ROMANIAN ACADEMY

School of Advanced Studies of the Romanian Academy

„Simion Stoilow" Institute of Mathematics

PhD Thesis Summary

Semi supervised learning of multiple tasks on concept neural graphs

PHD STUDENT

Pîrvu Mihai-Cristian

THESIS ADVISOR

Prof. Dr. Marius Leordeanu

2025

Contents

Chapter 1

Introduction

In the last few years, Machine Learning has changed a great deal. Large neural networks, trained with methods like backpropagation and stochastic gradient descent (SGD), have become the standard for working with large, complex datasets. This approach, known as Deep Learning, has led to many advances in fields like Computer Vision and Natural Language Processing. When I started the work presented in this thesis, I had already seen this shift from older, manual methods to the new data-driven ones. It was clear that deep learning was very powerful, but it was less clear if it was the complete solution for creating intelligent systems.

Recent studies suggest that simply making models bigger or giving them more data leads to smaller and smaller gains in performance. Theoretical work also shows that current neural networks have fundamental limits and cannot solve certain types of complex problems. My own feeling was that pure deep learning is a very important tool, but it is only one piece of a bigger puzzle. It can be compared to the intuitive, pattern-recognition part of our thinking, which is separate from the more structured, reasoning part. This thesis explores how we can build on the strengths of deep learning while addressing some of its limitations.

Key questions and research goals

To do this, my research focused on the combination of five key areas: deep neural networks, graphs, prediction consensus through ensembles, iterative semi-supervised learning with model distillation, and the practical application of aerial image understanding. I chose neural networks as the core technology, but used graphs to model how different sensors and data types relate to each other, just as we see the world through different senses. To make the predictions of these models more reliable, I used ensemble learning, where multiple models vote on a final decision. To make the most of available data, I used semi-supervised learning and distillation, where a model trained on a small amount of labeled data teaches a new model using a large amount of unlabeled data. Finally, all this work was grounded in the real-world problem of aerial image understanding, with the goal of creating models that are efficient enough to be used on robots and drones.

All of these can be seen in Figure 1.1 below as well.

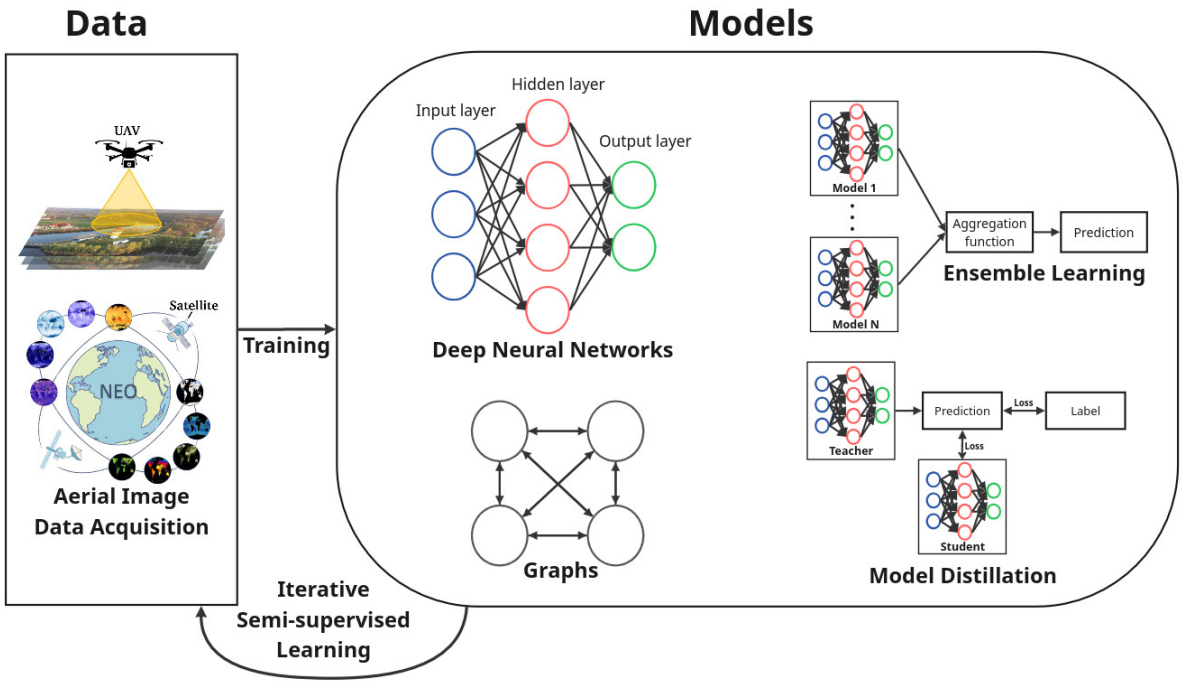


Figure 1.1: Main figure: the basic blocks of this thesis.

Datasets We introduced several new datasets to help the research community. These include a synthetic dataset for aerial scene understanding generated with the CARLA simulator, a real-world dataset called Dronescapes with flights over Romanian and Norwegian landscapes, and an aggregated dataset from NASA Earth Observations for predicting climate-related measurements. We also present an extension to Dronescapes and an open-source framework to help others create their own multi-modal datasets from videos.

Methods and Algorithms The methodological contributions focus on combining the core ideas presented earlier.

- We developed new methods for depth estimation by combining analytical and neural network approaches using ensembles.
- We introduced Concept Neural Hyper-Graphs as a way to perform multi-modal and multi-task learning, where each part of the graph represents a different view of the world.
- We proposed and validated iterative semi-supervised training methods that use graphs, ensembles, and distillation, providing both theoretical and practical evidence of their effectiveness on large aerial datasets.
- We showed that our methods produce more consistent and temporally coherent predictions, even when trained only on still images.
- We developed an efficient distillation approach to train simple, fast models from more complex ones, making them suitable for real-time use on hardware with limited resources.
- We presented a simplified and more efficient implementation of the Hyper-Graph con-

cept, showing it can be represented by a single neural network, which speeds up training and inference significantly.

The rest of the thesis is structured to build upon these ideas step-by-step. We will first cover the background concepts, then present our work on depth estimation, followed by the development of our graph-based models on both synthetic and real-world data, and finally conclude with our unified model and future directions.

Chapter 2

Background

This chapter gives a high-level introduction to the technical concepts used in the thesis. The goal is to define the standard terms and technologies we'll be using in later chapters. We will start with data for UAVs and multi-task learning in Section 2.1. Then, in Section 2.2, we will introduce basic concepts about Machine Learning and Neural Networks, explaining how they are defined, trained, and applied to deep learning problems. We will also discuss multi-task multi-modal learning and end the chapter with a brief introduction to graphs, as they are used a lot in the later chapters.

2.1 Data. The fuel that powers Machine Learning

Our work in [21] introduced a synthetic dataset with depth and "safe landing" maps. Lately, large vision-language datasets have also become popular, with billions of captioned images [30].

SafeUAV dataset

In our SafeUAV paper [21], we used Google Earth [13] to automatically create synthetic UAV-like images with labels. The idea was to use 3D reconstructions of real places to get as close as possible to actual flight scenarios. While the images weren't perfectly realistic, they were useful for training models that could transfer some of what they learned to real-world images.

The dataset has 11,907 samples, split into training (80%) and validation (20%) sets. We collected data from four different areas—two urban and two suburban—to make the dataset diverse. For each sample, we have a 640x480 RGB image, per-pixel depth data, and a semantic map that labels surfaces as horizontal, vertical, or other (HVO).

Scene	Surface Area	Training	Validation
Suburban A	1.7km^2	1966	492
Suburban B	1.1km^2	1049	263
Urban A	3.5km^2	3636	909
Urban B	3.3km^2	2873	819

Table 2.1: Distribution of the scenes in the SafeUAV dataset.

