SAPIENZA
UNIVERSITÀ DI ROMA

From Single to Many: Virtual Subnetworks for Vision Transformer Ensembles

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Engineering in Computer Science

Candidate

Alessandro Ferrante

ID number 1849675

Thesis Advisor

Prof. Simone Scardapane

Co-Advisor

Prof. Danilo Comminiello

Academic Year 2023/2024

From Single to Many: Virtual Subnetworks for Vision Transformer Ensembles
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: aferrante99@gmail.com

Abstract

Machine learning models, particularly Vision Transformers (ViTs), have revolutionized image understanding in computer vision tasks. However, their high computational cost and reliance on large datasets present significant challenges for real-world applications.
The goal of this thesis project is to develop a model based on a state-of-the-art ViT, called Custom ViT, which can be enhanced through ensemble learning and virtual subnetworks to improve accuracy and robustness in various scenarios.
The proposed solution modifies the standard ViT architecture by adding a selected number of additional tokens, which enable the extraction of virtual subnetworks. These subnetworks are essentially distinct embeddings whose predictions can be combined using various ensemble techniques.
The solution is evaluated using standard datasets and tested against noisy input, adversarial attacks, as well as additional tests for out-of-distribution (OOD) detection.
The Custom ViT models, generated with varying numbers of subnetworks and different ensemble techniques, demonstrated significant improvements over the standard ViT, both in terms of accuracy and robustness. These improvements are particularly noteworthy considering the minimal training cost required for deploying each model.

Contents

Chapter 1

Introduction

The project proposed in this thesis arises from the need for more robust yet efficient machine learning models, especially in computer vision tasks. Vision Transformers (ViTs) have significantly advanced state-of-the-art vision models, revolutionizing how image and video understanding are approached. Their ability to capture global dependencies and their inherent flexibility have driven considerable progress in the field.

However, a key challenge in the application of ViTs lies in their computational cost and data efficiency. Despite their revolutionary nature, these models come with substantial computational complexity, primarily due to the self-attention mechanism, which scales quadratically with the sequence length. Additionally, ViTs often require large datasets to achieve optimal performance, which can limit their applicability in resource-constrained environments.

The proposed solution, Custom ViT, aims to address these challenges by offering an innovative system of subnetworks derived from an existing pre-trained model. This approach allows for extended input representations without significant additional costs, leading to improved performance in terms of both accuracy and robustness across various tasks.

The strength of the proposed solution lies not only in the model modifications but also in leveraging the diverse subnetworks generated by performing ensemble learning among them. Ensemble learning is a widely researched area as it enables the creation of more accurate and robust models by combining individual models that would otherwise yield lower results on their own. However, this often leads to increased computational costs due to the need to train multiple models. In contrast, the solution presented in this thesis ensures that all ensemble components are generated from the same training process as a standard Vision Transformer, avoiding additional training costs.

The pursuit of robustness and efficiency is critical in modern machine learning models, particularly in real-world applications such as autonomous driving, medical imaging, and mobile devices. However, this demand must address the limitations of current ViT models, including scalability challenges, computational overhead, and the need for large datasets.

This work aims to leverage the advancements in state-of-the-art ViTs to harness their potential while mitigating their computational drawbacks. The focus is on optimizing

these models to enhance their efficiency and scalability without compromising their powerful capabilities.
The proposed solution is first explained in detail, followed by a series of tests conducted under standard conditions and with inputs subject to corruption from Gaussian noise or real adversarial attacks. Additionally, an analysis of OOD detection is performed to evaluate the model's robustness in various scenarios.
The structure of the thesis is outlined below, providing an overview of the chapters and the topics covered in each.

Chapter 2 offers a theoretical background on the foundations of Custom ViT, focusing on Vision Transformers (ViTs), their capabilities, and an in-depth exploration of ensemble learning.

Chapter 3 examines the latest research trends and cutting-edge solutions in the fields of ViTs and ensembling, as well as approaches related to subnetworks.

Chapter 4 provides a detailed description of the proposed solution, including modifications to the standard ViT architecture, the training process, and the ensembling techniques explored.

Chapter 5 presents the results of all tests conducted on the proposed models, comparing their performance with that of the vanilla ViTs analyzed.

The Custom ViT implementation is available in the GitHub repository at the following link: `https://github.com/aferrante99/Custom-ViT`

Chapter 2

Background

2.1 Vision Transformer (ViT)

Introduced in the 2021 article by Dosovitskiy et al., 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale' [1], the Vision Transformer (ViT) offers a novel approach to image analysis by segmenting images into smaller patches and utilizing self-attention mechanisms. This approach builds on the success of Transformers, which were introduced in 2017 in 'Attention is All You Need' [2].

The breakthrough in [1] demonstrated that reliance on CNNs is not essential, as a pure transformer can perform exceptionally well in image classification tasks. Compared to state-of-the-art Convolutional Neural Networks, ViT achieves excellent results while requiring fewer computational resources.

The difference between the two approaches could be highlighted in the following points:

- Input Representation: Instead of processing raw pixels directly, ViT divides the input image into patches which are eventually transformed into tokens.
- Processing: CNNs hierarchically capture features at different spatial scales using convolutional and pooling layers, therefore relationships between distant pixels, such as the first and last, are compared indirectly through successive layers. In contrast, ViT's self-attention mechanism directly considers the relationships between all patches, regardless of spatial distance, enabling it to capture global context in a single step. In summary, CNNs depend exclusively on pooling layers to capture global information.
- Efficiency: CNNs typically need large quantities of labeled data for training, while ViT can leverage pre-training on extensive datasets followed by fine-tuning for specific tasks.

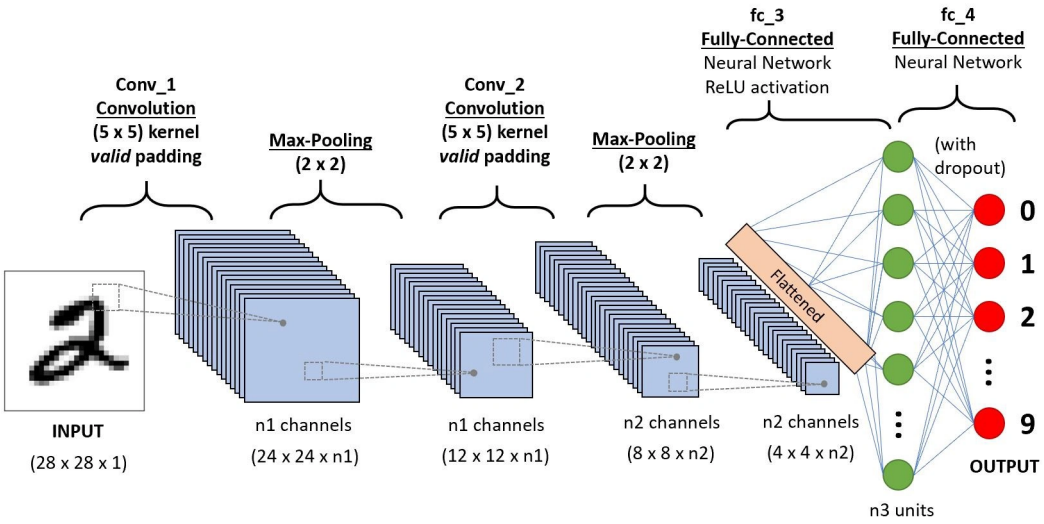


Figure 1. CNN Architecture

The functioning of ViTs can be summarized in the following steps:

1. The input image is split into fixed-size square patches, and each patch is linearly transformed into a vector through a learnable linear projection. This process generates a sequence of patch embeddings, which are used as input tokens for the following layers.

2. Positional encodings are added to the patch embeddings since no spatial information is inherently grasped by the Vision Transformer per se. These encodings provide positional relationships and are typically learned.

3. The central component of the Vision Transformer is composed of several encoder layers, each featuring two main sub-layers: multi-head self-attention and feedforward neural networks.

3a. The self-attention mechanism captures relationships between patches in the input sequence by computing a weighted sum of all patch embeddings, with weights based on the relevance of each patch to the current one. This enables the model to focus on important patches while considering both local and global contexts. Multi-head attention uses multiple sets of learnable parameters (attention heads) to capture diverse types of relationships.

3b. Feedforward neural networks process the output from each patch's self-attention mechanism. Those are typically made of a fully connected layer followed by an activation function like ReLU, these networks introduce non-linearity, allowing the model to learn complex relationships between patches.

4. Layer normalization and residual connections follow the outputs of both the self-attention mechanism and the feedforward network. Layer normalization

stabilizes and accelerates training by normalizing the inputs to each sub-layer. Residual connections, or skip connections, add the original input embeddings to the output of each sub-layer, facilitating gradient flow during training and mitigating the vanishing gradient problem.

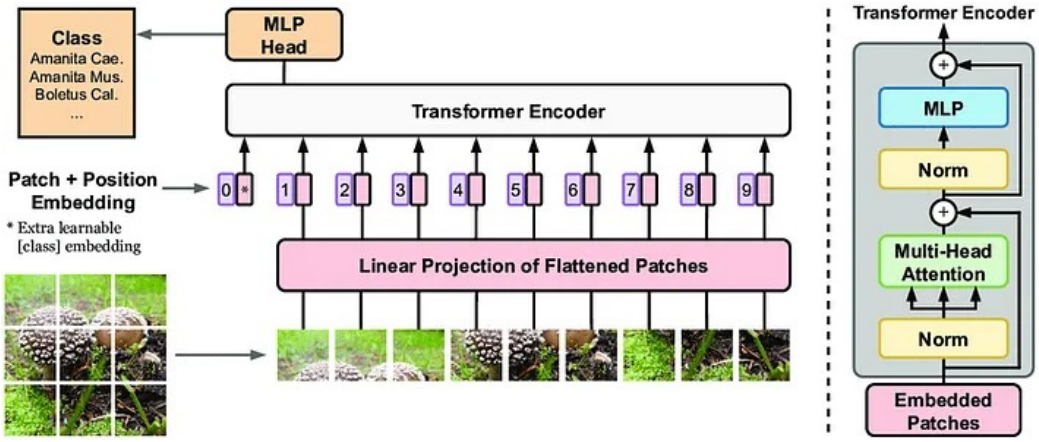
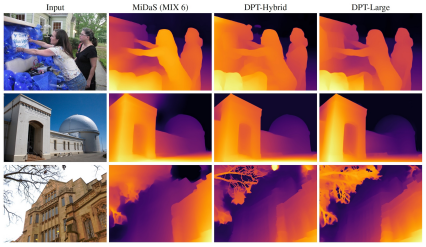


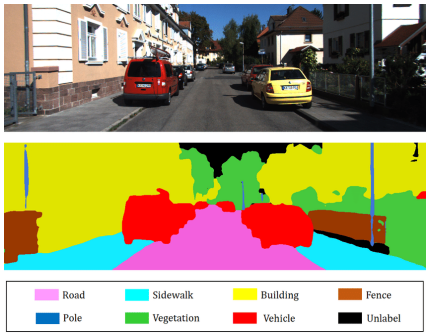
Figure 2. ViT Architecture

The Vision Transformer represents a significant step toward developing scalable, generic models that can tackle a wide range of vision tasks. Its broad applicability and scalability have led to its increasing prominence in research.
For image classification tasks, CNN-based methods are considered state-of-the-art, particularly on small to medium-sized datasets. However, ViTs, while not matching the performance of CNNs on smaller datasets, have shown superior results on very large datasets. This is because CNNs are more effective at encoding local image information, thanks to their use of locally restricted receptive fields. On the other hand, ViTs excel in capturing global context and relationships across the entire image, making them more suitable for large-scale datasets where global dependencies play a crucial role.
For example, in tasks like large-scale image recognition on datasets such as ImageNet, ViTs can leverage vast amounts of data to achieve better generalization and performance compared to CNNs. The difference and usefulness of ViTs compared to CNNs is even more evident in fields such as image captioning and semantic segmentation. ViTs enable more advanced image categorization by generating descriptive captions instead of simple labels. Models like Dense Prediction Transformers (DPT) [3] have achieved strong results in image semantic segmentation and monocular depth estimation, outperforming CNN-based methods in these tasks. Additionally, ViTs are increasingly used in areas like autonomous driving, anomaly detection [4] and action recognition [5], contributing to better understanding and decision-making.

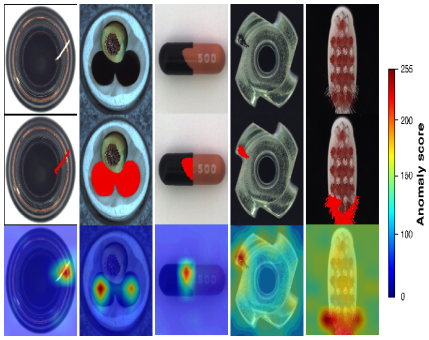


(a) DPT

(b) Autonomous Driving



(c) Semantic Segmentation



(d) Anomaly Detection

Figure 3. Latest applications of Vision Transformers in various domains.

2.2 Ensemble Learning

In machine learning, two approaches often outperform traditional algorithms: Deep Learning and Ensemble Learning.
Modern deep learning techniques enable tackling complex problems across a wide range of architectures, significantly improving performance across various tasks and domains. However, despite their outstanding performance and scalability, deep learning methods require substantial effort. Finding the optimal set of hyperparameters demands both expertise and extensive experimentation, making the process tedious and time-intensive.
Ensemble learning, on the other hand, is a methodology that combines or aggregates multiple base models into a single, more powerful model that outperforms the individual models it is composed of.
Historically, ensemble learning embodies the machine learning interpretation of the *wisdom of the crowd*. Surowiecki [10] outlined guidelines to ensure that the collective wisdom of a group can outperform individual decisions. These principles strongly align to the criteria applied in practical machine learning scenarios, making them relevant for ensemble methodologies.

- **Independence:** One's opinion is not affected by other opinions.
- **Decentralization:** One is capable of specializing and making conclusions based on local information.
- **Diversity of opinions:** One should hold private information, even if it is just an eccentric interpretation of the known facts.
- **Aggregation:** Some mechanism exists for turning private judgments into a collective decision.

Practically speaking, given a dataset, the general architecture of a classifier ensemble involves combining multiple classifiers $c_1, c_2, \ldots, c_n$ through an aggregation function G, which produces a single output. This structure is illustrated in the following image.

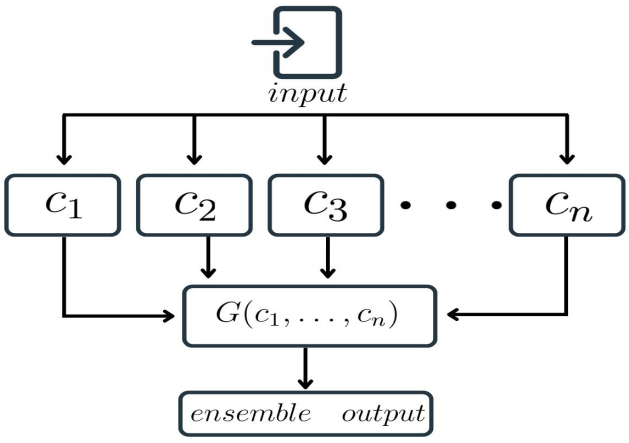


Figure 4. Typical Ensemble flowchart

Within this general framework, constructing an ensemble model requires selecting a training methodology for the individual models and determining an appropriate approach to combine their outputs. As mentioned earlier, there are general guidelines that should be followed to achieve an efficient and high-performing ensemble.

- **Diversity:** The superior performance of ensemble models is achieved mainly due to the use of various "inductive biases". The participating classifiers should be sufficiently diverse to achieve the desired predictive performance.
- **Predictive performance:** The individual inducer's predictive performance should be as high as possible and at least as good as a random model. This ensures that each classifier contributes valuable information to the ensemble.

