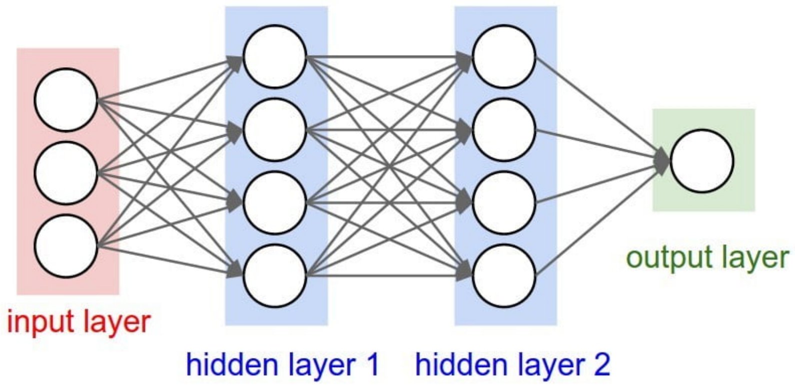

***Figure 2.3:*** *A deep neural network with an input, output, and two hidden fully connected layers.*

The image above represents an input layer, the hidden layers and an output layer. As mentioned above, all these layers apply an operation to their input data, transforming it slightly. In the network shown above, these layers are the hidden layers applying the weights and the biases then activation on the input:

$$x_o = ReLu((x_1 \cdot \vec{w_1}) + \vec{b_1}) \tag{2.1}$$

## 2.3 Convolutional Neural networks (CNNs)

Convolutional neural networks, unlike traditionally used neural networks, do not use fully connected layers as their primary type of layer but rely on convolution layers, which utilise kernels that slide over the input data. It is essentially a sliding window, which, given the filters, produces different outcomes. This operation is called the convolution operation:

$$\begin{aligned} &\text{Data:} \quad [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]] \\ &\text{Filter:} \quad [[1,0,1],[0,-1,1],[-2,1,1]] \\ &(1\cdot 1)+(2\cdot 0)+(3\cdot 1)+ \\ &+(5\cdot 0)+(6\cdot -1)+(7\cdot 1)+ \\ &+(9\cdot -2)+(10\cdot 1)+(11\cdot 1)=2 \qquad \text{Element-wise multiplication and sum} \\ &\text{shift filter, and repeat} \end{aligned} \tag{2.2}$$

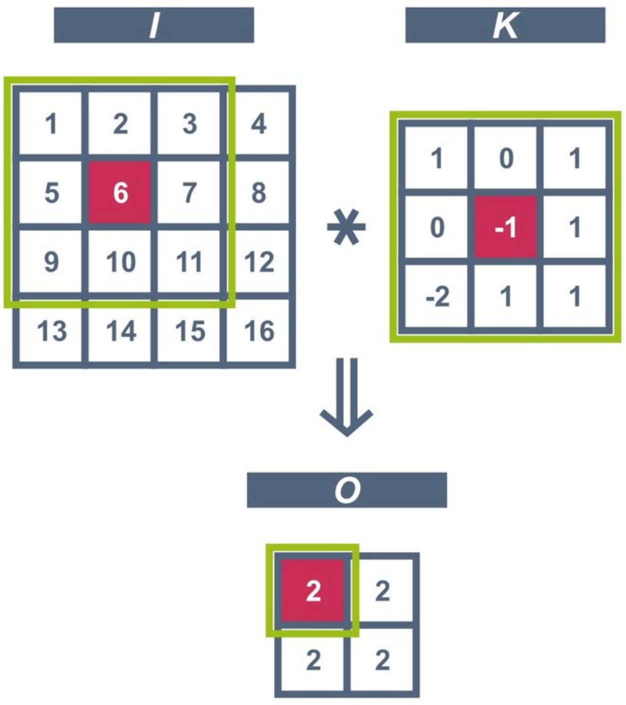

***Figure 2.4:*** *2D convolution operation, I - for input, K - for the convolution kernel, O - for output of the kernel (Roberto D'Autilia 2019). Demonstrated by equation eq.* (2.2)

This convolution operation results in a map (figure 2.4) that can extract specific features from data, such as horizontal or vertical lines. In convolutional neural networks, the optimiser optimises this extraction procedure or the filter's content for more efficient feature extraction. Further increasing the effectiveness of the convolution layers is not fully connected, as the filters only focus on a small sample of the data at a time, requiring fewer parameters and hence computational resources for their calculation. This capability of efficiently extracting features allowed convolutional networks historically to become the standard in computer vision for the coming decade, boasting new architecture, such as the VGG (Simonyan and Zisserman 2015) or the Inception (Szegedy et al. 2014), and techniques, such as using the residual in the (He et al. 2015).

One major drawback of the convolution neural networks comes from one of its benefits, which is the convolution operation itself. Due to the convolution operation, only a small window of data is visible to the network at a time, such as only having very localised feature extraction. Afterwards, pooling layers drop the resolution of the data so that the convolution layers can gain broader coverage over the data, practically growing to global feature extraction. This leads to the first layers not having context information on the data's whereabouts but also requiring the removal of fine-grained data, losing out on many details at deeper stages of the NN.

## 2.4 Vision Transformers

Convolutional neural networks have dominated the landscape of computer vision for around a decade now (Wikipedia contributors 2024). However, as mentioned earlier, due to their limitations and because they are challenging to scale, they never evolved further than the original counterparts, implementing only a few additional ideas, such as attention-guided convolution. This all changed with a new encoder architecture. A new network backbone emerged from an originally language-oriented network the transformers proposed by Vaswani et al. (2017), and then converted into a vision model using the same core of the backbone (Dosovitskiy et al. 2021), providing unparalleled global feature extraction capabilities, and hence became the focus point of most cutting-edge computer vision research.

### The Transformer architecture

Initially introduced by Vaswani et al. (2017), transformer networks became the cornerstone of natural language processing (NLP) due to their global extraction capabilities, such that allowing NLP networks to identify links between the words, thanks to a new architecture based on mainly a self-attention and a feed-forward block. These networks excel at long-range feature extraction, making them a great candidate for NLP applications.

Transformer networks are very scalable due to their being easily parallelisable, which allows better distribution among Graphics Processing Units (GPUs). Furthermore, transformer blocks are easily stackable, allowing them to increase their effective representation size. This allows the networks to encode much more information and generalise on datasets with hundreds of millions of entries.

**Self-Attention:** The self-attention is one of the two main building blocks of the transformer. Self-attention provides a way for the transformer to compare elements of the input sequence and calculate their relevancy. For each word and the dependencies to other words, we calculate the attention resulting in the attention matrix, which provides transformers' global feature extraction ability, overstepping the traditional convolutional methods for linking long-range dependencies. Self-attention also allows transformers to extract inter-data dependencies from the moment the data enters the network for which traditionally pooling is used in convolutional networks, losing out on finer information.

$$\begin{aligned} Q &= \text{Query matrix} \\ K &= \text{Key matrix} \\ V &= \text{Value matrix} \\ d_k &= \text{Dimensionality of } K \\ Attention(Q,K,V) &= Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right) \cdot V \end{aligned} \tag{2.3}$$

The above operation and the three initial matrices ($Q$, $K$, $V$) are optimised during the training of the DNN.

**Feed-Forward:** Feed-forward layers are, at their core, densely connected network layers, essentially taking the self-attention's output and processing it for each position independently. The main goal of the feed-forward block is to apply non-linear transformations on the output of the self-attention, which is primarily a linear operation (Zhang 2023).

$$\begin{aligned} W_{ff} &= \text{The weights for the fully connected layer} \\ b_{ff} &= \text{The biases for the fully connected layer} \\ X &= \text{The input of the feed forward layer} \\ FeedForward(X) &= ReLU(W_{ff} * X + b_{ff}) \end{aligned} \tag{2.4}$$

## How does a vision transformer work?

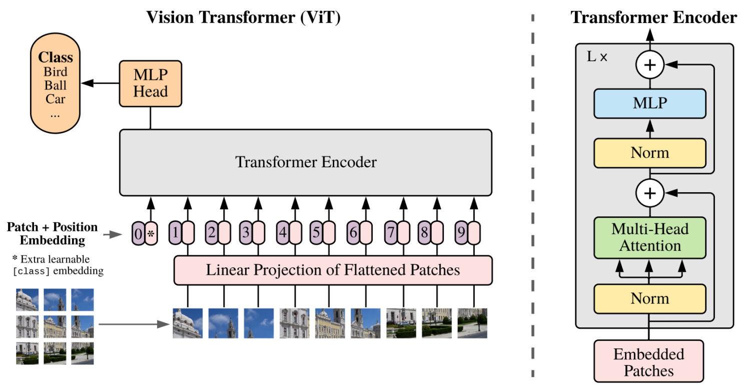

*Figure 2.5: Vision transformer architecture, showing linear projection, patch embeddings (Dosovitskiy et al. 2021).*

Vision transformers build on transformer networks; however, they use image patches instead of text tokens (the vector representation of words) and sentence vectors as input to the DNN. This allows vision transformers to, instead of processing the whole image using fully connected feed-forward blocks, use those image patches to first calculate the interrelationship between one patch, through the self-attention and the others and then only process a small section in relation to other segments, through the feed-forward. Vision transformers have no sense of space, so positional encoding is used for the network to learn where each image patch belongs in the image. Due to their ability to scale well, and as it is going to be explained later, the flexibility, such as the ability to manipulate or even remove patches while also providing state-of-the-art performance over image tasks, allowed vision transformers to become the main focus of computer vision research over traditional CNNs.