The goal of the original work was to find safe landing areas from RGB images. We framed this as a pixel-wise semantic segmentation task with three classes: 'horizontals' (safe to land), 'verticals' (obstacles to avoid), and 'others' (sloped or irregular surfaces). While this is a simplified view of the world (e.g., a horizontal water surface is not safe), it provides a good geometric understanding of the scene from just a camera feed. Figure 2.1 shows a sample with its corresponding HVO and depth labels.

Figure 2.1: SafeUAV dataset sample. Columns: RGB, HVO and metric depth

2.2 Models. Modeling data distributions with Machine Learning

Machine Learning has become a standard tool for modeling data. The approach has shifted over the years. Early on, experts would hand-craft features from the data, which were then fed into a simple learning algorithm. After 2012, end-to-end models that learn the features directly from raw data became popular. This data-driven approach, powered by large datasets and better hardware, has proven to be much more successful. Today, the focus is on building massive datasets and using self-supervised methods, though there is a growing interest in combining data-driven learning with fundamental domain knowledge.

Classical problems in Machine Learning

At its core, Machine Learning is about learning patterns from data. A dataset contains different variables or features. If a variable is a real number (like temperature), it's a regression problem.

If it's one of a fixed set of categories (like 'cat' or 'dog'), it's a classification problem. Some features are inputs (what we have), and others are outputs (what we want to predict).

In this thesis, we focus on two computer vision problems: semantic segmentation and depth estimation. Both are dense prediction tasks, meaning we have to make a prediction for every pixel in an image.

Semantic Segmentation This task involves assigning a class label to every pixel, like identifying all the pixels that belong to buildings, roads, or trees. It's a key part of scene understanding. A challenge is that the set of classes is usually fixed by the dataset creators.

Depth Estimation Here, the goal is to estimate the distance from the camera to each pixel in an image. This 3D information is very useful for robotics and navigation.

Neural Networks

Neural networks are the main technology used in Machine Learning today as well as this thesis. To use one, you need a labeled dataset, a model architecture, and a training algorithm. We can think of a neural network as a function $y = f(x)$, where x is the input data, y is the model's output, and the function f contains learnable weights, W, that are adjusted during training.

A network can be *discriminative*, meaning it predicts an output (like a class label or a value). Or it can be *generative*, meaning it creates new data similar to the input, like generating an image from noise. For classification (discriminative) problems, it's common to use a *one-hot* vector to represent classes. For example, with three classes, 'cat', 'dog', and 'duck', we would represent them as $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$. This way, the model learns that all wrong predictions are equally wrong.

Training Neural Networks

Training a neural network means adjusting its weights W based on a dataset. We use an algorithm called backpropagation. The process is:

1. Pass an input X_i through the model to get a prediction Y_i.
2. Calculate the error (or loss) between the prediction Y_i and the true label GT_i.
3. Update the weights W to reduce this error.

The update step is typically done using *gradient descent*, where we adjust the weights in the direction that most rapidly decreases the error. The goal is not just to memorize the training data (*over-fitting*), but to learn patterns that *generalize* to new, unseen data. To ensure this, we split our data into training, validation, and test sets. The model is trained on the training set, its performance is monitored on the validation set to prevent over-fitting, and the final evaluation is done on the test set.

Deep Neural Networks and various model architectures

A deep neural network is simply a neural network with multiple layers. Each layer applies a transformation to the output of the previous one. This layered structure allows the network to learn a hierarchy of features. Early layers might learn simple patterns like edges and corners, while deeper layers combine these to recognize more complex objects. Figure 2.2 shows how each layer in a network transforms the data, eventually making a complex, non-linear problem easy to solve for the final layer.

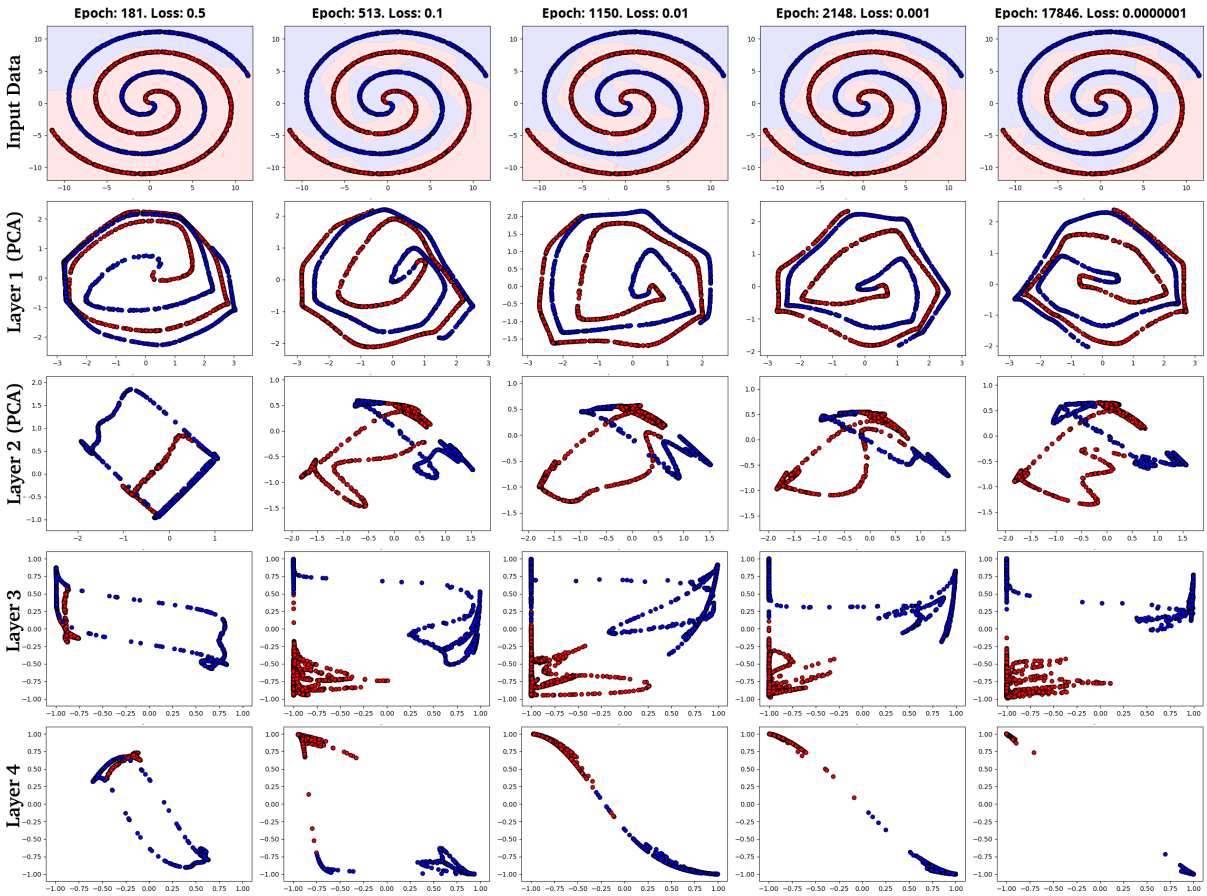


Figure 2.2: A four-layer neural network learning to classify a spiral dataset. We can see how the data is transformed at each layer, becoming linearly separable by the final layer.

Convolutional Neural Networks For image data, a special type of deep network called a Convolutional Neural Network (CNN) is very effective. We use this type of neural network across the entire thesis. Instead of connecting every input pixel to every neuron in the first layer, a CNN uses small, shared filters (or kernels) that slide across the image, reducing the number of operations. These filters are learned during training and are good at detecting local patterns like edges, textures, and shapes, regardless of where they appear in the image. These inductive biases of natural images make CNNs much more efficient, requiring less data to train complex data distributions.