Diversity in ensemble models can be achieved through various techniques that aim to improve both robustness and accuracy, and these can be applied at different stages of the pipeline. *Input manipulation* involves training base models on different subsets of data, introducing variations in samples or class distributions. By altering hyperparameters — such as learning rates, number of layers, or activation functions — each neural network in the ensemble can converge to different solutions within the hypothesis space. This process is known as *learning algorithm manipulation*. Other techniques include *dataset partitioning*, which enables models to focus on distinct aspects of the data, either by dividing instances (horizontal) or features (vertical). *Output manipulation* methods transform multiclass classification problems into simpler binary tasks and aggregate predictions to improve accuracy. Finally, *hybrid approaches*, which combine strategies like instance and feature manipulations, further enhance the effectiveness of ensemble models.
After this critical phase of looking for diversity and improvement at each stage of the pipeline, ensemble learning can also vary at the final stage of predictions, when the results of individual models are available and need to be combined. This step, known as *output fusion*, integrates the outputs of the models. The two primary methods employed at this stage are *weighting methods* and *meta-learning methods*.

- *Weighting Methods:* The outputs of base models are combined by assigning weights to each model. The weighting approach is particularly effective when the base models have comparable performance. For classification tasks, the simplest method is majority voting, where the final class is determined by the one receiving the most "votes". Alternatively, weights can be assigned based on the relative strengths of the models, often determined by their predictive performance on a validation set. Additionally, a more sophisticated weighting method is the Bayesian combination method, where the weights assigned to each base model are determined based on its probabilistic likelihood given the entire dataset. This means assessing the probability that a base model provides the best explanation of such dataset, relying on its performance and alignment with the observed data.
- *Meta-Learning Methods:* In this approach, outputs from base models serve as inputs to a meta-learner, which generates the final prediction. The most well-known meta-learning technique is Stacking, where a meta-dataset is created using the outputs of base models as input features, while the target attribute

remains the same. Typically, the original dataset is split into two parts: one for training the base models and the other for building the meta-learner. For new instances, base models generate predictions, which are then passed to the meta-learner to produce the final output. Another notable meta-learning method is the Mixture of Experts (ME), which divides the problem space into subspaces and assigns the most suitable base model to each region. Subspaces can overlap, and an additional learning algorithm often determines the best base model for each.

Given the previously mentioned methodologies and techniques for creating efficient and effective ensembles, the state-of-the-art frameworks for ensemble methods can be categorized into two main types: the *dependent framework* and the *independent framework*.
In the dependent framework, the output of each base model influences the construction of subsequent models. In this setup, knowledge generated in earlier iterations is used to guide the learning process in the next stages, creating a sequential and interconnected training pipeline. A well-known example of this framework is AdaBoost [11] [12]. Its core principle is to focus on instances that were previously misclassified when training a new classifier. This is achieved by assigning a weight to each instance in the training set. Initially, all instances are given equal weight. With each iteration, the weights of misclassified instances are increased, directing more attention to them, while the weights of correctly classified instances are reduced. Other implementations and variations of AdaBoost focus on addressing specific challenges, such as improving robustness to outliers [13], enhancing generalization at the expense of training accuracy [14], incorporating spatial reasoning [15], or adapting the algorithm for parallel processing to overcome its sequential nature [16].

On the other hand, in the independent framework, each inducer is built independently from the others. The simplest yet highly effective example of this framework is Bagging. This method generates an ensemble of independent models by training each inducer on a sample of instances drawn with replacement from the original dataset. To ensure sufficient data for training each inducer, these samples typically contain the same number of instances as the original dataset. The final prediction for an unseen instance is determined by the majority vote of the classifications from the inducers [17]. Since sampling is performed with replacement, some instances from the original dataset may appear multiple times in the training set of a single inducer, while others might be excluded entirely. This randomness helps to increase diversity among the inducers. Due to the independent training of the base models, Bagging can be efficiently implemented in parallel, with each inducer trained on separate computational units. Several variations have also been implemented in this type of framework. Some aim to improve generalization by refining the resampling process or incorporating information entropy [18]. Others focus on increasing computational efficiency or training multiple classifiers using different subsets of data, such as in Online Bagging [19] or Double-Bagging [20].

When selecting the best ensemble method, several factors should be considered beyond predictive performance. These include the suitability of the method to

the specific problem context, such as class imbalance or noisy data, as well as its computational cost, especially in real-time systems. Software availability and ease of integration across platforms also play a crucial role. Additionally, usability is important, as users often prefer models with clear, understandable parameters for tuning. Taking all these factors into account ensures the selection of the most appropriate ensemble approach for a given task.

Chapter 3

Related Works

3.1 Leveraging ensembles: trends and applications

In recent years, ensemble learning has become one of the most researched and analyzed methods in the field of artificial intelligence and machine learning. With state-of-the-art models and networks currently achieving exceptionally high levels of precision and accuracy, there is a growing need for a different type of research, focused on enhancing efficiency in new ways. Ensemble learning stands out among these approaches, both as a mean to enhance robustness for specific tasks and settings, and as a strategy to reduce the computational costs associated with training large models.

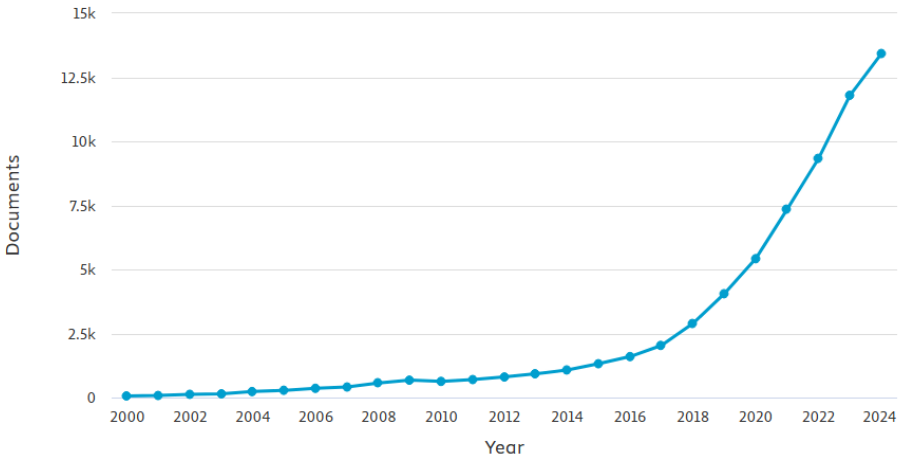


Figure 5. Ensemble Learning research trend

An interesting application that combines Vision Transformers (ViTs) with ensemble learning is the recent work presented in "Vision Transformer-Based Ensemble Learning for Hyperspectral Image Classification" [22]. In this study, an ensemble learning framework is employed for hyperspectral image (HSI) classification by integrating multiple variants of ViTs. Hyperspectral Imaging (HSI) is a technique that captures and processes information across a wide spectrum of wavelengths, providing detailed spectral information for each pixel in an image, which is used

for applications such as remote sensing, agriculture, and medical diagnostics. The approach proposed in the article mentioned above achieves high-precision pixel-level classification, incorporating as many as seven different types of ViTs.
Considering the high-dimensional nature of HSI samples, over the years, researchers have applied various machine learning methods to HSI classification [23] [24] [25] [26]. Due to the limited number of available training samples for HSI data, which often fail to meet the requirements of the spectral dimension, these classifiers still exhibit inadequate performance on the original data. To address these issues, HSI feature extraction methods have emerged. The literature includes experiments with CNNs and RNNs [27], leading up to the advent of Transformers.
A critical challenge in HSI classification arises when training samples are limited, as a single classification network may perform poorly. Clearly, improving the model structure and adding feature extraction modules can theoretically enhance classification accuracy. However, this often leads to higher requirements in terms of data and computational cost. This is where ensemble learning comes into play, as the individual models in an ensemble have significantly lower demands. Each ViT in the ensemble has different feature extraction capabilities, allowing for multiple perspectives and significantly enhancing diversity.
As previously mentioned, the models used are seven, and they are:

- ViT.
- Simple ViT [28] : A streamlined Vision Transformer with a simplified architecture.
- CaiT [29] : A ViT variant with enhanced attention mechanisms for deeper network capabilities.
- DeepViT [30]: A ViT designed for improved feature extraction through deeper layers.
- ViT with Patch Merger (ViTPM) [31] : A ViT that merges patches to reduce computational complexity.
- Learnable Memory ViT [32] : A ViT incorporating a learnable memory module to enhance feature retention.
- Adaptive Token Sampling ViT (ATSViT) [33] : A ViT that dynamically adjusts token sampling to optimize performance.

Another key aspect is fusing spatial and spectral features. By processing the pixel information within the $N \times N$ neighborhood surrounding the current pixel, spatial features can be extracted and combined with spectral features to achieve high-accuracy classification.
The ensemble strategy proposed in the article is twofold: in the first approach, classification results are obtained through a round of ensembling using majority voting. In the second approach, another pass is made through the models, followed by an additional ensembling step, again using majority voting.

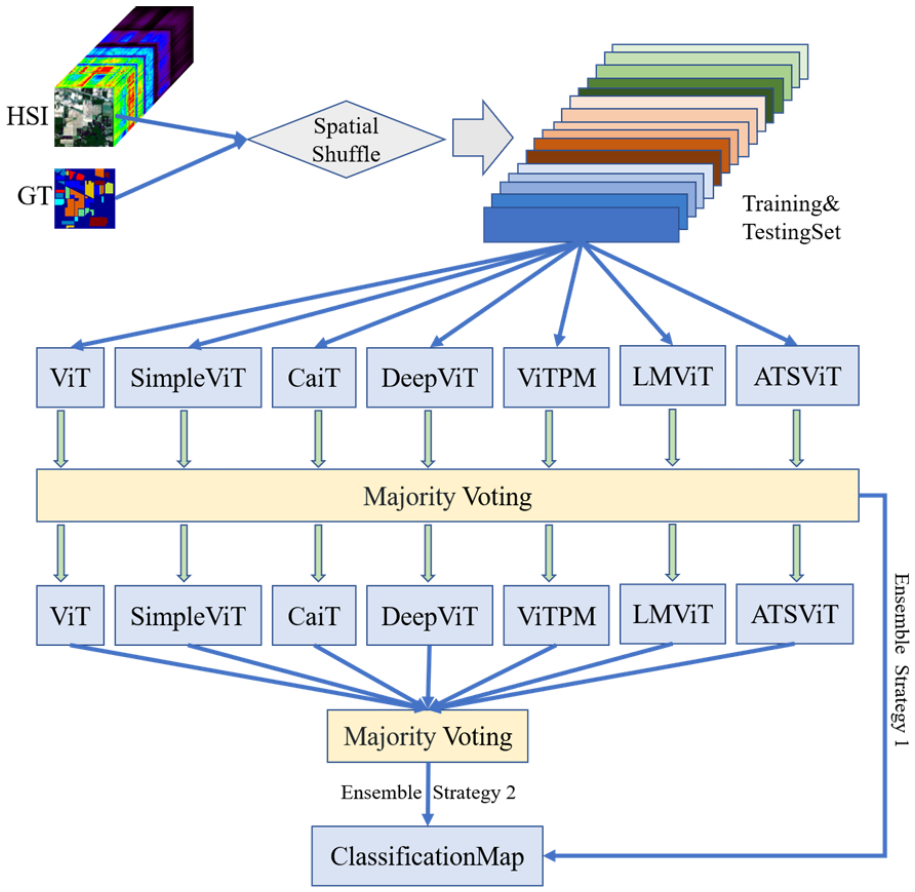


Figure 6. Ensemble method for HSI classification

This ensemble technique has produced excellent results, particularly highlighting how, by varying the datasets, each individual model experiences a performance drop in some cases. However, the ensemble of all models ensures stable results, indicating the robustness of the set. In this specific field, given the scarcity of training samples, it is clear that results improve as the number of samples increases. However, it can be observed that, thanks to the proposed method, ensemble learning achieves results that are only attainable once a reasonable threshold of sample proportion is surpassed.

Another application of ensemble learning with Vision Transformers is SEViT [34]. This research originated for clinical settings, as a self-ensembling system designed to improve robustness against adversarial attacks. Indeed, despite the well-established superiority of ViTs over CNNs in various tasks, it is well known that ViTs are susceptible to certain types of attacks [35][36]. In this analysis, it is shown that the impact of perturbations is significantly greater on the final set of blocks in the model, while the patch tokens learned in the initial blocks remain relatively unaffected.
For this reason, instead of focusing solely on the final class token, as standard ViTs do, an MLP classifier is added at the end of each block. Each intermediate MLP classifier produces a probability distribution based on the patch tokens of the corresponding block. This results in a self-ensemble (called "self" because different

models are not fused) of L classifiers, which together produce the final classification. Additionally, it is highlighted that the complexity can be further reduced by adding MLPs only in the first $m < L$ blocks, where adversarial attacks have less impact.

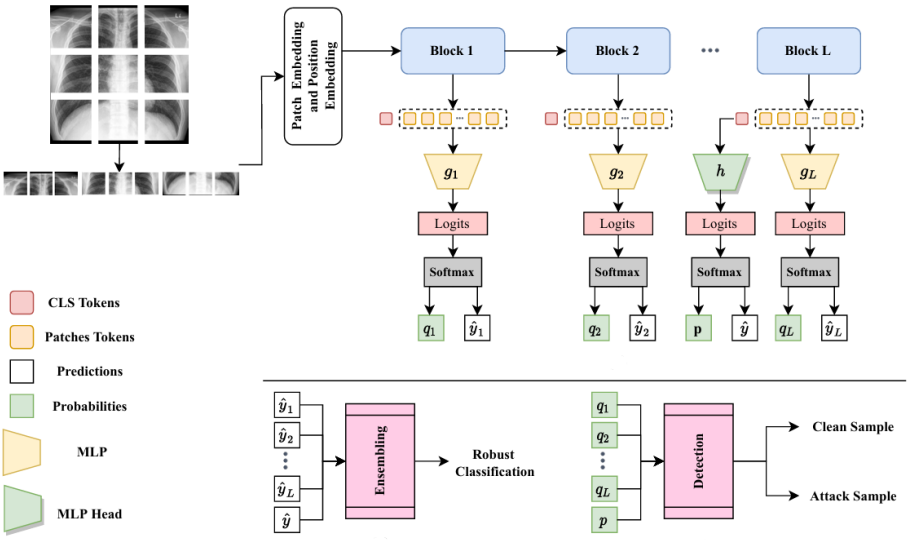


Figure 7. SEViT architecture, with its applications

In addition to more robust classification, another application of SEViT is the detection of adversarial and clean samples. Tests were conducted with inputs corrupted by various attacks, and the results were optimal. SEViT not only improves accuracy on clean samples (on which the standard ViT already performed well), but, most importantly, significantly increases precision on corrupted inputs.

In the first example, the ensemble technique combines multiple independent models, while in the second case, the additional cost is reduced by adding a limited number of MLPs to a single model. As a matter of fact, like the two previous ones, most prior studies focus on improving generalization performance, while few address the cost of training ensembles. In the next case, although it is an ensemble learning work that does not rely on ViTs, the reduction of computational costs is emphasized, which makes it particularly interesting to mention in the context of this thesis.
Indeed, in Snapshot Ensembles [37], a method is proposed to ensemble multiple neural networks with no additional training cost. This is achieved through an out-of-the-box insight into stochastic gradient descent (SGD). Although it was initially designed to avoid local minima [38], the paper focuses on improving performance by leveraging the information contained within these very local minima. The innovative technique consists of avoiding the training of M different models, instead making SGD converge M times to local minima along its optimization path. Each time the model converges, the weights are saved and the corresponding network is added to the ensemble through snapshotting (hence the name). The optimization is then resumed by increasing the learning rate to escape the local minimum where the model has converged. This allows the entire ensemble to be trained in the same

time as a single traditional model; moreover, it is possible to limit the number of models by selecting only the m most accurate ones. As an additional abstraction, it is theoretically possible to create ensembles of Snapshot Ensembles.