## 2.5 Masked learning

### Auto-Encoders

The idea of auto-encoders, which are neural networks (NN) that learn feature representations by first converting the image into a lower dimension and then attempting to rebuild it, has been used for a long time now for feature-learning purposes going back to Le et al. (2011). Auto-Encoders utilise the idea of bottlenecking (Scholz et al. 2005) to transform the data in a non-linear way into latent space, an alternative representation which is a compressed representation of the input data, such that the network learns much more robust mapping of input x to output y, and hence allowing the training of much larger networks. The idea of auto-encoders also allowed encoders to train on unlabeled data, thus creating the development stage of DNNs called pre-training and self-supervised learning.

### Encoder-Decoder design

The encoder-decoder design gained popularity, first being standardised in sequence to sequence tasks (Sutskever et al. 2014), then later applied in vision tasks. The idea behind the encoder-decoder design is that by training the encoder, which is the part of the network that turns the input into the latent space Figure 2.6, is trained in a self-supervised manner using an auto-encoder, while the decoder, the part of the network that turns the latent into useful output, is only trained for down-stream applications, reducing the labelled data required. During the training of the encoder (pre-training), the network is trained on generic data, allowing it to learn better general representations of objects (Hendrycks et al. 2019), improving the feature extraction abilities. After the pre-training, the decoder can be added or changed depending on the application.

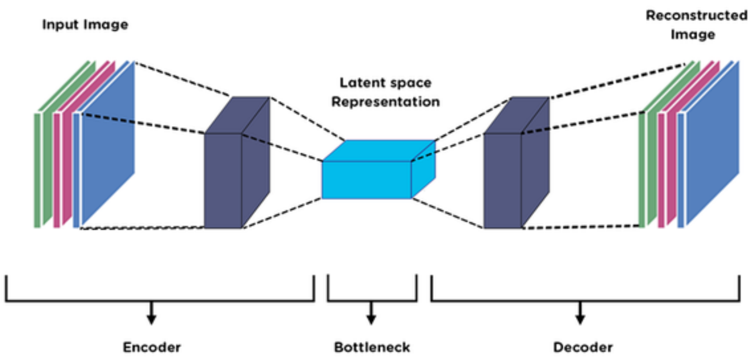

*Figure 2.6: The representation of the auto-encoder architecture (Birla 2019).*

### The masked-autoencoder

The masked auto-encoder (He et al. 2021) provided a revolutionary method for pre-training vision networks. As mentioned initially by the paper, the main problem when trying to train very large networks is that they require large amounts of data to train and even more data to scale, drastically limiting the learning capacity of vision networks. The masked auto-encoder builds on top of the previously mentioned self-supervised learning method originating from NLP but also incorporates another critical component from the NLP world from BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2018), which is the idea of masking some part of the input sequence. Hence, the model has to try regenerating it. The masked pre-training allowed the scaling of NLP networks into the 100 Billion parameter size. Thus, it is also trying to leverage the benefits of such training in vision applications with the introduction of vision transformers.

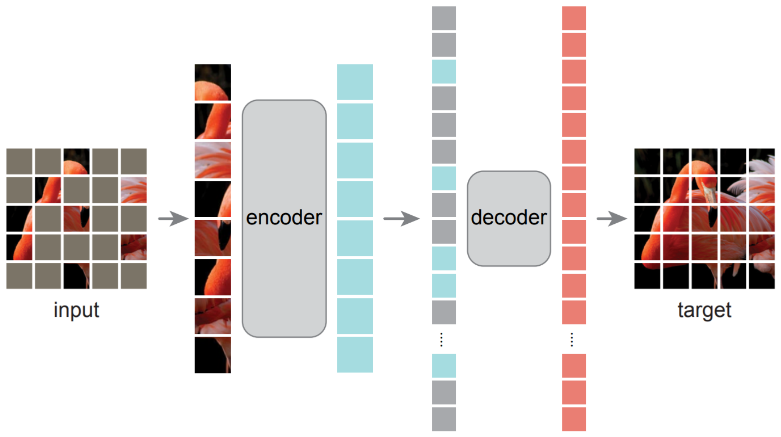

***Figure 2.7:*** *The masked auto-encoder (He et al. 2021).*

The masked auto-encoder study builds on top of the previously talked about ViT. It does not provide a new network architecture but an example of how current neural networks can benefit from NLP pretraining methods. The traditional ViT encoder network is paired with a smaller decoder and a transformer-based network in the masked auto-encoder. The 75%input of the encoder is randomly masked out 2.7 such that the encoder has no information on those missing patches, while the decoder only receives the latent representation for those patches, but also receives the number of masked patches such that it can regenerate the image from the latent space provided, and thus drastically increasing the reliance of interconnections of patches. The paper reported drastically faster learning times and higher accuracies using the masked auto-encoder.

## 2.6 Distillation with no labels

While the masked auto-encoders excel at image representation extractions, a new method of pre-training networks was released (Caron et al. 2021) by Meta FAIR focusing on an encoder network to be trained, rather than an encoder-decoder. This style of pre-training is called DINO (DIstillation with NO labels).

Traditionally, distillation in deep learning refers to the process whereby first training a very large DNN called the *'Teacher'*, that performs well on our data, we can 'Distill' that knowledge onto a smaller, more efficient DNN called the *'Student'* (Hinton et al. 2015). It has been shown by research the networks perform better with distillation than by training them from scratch Potrimba (2023), due to the output containing more information about the input data using the predictions of the network than just using the straight labels, which is likely due to the soft nature of the teacher's output.

However, DINO uses a different approach. Instead of using a larger and a smaller DNN, it uses the same network to perform the distillation, so there is no the traditionally mentioned *'Student'*

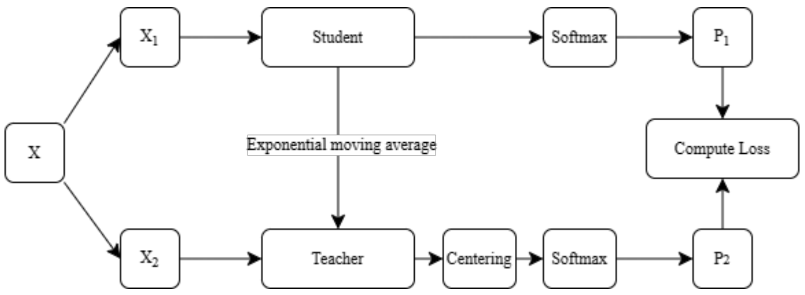

***Figure 2.8:*** *The high-level overview of the DINO trainer, showing the flow of operations. Caron et al. (2021)*

and *'Teacher'*. However, the relationship between the two networks still stands. The networks are trained at the same time, where the student network is the network that is being optimised by the optimiser. The teacher network acts as a copy of the student network, where the copied data is calculated with an exponential moving average 2.8, and uses special pre-processing to strengthen the teacher model's output features, which is what the student tries to replicate. The paper shows the state-of-the-art results achieved but, more importantly, shows that vision transformers can understand the subjects of the images without labels, creating much better latent representations and much smaller decoder networks. DINO revolutionalised the pre-training and contrastive learning domain, showing state-of-the-art performance with unparalleled inner representations.

## 2.7 Seeing the invisible

With vision transformers being in focus, as mentioned above, they have become the centre of attention for much of computer vision research. Their unparalleled global feature extraction abilities make them show, before unseen potential, rapidly closing the gap between vision transformers and state-of-the-art CNNs. Some earth observation research also started experimentation with vision transformers, many focusing on processing Hyper-Spectral images. As mentioned above, vision transformers are data-hungry and rely on embeddings to understand relationships between patches, such that many of the networks in hyperspectral data processing utilise a dual encoder model where one encoder is responsible for the encoding of the spatial data. The other encoder is responsible for the spectral data, cutting the data required for training and the compute time needed for training such a model (Chen et al. 2023). Remote sensing research has focused so far on how different architectures improve performance.

## 2.8 Gap

### Spectral training updated

As we described above, since the introduction of the vision transformer (Dosovitskiy et al. 2021), much of the research has started focusing on the benefits and potential of these networks. As we discussed, significant advancements in self-supervised training have left a gap between our current methods to train Hyperspectral optimised earth observation models and the state-of-the-art pre-training methods from which remote sensing applications could benefit.