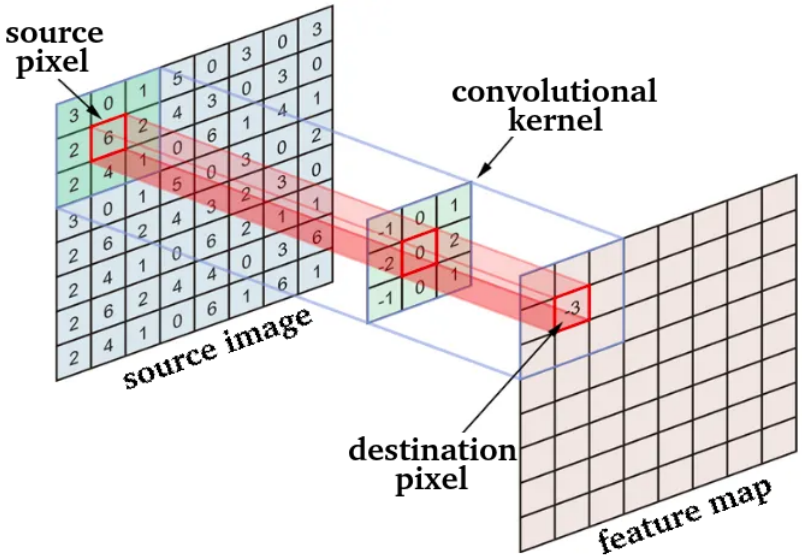


Figure 2.3: The convolutional operator. A small kernel (filter) slides over the input image to produce a feature map.

SafeUAV: Convolutional Neural Network architecture

One of our early contributions was an embeddable CNN for estimating depth and identifying safe landing areas (HVO segmentation). We designed a network based on the popular U-Net architecture. We created two versions: *SafeUAV-Net-Large* for higher accuracy and *SafeUAV-Net-Small* for real-time performance on embedded hardware like the NVIDIA Jetson TX2.

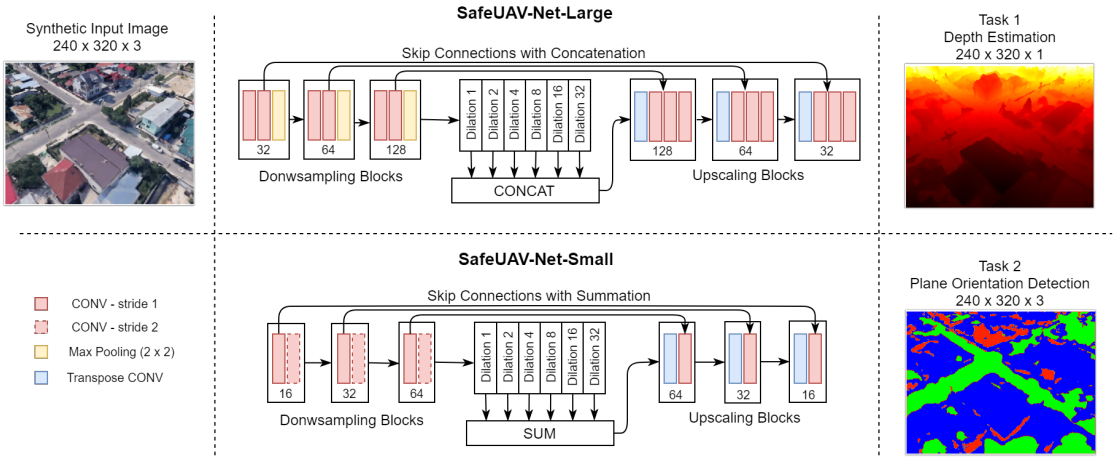


Figure 2.4: The proposed SafeUAV network architectures, used for tasks like depth estimation and semantic segmentation.

Both networks use an encoder-decoder structure with skip connections, which is typical for U-Nets. The smaller version uses fewer filters and a more efficient bottleneck design to reduce

computational cost. Table 2.2 shows a comparison of their size and speed.

Network	Number of Parameters	Memory Usage	FPS (Jetson TX2)
U-net [28]	31,031,745	1.7GB	37
DeepLabv3+ [5]	53,549,729	1.9GB	n/a
SafeUAV-Net-Large	23,896,129	927MB	35
SafeUAV-Net-Small	1,029,537	433MB	138

Table 2.2: Inference statistics for the SafeUAV networks compared to standard models.

Our experiments showed that these custom architectures performed competitively, and even better than some standard models at the time, on both depth estimation and HVO segmentation tasks on our SafeUAV dataset.

Multi-Modal Multi-Task Learning

In this thesis, we tackle the problem of multi-modality and multi-task learning. In a Multi-Modal Multi-Task (MTL) setting, we have multiple types of inputs (*modalities*) and/or multiple prediction outputs (*tasks*). For example, we might use both RGB images and depth data as input to predict both semantic segmentation and object locations as output.

A key challenge in MTL is deciding how to combine the inputs and how to structure the outputs. We can fuse inputs early (e.g., stacking image channels) or late (processing each input with a separate network before combining). For the outputs, we can use a shared decoder for all tasks or separate, specialized decoders for each one. These choices depend on the specific problem. One issue that can arise is *negative transfer*, where training on one task hurts performance on another.

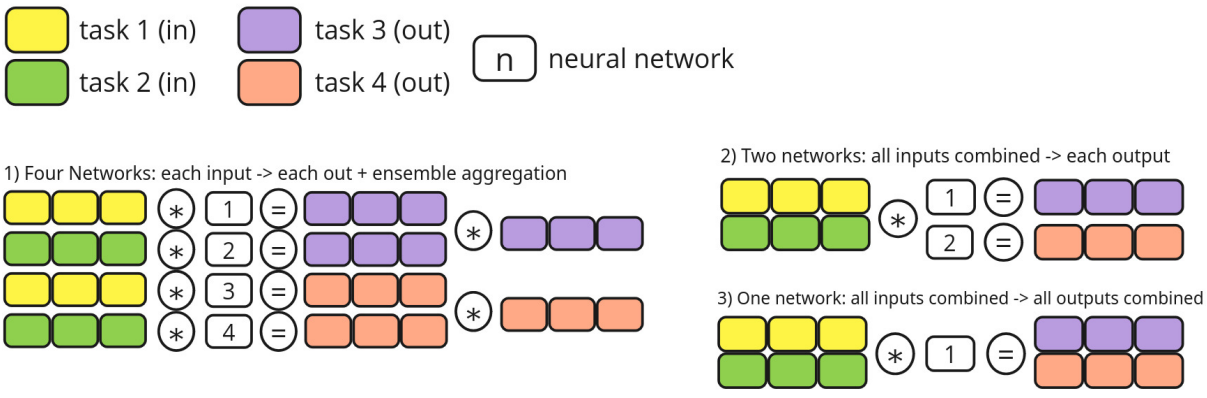


Figure 2.5: Different ways to model a 2-input, 2-output problem. Left: Separate models. Top right: Fused inputs, separate outputs. Bottom right: Fused inputs, shared output network.

Ensemble Learning

A standard neural network gives a prediction but doesn't tell us how confident it is. Ensemble Learning is a way to address this. The idea is to train multiple independent models on the same task and then combine their predictions. If the models are diverse and make different kinds of errors, the combined prediction is often more accurate and reliable than any single model's prediction.