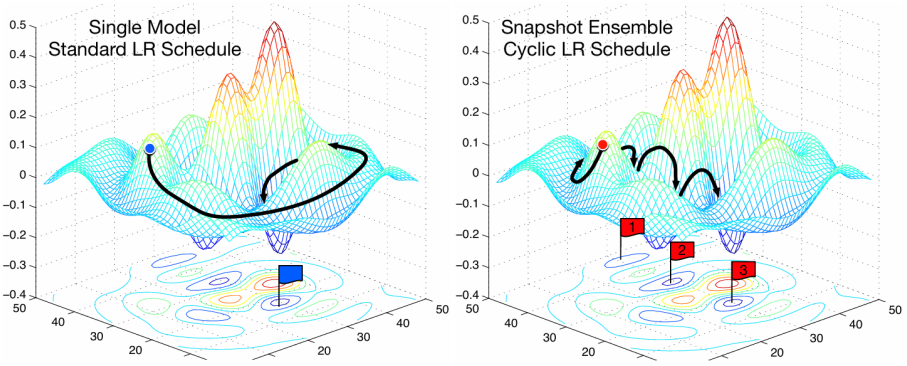


Figure 8. Comparison between a typical optimization schedule (left) and Snapshot Ensemble (right)

To converge to multiple local minima, Snapshot Ensemble follows a cyclic annealing schedule as proposed by Loshchilov & Hutter [39]. The learning rate is lowered at a very fast pace, encouraging the model to converge towards its first local minimum after a few epochs. The optimization is then continued at a larger learning rate, which perturbs the model and drives it away from the minimum. The process is repeated several times to obtain multiple convergences.

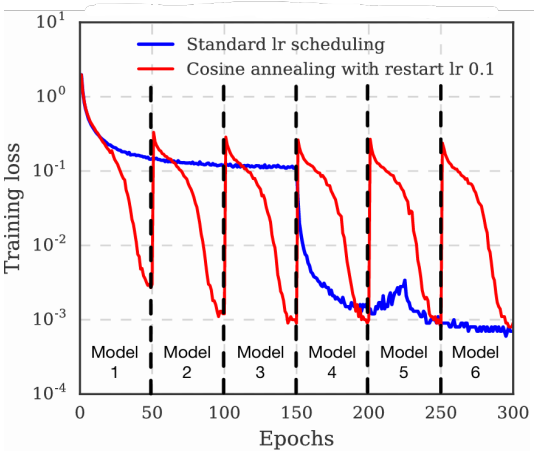


Figure 9. Training loss progression in the two modes

Although clearly the standard learning rate schedule achieves lower training loss than the cyclic schedule in some cases, when it comes to ensembling, the effectiveness of this procedure becomes apparent. The ensemble procedure is remarkably simple. Once the best m models are determined, they are fused by averaging. As further considerations, the paper notes that it may be necessary to dynamically adjust the number of models at test time. Additionally, to increase diversity, the pairwise

correlation of the softmax outputs for every pair of snapshots is computed. The results are excellent. Snapshot Ensemble not only improves the error of the state-of-the-art models it is compared to, but it even approaches the results achieved by the traditional ensemble of independent models.

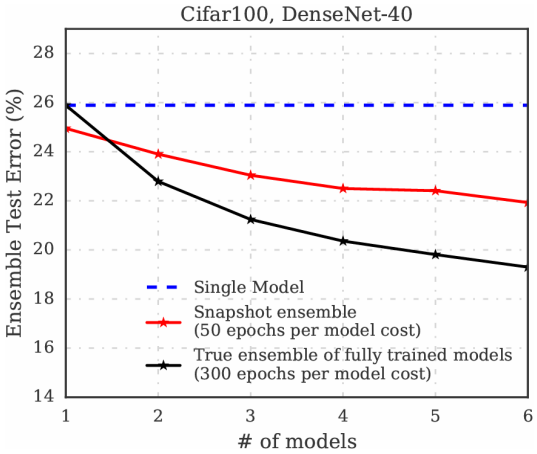


Figure 10. Test results comparison: Snapshot Ensemble vs. Standard Model vs. Standard Ensemble

3.2 The Subnetwork Paradigm

Even though this work does not directly follow the traditional subnetwork approach, it is important to acknowledge and discuss the foundational research in subnetwork techniques. Given the relevance of these approaches in improving model efficiency, robustness, and generalization, understanding the methods outlined in the following section provides valuable insights. These techniques have played a significant role in advancing the field of deep learning and continue to inspire alternative methods for enhancing network performance, including those explored in this thesis.

The idea of subnetworks, or at least neural network compression techniques, dates back to 1989 with the work of LeCun et al. [40] The idea was to reduce the number of weights without sacrificing performance, effectively creating a smaller and less costly network. These early methods and techniques, mostly related to pruning, laid the groundwork for later studies on subnetworks.
The most important innovation in this regard is certainly the formulation of the lottery ticket hypothesis in 2018 [41]. It states that:

LTH: *A randomly initialized, dense neural network contains a subnetwork that is initialized in such a way that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.*

The subnetwork with these characteristics is called the *winning ticket*. In the work, it is identified by training a network and pruning its smallest-magnitude weights.

The remaining, unpruned connections constitute the architecture of the winning ticket. After that, each unpruned connection's value is reset to its initialization from the original network before it was trained.
This procedure can be summarized in the following steps:

- A Neural Network and its weights are randomly initialized: $\theta^{(0)}$
- The Neural Network is trained for j iterations, arriving at parameters θ^*
- Prune the lowest-magnitude weights in the network, creating a mask m
- Reset the remaining parameters to their values in $\theta^{(0)}$, creating the *winning ticket* $m \odot \theta^{(0)}$

This method is then applied both in a one-shot manner, with a single training cycle, and iteratively, up to n training cycles in which all phases (pruning, training, and resetting) occur. As a result, the identified winning tickets constitute 10-20% (or less) of the original network's size. At this reduced size, they achieve or surpass the test accuracy of the original network within the same or fewer training iterations. Weight initialization plays a crucial role in the efficiency of winning tickets. In fact, even with the same pruned architecture, if the weights are reinitialized randomly, the performance of the winning tickets deteriorates significantly.

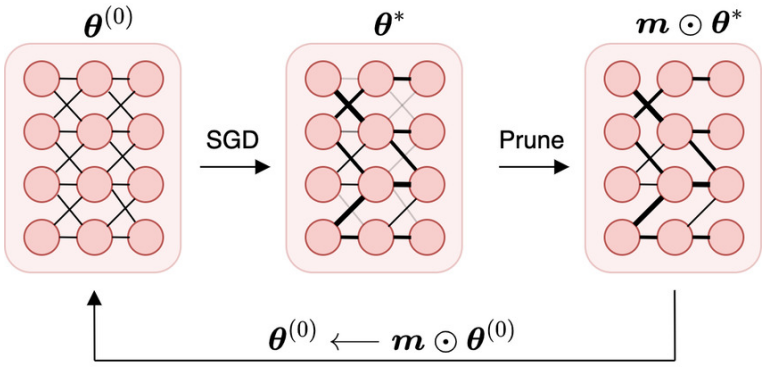


Figure 11. An iteration of the Lottery Ticket Hypothesis

Pruning has therefore become a popular approach due to its ability to preserve the accuracy of a network while reducing its size. This reduction allows for cutting down the computational and memory requirements that modern deep neural networks demand, both for creation and training, but especially for deploying them on mobile devices. As explained in the survey "What is the state of Neural Network pruning" [42], the method involves taking a model $f(x; W)$ and producing a new model $f(x; M \odot W')$, where W' is a set of parameters that may be different from W, and $M \in 0, 1^{|W'|}$ is a binary mask.
Pruning can achieve various objectives, such as reducing the storage size of a neural network and lowering the computational cost of inference. Every time this method

is used, the trade-off between resource savings and maintaining the performance of the original model must be considered. For this reason, each of these goals requires different design decisions and evaluation metrics. The main methods can be summarized as:

- **Structure:**
  - Unstructured pruning removes individual parameters, creating a sparse model.
  - Structured pruning removes entire groups of parameters, like neurons, filters, or channels, to better leverage hardware and software optimizations.
- **Scoring:** Parameters can be scored based on their absolute values, trained importance, or contributions to activations and gradients.
  - Local scoring prunes the lowest-scoring parameters within specific network components, like layers.
  - Global scoring compares parameters across the entire network, without considering their specific location.
- **Scheduling:** As mentioned before, the pruning can happen at different stages.
  - One-step pruning removes all selected weights at once.
  - Iterative pruning prunes a fraction of the weights over several steps.
  - Dynamic scheduling adjusts the pruning rate over time.
- **Fine-tuning:** After pruning, networks are typically fine-tuned by either continuing training with the pruned weights, rewinding the network to an earlier state, or reinitializing it entirely.

The pruning method has been applied in the literature to various models, including ViTs. In a recent paper [43], a method is proposed for slimming ViTs by pruning the dimensions of the Multi-Head Self Attention (MHSA) and Multi-Layer Perceptron (MLP) blocks. Feature importance scores are learned and used to prune the least important features, with ℓ_1 regularization applied to enforce sparsity. The relaxed continuous representation of the scores is optimized during training. After pruning, a threshold is applied to convert the continuous importance scores into binary values, indicating which features to keep. The model is then fine-tuned to minimize accuracy loss. This approach effectively reduces model size while maintaining performance, offering a valuable method for deploying ViTs in resource-constrained environments.

Another approach related to subnetworks is Mixture-of-Experts (MoE) models [44], which have gained significant attention in recent years, especially for large language models (LLMs) and vision tasks. MoE layers allow for scaling up a model's parameters while maintaining a fixed computational budget. This is achieved by selectively activating a small subset of "expert" subnetworks during the forward pass, with each expert specializing in a specific part of the input space. The selection of these experts is performed by a routing mechanism, which dynamically chooses which experts to activate based on the input. This approach not only enables

efficient utilization of computational resources but also supports specialization of subnetworks, enhancing performance across diverse tasks. MoE models have been successfully implemented in transformer architectures for NLP [45][46], vision [47], and multimodal tasks [48], offering a flexible and scalable solution for large models.

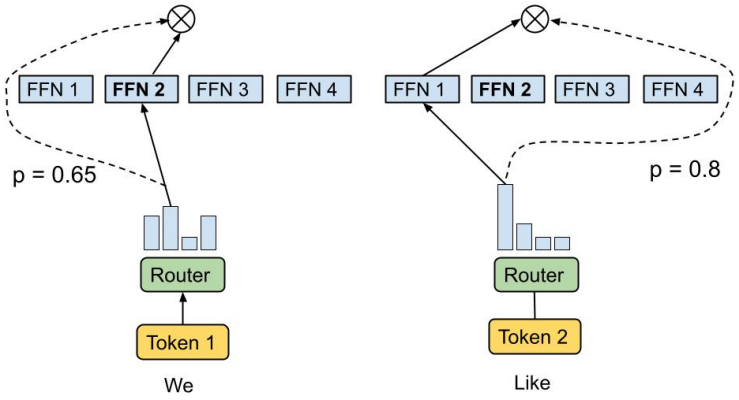


Figure 12. MoE's routing algorithm picks the experts with highest affinity scores for each token

In contrast, capsule networks [49] represent a modular approach where capsules—groups of neurons—encapsulate the instantiation parameters of entities, such as objects or object parts. These capsules work together in a hierarchical structure, with each capsule at a lower level passing information to higher-level capsules through a dynamic routing mechanism. This routing-by-agreement process allows capsules to form a modular network that preserves spatial relationships and instantiation parameters, which is crucial for recognizing complex objects and handling transformations like rotation or scaling. Although the capsule network approach does not rely on subnetworks, it still preserves the concept of modularity, which allows for computational resource savings by focusing computations where needed and avoiding redundant operations.

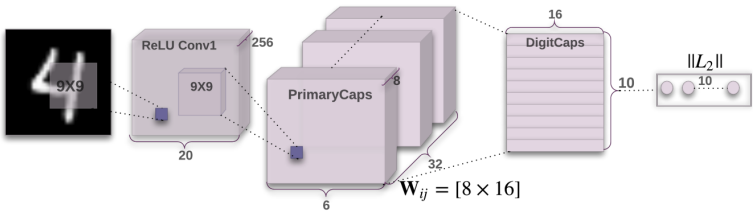


Figure 13. The input goes through the primary capsule, then the digit capsule performs the routing re-adjusting the weights according to the feature of the input

Chapter 4

Custom ViT

As discussed in the background chapter, Vision Transformers (ViT) have revolutionized the way visual data is processed by leveraging the self-attention mechanism from natural language processing.
While Vision Transformers (ViT) have demonstrated impressive performance, this thesis explores opportunities to further enhance their functionality. The primary focus is on improving efficiency and, most importantly, robustness, particularly through the use of ensemble learning techniques.

4.1 Custom ViT Class

In this work, it was not necessary to build a custom classification model from scratch. Instead, the proposed solution is based on a pre-trained Vision Transformer (ViT) model. The `timm` library in PyTorch was utilized to access the *DeiT Tiny* [21] model as the foundation for this research. This model belongs to the DeiT family of image classification models, known for their ability to achieve high performance with reduced data requirements, even with less training data compared to traditional methods. It has been pre-trained on ImageNet-1k, further enhancing its ability to generalize. The 'tiny' configuration was chosen to meet computational constraints, making it a more efficient option. It accepts input images with a resolution of 224x224 pixels, which are divided into patches of 16x16 pixels, following the standard ViT approach.

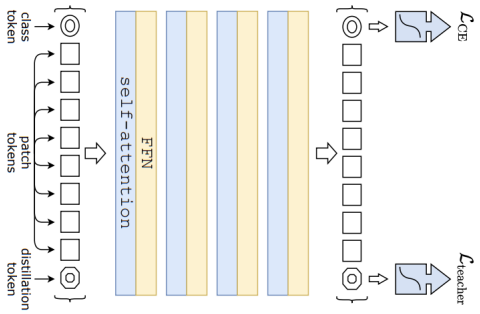


Figure 14. DeiT Architecture

The DeiT model is then encapsulated within a custom class, which forms the core of this work. This class is referred to as *Custom Vit*, and the key features and functions are outlined below.

The primary concept of the CustomViT class centers on the use of **additional tokens**, which are introduced into the model as learnable parameters during training. These tokens are concatenated with the input to facilitate the classification task. Each additional token generates a distinct embedding for the input, effectively creating diverse internal representations. This mechanism can be seen as generating **virtual subnetworks**, where the term "virtual" highlights that the model's architecture remains unchanged, but the input representation varies based on the specific additional token used. By diversifying these embeddings, the approach aims to enhance the overall robustness and maintain high accuracy through ensembling. This diversity is crucial for improving the variance between outputs, which is managed and encouraged through mechanisms like KL divergence.

On a practical level, these additional tokens are initialized randomly and scaled by a factor (namely of 0.02) in order to prevent them from initializing with excessively large values, which could distort the early learning.