# 3 | Methodology

## 3.1 Dataset

The dataset used for this experiment was the WHU OHS (Li et al. 2022) hyper-spectral segmentation benchmark dataset. The dataset was initially constructed to help the growing need for hyper-spectral data for remote sensing-related tasks. The dataset comprises 25 classes and 32 spectral channels ranging from 466*nm* to 940*nm* as shown in 3.1.

| band no. | nm | band no. | nm | No. of bands | nm | band no. | nm |
|---|---|---|---|---|---|---|---|
| 1 | 466 | 9 | 569 | 17 | 716 | 25 | 836 |
| 2 | 480 | 10 | 610 | 18 | 730 | 26 | 850 |
| 3 | 500 | 11 | 626 | 19 | 746 | 27 | 866 |
| 4 | 520 | 12 | 640 | 20 | 760 | 28 | 880 |
| 5 | 536 | 13 | 656 | 21 | 776 | 29 | 896 |
| 6 | 550 | 14 | 670 | 22 | 790 | 30 | 910 |
| 7 | 566 | 15 | 686 | 23 | 806 | 31 | 926 |
| 8 | 580 | 16 | 700 | 24 | 820 | 32 | 940 |

***Figure 3.1:*** *The spectral channels corresponding to the central wavelengths.*

| ID | Category | ID | Category | ID | Category |
|---|---|---|---|---|---|
| 1 | Paddy field | 9 | Low-covered grassland | 17 | Other construction land |
| 2 | Dry farm | 10 | River canal | 18 | Sand |
| 3 | Woodland | 11 | Lake | 19 | Gobia |
| 4 | Shrubbery | 12 | Reservoir pond | 20 | Saline-alkali soil |
| 5 | Sparse woodland | 13 | Beach land | 21 | Marshland |
| 6 | Other forest land | 14 | Shoal | 22 | Bare land |
| 7 | High-covered grassland | 15 | Urban built-up | 23 | Bare rock |
| 8 | Medium-covered grassland | 16 | Rural settlement | 24 | Ocean |

***Figure 3.2:*** *The labels of the dataset and their corresponding index.*

The dataset's images were captured in forty locations in different regions of China and published by Wuhan University. The dataset initially contains $9,000$ labelled images, each with a resolution of $512 \times 512$, where 5000 of those samples are allocated for training, 3500 is to test, and 500 to validate. It is important to note that due to the nature of the research reported in this dissertation, the pre-training of networks, and the requirements that would have had in terms of compute power on high-resolution data, the dataset had to be cut up into smaller pieces of images, providing approximately $300,000$ images.

## 3.2 Networks

### Vision Transformer (ViT)

The ViT architecture, initially introduced in Dosovitskiy et al. (2021), served as the backbone for the study. The architecture was chosen due to its simplicity in the vision transformer domain,

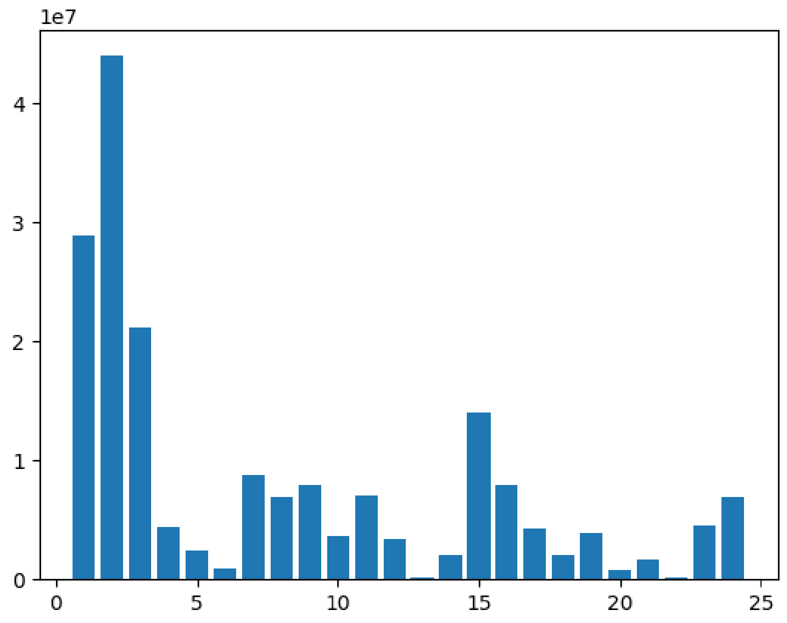

***Figure 3.3:*** *Class labels excluding background class. The labels are positioned by their ID such that refer to the table above to identify 3.2*

which allows faster experimentation. Furthermore, the architecture is widespread among studies focusing on pre-training vision transformers such as the DINO and Masked Auto-Encoder. As our research aims, using the ViT is more than justified, providing a baseline for evaluating the performance of the networks. The architecture was modified to use 32 input channels instead of the originally used 3.

| **Hyperparameters** | **value** |
|---:|:---|
| No. of transformer blocks | 12 |
| No. of attention heads | 12 |
| Patch size | 8 (Total number of patches: 64) |
| Embedding dimension | 768 |
| Input dimensions | 64 x 64 (pixels) |
| Input depth | 32 channels |
| **Hyperparameters used only in the second experiment** | |
| Number of register tokens | 4 |
| Feed-Forward layer | SwigluFused |

***Figure 3.4:*** *Hyperparameters of the ViT encoder*

The smaller input size (64 compared to the introduced initially ViT-B, which had an input size of 224) was chosen so that a larger batch size could fit into the memory of the used GPUs, speeding up the training speed and also to try mitigate the problem of the limited available data mentioned above, it is essential to say that due to the drastic drop of image size (over $\times$12.25), the embedding size left unchanged to accommodate for the additional complexity over the increased depth of data (over $\times$10). Patch size eight was chosen to improve the network's accuracy at the cost of increased memory usage, which would not have been possible at the original image size. The original paper evaluated and found that a smaller number of patches increases the overall accuracy of the network. The rest of the parameters were left, as it was in the original paper.

As shown in 3.4, some parameters and the original ViT parameters were used in the second experiment. This is due to the changed trainer, from a masked auto-encoder to a DINOv2 trainer; the register tokens help in the stabilisation of self-distillation-based self-supervised learning (Darcet et al. 2023), while the SwigluFused Feed-forward network layer (FFN layer), changes the original Multi-Layer Perceptron (MLP) in the FFN layer achieving higher performance (Oquab et al. 2024).

## Spatial-Spectral Vision Transformer

The above network was modified, based on a paper on HSI data sharpening (Chen et al. 2023), for the last experiment to use a new encoder design, such that it can leverage the depthwise information in the HSI data. This encoder tries to learn the relation spatially and across the spectral channels such that the network utilises two encoders, one for extracting the spatial information similarly to how the ViT extracts information from images and a spectral encoder that extracts information along the spectral channels 3.5.

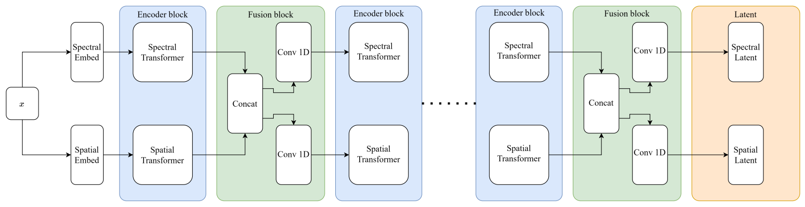

***Figure 3.5:*** *Spatial-Spectral encoder network. Green boxes mark the fusion blocks 3.6, while blue boxes mark the group of encoder blocks*

In between the encoders' processing blocks, the features are concatenated and fused. Fusion is done through a 1x1 convolution layer mapping the concatenated state of features to the input of the following processing block.

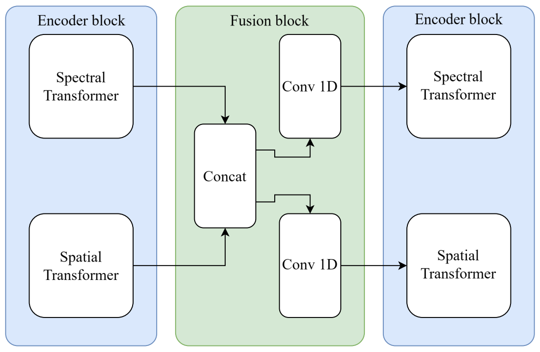

***Figure 3.6:*** *Spatial-Spectral Feature fusion block*

In order to fit the patch sizes, so in the case of the spectral transformer, one spectral channel, its dimensionality is reduced using 1x1 convolution to fit the embedding size that matches the spatial-encoders embedding size and therefore input shape, dropping the dimensions of a channel by 5.3 times. This is both used to increase memory efficiency and also to allow the concatenation of the data during fusion.

| Hyperparameters | value |
|---|---|
| No. of transformer blocks | 12 |
| No. of attention heads | 12 |
| Patch size | 8 (Total number of patches: 64) |
| Spectral Patch size | 2 (every two channels is grouped, Number of patches: 16) |
| Embedding dimension | 768 |
| Spectral Embedding dimension | 768 |
| Input dimensions | 64 x 64 (pixels) |
| Input depth | 32 channels |
| Number of register tokens | 4 |
| Feed-Forward layer | SwigluFused |
| Total number of tokens: 90 (64 spatial + 4 register + 1 CLS, 16 spectral + 4 register + 1 CLS) | |

***Figure 3.7:*** *Hyperparameters of the Spatial-Spectral transformer*

## 3.3 Trainers

### The masked auto-encoder

The masked auto-encoder (MAE) was among the most important studies that have affected how vision transformers are being trained nowadays. Building on the novel mechanics of the ViT, the Masked Auto-Encoder utilises random masking of the input, around 75% of it (similarly to the BERT sequence model (Devlin et al. 2018)), such that the encoder network in the Auto-Encoder only receives a limited amount of data, emphasising the usage of different paths in the network. The decoder gets the number of patches masked but no more information about them, using masking tokens, which are appended to the series of tokens received from the encoder. The objective is to regenerate the missing patches from the existing patches of the Auto-Encoder. The paper shows higher accuracies and significantly faster training times. The MAE in this study is to be used as the baseline pre-training pipeline specifically built for transformer networks. This is further supported by the fact that the backbone network for the MAE is the originally introduced ViT. Thus, the changes are focused on the encoder's pre-training.

Preprocessing has also been done on the input during training, which, following the original paper, was normalisation of the image data, random cropping, which essentially randomly selects a ratio to zoom into the image, and random flip; however, the trainer would not require by design the random flip augmentation, it has been added to increase the variability of the dataset.