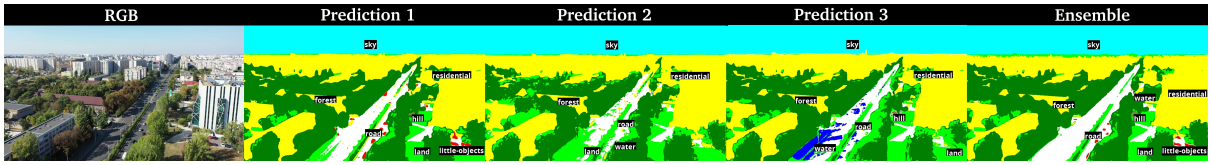


Figure 2.6: An example of Ensemble Learning. Three different models make predictions with different errors. By averaging them, we get a final prediction that is cleaner and more accurate.

The predictions are combined using an aggregation function. For regression tasks, this can be a simple average. For classification, it could be a majority vote. The key is that the models should be diverse; if all models make the same mistakes, the ensemble won't help.

Graphs

A graph is a mathematical structure made of nodes (or vertices) and edges that connect them. Graphs are a powerful way to model relationships between things. In Machine Learning, Graph Neural Networks (GNNs) have been developed to apply neural network concepts to graph-structured data, like social networks or molecular structures.

In our work, we use *graphs of Representations*. Here, each node in the graph represents an entire data view or modality (like RGB images or depth maps), and each edge is a neural network that transforms one view into another. This framework allows us to model the relationships between different tasks and modalities in a structured way and provides a natural bridge to Ensemble Learning, as multiple paths to the same node create an ensemble of predictions. It can also be seen as a special case of Probabilistic Graphical Modeling [19, 3].

Unsupervised and Self-supervised Learning

What if we have a lot of data but no labels? This is where unsupervised learning comes in. These algorithms try to find patterns and structure in the data on their own, for example by clustering similar data points together.

A powerful subset of this is *self-supervised learning*, where the model creates its own supervision from the data. A common technique is the autoencoder, which learns to compress data into a smaller representation (encoding) and then reconstruct the original data from it (decoding). By learning to reconstruct the input, the model learns meaningful features about the data without needing any external labels.

Another popular method is the Masked Autoencoder (MAE). In an MAE, parts of the input (like patches of an image or words in a sentence) are randomly hidden, and the model's task is to predict the missing parts based on the visible context. This forces the model to learn deep contextual relationships within the data.

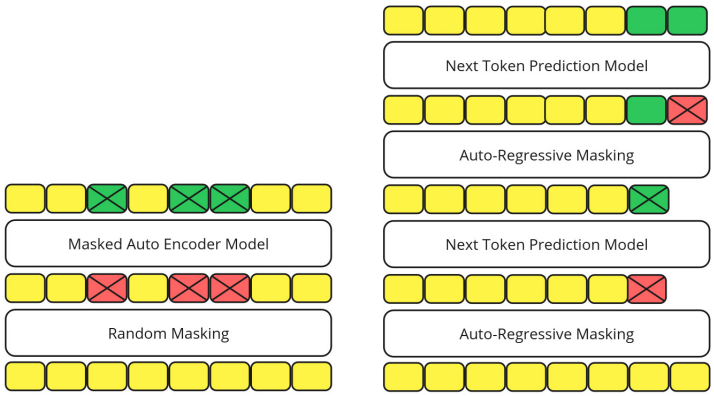


Figure 2.7: Two self-supervised learning methods. Masked Autoencoders (left) reconstruct missing parts of the input. Next Token Prediction (right) predicts the next item in a sequence.

Model Distillation

Model Distillation is a technique where knowledge from a large, complex model (*teacher*) is transferred to a smaller, more efficient model (*student*). First, the teacher model is trained on a task. Then, the student model is trained not on the original data labels, but on the predictions of the teacher model.

This process can be used to compress large models into smaller ones that can run on devices with limited resources. It can also sometimes lead to a student model that performs better than if it were trained on the original labels from scratch, as the teacher's "soft" predictions (probabilities rather than hard labels) can provide a richer training signal.

Semi-supervised Learning

Semi-supervised learning is a middle ground between supervised and unsupervised learning. It's used when you have a small amount of labeled data and a large amount of unlabeled data. The typical process is:

1. Train a *seed model* on the small labeled dataset.
2. Use this seed model to make predictions on the large unlabeled dataset. These predictions are called *pseudolabels*.
3. Combine the original labeled data with the newly pseudolabeled data.
4. Train a final, often larger, model on this combined dataset.

This approach can be very effective, especially when the pseudolabels are generated by a reliable model, such as an ensemble. It allows us to leverage vast amounts of unlabeled data to build better models than we could with the labeled data alone. This is a key technique we use in the later chapters.

Chapter 3

Ensemble learning and knowledge distillation of neural and analytical methods for depth estimation

This chapter introduces our work on unsupervised metric depth estimation for UAVs, which was first presented in our paper *Depth Distillation: Unsupervised Metric Depth Estimation for UAVs by Finding Consensus Between Kinematics, Optical Flow and Deep Learning* [25].

Estimating precise, real-world (metric) depth is important for UAVs to navigate safely. Doing this without direct supervision or odometry data is a hard problem. On the other hand, we can calculate depth using math from camera motion (kinematics) and optical flow. This analytical method is exact in theory, but it can be unstable and fails completely in some areas, like the focus of expansion. We propose a model that combines the best of both worlds: the precision of analytical methods and the robustness of unsupervised deep learning.

Our main contributions are:

1. A new method for estimating metric depth for UAVs using only an RGB camera. It learns in an unsupervised way from a single flight video by using an ensemble of an analytical method and a deep learning method as a "teacher" to distill knowledge into a final network.
2. An improved analytical method for depth estimation that is more robust because it simultaneously estimates depth and corrects for errors in the camera's angular velocity.
3. A new UAV dataset with nearly 20 minutes of flight video, including vehicle kinematics and GPS. This dataset is extended later in this thesis, creating the Dronescapes dataset.

As shown in Figure 3.1, we use two paths to estimate depth. The *analytical path* uses odometry and optical flow. The *data-driven path* uses an unsupervised deep network. These two paths supervise a final, single deep network. Then, through model distillation, the final network learns to estimate metric depth based on a consensus between geometry, camera motion, and the input image. The resulting network is small and fast, making it perfect for use on embedded devices. To get accurate metric depth, we use several methods. An unsupervised

network provides non-metric depth (D_{Unsup}), while another method uses odometry and optical flow to get metric depth ($D_{OdoFlow}$). We use $D_{OdoFlow}$ to give a real-world scale to D_{Unsup}. Together, these two form a "teacher" ensemble that trains a final student network. For evaluation only, we use a Structure from Motion (SfM) pipeline to generate a third depth map, D_{SfM}, which serves as our ground truth. While a SfM pipeline is slow, it can produce accurate depth maps.

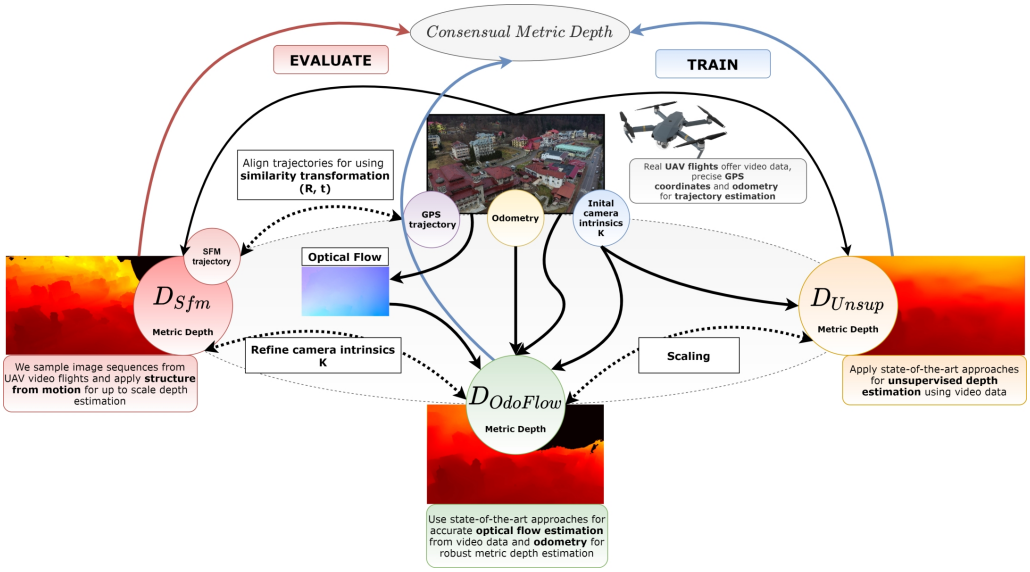


Figure 3.1: Overview of our method, combining analytical and data-driven pathways.