$\mathbf{t}_{\text{add}} = \mathcal{N}(0, 0.02^2)$ (random initialization scaled by a factor of 0.02)

The number n of additional tokens to be generated is a parameter within the CustomViT class, initialized during the model's construction. The tokens are stored in a ParameterList, each accessible through a unique index. These indices represent the different virtual subnetworks. The index is passed as a parameter to the forward method of the CustomViT class. By specifying the index number during execution, one can retrieve the prediction from a specific subnetwork.
In the forward pass, the embedding derived from the chosen additional token is combined with the image patches. The result is a final embedding, which is concatenated with the input features. This forms a new input for the rest of the network, consisting of both the original image patches and the additional token embedding. This allows the network to generate a different representation of the image based on the selected additional token.

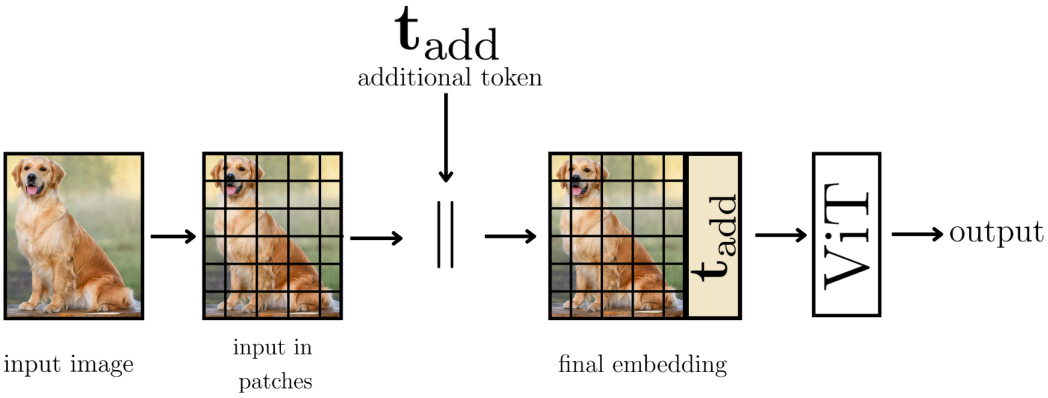


Figure 15. Additional token integration process

Given the Custom ViT's ability to initialize an arbitrary number of subnetworks, the advantage is that, at inference time, the model can be run multiple times (for instance, as many times as the number of additional tokens instantiated), each time obtaining the predictions from a different virtual subnetwork. This allows for the benefits of an ensemble at the cost of training only one model, leveraging the diversity among the subnetworks to enhance robustness and improve performance.

4.2 Training

Regarding the model training, as previously mentioned, for this work, a model pre-trained on ImageNet1k is used. The training conducted for the purpose of this thesis is done on Imagenette, a subset with fewer classes. To allow for an analysis and exploration of different ensemble techniques, the training was performed many times, once for each technique used. For all types of ensembles, due to computational limitations, no more than 5 virtual subnetworks per model were ever instantiated. To ensure the comparison was as fair as possible, the number of training epochs was kept consistent across the different methods, as well as the `Adam` optimizer and the `CosineAnnealingLR` scheduler. The data augmentation techniques applied in this work are commonly used in image classification tasks and include random transformations, color adjustments, resizing, random horizontal flipping, as well as tensor conversion and normalization.

Figure 16. Imagenette training set images (10 classes)

In the following sections, the different ensembling methods used are listed. Apart from the variation in the techniques employed, the fusion of predictions was always done using a simple average of the subnetworks' predictions, both during training and testing, without resorting to weighted averages or more complex methods.

4.2.1 Ensemble by Averaging

The first type of ensemble created is the simple yet effective ensemble averaging. Depending on the number of virtual subnetworks instantiated in CustomViT, during the training loop, all subnetworks of the model are run on the same inputs for each batch. The logits for each subnetwork are stored in a list throughout the process, and then the cross-entropy loss is computed on the average of the logits from these subnetworks.

$$Ensemble\ Logits = \frac{1}{n}\sum_{1}^{n}\ Logits\ list$$
$$Ensemble\ Loss = CE(Ensemble\ Logits\ ,\ labels)$$

In addition to minimizing the cross-entropy loss, another crucial aspect during training is to enforce as much diversity as possible between the predictions of the subnetworks, without sacrificing accuracy. This can be achieved by leveraging the Kullback-Leibler (KL) divergence as a measure of this diversity. Specifically, the probabilities of the subnetworks are cyclically compared with those of the ensemble logits. Once the divergences are accumulated, they are normalized by the number of subnetworks and added to the final loss function, with a scaling parameter lambda to control their contribution. Since the KL divergence is not symmetric (i.e. $D_{KL}(P(x)\|Q(x)) \neq D_{KL}(Q(x)\|P(x))$), the choice to compare the probabilities produced by the subnetworks against the average of the logits (i.e., the ensemble logits) is driven by the fact that, to obtain a finer-grained calculation of all combinations, with n subnetworks, $\binom{n}{2}$ comparisons would be required. This results in a computational cost of $O(n^2)$ due to the double loop that would be needed. For completeness, experiments were conducted, and in the context of training, there were no significant differences between the two methods. This allowed for the selection of the less computationally expensive method.
Therefore, for each subnetwork i, the following is calculated:

$$KL\ Divergence = \frac{1}{n}\sum_{i=1}^{n}\mathcal{D}_{\mathrm{KL}_i}(P_{Logits_i}\|P_{Ensemble\ \ Logits})$$

The CE loss and the KL divergence are the two components of the final loss that needs to be minimized, which takes the following form:

$$\mathcal{L} = \mathcal{L}_{CE_{\text{ensemble}}} - \lambda_{\mathrm{KL}} \cdot \mathcal{D}_{\mathrm{KL}}$$

Where λ_{KL} is a weighting factor (empirically set to 0.3) to prevent the KL divergence term from becoming too large and causing numerical instability, and the contribution of the KL divergence has a negative value because it is defined between 0 and $+\infty$, so it needs to be maximized by inverting the sign.
The training results for this method are summarized in the plots below.

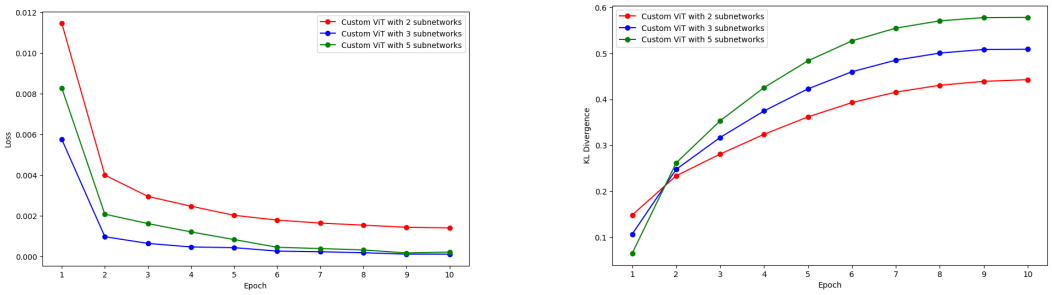


Figure 17. Plot of the training loss (left) and the increment of KL divergence (right) for the training related to ensemble averaging

4.2.2 Ensemble by Data Partitioning

The second method used in training is similar to the Bagging (or bootstrap aggregating) approach. In traditional bagging, the method involves creating n subsets (each at most as large as the original training set) through random sampling with replacement from the original training set. This approach allows the models to be independent of each other since, given that the size of each subset is equal to the size of the original training set, the unique samples (non-duplicates) in each subset would be about $1 - \frac{1}{e}$ (approximately 63.21%) [50]. This works well for large datasets like ImageNet-1k, which has over 1.2 million images in its training set. However, due to computational constraints in this work, the CustomViT model is trained on Imagenette, a smaller subset containing just over 10,000 samples in the training set. Using the bagging technique on such a relatively small dataset would not provide sufficient diversity among the subsets, leading to limited variability in the trained models.
This consideration led to the choice of a similar method, partitioning. The original training set was randomly divided into as many equal-sized subsets as there are virtual subnetworks in the model to be trained. Once the subsets are created, the training proceeds by training each of the subnetworks on their corresponding subset for each epoch. Unlike the first method, in this case, the KL divergence of the predictions is not tracked, as the diversity required for the generalization capability of the ensemble is sufficiently ensured by the difference in the datasets on which the subnetworks are trained. For this reason, only the cross-entropy (CE) loss is calculated for each subnetwork and then averaged and minimized in the final loss.

$$Subnetwork\ i\ Loss = CE(Logits_i\ ,\ labels)$$
$$\mathcal{L}_{tot} = \frac{1}{n}\sum_{i=1}^{n} Subnetwork\ i\ Loss$$

Where n is the number of subnetworks.
As explained in the section dedicated to experiments and results, partitioning the dataset instead of following traditional bagging will, in particular on certain tasks, lead to a significant improvement in performance when using 5 subnetworks.
Below is the progression of the training loss during the training.

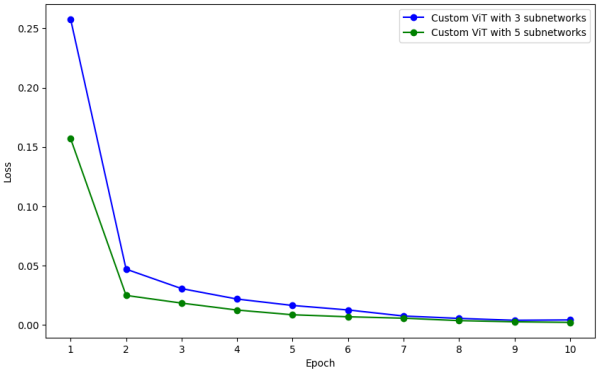


Figure 18. Plot of the training loss for the training related to data partitioning ensemble

4.2.3 Snapshot Ensemble

The third and final approach used to perform ensemble learning is the one already discussed in the related work section on Snapshot Ensemble. Due to computational limitations, it was not possible to extend the number of subnetworks beyond 5. However, the technique used in Snapshot Ensemble, combined with the creation of virtual subnetworks proposed in this thesis, allows for achieving 9 subnetworks "at the cost" of 3. This is because, by keeping the training epochs the same as in the previous cases (10 epochs), it was possible to divide this number into 3 cycles of 3 epochs each.
Following the method proposed in the original Snapshot Ensemble, it was possible to obtain 3 snapshots starting from the CustomViT model with 3 virtual subnetworks using the averaging ensemble. This resulted in 3 snapshots x 3 subnetworks, totaling 9 predictions per run.
To allow for rapid convergence to a local minimum, the starting learning rate was set to $5 \cdot 10^{-4}$, with the final learning rate set to $1 \cdot 10^{-5}$. At the end of each cycle (3 epochs), the optimizer and the scheduler are reinitialized with the learning rates mentioned above. This allows the SGD to escape from a local minimum and continue learning, decreasing the training loss. Once a cycle is completed, a snapshot of the model, along with its weights, is saved.
The body and method used in the training loop are the same (including the λ_{KL} parameter) as those used in the Averaging Ensemble case.
The logic behind this technique can be summarized as follows.

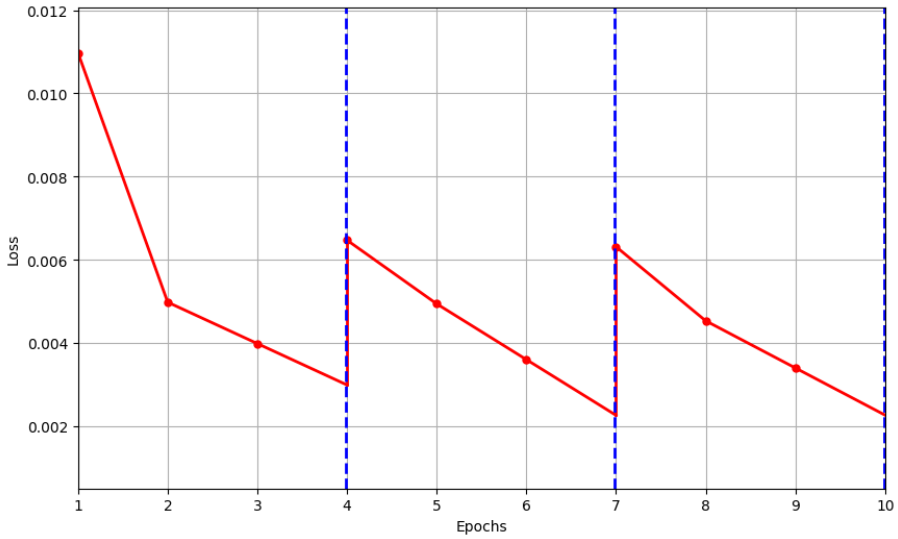
Training Custom Vit with Snapshot Ensemble approach

For each cycle:
    Re-initialize Optimizer and Scheduler with:
    $lr_{initial} = 5 \cdot 10^{-4}$ and $lr_{final} = 1 \cdot 10^{-5}$

    For each epoch in the cycle:
        Train Custom Vit:
            ⋆ Maximize KL Divergence
            ⋆ Minimize CE Loss

Save a Snapshot of the model

The training loss followed the expected trend, as in the original Snapshot ensemble research, as shown in the plot below.



It can be observed that, at the end of the 3rd and 7th epoch, when a cycle ends and the optimizer and scheduler are re-initialized with the parameters mentioned above, the training loss increases before continuing to decrease and find the next minimum, as seen from the vertical step. In the plot above, the dashed blue lines represent the moments when the model snapshots were saved, for a total of three times.

Chapter 5

Experiments and results

In this chapter, the various ensemble methods are tested on different tasks, including the Imagenette test set and data subjected to perturbations or out-of-distribution (OOD) datasets. The goal is to assess the performance and robustness of the models in various configurations, to determine which method is most suitable for each task, and to verify whether the application of the additional token concept is valid in all contexts or has certain limitations.
In addition to the models resulting from the ensemble techniques, a standard ("vanilla") ViT trained on the same training set and an ensemble of two "vanilla" ViTs, also trained on the same training set, were tested as baselines for comparison.

5.1 Performance metrics analysis

In the following section, the models are analyzed on image classification from Imagenette's test set using standard metrics. In the table below, it's presented a comparison of Accuracy and ECE (Expected Calibration Error), which measures how well the predicted probabilities of the models align with actual outcomes.

Model	**Accuracy**	**ECE**
ViT Vanilla	0.9437	0.0422
Ensemble 2 ViT Vanilla	0.9480	0.0397
Custom ViT - 2 Subnetworks (Average)	0.9786	0.0453
Custom ViT - 3 Subnetworks (Average)	0.9855	**0.0139**
Custom ViT - 5 Subnetworks (Average)	0.9822	0.0177
Custom ViT - 3 Subnetworks (Partitioning)	0.9840	0.0150
Custom ViT - 5 Subnetworks (Partitioning)	**0.9857**	0.0148
Custom ViT - 9 Subnetworks (Snapshot)	0.9572	0.1028

In the table below, the other standard metrics for classification, including precision, recall, and F1 score, are shown.