### DINOv2

As mentioned in previous sections, DINO marked a significant leap in the domain of pre-training vision transformers. DINO heavily relies on data augmentation, compared to the MAE, which almost requires none. As a self-distillation trainer, DINO showed that vision transformers can pick up features without labelling the data and focusing on essential details. The importance of those features is that without any particular labelling of the data and given enough training samples, the network learns to understand the subject of an image by exploiting the fact that images are most often taken of a subject.

DINOv2 builds on top of the idea of the original DINO by further enhancing the self-distillation-based training. However, unlike the original DINO, it utilises a new discriminative objective function that uses both an image level (DINO head) and patch level (iBOT head) objective. DINO head utilises the CLS tokens of both the teacher and student network, which are then sent through the DINO head, outputting a vector of scores, or as the paper calls it *"prototype"* scores, on which a softmax and finally a centring is applied. In our experiment, the Sinkhorn-Knopp centring (Ruan et al. 2023) replaces the original softmax centring of the DINO. Patch level loss is calculated using the iBOT head, one in which random masking is applied to some percentage of the patches for the student network, such that the objective of the self-distillation, unlike the DINOs feature replication procedure, is to regenerate the missing latent space patches of the teacher module (Zhou et al. 2022) Figure 3.8.

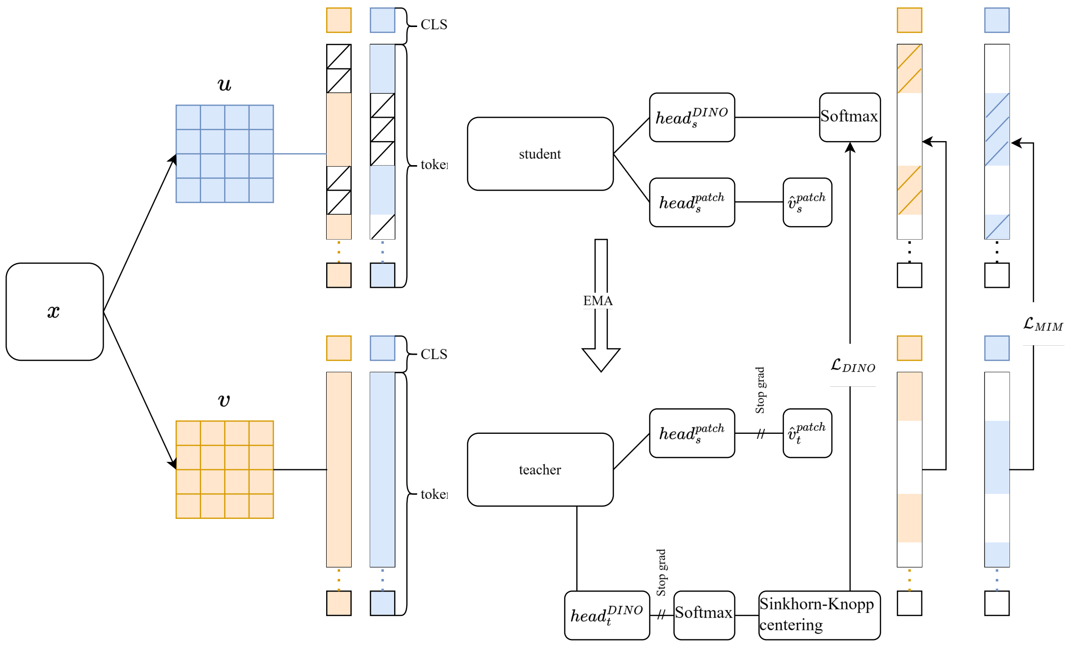

*Figure 3.8: DINO v2 trainer high-level diagram, showing the links between the different losses and loss heads such as the iBOT and DINOv2 head.*

During training, the parameters of the student are copied over to the teacher using an exponential moving average (EMA). As seen in Figure 3.8, the loss for the iBOT head is acquired through the patch regeneration and for the DINO head through the CLS token. We can define the objective functions in the following way:

Patch level, regenerative objective:

$$\begin{aligned} \theta' &= \text{parameters of the teacher} \\ \theta &= \text{parameters of the student} \\ \boldsymbol{u} &= \text{Non masked view of image } x \\ \hat{\boldsymbol{u}} &= \text{Masked view of image } x \\ \boldsymbol{v} &= \text{Non masked view of image } x \\ \hat{\boldsymbol{v}} &= \text{Masked view of image } x \\ N &= \text{Number of tokens} \\ P(\cdot) &= \text{Transforms input to a probability distribution} \\ \mathcal{L}_{MIM} &= -\sum_{i=1}^{N} m_i \cdot P_{\theta'}^{\text{patch}}(\boldsymbol{u}_i)^{\mathrm{T}} \log P_{\theta}^{\text{patch}}(\hat{\boldsymbol{u}}_i) \\ \mathcal{L}_{MIM} &= -\sum_{i=1}^{N} m_i \cdot P_{\theta'}^{\text{patch}}(\boldsymbol{v}_i)^{\mathrm{T}} \log P_{\theta}^{\text{patch}}(\hat{\boldsymbol{v}}_i) \end{aligned} \tag{3.1}$$

Image level objective:

$$\begin{aligned} p_t &= \text{Prototype scores obtained from teacher} \\ p_s &= \text{Prototype scores obtained from student} \\ \mathcal{L}_{DINO} &= -\sum p_t \log p_s \end{aligned} \tag{3.2}$$

These objective functions provide the basis for DINOv2, which was used in the experiment and modified in a later one; more on that in section 3.3. Due to the nature of DINOv2 and many other contrastive learning methods, training large vision models also introduces artefacts in the attention. For those cases, register tokens help mitigate that problem by introducing a token covering additional information about the input's global state, assisting in stabilising the training. Even though our network is not amongst the networks that are in danger of forming artefacts, for better scalability, 4 register tokens are included in the DINOv2 trained ViT backbones Figure 3.4 Figure 3.7. As mentioned above, the DINOv2 ViT backbone, similarly to the MAE ViT backbone, was adapted to operate on 32 input channels instead of the originally used 3, requiring the change of most pre-processing functions. One additional benefit included in DINOv2 is the usage of local and global crops, which essentially provides two separate views, a small and large image for the network to learn on, allowing for later a change in resolution to higher dimensions, significantly speeding up training times compared to, only high-resolution pre-training.

## DINOv2 on HSI data

However, DINOv2 required additional modifications to accommodate the proposed spatial-spectral backbone to exploit the increased data depth in our HSI data adequately. To make the backbone swap possible, the pre-processing step of DINO had to be further updated, as now global and local data crops have to be compared both in spatial and spectral view of the data, to ensure that iBOT loss is calculated during training, an additional masking generator is added to the trainer to provide masking on the spectral patches, as seen in Figure 3.5.

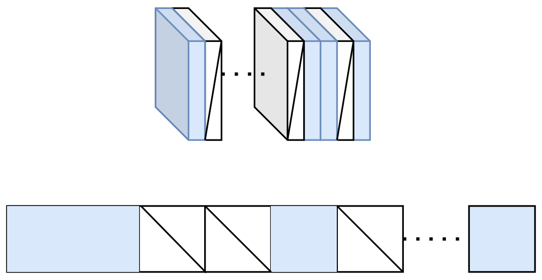

*Figure 3.9: Spatial spectral masking for the two encoder modules.*

On top of the modifications mentioned above, the output of the spectral encoder, similar to the spatial encoder (The original ViT), was sent through both the DINO head and the iBOT loss head, providing additional information on the learnt spectral features. The losses are combined afterwards to be used in the optimisation process. As mentioned in Figure 3.7, both encoder modules use register tokens 4-4 each for feature stabilisation, giving 69 tokens for the spatial encoder and 21 tokens for the spectral encoder.

| Trainer | Encoder | Complexity | Parameters | Total parameters |
|---|---|---|---|---|
| MAE | ViT-B | 11.4 GFlops | 113,378,560 | 113,378,560 |
| DINOv2 | ViT-B | 12.1 GFlops | 109,853,440 | 219,706,880 |
| DINOv2 | HSI-ViT-B | 15.8 GFlops | 202,915,984 | 405,831,968 |

***Figure 3.10:*** *Trainer, and encoder network size, and complexity comparison.*

## 3.4 Downstream

The latent space generated by the Masked Auto-Encoder, or DINOv2, is a compressed input data representation. While this representation is crucial for achieving high downstream performance, there is no straightforward way to evaluate it due to the lack of human-understandable features inside the latent space. Other evaluation methods have to be used, including regeneration loss (which, in our case, is not applicable), downstream evaluation (used in this dissertation), and visual evaluation of the latent space. For evaluation, we used a semantic segmentation task to evaluate the encoders.

### Semantic Segmentation

Semantic segmentation is a complex task for any DNN to perform, and the output dramatically depends on the quality of the latent features. This is because semantic segmentation is a pixel-level task. For the performance to be good, the network needs to be aware of the inter-pixel dependencies, such as the output depending on that specific pixel and all the pixels neighbouring it. On top of this, the difficulty level shifts dramatically in satellite applications. Even though there are fewer classes to identify, at least in our dataset, the ambiguity of classes in terms of shape and size and seasonal variation in the images all introduce an additional layer of difficulty. Semantic segmentation is an excellent downstream task to evaluate on due to the task's heavy reliance on the latent space.

**Linear Segmentation Decoder** The implementation is very limited in size. The only learnable parameters in it are the classification layer, such that the 1x1 convolution produces the logits of the segmentation decoder. The segmentation decoder consists of one resize operation Figure 3.11 using the outputs of the last four encoder blocks, then concatenates the patch tokens and performs the upsampling.