3.1 Metric depth distillation overview

Trajectory estimation from GPS We smooth the noisy GPS data by fitting 3rd-degree polynomials over a sliding window of measurements. This gives us a more stable trajectory estimate.

Analytical depth from odometry, flow and trajectory The relationship between camera motion, optical flow, and metric depth is described by the *Image Jacobian*. If we know the camera's linear and angular velocities and can compute the optical flow between frames, we can solve for a dense metric depth map. We also introduce a more robust way to do this, less sensitive to optical flow errors.

The motion of a pixel is related to the camera's linear velocity ν and angular velocity ω through the Image Jacobian matrix, as shown in the equation below. From this relationship, we can form a linear equation in terms of inverse depth $1/Z$: $(J_\nu \nu)\frac{1}{Z} + J_\omega \Delta\omega = \dot{p} - J_\omega \omega$, where $\dot{\tilde{p}}$ is the optical flow. The least squares solution for depth Z is then: $Z = \frac{\|A\|^2}{A^T b}$

However, this solution can be sensitive to errors in the estimated angular velocity. To make it more robust, we introduce a correction term $\Delta\omega$ and solve for both the depths Z_i and the correction simultaneously over multiple points. This is one of our theoretical contributions.

Post-processing for depth computation The solution for Z in the equation above becomes unstable near the focus of expansion, where the optical flow is close to zero. We filter out these unreliable depth values by applying thresholds on the optical flow magnitude and the angle between the motion vectors.

SfM alignment For evaluation, we use an offline SfM tool to reconstruct a 3D model of the scene. Since this model is not at a real-world scale, we align its trajectory with our GPS trajectory to make it metric. This gives us a high-quality depth map (D_{SfM}) to use as ground truth for our experiments.

Our system, shown in Figure 3.2, combines the outputs of the analytical method ($D_{OdoFlow}$) and the unsupervised network (D_{Unsup}) to form an ensemble teacher. A student network is trained to mimic this teacher, learning to predict metric depth from a single RGB image. The student's predictions are evaluated against the offline D_{SfM} ground truth, using an L1 loss that ignores invalid pixels in the SfM map.

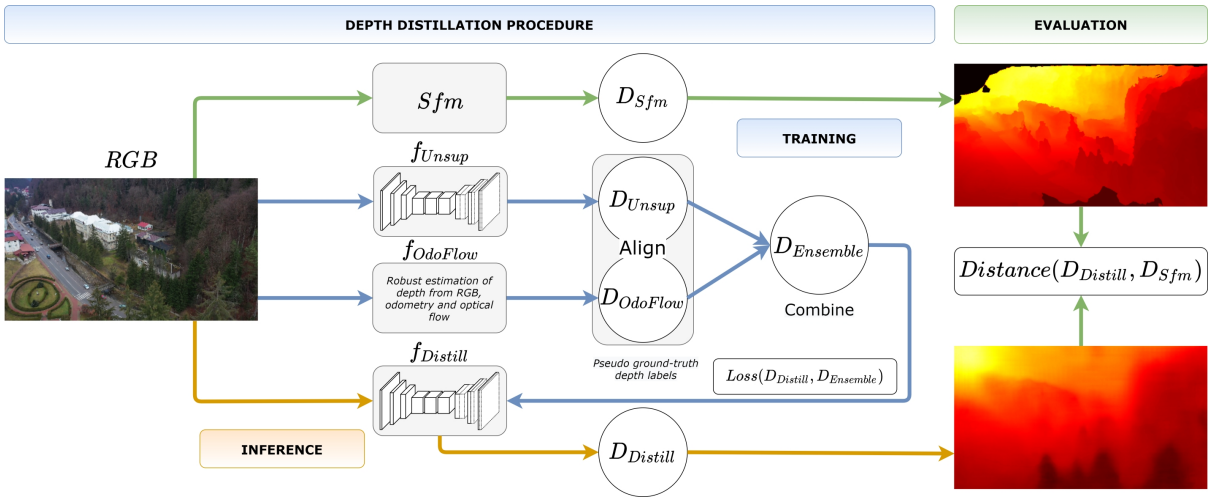


Figure 3.2: The distillation process. The analytical ($D_{OdoFlow}$) and unsupervised (D_{Unsup}) methods are combined into an ensemble teacher, which trains a student network to predict metric depth from a single RGB image.

3.2 Experimental Analysis

We introduce a new dataset from two UAV flights over mountain towns, which we call **Slanic** and **Herculane** (see trajectories in Figure 3.3). Slanic is split into training and testing sets to evaluate performance in a familiar scene. Herculane is used only for testing to see how well our method generalizes to a new environment.

Figure 3.3: UAV flight trajectories for the Slanic and Herculane datasets.

Setup and Results

We use the Meshroom tool [14] for SfM reconstruction, RAFT [31] for optical flow and DPT [27] for unsupervised depth. The student networks are two variants of the SafeUAV architecture, Tiny and Large, which are designed for real-time performance on embedded systems. We train the student using an L2 loss against the teacher ensemble, which is created by taking a pixel-wise average of $D_{OdoFlow}$ and the scaled D_{Unsup}.

Our results show that the distilled student network can outperform its teachers. As shown in Table 3.1, on the Slanic test set, the student network achieves a lower error (21.58 m) than the unsupervised method (27.28 m), the analytical method (26.05 m), and even the teacher ensemble itself (25.63 m). The same trend holds when we only look at the "good" areas where the analytical method is valid.

	Slanic		Herculane	
	Metric	Relative	Metric	Relative
D_{Unsup}	27.28 m	17.10 %	44.39 m	20.29 %
$D_{OdoFlow}$	26.05 m	16.34 %	**39.67 m**	**17.53 %**
$D_{Ensemble}$	25.63 m	15.88 %	41.18 m	18.29 %
$Tiny-16$	**21.58 m**	**14.58 %**	46.77 m	24.09 %
$Large-16$	21.84 m	14.65%	48.00 m	23.97 %

Table 3.1: Mean absolute and relative errors against D_{SfM} ground truth. The distilled student models outperform their teachers on the Slanic test set.

We also found that errors are much smaller for objects that are closer to the UAV, which is the most important region for tasks like obstacle avoidance. Qualitative results in Figure

3.4 show that the student network learns to combine the strengths of both teacher methods, producing smooth and detailed depth maps.