Model	Precision	Recall	F1 Score
ViT Vanilla	0.9434	0.9439	0.9435
Ensemble 2 ViT Vanilla	0.9480	0.9485	0.9481
Custom ViT - 2 Subnetworks (Average)	0.9789	0.9785	0.9787
Custom ViT - 3 Subnetworks (Average)	**0.9856**	0.9855	0.9855
Custom ViT - 5 Subnetworks (Average)	0.9824	0.9821	0.9822
Custom ViT - 3 Subnetworks (Partitioning)	0.9841	0.9840	0.9841
Custom ViT - 5 Subnetworks (Partitioning)	**0.9856**	**0.9858**	**0.9857**
Custom ViT - 9 Subnetworks (Snapshot)	0.9575	0.9575	0.9573

To conclude, the Confusion Matrices of the models:

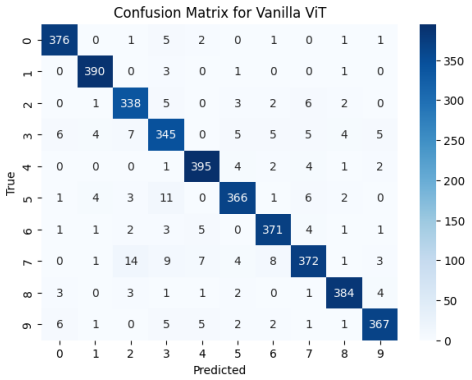


Figure 19. Confusion Matrix for Vanilla ViT

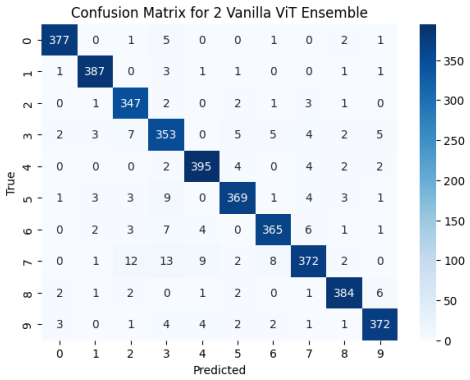


Figure 20. Confusion Matrix for 2 Vanilla ViTs Ensemble (Average)

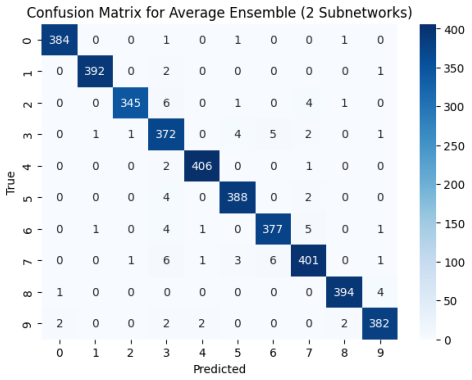


Figure 21. Confusion Matrix for Average Ensemble (2 Subnetworks)

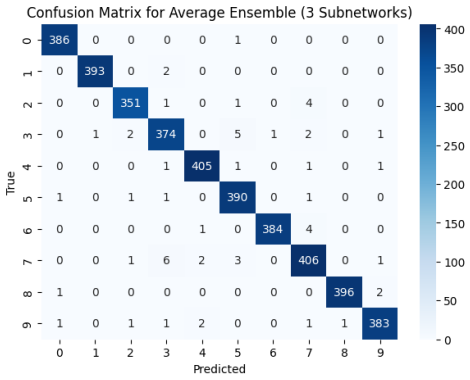


Figure 22. Confusion Matrix for Average Ensemble (3 Subnetworks)

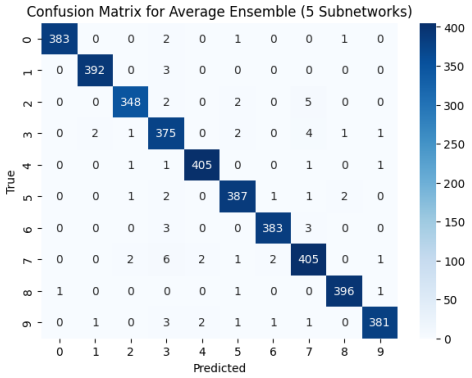


Figure 23. Confusion Matrix for Average Ensemble (5 Subnetworks)

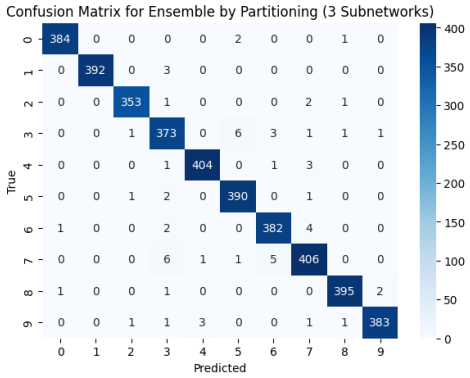


Figure 24. Confusion Matrix for Ensemble by Partitioning (3 Subnetworks)

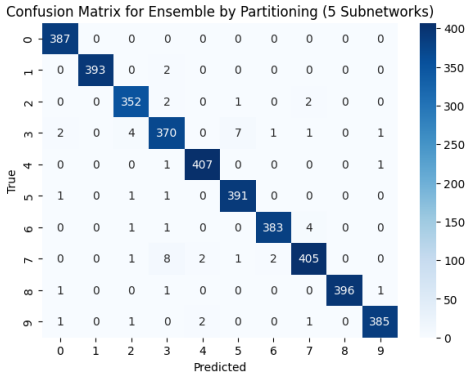


Figure 25. Confusion Matrix for Ensemble by Partitioning (5 Subnetworks)

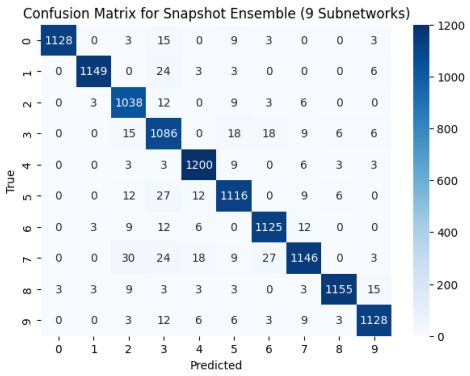


Figure 26. Confusion Matrix for Snapshot Ensemble (9 Subnetworks)

As seen from the results, all models built upon the foundation of Custom ViT outperform both the Vanilla ViT and the ensemble composed of two of these models. Interestingly, despite Vanilla ViT having a high accuracy, it is still surpassed by the others, which were trained for the same number of epochs, with the only difference being the additional inferences corresponding to the number of subnetworks, confirming the efficiency of the method proposed in this thesis.

5.2 Gaussian Noise Test

In this section, the models are subjected to varying levels of Gaussian noise to assess their robustness against perturbations. While Gaussian noise serves as a starting point for evaluating robustness, it represents only one type of corruption that can affect the quality of input data. Testing against Gaussian noise is a fundamental step in understanding a model's ability to handle distortions, as it simulates real-world scenarios [51] where data may not always be clean or perfectly structured. Models that perform well under such conditions demonstrate their capacity to handle slight variations in input without significant performance degradation. This type of robustness is essential for applications where inputs may be subject to various forms of noise or distortion, such as sensor inaccuracies, environmental factors, or transmission errors. In the case of Vision Transformers (ViTs), it is particularly important to evaluate how well these models generalize and maintain high accuracy when faced with noisy or incomplete data, with Gaussian noise being just one example of the many potential corruptions they may encounter.
For instance, in autonomous vehicles, noise from weather conditions or sensor limitations can affect the vehicle's ability to accurately interpret its surroundings [52], which is crucial for safe navigation. In healthcare imaging, noise in scans like MRIs or CTs can lead to distorted features [53], potentially causing incorrect diagnoses. Noise in remote sensing imagery degrades data interpretability, arising from factors such as thermal effects, sensor saturation, quantization errors, and transmission issues. This noise is typically independent of the data and generally additive, impacting the quality of satellite or drone-captured images and affecting critical applications like disaster response [54].
The process of adding noise to the input data can be described as follows:

$$noise \ = \ \mathcal{N}(\mu, \sigma^2)$$
$$\tilde{x} = x + noise$$

Here, σ is varied with increasing values, and x represents the original input. The perturbed input $\tilde{x}$ is subsequently clamped to ensure that the resulting noisy values remain within the valid range of $[0, 1]$.
Below is the visualization of the standard deviations applied to each input image.

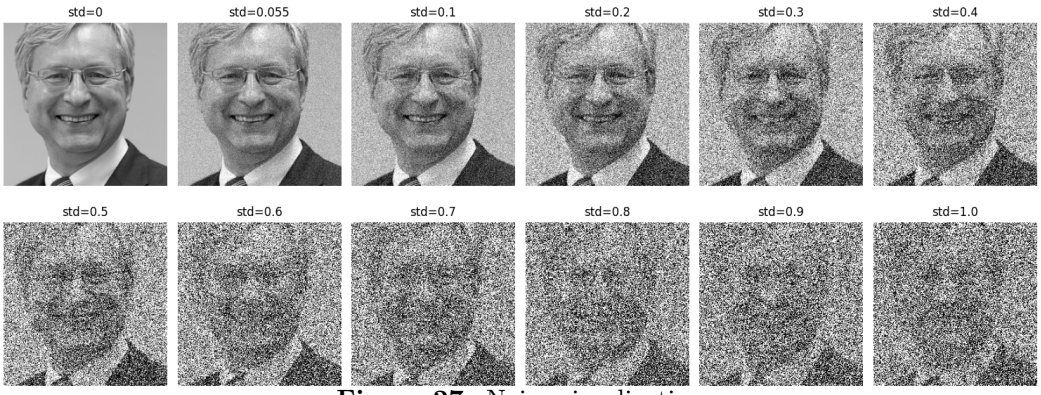


Figure 27. Noise visualization

The following table presents the accuracy and ECE results of various models as the noise added to the inputs increases.

Tables for accuracy:

Model \ std	**0.01**	**0.06**	**0.10**	**0.20**	**0.30**	**0.40**
Vanilla	0.7725	0.7727	0.7794	0.7794	0.7758	0.7687
Vanilla x2	0.7755	0.7771	0.7766	0.7845	0.7837	0.7806
Custom x2 (Avg)	0.8950	0.8973	0.8940	0.8904	0.8813	0.8718
Custom x3 (Avg)	**0.9039**	**0.9065**	**0.9070**	**0.9065**	**0.8930**	**0.8874**
Custom x5 (Avg)	0.8690	0.8734	0.8729	0.8713	0.8650	0.8517
Custom x3 (Part)	0.8813	0.8869	0.8851	0.8775	0.8688	0.8619
Custom x5 (Part)	0.8792	0.8798	0.8773	0.8731	0.8610	0.8504
Custom x9 (Snap)	0.8845	0.8854	0.8826	0.8758	0.8675	0.8563

Model \ std	**0.50**	**0.60**	**0.70**	**0.80**	**0.90**	**1.00**
Vanilla	0.7671	0.7554	0.7450	0.7169	0.6823	0.6359
Vanilla x2	0.7781	0.7674	0.7516	0.7317	0.6938	0.6466
Custom x2 (Avg)	0.8601	0.8397	0.8275	0.8163	0.8036	0.7761
Custom x3 (Avg)	**0.8782**	**0.8752**	**0.8599**	**0.8520**	**0.8461**	**0.8298**
Custom x5 (Avg)	0.8413	0.8283	0.8194	0.8008	0.7837	0.7740
Custom x3 (Part)	0.8530	0.8413	0.8400	0.8247	0.8104	0.7941
Custom x5 (Part)	0.8574	0.8540	0.8455	0.8391	0.8269	0.8135
Custom x9 (Snap)	0.8760	0.8649	0.8526	0.8404	0.8250	0.8136

Tables for ECE:

Model \ std	0.01	0.06	0.10	0.20	0.30	0.40
Vanilla	0.1377	0.1403	0.1353	0.1349	0.1318	0.1314
Vanilla x2	0.1271	0.1278	0.1241	0.1199	0.1208	0.1179
Custom x2 (Avg)	0.0706	0.0734	0.0728	0.0712	0.0733	0.0764
Custom x3 (Avg)	0.0672	0.0637	0.0650	0.0658	0.0733	0.0780
Custom x5 (Avg)	**0.0598**	**0.0609**	**0.0599**	**0.0608**	**0.0651**	**0.0686**
Custom x3 (Part)	0.0828	0.0807	0.0830	0.0872	0.0913	0.0968
Custom x5 (Part)	0.0854	0.0851	0.0847	0.0881	0.0925	0.0962
Custom x9 (Snap)	0.0787	0.0816	0.0803	0.0837	0.0881	0.0909

Model \ std	0.50	0.60	0.70	0.80	0.90	1.00
Vanilla	0.1260	0.1335	0.1416	0.1569	0.1780	0.2075
Vanilla x2	0.1186	0.1190	0.1301	0.1404	0.1572	0.1835
Custom x2 (Avg)	0.0791	0.0875	0.0918	0.0948	0.0945	0.1114
Custom x3 (Avg)	0.0810	0.0869	0.0916	0.0942	0.0992	0.1081
Custom x5 (Avg)	**0.0770**	**0.0762**	**0.0773**	**0.0795**	**0.0932**	**0.0896**
Custom x3 (Part)	0.1030	0.1088	0.1125	0.1239	0.1316	0.1462
Custom x5 (Part)	0.1079	0.1113	0.1162	0.1234	0.1289	0.1374
Custom x9 (Snap)	0.0941	0.0994	0.1023	0.1097	0.1148	0.1213

Analyzing specifically the top performers based on Custom ViT and vanilla ViT, it can be noted that in terms of accuracy, the Custom ViT with 3 subnetworks (Averaging) performs on average +17.54% better than the single vanilla ViT and +16.28% better than the ensemble of 2 vanilla ViTs. Meanwhile, for ECE, the Custom ViT with 5 subnetworks (Averaging) manages to lower this metric on average by -50.54% compared to the vanilla ViT and -45.29% compared to the ensemble of 2 vanilla ViTs.
Below, the plots of the top-performing Custom ViT models against the vanilla models.

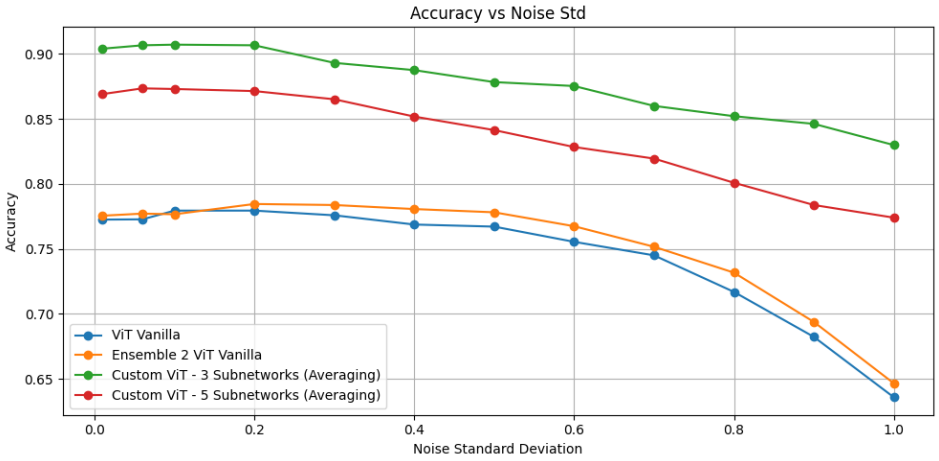


Figure 28. Trend of accuracy as the noise changes

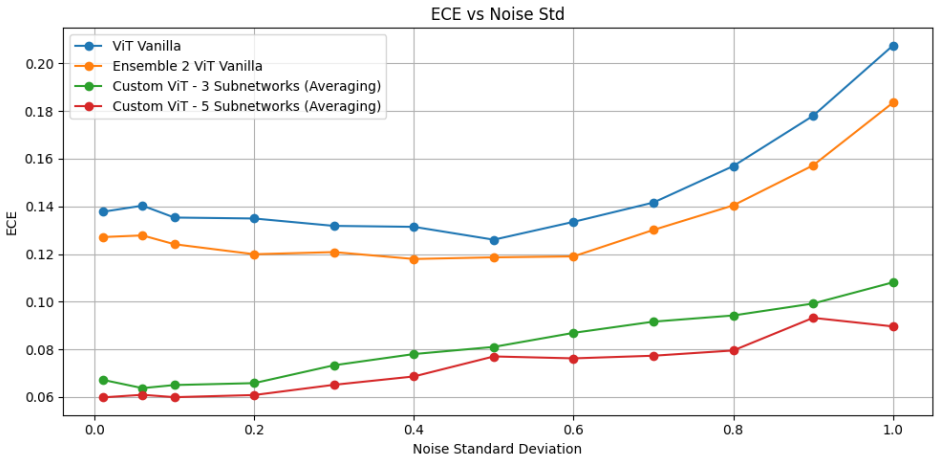


Figure 29. Trend of ECE as the noise changes

5.3 Adversarial Robustness

This section evaluates model performance when exposed to various adversarial attacks. Testing Vision Transformers (ViTs) under these conditions is essential for assessing their robustness and their ability to withstand real-world threats. Adversarial attacks simulate scenarios where models face deliberately crafted perturbations aimed at misleading them.
Robustness against such attacks is not merely a theoretical issue but a practical requirement. It ensures that ViTs are both reliable and secure in the unpredictable and varied environments they are likely to encounter.
In autonomous vehicles, for instance, adversarial attacks could distort the vehicle's interpretation of its surroundings, leading to the misreading of road signs or other vehicles, with potentially hazardous outcomes. In healthcare diagnostics, subtle alterations to medical images could result in misdiagnoses, endangering patient safety. In defense and military operations, ViTs are pivotal for surveillance and intelligence gathering. Adversarial manipulations in these areas could compromise critical data, jeopardizing national security. Similarly, in financial fraud detection, adversarial examples could exploit vulnerabilities in ViTs, allowing fraudulent activities to bypass detection and causing substantial financial damage.