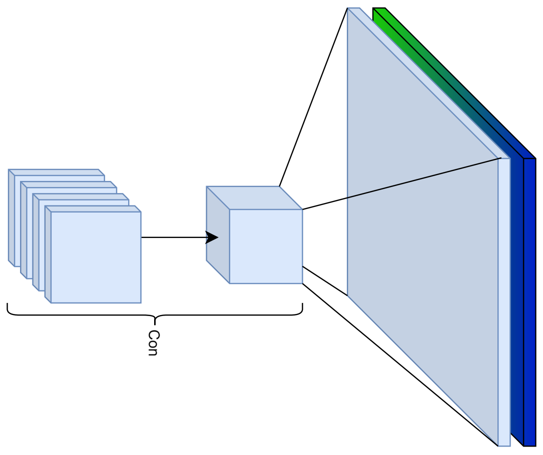

***Figure 3.11:*** *Multiscale Linear decoder, high-level illustration.*

**Convolution Segmentation Decoder** This basic convolution segmentation decoder works similarly to a UNet (Ronneberger et al. 2015) decoder and uses the output of the encoder blocks as skip connections for the upsampling. The features in between are upscaled using multiple steps of upscaling with deconvolution and convolution between deconvolution layers.

**Transformer Segmentation Decoder** This segmentation decoder was based on the Strudel et al. (2021) paper. The idea behind the segmenter is it uses a transformer-based decoder to perform the segmentation compared to the previously mentioned approaches while showing state-of-the-art performance. Our implementation used a decoder depth of 2 with 12 attention heads, making a reasonably complex decoder to capture more complex information.

**DeepLabV3 plus Degmentation Decoder** This decoder was only used for the evaluation of our baseline CNN network, which used a ResNetV1c as its backbone, and not to perform any additional review. The segmentation decoder uses a particular module called atrous spatial pyramid pooling module (ASPPM)(Chen et al. 2018), which uses different stages of dilated convolution, essentially leaves a gap in the convolution kernel, broadening its coverage without introducing new parameters Figure 3.12, and upsampling before receiving the final layer; those upsampled stages are concatenated together and used to produce the logits for the final result.

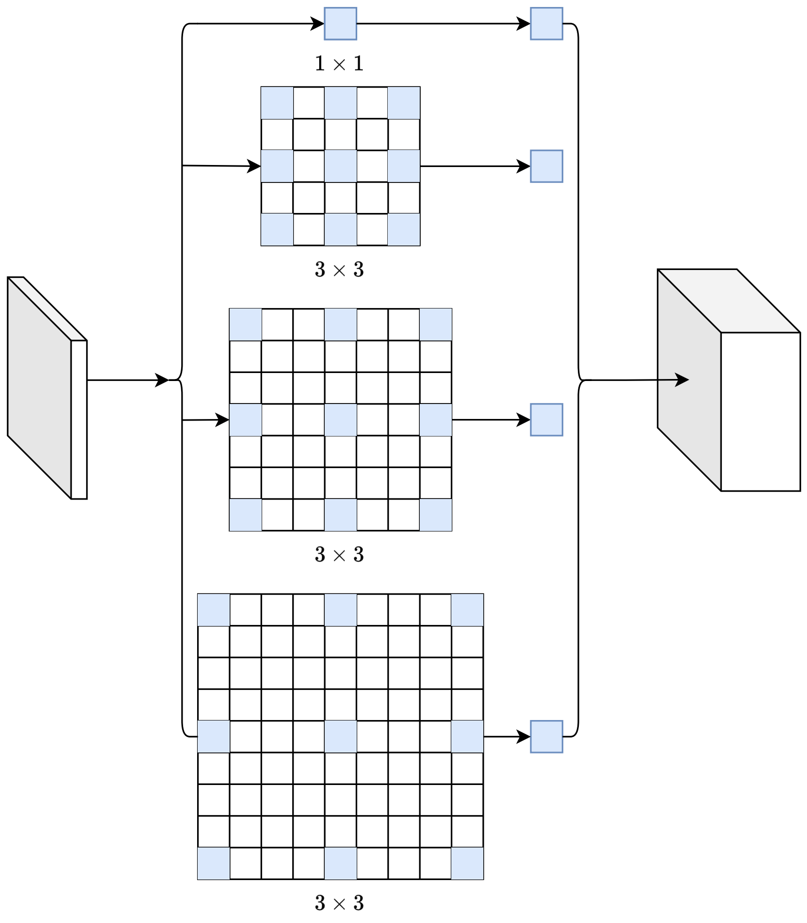

*Figure 3.12: Atrous Spatial Pyramid Pooling Module used in DeepLabV3plus, demonstrating dilated convolution.*

# 4 Experimental design

The main goal of the experiments was to systematically evaluate the performance of encoder networks trained using different self-supervised trainers, using the DINOv2 as mentioned earlier and MAE, and whether they improve performance overall. Due to the lack of straightforward options in evaluating pre-trained encoders, the pre-training performance will be reviewed on a downstream segmentation performance using a set of defined hyper-parameters and metrics shared across the experiments.

For each training run, seeds were set (explained later in section 4.1), run in a containerised environment, and logged by weights and biases (a data logger website (wandb 2024)), including the configuration used for each run. This is to support the experiments' reproducibility in terms of the libraries' randomness and to make it easier to load and run the experiments and retrieve their results, thanks to the containers and backed-up hyper-parameters.

## 4.1 Experimental setup

### Proof of concept

The first step in our experimentation process was to create a proof of concept. Adapting the Masked Auto-Encoder for the new input data provided the the proof of concept in this scenario. Firstly, we used pre-trained weights for the encoder and Principal component analysis to reduce the number of channels of the input images from 32 to 3 and trained the decoder to generate that back. Secondly, the Masked Auto-Encoder was adapted to use 32 input channels and 32 output channels to learn the representation of the HSI data without using any down-sampling method. This way, we could ensure that the MAE's encoding capacity is adequate and such there are enough parameters to represent the HSI data in the latent space.

### Masked Auto-Encoder based training

For this experiment, the Masked Auto-Encoder was used to pre-train the ViT-B, and then the network was further tuned for downstream evaluation. The MAE network was trained for 100 epochs, while the dropout was set to 0.1 to avoid overfitting on the small amount of available data. The input data was $64 \times 64$ to preserve memory, allowing the increased batch size of 128 and, as mentioned above in Figure 3.4, increase the patch size of the network.

After the pre-training, the DNN was fine-tuned on segmentation data using the decoder modules, as mentioned earlier, the linear segmentation decoder, convolutional segmentation decoder, and transformer segmentation decoder, on similarly $64 \times 64$ images to produce the final result. Both during pre-training and during fine-tuning, the following seeds were set:

```python
import random
import torch
import numpy as np

def train():
```

```
'''
Function representing the main pre-trainer
function structure for the MAE
'''
torch.manual_seed(0)
np.random.seed(0)
random.seed(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

# initialise data loaders

# initialise model

# Initialise criterion, and optimiser

# train, eval, checkpoint

# save network
```

This was done to improve the reproducibility of our findings. Data augmentation during training has been limited to normalisation, random flip, and resize-crop.

### DINOv2 based training

For this experiment, DINOv2 was used to pre-train the ViT-B backbone. Similarly to the MAE-based experiment, the network was further trained on downstream segmentation for evaluation (the actual models are frozen during evaluation; only the decoder runs with gradient calculations).

The pre-training was done on batch size 128 for 100 epochs over the cut-up dataset, using the same seed configuration as before to support reproducibility. At the end of the pre-training, the resolution of the network was changed to $512 \times 512$, similarly to the original study, to improve the performance of the backbone on downstream segmentation.

Augmentation in this experiment was more extensive, including geometric transforms and colour transforms, such as random flip, resize-crop, colour jitter, random solarise, Gaussian blur, and normalisation. All of them have been reimplemented to work on Hyperspectral image data.

### DINOv2 based spatial-spectral training

This experiment used the hyperspectral-adapted DINOv2 to pre-train the Spatial-Spectral ViT-B backbone. Similarly to the previous two experiments, the results are also further trained on downstream tasks for evaluation (The actual models are frozen during evaluation, and only the decoder runs with gradient calculations).

The pre-training was done on batch size 64 for 100 epochs over the cut-up dataset; this change in batch size was introduced compared to the previous networks due to the increase in the size of the now two utilised encoder networks, effectively doubling the size of the encoder module. The experiment uses the previously mentioned seed configuration to support reproducibility. In this case, the upscaling mechanism was not in use due to the results of the previous experiment; more on that later section 5.1.1. Similarly to the previous experiment, augmentation enhanced the data further. However, it had to be further modified to work on both the Spatial and the Spectral encoder. The same is true for the losses used in DINOv2, for which both pre-processing and loss calculation had to be updated to work on both encoder modules.

### Baseline evaluation

To relate the experiment to previous experiments done in the domain, a baseline measurement was also performed. This provides a clear understanding of the dataset's difficulty and how the networks' performance scales to external datasets. The baseline was measured on a ResNetV1c-50 backbone using a DeepLabV3plus decoder architecture and an FCN decoder for auxiliary output. This guides the efficient training of the latent space. The network was trained on uncut images with a total resolution of 512 × 512, with a batch size of 8, which had to be done to eliminate the root cause of problems with the small image size and thus to fit in memory. Seeds were set similarly to the previous experiments for reproducibility.