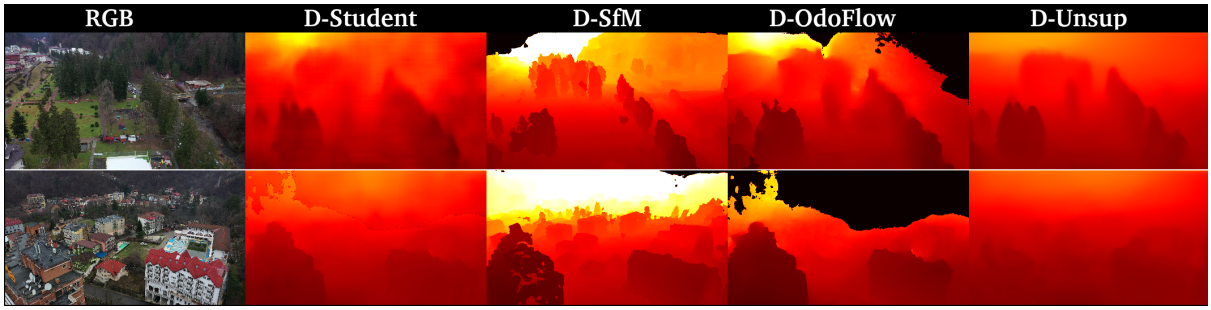


Figure 3.4: Qualitative results. From left: RGB, Student, SfM, OdoFlow, Unsupervised. The student network produces dense and accurate depth maps, learning to correct for the errors in its teachers.

Finally, our student networks are very efficient. Notably, the Tiny-16 model runs at over 10 FPS on an embedded Jetson TX2 platform, making it suitable for real-time deployment on a UAV.

3.3 Conclusions

No single method for metric depth estimation is perfect. However, by combining an analytical method with an unsupervised deep learning approach, we can create a powerful ensemble teacher. This teacher can then distill its knowledge into a single, lightweight student network that is fast, robust, and often more accurate than its teachers. Our evaluation on a new, challenging UAV dataset shows that this self-supervised training is effective and practical for real-world applications.

Chapter 4

Multi-Layer Neural Graph Consensus for Semi-supervised Learning

This chapter summarizes the work from our paper, *Semi-Supervised Learning for Multi-Task Scene Understanding by Neural Graph Consensus* [20], which was accepted at the *35th AAAI Conference on Artificial Intelligence (AAAI 2021)*.

We address the problem of semi-supervised learning when we have several different ways to interpret a visual scene, specifically for aerial image understanding. Our method uses a graph of neural networks to find a consensus between these interpretations. In this graph, each node represents an interpretation layer (like depth or semantic segmentation), and each edge is a deep network that transforms one layer into another.

The learning process has two phases:

- supervised phase: we train the edge networks using the available ground truth.
- semi-supervised phases (≥ 1) we use unlabeled data generated from the previous phase, supervised or semi-supervised on newly added data

The pseudo-labels are generated by finding a consensus among multiple paths in the graph that all lead to the same output. They act as ensemble teachers and the pseudo-labels are used to train a student model for the next phase. Through this iterative process, the entire graph becomes more consistent, even without new labels.

This method is called **Neural Graph Consensus (NGC)** and brings together the strengths of deep nets and graphs. Deep nets are powerful but need a lot of labeled data. Graphs can find global solutions from local information. As shown in Figure 4.1, NGC can connect many different tasks, such as predicting 3D structure, pose, and semantic classes, into a single framework. By forcing these tasks to agree with each other, they can effectively learn from unlabeled data. Our main contribution is the NGC model, a new framework for semi-supervised learning of multiple scene interpretations. We show how different tasks can teach each other through consensus, provide theoretical support for our approach, and demonstrate its effectiveness on a large-scale dataset.

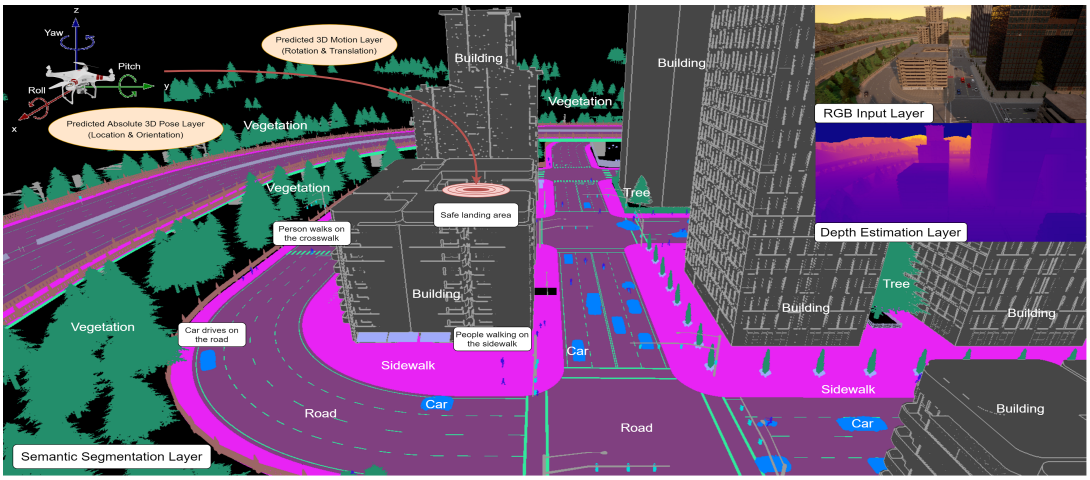
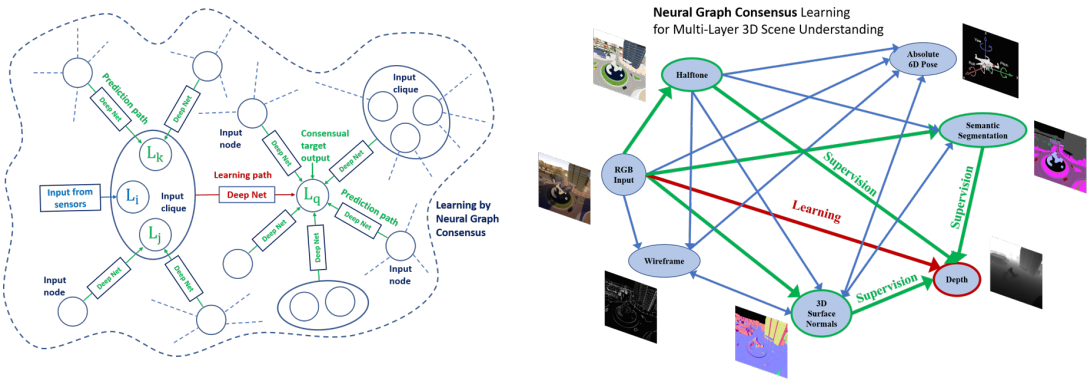


Figure 4.1: Different scene interpretations like 3D structure, pose, and semantics are connected in a neural graph. Multiple paths that reach the same node can act as a "teacher" through consensus to train a single edge network. This allows the model to learn robustly from unlabeled data.

4.1 NGC: Neural Graph Consensus model

In the NGC model, each node i holds a layer L_i, which is a specific interpretation of the world (e.g., a depth map or semantic segmentation). Edges in the graph are deep nets that predict a layer at one node from layers at other nodes. This model is show in Figure 4.2.



A. General Conceptual NGC Model

B. Actual NGC model implemented

Figure 4.2: NGC model: generic and implemented.

On the left, the theoretical graph model can have complex connections where many paths can reach a given node. The model works by alternating roles between edge networks. One network becomes the 'student' and is trained using the pseudo-ground truth generated by the consensus of all other paths reaching the same output node. These other paths act as

'teachers'. On the right we present the specific structure we used in our experiments to learn depth, semantic segmentation, and the position of a drone in a simulated environment.

Theoretical Analysis

The iterative semi-supervised learning process of NGC consists of three steps:

1. Pre-train a set of seed edge-networks on available labeled data.
2. Form the NGC graph by connecting the individual edges.
3. On a new unlabeled set, re-train the edge-networks using the consensual output of all paths reaching a specific node as pseudo-ground truth. Repeat Step 3 iteratively until no more new data is available.