For this thesis, the generation of adversarial attacks was conducted using the `torchattacks` library for PyTorch [55].

5.3.1 FSGM Attack

The Fast Gradient Sign Method (FGSM) [56] is a white-box attack, meaning it requires knowledge of the model's architecture and parameters. For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image. This can be summarized with the following expression:

$$adv_x = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

Where:

- x is the original input image
- y is the original input label
- ϵ is a multiplier to ensure that perturbations aren't too big to be discovered
- ∇ is the Loss function
- θ are the model's parameters

Putting it all together, the adversarial example is crafted to maximize the loss, causing misclassification while ensuring the perturbation is small enough to remain imperceptible.

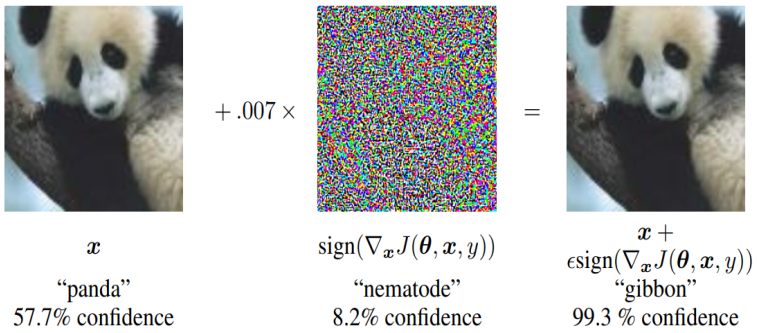


Figure 30. Demonstration of FGSM attack

The following results were obtained by generating adversarial examples using the FGSM attack with $\epsilon = \frac{8}{255} \simeq 0.03$.

Model	**Accuracy**	**ECE**
ViT Vanilla	0.6170	0.2890
Ensemble 2 ViT Vanilla	0.6349	0.3048
Custom ViT - 2 Subnetworks (Average)	0.7900	**0.1620**
Custom ViT - 3 Subnetworks (Average)	**0.7964**	0.1783
Custom ViT - 5 Subnetworks (Average)	0.7819	0.1913
Custom ViT - 3 Subnetworks (Partitioning)	0.7659	0.2080
Custom ViT - 5 Subnetworks (Partitioning)	0.7857	0.1872
Custom ViT - 9 Subnetworks (Snapshot)	0.6194	0.3219

It is worth noting that the models built on the Custom ViT approach perform significantly better than the Vanilla ViT, except for the model following the Snapshot Ensemble.

5.3.2 PGD Attack

The Projected Gradient Descent (PGD) attack [57] is an iterative white-box attack that extends the idea of FGSM by performing multiple steps of gradient updates with a smaller step size. It iteratively perturbs the original input image in the direction of the gradient of the loss with respect to the input, then projects the resulting image back into a valid image space within a defined perturbation bound. The projection step ensures that perturbations stay within a defined limit, preventing adversarial examples from becoming too large or falling outside valid input space, therefore maintaining their stealth and effectiveness. Moreover the iterative process allows the attack to explore the space of adversarial examples more thoroughly, making PGD stronger than FGSM.

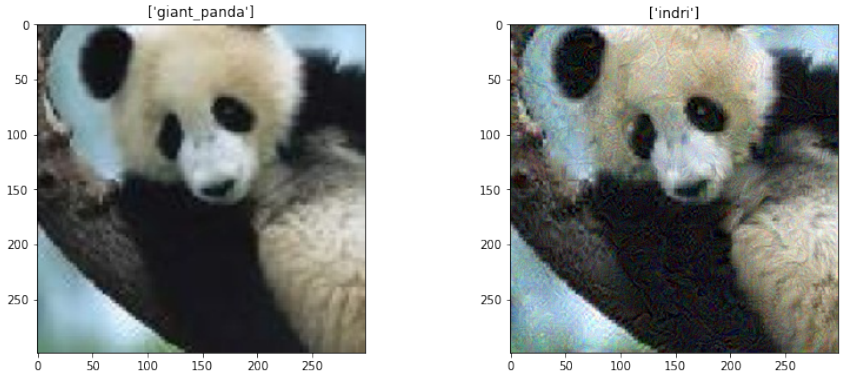


Figure 31. Demonstration of PGD attack

The following results were obtained by generating adversarial examples using the PGD attack with $\epsilon = \frac{1}{255}$, $\alpha = \frac{1}{510}$ and 5 steps, where ϵ has the same function as in the FGSM attack while α is a scaling factor for the perturbation.

Model	**Accuracy**	**ECE**
ViT Vanilla	0.6820	0.2210
Ensemble 2 ViT Vanilla	0.7004	0.2582
Custom ViT - 2 Subnetworks (Average)	0.7750	**0.1640**
Custom ViT - 3 Subnetworks (Average)	**0.7834**	0.1953
Custom ViT - 5 Subnetworks (Average)	0.7783	0.2024
Custom ViT - 3 Subnetworks (Partitioning)	0.7389	0.2368
Custom ViT - 5 Subnetworks (Partitioning)	0.7740	0.2056
Custom ViT - 9 Subnetworks (Snapshot)	0.7139	0.2519

As can be seen from the table, these results also reflect the previous ones.

5.3.3 Square Attack

Square Attack [58] is a black-box adversarial attack that aims to generate adversarial examples by perturbing the input image in a more structured way compared to gradient-based attacks like FGSM or PGD. Instead of relying on the model's gradients, Square Attack uses a random search strategy to iteratively modify small patches of the input image. The method works by selecting random regions of the image and adjusting their pixel values, gradually refining the perturbation to cause misclassification.
Square Attack reflects situations where the adversary has limited access to the model. It works by using feedback from the model's predictions to guide the attack, making it a versatile and efficient method in real-world settings where the model's details are often unknown.
Square Attack can be implemented both with l_∞ and l_2 norms. The l_∞ norm limits the perturbation on a per-pixel basis, allowing for more targeted and aggressive modifications, while the l_2 norm constrains the overall magnitude of perturbation across the entire image, leading to more subtle, widespread changes.

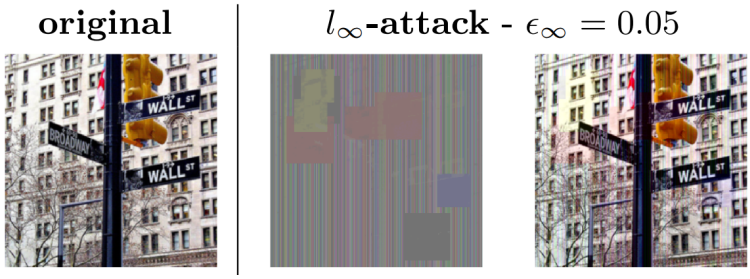


Figure 32. Demonstration of Square attack

The following results were obtained by generating adversarial examples using the Square attack with l_∞ norm, $\epsilon = \frac{4}{255}$, 50 iterations for each clean input image and finally $p = 80\%$ initial probability for each pixel to be attacked.

Model	**Accuracy**	**ECE**
ViT Vanilla	0.7460	0.1790
Ensemble 2 ViT Vanilla	0.7498	0.1807
Custom ViT - 2 Subnetworks (Average)	0.8820	0.0900
Custom ViT - 3 Subnetworks (Average)	**0.8882**	**0.0809**
Custom ViT - 5 Subnetworks (Average)	0.8800	0.0913
Custom ViT - 3 Subnetworks (Partitioning)	0.8624	0.1032
Custom ViT - 5 Subnetworks (Partitioning)	0.8759	0.0926
Custom ViT - 9 Subnetworks (Snapshot)	0.7824	0.1578

As it's possible to see from the table, these results are consistent with the previous ones, showing similar trends in accuracy and ECE across the different models.

5.3.4 CW Attack

The Carlini-Wagner (CW) Attack [59] is a powerful adversarial attack designed to create adversarial examples that are particularly difficult for the model to detect. It is a white-box attack, and formulates the generation of adversarial examples as an optimization problem, seeking to find the smallest perturbation to the input data that causes a misclassification by the target model. The optimization problem is carefully crafted to balance the imperceptibility of the perturbation with the effectiveness of inducing misclassification. Unlike other adversarial attacks that use simple gradient-based methods, the CW attack leverages a more sophisticated optimization technique by introducing a loss function that takes into account both the magnitude of perturbation and the misclassification of the model.
As for the Square attack, it can be implemented both with l_∞ and l_2 norms. The CW attack is often considered one of the strongest and most reliable adversarial attack methods due to its adaptability to a variety of scenarios and evaluation criteria, making it a standard benchmark for evaluating the robustness of models against adversarial examples.

The objective function has the following form:

$$J(x') = c \cdot l_2(x, x') + (1 - c) \cdot \mathcal{L}(f(x'), y_t)$$

Where:

- x is the original input
- x' is the perturbed input
- $l_2(x, x')$ measures the perturbation (this can be done also with l_∞ norm)
- $\mathcal{L}(f(x'), y_t)$ represents the misclassification loss of the model operating on the adversarial input with respect to the original labels
- c is a weighting factor that balances the perturbation and the misclassification

In the following image, the subtle yet important perturbation caused by a CW attack can be observed.

Figure 33. Demonstration of CW attack

The following results were obtained by generating adversarial examples using the CW attack with l_2 norm, $c = 0.1$, 5 steps for each clean input image and $lr = 0.01$ as the learning rate for the optimization process.

Model	**Accuracy**	**ECE**
ViT Vanilla	0.6820	0.2110
Ensemble 2 ViT Vanilla	0.7042	0.2038
Custom ViT - 2 Subnetworks (Average)	0.7810	0.1420
Custom ViT - 3 Subnetworks (Average)	**0.8132**	**0.1399**
Custom ViT - 5 Subnetworks (Average)	0.8071	0.1525
Custom ViT - 3 Subnetworks (Partitioning)	0.7781	0.1711
Custom ViT - 5 Subnetworks (Partitioning)	0.8041	0.1495
Custom ViT - 9 Subnetworks (Snapshot)	0.7213	0.1978

The results reflect the tests conducted on the other configurations, once again favoring the setup with average ensembling using 3 subnetworks.

5.4 Out-of-Distribution (OOD) Analysis

The closed-world assumption in neural networks is based on the belief that a model will predominantly encounter data similar to the one it has been trained on, often disregarding the possibility of data that deviates significantly from the training distribution. Out-of-Distribution (OOD) detection addresses this issue by enabling models to recognize and appropriately handle data that falls outside their training set, ensuring that they do not make unreliable predictions when faced with unfamiliar or unseen data. This ability is crucial in applications where data can vary over time or across different environments. For example, in medicine, an AI system with OOD brittleness might fail to detect rare diseases or misdiagnose patients, leading to incorrect treatments. Similarly, in home robotics, a robot might misinterpret an object or command that deviates from its expected set, resulting in unintended or even harmful actions. Given the increasing reliance on machine learning in safety-critical domains, improving OOD robustness is essential for ensuring the reliability and safety of AI systems.

OOD detection serves to identify when input data deviates from the distribution the model was trained on. Domain shift occurs when the distribution of data at test time differs from that used during training, potentially leading to performance degradation. In the context of OOD, the model must determine whether an image is in-distribution (ID) or out-of-distribution (OOD). This distinction is crucial to prevent incorrect predictions on unseen data. The model's behavior in such scenarios reflects its ability to generalize beyond IID conditions—i.e., how well it can adapt to data it has never encountered before, even from a different domain.

In evaluating Out-of-Distribution (OOD) detection models, two key metrics are often used: AUROC (Area Under the Receiver Operating Characteristic curve) and FPR95% (False Positive Rate at 95% True Positive Rate). In this thesis, these two metrics were also employed to assess the performance of the models.

- **AUROC:** It quantifies the model's overall ability to discriminate between in-distribution (ID) and out-of-distribution (OOD) data. It measures the area under the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds.
- **FPR95% :** It focuses on the model's performance at a specific point in the ROC curve. It measures the False Positive Rate (FPR) when the True Positive Rate (TPR) is fixed at 95%. This means that, assuming the model is correctly identifying 95% of the OOD samples, FPR95% indicates the percentage of false positives, i.e., how many ID samples are incorrectly classified as OOD when the model is performing as well as 95% TPR.