## 4.2 Evaluation

### Linear Segmentation Decoder

One stage of the evaluation used the linear segmentation decoder. Due to its very low complexity, it provides a comprehensive overview of how much the features in the network represent easily classifiable features. The goal was to segment 24 classes without the background label, as it is commonly excluded in land-cover classification tasks. The segmentation decoder takes the output of the last four blocks of the ViT encoder and uses the concatenated and then upsampled feature map to perform the pixel-wise classification.

The performance was evaluated using standard metrics used in segmentation downstream evaluation, such as (mean) intersection over union ((m)IoU), mean accuracy and overall accuracy. Still, it was also evaluated qualitatively on visualised attention maps of the encoder networks. The latter is only valid for transformer-based experiments.

### Convolution Segmentation Decoder

The convolution segmentation decoder provides a more traditional approach to performing the segmentation. Unlike the previous solution, this decoder performs the upsampling in multiple separate stages. It uses the output of the last four blocks of the ViT, similarly to before, during upsampling. All convolution layers in the decoder used 3×3 convolution, reducing computational requirements. The goal was to segment 24 classes, with the background label not included.

Similarly, the performance was evaluated on (m)IoU, mean accuracy, and overall accuracy.

### Transformer Segmentation Decoder

The transformer segmentation decoder presented an alternative approach to performing downstream segmentation with increased complexity and learning ability. The segmentation decoder used 2 transformer blocks with 12 attention heads segmenting 24, with the background label not included. The performance was measured similarly to before.

| ID | Encoder | Decoder | Learning rate/schedule | Resolution |
|---|---|---|---|---|
| 1 | ViT - MAE | Multiscale Linear | Poly schedule/0.0001 | 64 × 64 |
| 2 | ViT - MAE | Multiscale Convolutional | Poly schedule/0.0001 | 64 × 64 |
| 3 | ViT - MAE | segmentation Transformer | Poly schedule/0.0001 | 64 × 64 |
| 4 | ViT - DINOv2 | Multiscale Linear | Poly schedule/0.0001 | 64 × 64 |
| 5 | ViT - DINOv2 | Multiscale Convolutional | Poly schedule/0.0001 | 64 × 64 |
| 6 | ViT - DINOv2 | segmentation Transformer | Poly schedule/0.0001 | 64 × 64 and 512 × 512 |
| 7 | HSI-ViT - DINOv2 | Multiscale Linear | Poly schedule/0.0001 | 64 × 64 |
| 8 | HSI-ViT - DINOv2 | Multiscale Convolutional | Poly schedule/0.0001 | 64 × 64 |
| 9 | HSI-ViT - DINOv2 | segmentation Transformer | Poly schedule/0.0001 | 64 × 64 |

***Figure 4.1:*** *Training parameters.*

## 4.3 Build, Train, Test cycle

To develop the modules, training pipelines, and networks quickly, it was essential for the project to have an established system through which work was distributed. This includes separate stages of ML development, such as prototyping, data exploration, and large-scale training, as well as CI/CD environments to automate build processes.

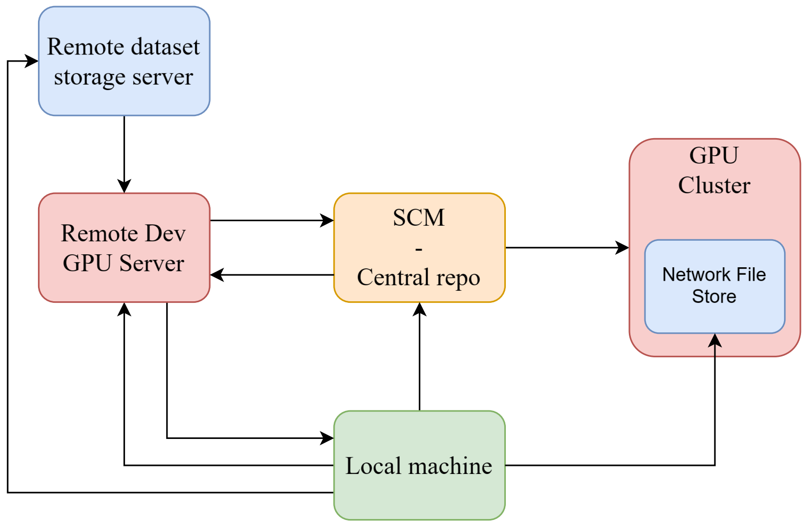

***Figure 4.2:*** *High-level architectural overview of environments.*

### Prototyping

In order to allow the quick exploration of modules, functions, and data, a GPU server was used with a remote Jupyter kernel, allowing fast iterations of elements due to the global state of Jupyter notebooks and easy data exploration for similar reasons. Each of the networks, both for pre-training and fine-tuning, was tested on the prototyping server before it was pushed for a large-scale training job to reduce development time. The prototyped codebase was synced with a Source Control Management (SCM) system.

### Continuous Build

To reduce the manual work required to start a training job, we utilised the events available on the University's GitLab instances. The SCM was set up in a way when there was a push received to the central repository it sent a webhook to the Computing cluster to request a build for a training docker container built on a GPU such that the correct code binaries are being compiled for the project's dependencies. The latter was required due to the specialised modules used for the experiment, such as deformable attention and other non-standard transformer blocks contained in MMCV (openmmlab 2024) (a computer vision library for benchmarking). As the containers are being built on the Computer Cluster, images do not have to be pulled from docker hub but can be pulled from the cluster's built-in image store, allowing the fast iterative updates of the containers. As such, the training code base can also be embedded into the container.

### Training job

As the codebase and dependencies were stored in the container, executing the training job required specifying the entry point and the resources necessary. As mentioned above, all jobs were logged to weights and biases to log training runs, and seeds were set in all jobs to make the runs reproducible. This allowed all the experiments to be tracked, giving a very large table of results to filter over.

# 5 Results

The above experiments shed some light on how different architectures and encoders perform on the WHU-OHS dataset, and even though the results are not great, there are a lot of hidden details in them that are worth paying attention to. This information is most interesting for comparing the relative performance of these transformer networks to existing methods, establishing minimum viable performance and state-of-the-art results.

## 5.1 Segmentation performance

### Performance on the WHU-OHS

As mentioned previously, the WHU-OHS remote sensing dataset is challenging for neural networks in the hyper-spectral domain.

Our proposed adaptation of the trainer and Spatial-Spectral ViT show deficient performance, as can also be seen in Figure 5.1. Concerning the original paper published with the dataset, their proposed mean intersection over union mIoU performance on 1D CNN was 22.7% on a 3D CNN 30.5, and their state-of-the-art 48.0% Achieved with FreeNet. However, our networks struggled with the segmentation of this dataset even after increasing the number of parameters. This was most obvious in the underrepresented classes of the dataset, as none of the networks could predict those correctly.

| ID | Encoder | overall Acc | mIoU | mean Acc |
|---|---|---|---|---|
| 1 | ViT - MAE | 11.86 % | 1.35 % | 3.60 % |
| 2 | ViT - MAE | 12.26 % | 1.21 % | 3.03 % |
| 3 | ViT - MAE | 11.73% | 1.19 % | 3.50 % |
| 4 | ViT - DINOv2 | 12.50 % | 1.40% | 3.55% |
| 5 | ViT - DINOv2 | 11.08 % | 1.03 % | 3.11 % |
| 6 | ViT - DINOv2 | 11.01 % | 1.58% | 3.54% |
| 7 | HSI-ViT - DINOv2 | 11.12 % | 1.16% | 3.85% |
| 8 | HSI-ViT - DINOv2 | 12.89 | 1.45% | 3.45 % |
| 9 | HSI-ViT - DINOv2 | 13.15 % | 1.34% | 3.14% |
| 10 | DeepLabV3plus | 12.56 % | 1.49% | 3.42% |

***Figure 5.1:*** *Performance of different encoder networks, using different trainers, and decoders (refer ID to fig. 4.1, for exact decoder).*

This behaviour is further accentuated when comparing the segmentation decoders' performance to the reported performance in the DINOv2 paper, which achieved an impressive 53.0% mIoU and 86.2.0% close to state-of-the-art on the Pascal VOC dataset (Everingham et al. 2010). This shows that none of the models with any decoders could capture the information well, including the baseline. And such suggests some underlying problems explained section 5.3 and section 6.1.

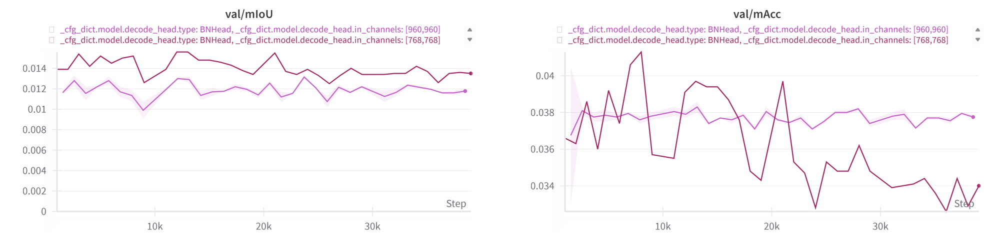

***Figure 5.2:*** *Linear layers performance over time, on ViT (bordeaux) and Spatial-Spectral ViT (Lighter pink).*

| Backbone | overall Acc | mIoU | mean Acc |
|---|---|---|---|
| ResNet50 $64 \times 64$ | 12.56 % | 1.49% | 3.42% |
| ResNet50 $512 \times 512$ | 11.36 % | 1.51 % | 3.50% |

***Figure 5.3:*** *Segmentation performance of the baseline network at different resolutions.*

### 5.1.1 Effects of Resolution

As segmentation is a pixel-level task for the model, the resolution at which this task is performed drastically affects the outcome of the experiments. As mentioned, this resolution had to be dropped dramatically to adapt to the computing requirements, which likely impacted the results. However, it is essential to note that even with the scaling of the resolution of the baseline from $64 \times 64$ to $512 \times 512$, the metrics did not improve, as shown in Figure 5.3. This behaviour was also observed with other backbones after performing high-resolution training on the ViT backbone on DINOv2 trainers in the last pre-training stage. This was the leading cause of stopping the high-resolution experimentation on the other backbones and purely focusing on how data processing and the data itself have underlying issues.