Our analysis shows that for regression tasks, this ensemble learning approach minimizes the variance between the outputs of different paths. This leads to our first proposition:

Proposition 1 In a densely connected NGC graph, we expect the variance over the outputs reaching a given node to decrease during ensemble learning.

We validate this proposition experimentally, showcasing reduced variance, thus increased consistency in predictions for regression tasks. For classification tasks, where consensus is achieved through voting, we show that the accuracy of the ensemble teacher improves as we add more independent paths.

Proposition 2 If the success probability p for an edge net is better than random, the probability of success of the teacher ensemble using majority voting approaches 1 as the number of paths $N \to \infty$.

Our simulations, support these theoretical findings, indicating that performance improves with more paths in the ensemble and can continue to improve over multiple iterations. This analysis makes a few assumptions, such as the fact that each network learns complementary patterns. Obviously, if all the candidates produced the same result, the variance would be 0, but the added benefit of each candidate is also null. The corollary is that we need diverse candidates, which can be achieved by having multiple representations in our datasets, such as RGB (colors), edges (low-level textures), depth (mid-level vision) and semantics (high-level).

4.2 Experimental analysis

Dataset and Implementation

We created a large-scale dataset from a virtual environment using CARLA [9] where a drone flies over a city, as seen in Figure 4.3. For each image, we have ground truth for scene

depth, 3D surface normals, 6D pose, wireframes, and semantic segmentation into 12 classes. We implemented our NGC framework in PyTorch, using lightweight networks of about 1.1M parameters for each edge. The complete graph consists of 27 such networks.

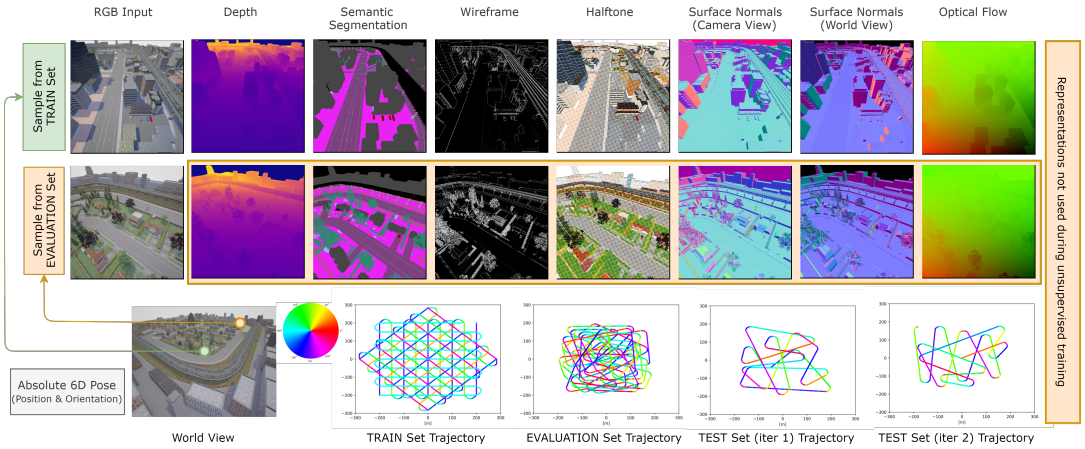


Figure 4.3: Samples from our synthetic dataset, showing RGB images and drone trajectories.

Iterative semi-supervised ensemble learning

We followed the iterative training process for three iterations. The first iteration was supervised training. The next two were semi-supervised, using unlabeled data. The results, presented in Table 4.1, show a consistent improvement across all tasks and for both the single "distilled" networks and the full NGC ensembles. This demonstrates that our consensus-based approach effectively leverages unlabeled data to enhance performance. Qualitative results in Figure 4.4 further illustrate these improvements.

		Iteration 1	Iteration 2		Iteration 3	
Representation	Evaluation Metric	EdgeNet	NGC	Distil. EdgeNet	NGC	Distil. EdgeNet
Depth	L1 (meters)	4.9844	3.4867	4.2802	**3.2994**	**3.9508**
	Pixels ↑ (%)	-	79.30	60.66	**79.69**	**61.90**
Semantic Segmentation	mIOU	0.4840	0.4978	0.4980	**0.5258**	**0.5159**
	Pixels ↑ (%)	-	79.46	69.62	**81.49**	**71.95**
Position	L2 (meters)	25.7597	15.5383	20.0204	**12.0764**	**15.5599**
Orientation	L1 (degrees)	3.8439	2.5001	3.3961	**2.2088**	**3.0005**

Table 4.1: Results over two iterations of unsupervised learning, showing consistent improvement for both NGC ensembles (**red**) and distilled single EdgeNets (**blue**). (Table abridged for brevity).

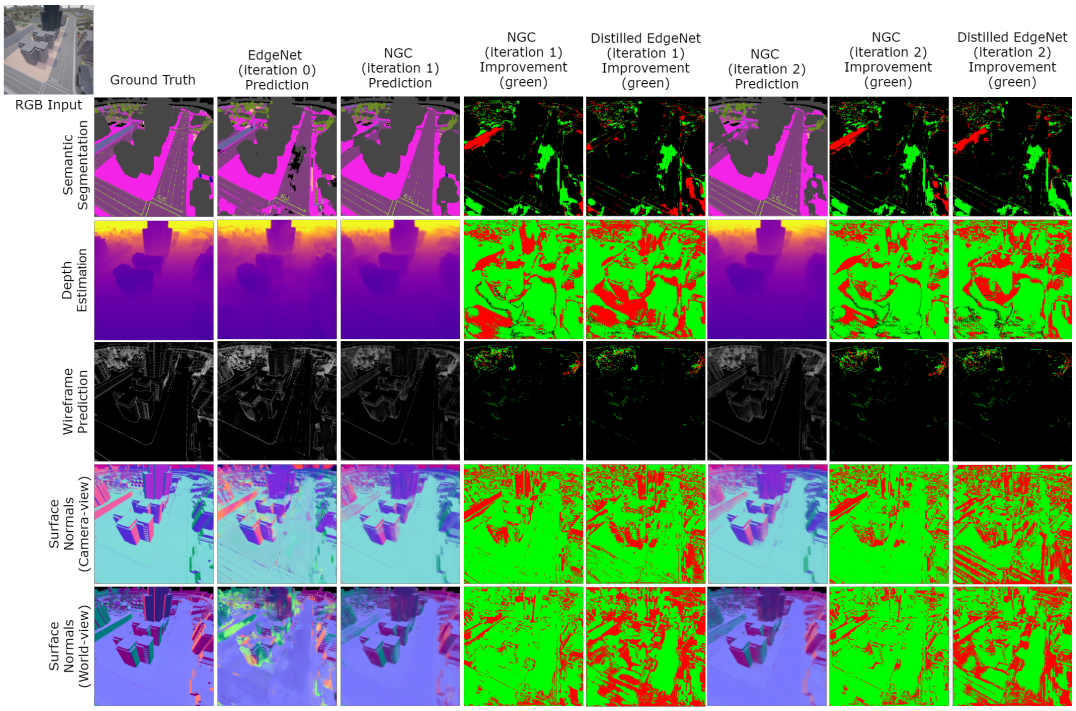


Figure 4.4: Qualitative results for NGC. Green shows areas of improvement after semi-supervised learning, while red shows areas of degradation.

Comparisons to State-of-the-art

We compared NGC against leading methods in several related areas.

- **Multi-Task Learning:** We tested against NDDR [11] and MTL-NAS [12]. While they performed well on one task (normals), our method showed better overall performance, especially on semantic segmentation.
- **Semi-supervised Learning:** We compared with CCT [24], a general semi-supervised method. Our NGC significantly outperformed CCT on semantic segmentation on our dataset, both in absolute scores and in the improvement gained from unlabeled data.
- **Ensemble Methods:** We compared NGC against standard ensembles of networks. NGC, which uses diverse intermediate representations, performed better than ensembles of networks all trained for the same task.