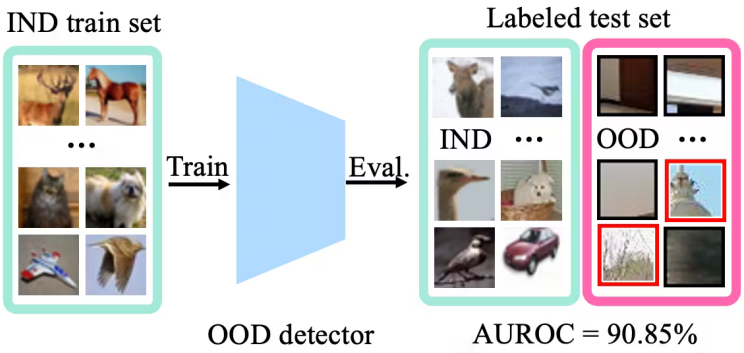


Figure 34. Example of OOD detection: the model classifies the images as either ID (in-distribution) or OOD (out-of-distribution)

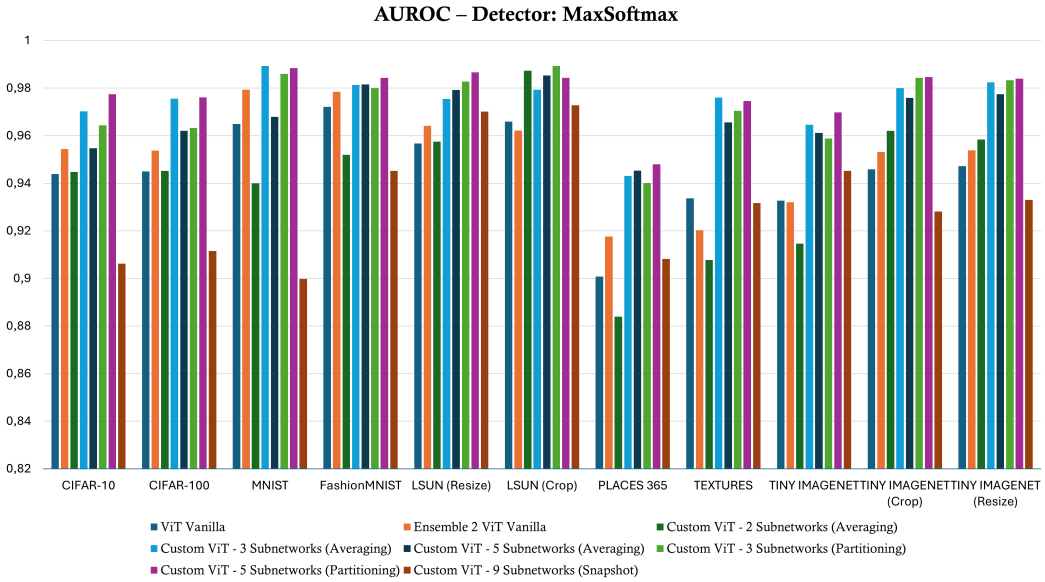
In this work, an analysis of OOD detection was performed on all the models proposed so far. These models were tested on 11 different datasets, each exhibiting varying degrees of domain shift compared to the Imagenette test set.
The following detectors were used:

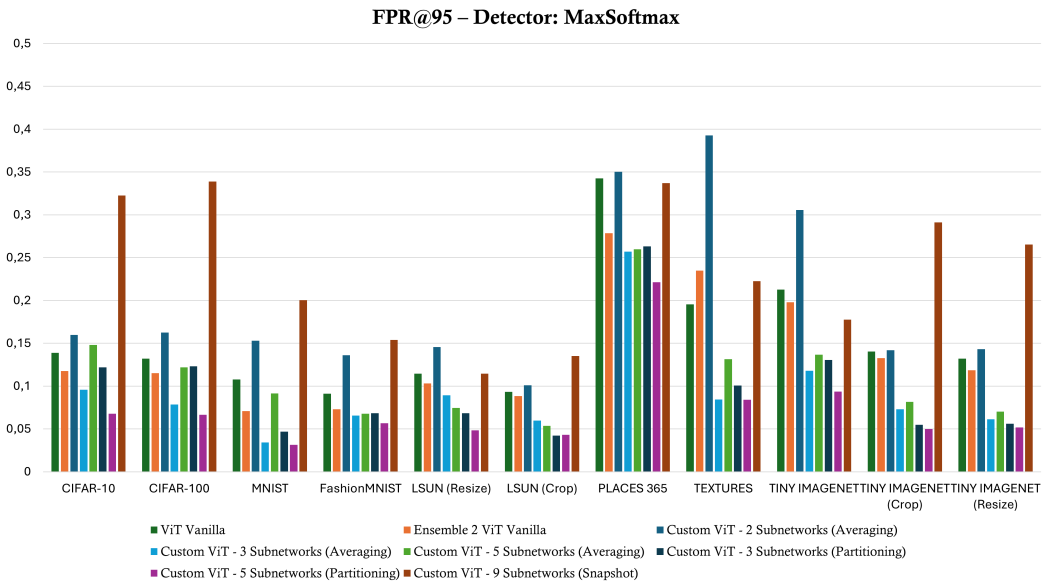
- **Max Softmax:** This method involves examining the model's output probabilities, which are obtained by applying the softmax function to the logits. For in-distribution data, the model typically assigns a high probability to the correct class, leading to a large maximum softmax value. Conversely, for out-of-distribution (OOD) data, the model often shows lower confidence across all classes, resulting in a more evenly distributed softmax output and a smaller maximum value. By comparing the maximum softmax probability of the model's output, we can effectively distinguish between in-distribution and OOD samples, as in-distribution samples will have a more pronounced peak in their probability distribution.

- **ODIN:** Out-of-DIstribution detector for Neural Networks is an OOD detection method that improves the logits by introducing a small temperature-scaled perturbation to the input. It applies a softmax temperature to smooth the logits, followed by adding a small perturbation based on the gradient of the softmax output with respect to the input. ODIN enhances the likelihood of in-distribution data through temperature scaling and input perturbation, thus making it easier to differentiate between in-distribution and OOD inputs.

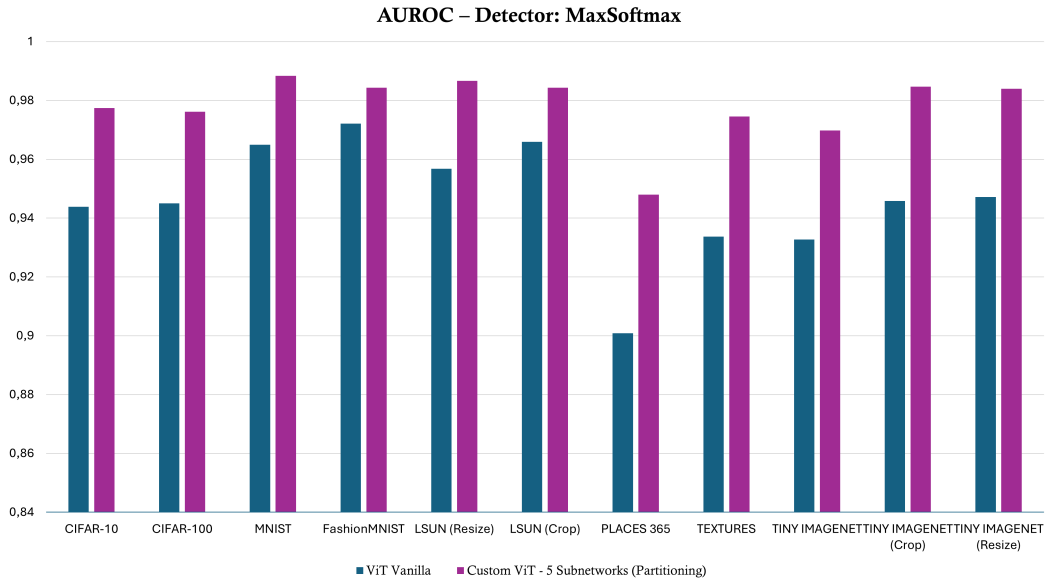
For both, the `pytorch-ood` library [60] was used.

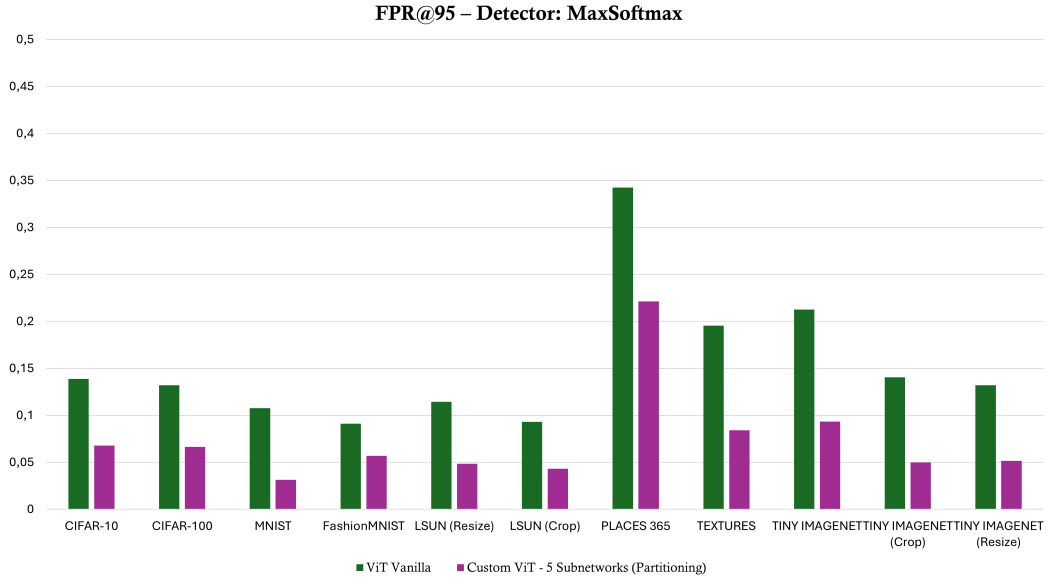
In the images below, the results of OOD detection using the MaxSoftmax detector are shown.





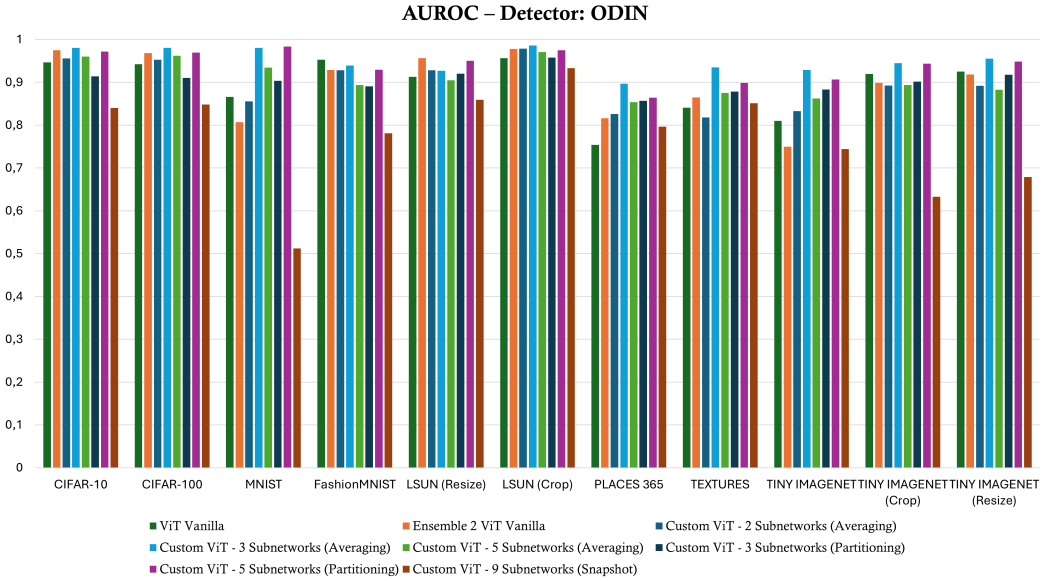
In order to improve clarity, given the large number of OOD datasets and models, the same results are presented, emphasizing the differences between the top-performing model based on Custom ViT (specifically the one obtained from 5 subnetworks with ensemble by partitioning) and the Vanilla ViT.

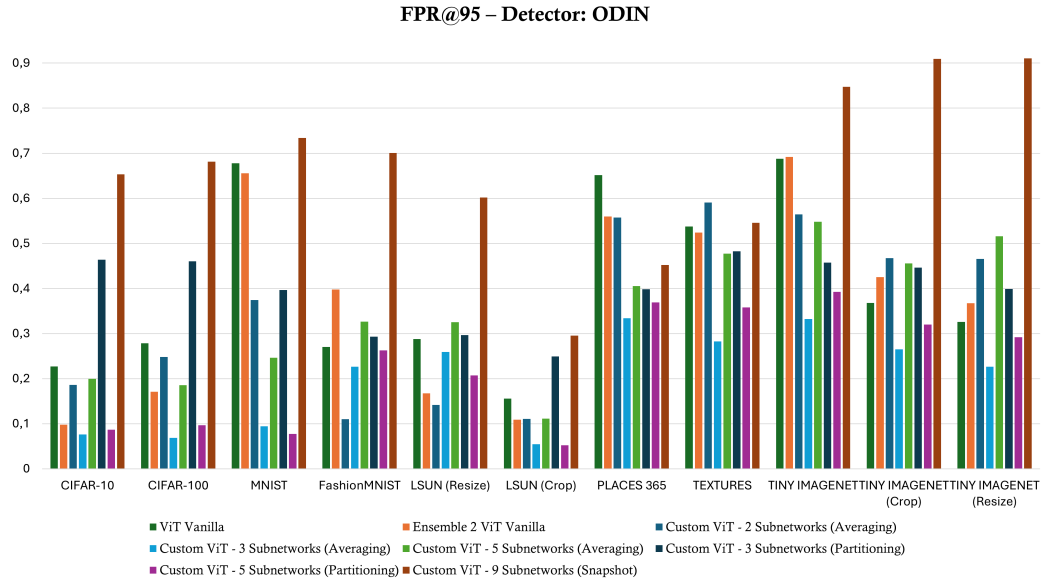




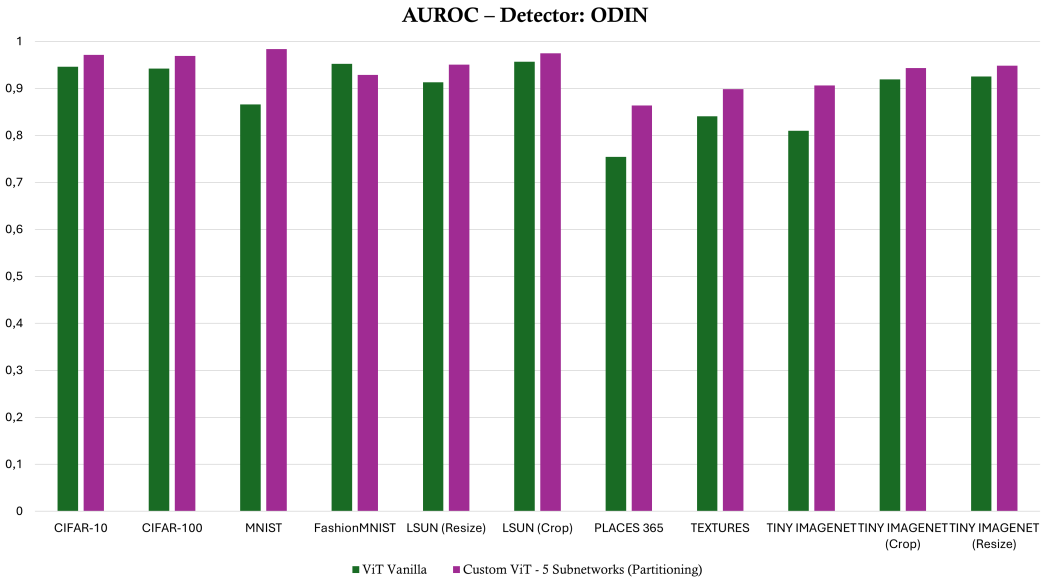
In particular, there is an average increase of 3.36% across the 11 datasets in AUROC between the top performer and the vanilla ViT. However, it is especially important to highlight a significant reduction of 52.11% in FPR@95%.

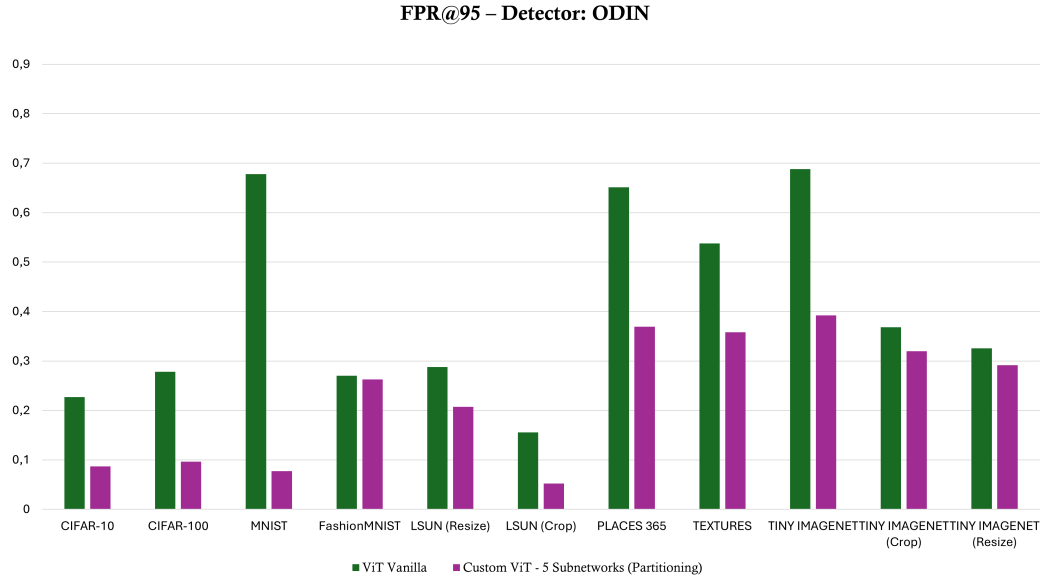
The images below present the results of OOD detection using the ODIN detector.