### 5.1.2 Feature representation

Using the lightweight linear classifier, we can similarly extract segmentation maps, showing the features' usability directly due to the minimal size of the decoder (only using the pixel-level class segmentation decoder). As it is visible in Figure 5.4, with the MAE output, the classifier could not pick up the underlying features in the encoder. However, the same is not true about either model using DINOv2. Looking at the outputs of DINOv2, we can see some noticeable differences in the networks predicting features resembling the ground truths structures and objects. And while the performance shown in Figure 5.1 is not close to the reported results in Oquab et al. (2024), this behaviour highlights how the DINOv2 was tuned originally to learn visual features and structures which are optimised for KNN inference performance. It is safe to assume that the underlying distribution representing the input in the latent space is much more representative of the input data than the features stored in the masked Auto-Encoder while still being more useful in downstream tasks.

### 5.1.3 Impact of Objective

We explored multiple loss functions during experimentation to help the model converge. Due to the significant variance between the features, the objective function impacts the outcome. This variance shows both in between the classes and their visual features, but also due to the number of samples available for each one of these classes as it is visible in Figure 3.3. This effect was mainly observed using Focal loss over cross-entropy loss.

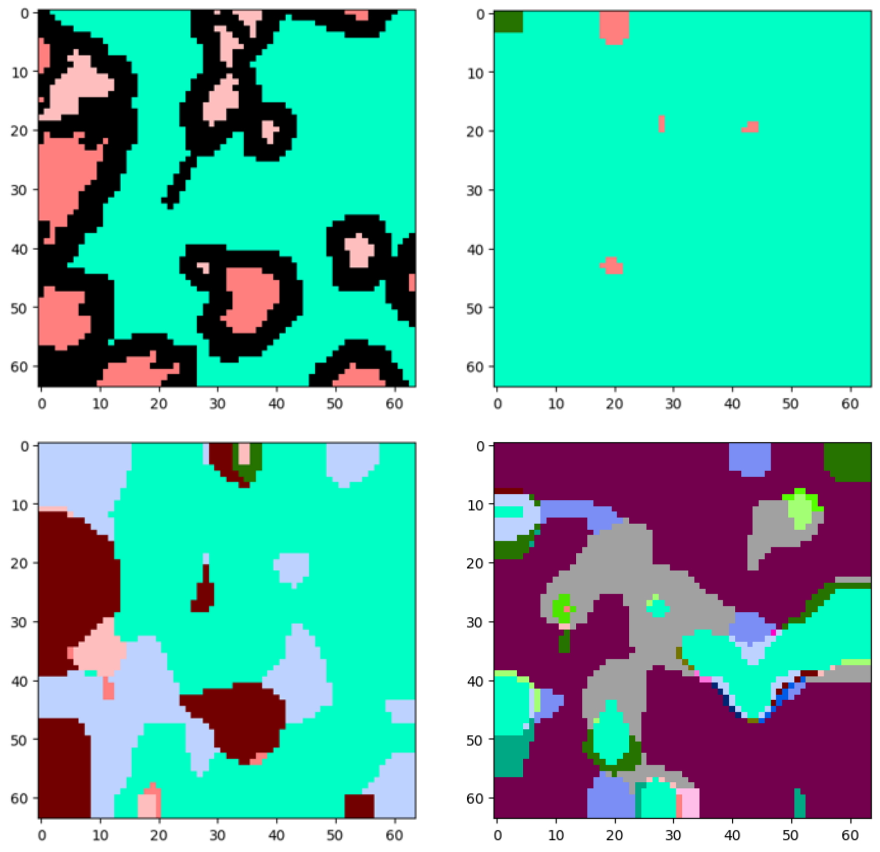

*Figure 5.4: Images showing segmentation mask for all backbones with the linear multiscale classifier (top left: ground truth, top right: MAE, bottom left: DINOv2, bottom right: HSI-ViT DINOv2).*

It was observed that during tuning downstream, some experiments achieved better performance than others; however, because the experiments were all performed only within the margin of error, the objective functions were not included in this evaluation explicitly.

## 5.2 Visualised attention

The encoders were also evaluated based on the output of their attention map. One big benefit of this type of evaluation is that, unlike the previous evaluation, which had to be performed downstream and hence largely dependent on the labelling of the data, it shows direct information from the last attention module of the encoder, supplying useful information on what sections the model considers useful in the image and what does not. Looking at the attention maps, shown at Figure 5.5, of the three backbones, we see significant differences in what features in the input each model considers useful based on their relevancy. Overall, the attention map of the DINOv2 trained - ViT is the worst, only showing minimal attention to most of the details, considering sections not relevant to each other, which is not true for either the MAE or the DINOv2 Spatial-Spectral ViT, where the first one shows more associations between patches. Meanwhile, the latter shows strong association between the patches while preserving solid boundaries between the related regions.

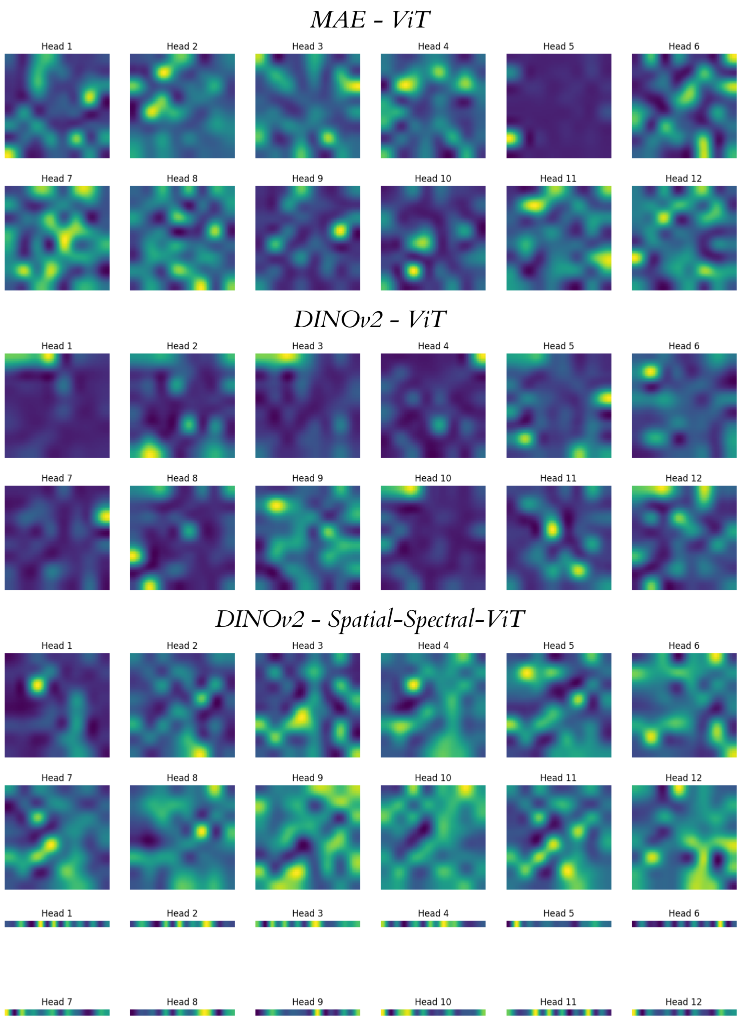

*Figure 5.5: Visualised attention of the MAE, DINOv2 ViT and Spatial-Spectral -ViT backbone (Top MAE-ViT, middle DINOv2 - ViT, DINOv2-Spatial-Spectral-ViT).*

## 5.3 Causes of problems

In the above sections, we highlighted the limitations of the backbones, used and pre-trained with different methods, and such that it is important to evaluate what went wrong during the experiment and did not work. As also mentioned above, there were many root causes identified:

**Resolution** As mentioned before, for testing on semantic segmentation downstream, the results are significantly impacted for mIoU metric at different resolutions, which could have caused additional problems during classification.

**Compute requirements** The pre-training of such networks requires tremendous compute power. As mentioned before, the encoders were each trained for 100 epochs. However, neither of the original studies used that number but used 800 and 500 for the MAE and DINOv2, requiring $\times 8$ and $\times 5$ the training time, respectively.

**Limited data** To perform the pre-training on the dataset, the networks used the WHU-OHS dataset cut up into $64 \times 64$ image patches, resulting in around $300,000$ images. However, most of the time, pre-training is performed on much more significant amounts of data, such that for the MAE, around 14 Million and the DINOv2, 142 Million images were used during the pre-training phase. This drastically limits the variety of the data presented to the model, encouraging overfitting. For this to get around, the MAE used dropout to discourage the behaviour. However, DINOv2 was not using any dropouts, and hence, this could have been a result of its poor performance. Overall, the data available was not enough for a full pre-training.

**Data complexity** HSI data stores much more information about the area presented to the model. However, this data requires additional space to be encoded, drastically scaling these networks. This was used on most transformer networks; hence, the proof of concept was built on whether the networks were complex enough to represent the data. However, this might not have been the case for the baseline CNN, which might not possess enough complexity to store all the required information.