Again, in order to improve clarity, only the results between the top-performing model based on Custom ViT (specifically the one obtained from 5 subnetworks with ensemble by partitioning) and the Vanilla ViT are presented.





The improvements in ODIN are also more pronounced for FPR@95%, with an average decrease of 17.76% and an average increase of 5.23% for AUROC.

As observed in the OOD analysis, all Custom ViT-based models achieved better performance in OOD detection, thereby confirming the excellent suitability of the proposed method for robustness in such scenarios.

Chapter 6

Conclusions and future works

This work proposes an enhancement to existing Vision Transformers (ViTs) by integrating additional tokens and forming an ensemble of the corresponding virtual subnetworks generated from them.

There is a growing body of research emphasizing the critical need for smaller, more efficient neural networks, driven by the limitations and demands of real-world applications. Deploying AI on resource-constrained devices, such as smartphones and embedded systems, necessitates models with minimal memory and computational requirements. Additionally, power efficiency is essential for battery-powered devices and environmentally sustainable AI. Smaller models also often lead to faster inference speeds, which are crucial for real-time applications. Lastly, the drive for AI explainability calls for simpler models that are easier to understand and interpret, fostering greater trust and transparency in decision-making processes.

This thesis, with its contribution, aligns with this trend. The primary objective was to demonstrate how state-of-the-art structures can be improved without incurring significant computational or time-based costs. The proposed solution effectively extracts virtual subnetworks with minimal additional cost, only requiring as many inferences as there are subnetworks. This approach does not impose a substantial burden on the overall training and testing costs. In other words, it provides a means to achieve more robust and comprehensive models without exceeding the computational effort required for standard training (or what has been referred to in this work as the "vanilla" model).

Starting from the analysis of the performance in the preliminary tests, it was observed that even with the introduction of just two subnetworks, the results were still superior to those of the standard ViT, and even to an ensemble of two standard ViTs. While a few percentage points might seem marginal, it's important to note that these improvements were achieved at virtually no additional cost, simply by leveraging additional tokens and the diversity in input processing that they introduce. In addition to higher accuracy and precision, the diversity provided by the subnetwork ensembles also results in improved calibration. This enhanced calibration translates into greater reliability of the model's predicted probabilities.

The same diversity also ensures excellent robustness. To verify this, a series of experiments were conducted by introducing perturbations in the input data. The models initially showed a good response to data corrupted by Gaussian noise, maintaining both higher accuracy and lower ECE compared to the vanilla models.
This capability of the custom ViT-based models was also evident when tested with input corrupted by various adversarial attacks, whether they were model-aware or not.
Finally, through an analysis of OOD detection, the models based on the proposed solution demonstrated an outstanding ability to generalize, achieving remarkable in the tests on more than 11 popular datasets.
It is clear that not all of the proposed models outperform the standard ViT in the same way. However, the variety of ensemble techniques and subnetworks used has highlighted certain models for specific tasks. The versatility of this solution allows for selecting the ensemble technique and the number of subnetworks tailored to the particular task at hand. An example of this is the case of Snapshot Ensemble. Despite the outstanding success of the related paper and subsequent applications, it is paradoxical that it appears less effective in this work. However, it should be noted that such a method can certainly be improved by increasing the number of training epochs, allowing individual snapshots to have a greater impact on the final classification. This adjustment was deliberately not included in the current work in order to keep the training process fair across all models, ensuring consistency with the training setup shared by all.

Given the promising results of this work, there is still room for optimization and further improvement to contribute to advancing the field of robust and efficient machine learning models. Efficiency could be enhanced by considering more structural modifications to the vanilla ViT model, upon which the Custom ViT is based. As highlighted in the related works, various state-of-the-art techniques, particularly pruning methods for Vision Transformers, could be applied to the proposed solution. These methods would also improve the interpretability of the individual subnetworks' contributions to the final predictions, addressing both explainability and interpretability issues.
Another potential direction involves extending the experiments to real-world datasets and applications. This would provide a deeper understanding of how the proposed model performs in practical scenarios, such as handling real-time data or operating on mobile devices with limited resources. Given the solution's lightweight nature, this could also include introducing budget constraints during the initialization and creation of tokens that generate the subnetworks.
Lastly, expanding the scope to multimodal data would not only broaden the range of potential applications but also further improve the robustness and performance of the solution by integrating multiple types of input.

Acknowledgements

Al termine del mio percorso universitario, desidero dedicare alcune righe di questa tesi alla mia famiglia. La vostra pazienza, comprensione e fiducia non sono mai passate inosservate, e mi hanno permesso di raggiungere questo traguardo.
In particolare, vorrei ringraziare i miei nonni.
È grazie ai loro sacrifici e alle loro rinunce se in questi anni ho avuto la fortuna di vivere l'Università godendomi ogni momento, e di potermi permettere distrazioni che loro hanno sempre messo in secondo piano.
Il vostro esempio è per me fonte di orgoglio e guida e mi accompagnerà sempre.

Bibliography

[1] Alexey Dosovitskiy, et al., "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*", https://arxiv.org/abs/2010.11929

[2] Ashish Vaswani, Noam Shazeer, et al., "*Attention Is All You Need* ", https://arxiv.org/abs/1706.03762

[3] René Ranftl, Alexey Bochkovskiy, Vladlen Koltun, "*Vision Transformers for Dense Prediction*", https://arxiv.org/abs/2103.13413

[4] Pankaj Mishra, Riccardo Verk, et al. ,"*VT-ADL: A Vision Transformer Network for Image Anomaly Detection and Localization*" , https://arxiv.org/pdf/2104.10036

[5] Anurag Arnab, Mostafa Dehghani, et al. ,"*ViViT: A Video Vision Transformer*" , https://arxiv.org/abs/2103.15691

[6] Sagi O, Rokach L., "*Ensemble learning: A survey*" WIREs Data Mining Knowl Discov. 2018; 8 , https://doi.org/10.1002/widm.1249

[7] Dong, X., Yu, Z., Cao, W. et al., "*A survey on ensemble learning* Front. Comput. Sci. 14, 241–258 (2020), https://doi.org/10.1007/s11704-019-8208-z

[8] Ammar Mohammed, Rania Kora, "*A comprehensive review on ensemble deep learning: Opportunities and challenges*", https://www.sciencedirect.com/science/article/pii/S1319157823000228

[9] R. Maclin, D. Opitz, "*Popular Ensemble Methods: An Empirical Study*", https://arxiv.org/abs/1106.0257

[10] Surowiecki, J., "*The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations*" Doubleday & Co.

[11] Freund, Y. , "*A Short Introduction to Boosting*", https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf

[12] Robert E. Schapire, "*Explaining AdaBoost* ", http://rob.schapire.net/papers/explaining-adaboost.pdf

[13] Rätsch, G., Onoda, T. & Müller, KR. , "*Soft Margins for AdaBoost*", https://doi.org/10.1023/A:1007618119488

[14] Alexander Vezhnevets, Vladimir Vezhnevets , "*'Modest AdaBoost' – Teaching AdaBoost to Generalize Better*" , https://calvin-vision.net/bigstuff/hp_avezhnev/Pubs/ModestAdaBoost.pdf

[15] Shai Avidan , "*SpatialBoost: Adding Spatial Reasoning to AdaBoost*", https://www.merl.com/publications/docs/TR2006-014.pdf

[16] I. Palit and C. K. Reddy, "*Parallelized Boosting with Map-Reduce*", https://ieeexplore.ieee.org/document/5693449

[17] Kuncheva, L. I., "*Combining pattern classifiers: Methods and algorithms*", 2004

[18] Jiang, T., et al., "*Improved bagging algorithm for pattern recognition in UHF signals of partial discharges*", 2011

[19] Oza, N. C., "*Online bagging and boosting*", 2005

[20] Hothorn, T.,& Lausen, B., "*Double-bagging: Combining classifiers by bootstrap aggregation*", 2003

[21] Hugo Touvron, Matthieu Cord, et al., "*Training data-efficient image transformers & distillation through attention*", https://arxiv.org/abs/2012.12877

[22] Jun Liu, Haoran Guo, Yile He and Huali Li," *Vision Transformer-Based Ensemble Learning for Hyperspectral Image Classification*", https://doi.org/10.3390/rs15215208

[23] Ghamisi, P.; Plaza, J.; Chen, Y.; Li, J.; Plaza, A.J., "*Advanced Spectral Classifiers for Hyperspectral Images: A Review*", https://ieeexplore.ieee.org/abstract/document/7882742

[24] Melgani, F.; Bruzzone, L. , "*Classification of Hyperspectral Remote Sensing Images with Support Vector Machines*" , https://ieeexplore.ieee.org/abstract/document/1323134

[25] Chen, Y.; Nasrabadi, N.M.; Tran, T.D. , "*Hyperspectral Image Classification Using Dictionary-based Sparse Representation*" , https://ieeexplore.ieee.org/abstract/document/5766028

[26] Ham,J.; Chen, Y.; Crawford, M.M.; Ghosh, J. , "*Investigation of the Random Forest Framework for Classification of Hyperspectral Data*", https://ieeexplore.ieee.org/abstract/document/1396322

[27] Wu, H.; Saurabh, P. "*Convolutional Recurrent Neural Networks for Hyperspectral Data Classification*", https://www.mdpi.com/2072-4292/9/3/298

[28] Beyer, L.; Zhai, X.; Kolesnikov, A., "*Better plain ViT baselines for ImageNet-1k*", https://arxiv.org/abs/2205.01580

[29] Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; Jégou, H., "*Going deeper with image transformers*", http://openaccess.thecvf.com/content/ICCV2021/html/Touvron_Going_Deeper_With_Image_Transformers_ICCV_2021_paper.html

[30] Zhou,D.; Kang, et al., "*Deepvit: Towards deeper vision transformer*", https://arxiv.org/abs/2103.11886

[31] Renggli, C.; Pinto, A.S.et al., "*Learning to merge tokens in vision transformers*", https://arxiv.org/abs/2202.12015

[32] Sandler, M.; Zhmoginov, A.; Vladymyrov, M.; Jackson, A. "*Fine-tuning image transformers using learnable memory*", http://openaccess.thecvf.com/content/CVPR2022/html/Sandler_Fine-Tuning_Image_Transformers_Using_Learnable_Memory_CVPR_2022_paper.html

[33] Fayyaz, M.; Koohpayegani et al., "*Adaptive token sampling for efficient vision transformers*", https://link.springer.com/chapter/10.1007/978-3-031-20083-0_24

[34] Faris Almalik, Mohammad Yaqub, Karthik Nandakumar , "*Self-Ensembling Vision Transformer (SEViT) for Robust Medical Image Classification*" , https://arxiv.org/abs/2208.02851

[35] Bhojanapalli, S., Chakrabarti, A., et al., "*Understanding robustness of transformers for image classification*" , http://openaccess.thecvf.com/content/ICCV2021/html/Bhojanapalli_Understanding_Robustness_of_Transformers_for_Image_Classification_ICCV_2021_paper.html

[36] Mahmood, K., Mahmood, R., van Dijk, M., "*On the robustness of vision transformers to adversarial examples*", http://openaccess.thecvf.com/content/ICCV2021/html/Mahmood_On_the_Robustness_of_Vision_Transformers_to_Adversarial_Examples_ICCV_2021_paper.html

[37] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, Kilian Q. Weinberger, "*Snapshot Ensembles: Train 1, get M for free*", https://arxiv.org/abs/1704.00109

[38] Léon Bottou , "*Large-Scale Machine Learning with Stochastic Gradient Descent*" , https://link.springer.com/chapter/10.1007/978-3-7908-2604-3_16

[39] Ilya Loshchilov, Frank Hutter, "*SGDR: Stochastic Gradient Descent with Warm Restarts*" , https://arxiv.org/abs/1608.03983

[40] Yann LeCun, John Denker, Sara Solla , "*Optimal Brain Damage*" , https://papers.nips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html

[41] Jonathan Frankle, Michael Carbin, "*The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*", https://arxiv.org/abs/1803.03635

[42] D Blalock, JJ Gonzalez Ortiz, J Frankle, J Guttag, "*What is the state of neural network pruning?*" , https://link.springer.com/chapter/10.1007/978-3-7908-2604-3_16

[43] Mingjian Zhu, Yehui Tang, Kai Han, "*Vision Transformer Pruning*", https://arxiv.org/abs/2104.08500

[44] S Scardapane, A Baiocchi et al., "*Conditional computation in neural networks: Principles and research trends*" , https://arxiv.org/abs/2403.07965

[45] W. Fedus, B. Zoph, and N. Shazeer "*Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*", https://www.jmlr.org/papers/v23/21-0998.html

[46] J. Puigcerver, C. Riquelme, B. Mustafa, C. Renggli, A. S. Pinto, S. Gelly, D. Keysers, and N. Houlsby, "*Scalable transfer learning with expert models*", https://arxiv.org/abs/2009.13239

[47] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto,D. Keysers, and N. Houlsby , "*Scaling vision with sparse mixture of experts*", https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html

[48] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby. , "*Multimodal contrastive learning with limoe: the language-image mixture of experts*, https://proceedings.neurips.cc/paper_files/paper/2022/hash/3e67e84abf900bb2c7cbd5759bfce62d-Abstract-Conference.html

[49] Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton, "*Dynamic Routing Between Capsules*", https://proceedings.neurips.cc/paper_files/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html

[50] JA Aslam, RA Popa, RL Rivest, "*On Estimating the Size and Confidence of a Statistical Audit*", https://www.usenix.org/event/evt07/tech/full_papers/aslam/aslam.pdf

[51] Alexey Kurakin, Ian Goodfellow, Samy Bengio, "*Adversarial examples in the physical world* " , https://arxiv.org/abs/1607.02533

[52] Vargas, J., Alsweiss, S., Toker, O., et al., *"An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions*", https://www.mdpi.com/1424-8220/21/16/5397

[53] B Goyal, S Agrawal, BS Sohi, "*Noise Issues Prevailing in Various Types of Medical Images* " , https://www.academia.edu/download/57505625/BPJ_Vol_11_No_3_p_1227-1237.pdf

[54] Corner, B. R., Narayanan, R. M., & Reichenbach, S. E. , "*Noise estimation in remote sensing imagery using data masking*" , https://www.tandfonline.com/doi/abs/10.1080/01431160210164271

[55] https://github.com/Harry24k/adversarial-attacks-pytorch

[56] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, "*Explaining and Harnessing Adversarial Examples*" , https://arxiv.org/abs/1412.6572

[57] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu, "*Towards Deep Learning Models Resistant to Adversarial Attacks* " , https://arxiv.org/abs/1706.06083

[58] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, Matthias Hein, "*Square Attack: a query-efficient black-box adversarial attack via random search* " , `https://arxiv.org/abs/1912.00049`

[59] Nicholas Carlini, David Wagner, "*owards Evaluating the Robustness of Neural Networks*" , `https://ieeexplore.ieee.org/document/7958570`

[60] `https://github.com/kkirchheim/pytorch-ood`