# 6 Conclusion

Vision transformers' sudden arrival into the computer vision space dominated by CNNs for a decade disturbed the waters that were still for a while now and received contributions in terms of ground-breaking research every month. Such research mainly focused on adopting state-of-the-art methods into a new, slightly different domain.

Overall, the research found that the methods in their current state could not perform adequately, and as such, the research did not prove successful. However, the given results suggest underlying issues requiring further exploration, issues that are outside the project's scope, such as data validation and results reproduction. However, even though the results of the networks were not up to the standards required by the state-of-the-art, there were still a few interesting points, such as the strong correlations extracted in the attention maps of the Spatial-Spectral encoder fig. 5.5, and the structural distributions represented in the patch tokens of the DINOv2 trained networks fig. 5.4, suggesting that there might be future applications of these methods.

## 6.1 Future work

There are many extra paths to explore regarding the domain, and even though the results were below expectations, especially compared to current published methods, the encoders showed promising signs, such as the structure retention of DINOv2 and the ability to adopt multiple resolutions and encode a wide variety of features by scaling.

### Dataset exploration

Hyperspectral data to train on is not available in vast amounts and is rather hard to gather. However, the same is not true for multispectral image data, which is still being collected by one of the most famous remote sensing satellites, Sentinel-2 (ESA 2017). It has over 14 million high-resolution images available, giving a solid foundation for the training of a large encoder.

However, due to the now changed input datatype, it would be an interesting path to explore how pretraining on multispectral data can be combined with downstream hyperspectral training.

### Underlying cause analysis

Further, evaluation of the dataset would shed some light on why the ViTs underperformed in all scenarios. This would include not only statistical analysis of remote sensing data but also reproducibility analysis of the original papers that this research relied on.

### Further downstream evaluation

After performing all the above two experiments, exploring additional evaluation downstream tasks would provide more in-depth information on the network's performance, such as what scenarios the models underperform and what scenarios they perform better in.

## 6.2 Lessons learned

I learnt a lot while working on this project. Besides the obvious, such as how transformer networks work under the hood, I learnt the importance of taking a step back instead of going in head first. Multiple times during the project, I found myself having problems with the data, which would have been tough to notice or understand if I had not taken a moment to look at the bigger picture.

Furthermore, I learnt that spending time on professional practices, such as using documentation, logging results, and plotting results constantly, provides a tremendous help later down the line when comparisons have to be drawn up and need to be made. Additionally, spending time setting up automated processes to perform repetitive tasks, such as compiling the docker images for the training environment or moving code between environments, the time not spent on those tasks is gained back that was paid for building the integration in the first place.

Lastly, I gained a lot of knowledge in the domain of vision transformers. Before starting this project, I had not heard of their existence yet, and now I was performing modifications on such networks. I found vision transformers fascinating due to their ability to capture and relate information unlike any other type of network before, which guided the project's main focus towards pre-training of vision transformers rather than fine-tuning.

Overall, I learnt a lot during the project, both about how to conduct research and why certain things are done in the way they are. I also gained much academic knowledge and insights into how vision transformers work.

# A | Appendices

# 6 Bibliography

D. Birla. Basics of autoencoders, 2019.

M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. URL https://arxiv.org/abs/2104.14294.

L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018. URL http://arxiv.org/abs/1802.02611.

L. Chen, G. Vivone, J. Qin, J. Chanussot, and X. Yang. Spectral–Spatial Transformer for Hyperspectral Image Sharpening. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2023.3297319. URL https://ieeexplore.ieee.org/document/10198668/.

T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. Vision transformers need registers, 2023.

DataCamp. Deep learning (dl) vs machine learning (ml): A comparative guide | datacamp. https://www.datacamp.com, 2023. URL https://www.datacamp.com/tutorial/machine-deep-learning. Accessed: March 6, 2024.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

O. ESA. Sentinel-2, 2017. URL https://developers.google.com/earth-engine/datasets/catalog/sentinel-2.

EUMeTrain. Training module on monitoring vegetation from space, 2010. URL https://resources.eumetrain.org/data/3/36/print_3.htm.

M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. URL http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. URL https://arxiv.org/abs/2111.06377.

C. Hecker, S. Hook, M. van der Meijde, W. Bakker, H. van der Werff, H. Wilbrink, F. van Ruitenbeek, B. de Smeth, and F. van der Meer. Thermal infrared spectrometer for earth science remote sensing applications-instrument modifications and measurement procedures. 2011. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3274325/.

D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. *CoRR*, abs/1901.09960, 2019. URL http://arxiv.org/abs/1901.09960.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.

S. Illarionova, D. Shadrin, A. Trekin, V. Ignatiev, and I. Oseledets. Generation of the nir spectral band for satellite images with convolutional neural networks. *Sensors*, 21(16), 2021. ISSN 1424-8220. doi: 10.3390/s21165646. URL https://www.mdpi.com/1424-8220/21/16/5646.

Q. V. Le, R. Monga, M. Devin, G. Corrado, K. Chen, M. Ranzato, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. *CoRR*, abs/1112.6209, 2011. URL http://arxiv.org/abs/1112.6209.

J. Li, X. Huang, and L. Tu. WHU-OHS: A benchmark dataset for large-scale Hersepctral Image classification. *International Journal of Applied Earth Observation and Geoinformation*, 113: 103022, 2022. ISSN 1569-8432. doi: https://doi.org/10.1016/j.jag.2022.103022. URL https://www.sciencedirect.com/science/article/pii/S1569843222002102.

T. Liu, Y. Liu, C. Zhang, L. Yuan, X. Sui, and Q. Chen. Hyperspectral image super-resolution via dual-domain network based on hybrid convolution. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. Preprint.

NASA. The hubble space telescope. https://www.nasa.gov/mission_pages/hubble/story/index.html, 1990. Accessed: yyyy-mm-dd.

openmmlab. mmcv, 2024. URL https://github.com/open-mmlab/mmcv.

M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

M. Pause, F. Raasch, C. Marrs, and E. Csaplovics. Monitoring glyphosate-based herbicide treatment using sentinel-2 time series—a proof-of-principle. *Remote Sensing*, 11(21), 2019. ISSN 2072-4292. doi: 10.3390/rs11212541. URL https://www.mdpi.com/2072-4292/11/21/2541.

P. Potrimba. https://blog.roboflow.com/what-is-knowledge-distillation/, 2023. URL https://blog.roboflow.com/what-is-knowledge-distillation/.

S. Republic. How nasa handles the world's largest data archive, 2023. URL https://www.siliconrepublic.com/enterprise/nasa-data-figures.

V. P. Roberto D'Autilia. Architecture recognition by means of convolutional neural networks. *ResearchGate*, 2019. URL https://www.researchgate.net/figure/A-2D-convolution-example_fig2_334416432.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Y. Ruan, S. Singh, W. Morningstar, A. A. Alemi, S. Ioffe, I. Fischer, and J. V. Dillon. Weighted ensemble self-supervised learning, 2023.

M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig. Non-linear PCA: a missing data approach. *Bioinformatics*, 21(20):3887–3895, Aug. 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti634. URL `https://doi.org/10.1093/bioinformatics/bti634`. _eprint: https://academic.oup.com/bioinformatics/article-pdf/21/20/3887/48980996/bioinformatics_21_20_3887.pdf.

H.-B. Shao, L.-Y. Chu, C. A. Jaleel, and C.-X. Zhao. Water-deficit stress-induced anatomical changes in higher plants. *Comptes Rendus. Biologies*, 331(3):215–225, 2008. doi: 10.1016/j.crvi.2008.01.002.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Specim. hyperspectral technology vs rgb, March 2021. URL `https://www.specim.com/hyperspectral-technology-vs-rgb/`. Accessed: 2024-03-01.

R. Strudel, R. G. Pinel, I. Laptev, and C. Schmid. Segmenter: Transformer for semantic segmentation. *CoRR*, abs/2105.05633, 2021. URL `https://arxiv.org/abs/2105.05633`.

I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. page 10, 09 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions, 2014.

H. K. Varchand, M. R. Pandya, J. A. Dave, H. J. Trivedi, and V. N. Pathak. Sensitivity study and intercomparison of co2 absorption bands in swir region. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 6610–6613, 2022. doi: 10.1109/IGARSS46834.2022.9883247.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

S. Voigt, F. Giulio-Tonolo, J. Lyons, J. Kučera, B. Jones, T. Schneiderhan, G. Platzeck, K. Kaku, M. K. Hazarika, L. Czaran, S. Li, G. Pedersen, G. K. James, J. Bequignon, and D. Guha-Sapir. Global trends in satellite-based emergency mapping. *Science*, 353(6296):247–252, 2016. URL `https://www.science.org/doi/10.1126/science.aad8728`.

J. J. Walsh, E. Mangina, and S. Negrão. Advancements in Imaging Sensors and AI for Plant Stress Detection: A Systematic Literature Review. *Plant Phenomics*, 6:0153. ISSN 2643-6515. doi: 10.34133/plantphenomics.0153. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10905704/`.

wandb. wandb, 2024. URL `https://wandb.ai/site`.

Wikipedia contributors. Alexnet — Wikipedia, the free encyclopedia, 2024. URL `https://en.wikipedia.org/w/index.php?title=AlexNet&oldid=1211899570`. [Online; accessed 5-March-2024].

M. Zhang. Neural attention: Enhancing qkv calculation in self-attention mechanism with neural networks, 2023.

J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong. ibot: Image bert pre-training with online tokenizer, 2022.