4.3 Conclusions

We introduced the Neural Graph Consensus (NGC) model, a new method for multi-task semi-supervised learning. NGC combines many deep networks into a single graph structure where they learn from each other through mutual consensus. Our experiments, supported by theoretical analysis, show that this approach is highly effective. The model successfully learns seven

different scene interpretations from single images with top performance, demonstrating that learning from consensus in large, collaborative graphs is a promising direction for unsupervised and semi-supervised learning.

Chapter 5

Multi-Layer Hyper-Graphs for Semi Supervised Learning

This chapter summarizes our work on extending graph-based learning to hyper-graphs, as introduced in our publications [22, 26]. We build upon the previous chapter by introducing more complex hyper-edges and learnable ensembles, and we validate our method on two challenging real-world scenarios: aerial image understanding and earth observation. The graph can now contain hyper-nodes and hyper-edges which consist of multiple representations at the same time (i.e. RGB plus edges) alongside regular edges. We apply the previously introduced iterative semi-supervised learning process. This process allows the system to improve over time as more unlabeled data becomes available, even when annotations are scarce.

We introduce two different datasets: Dronescapes, a new UAV video dataset, and the NASA Earth Observations (NEO) dataset, which contains 22 years of satellite data. As shown in Figure 5.1, our approach proves effective for both urban scene understanding and large-scale earth observation, improving accuracy and temporal consistency.

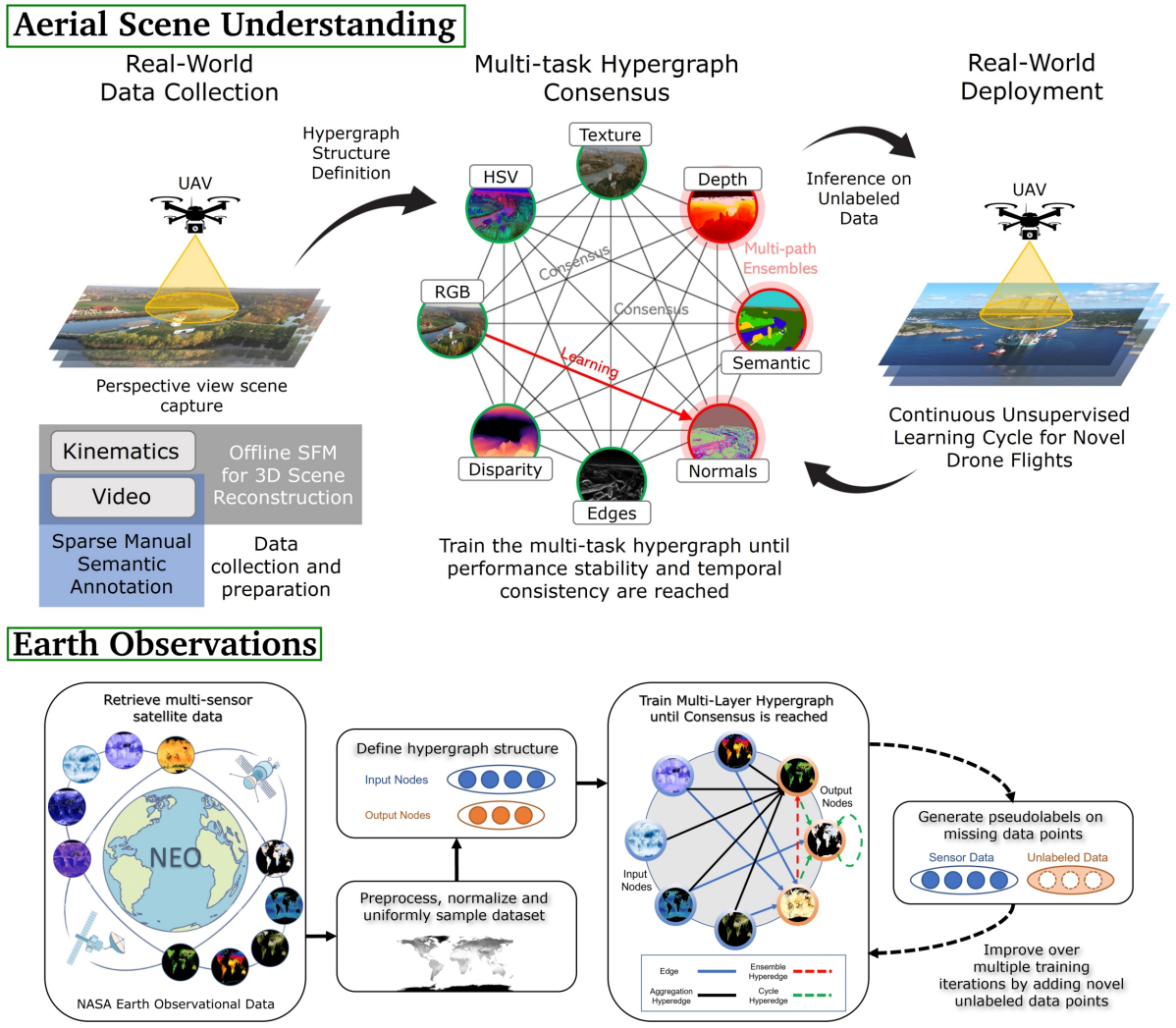


Figure 5.1: Our semi-supervised multi-layer Hyper-Graph in the context of real-world UAVs and Earth Observations. We define the Hyper-Graph structure in terms of input nodes (from sensors) and output nodes (those that will be predicted). We train in a semi-supervised manner over multiple iterations, which improves both accuracy and temporal consistency.

Our work is positioned at the intersection of several fields. While most unsupervised learning methods focus on a limited number of tasks, our model is general and can handle many tasks at once. Unlike consensus-based methods that use simple image transformations, we focus on meaningful scene representations and learn to balance them. In the area of graph-based learning, previous work often uses simpler graph structures with pairwise edges and non-learned ensembles. Our approach introduces higher-order hyper-edges and learns the ensemble mechanism itself. Finally, we use a form of self-distillation, where strong ensemble "teachers" from one iteration guide the training of simpler "student" edges in the next, a concept that builds on established knowledge distillation principles.

5.1 Multi-Layer Hyper-Graph model

Hyper-Graph Structure and Edge Types

Our Multi-Task Hyper-Graph, shown in Figure 5.2, consists of input nodes (N_i), representing known data from sensors, and output nodes (N_o), representing the world views we want to predict. These nodes are connected by different types of edges and hyper-edges, which are modeled by small U-Net neural networks.

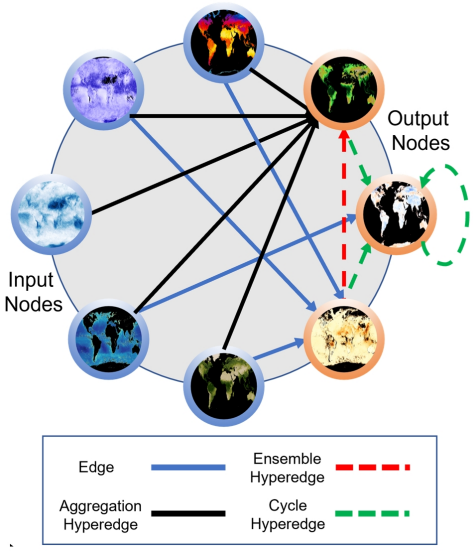


Figure 5.2: Multi-Task Hyper-Graph. Input and output nodes, edges and hyper-edges.

As an extension to the previous chapter which only used simple edges, we introduce several new types of hyper-edges to capture more complex relationships between layers, as illustrated in Figure 5.3:

- **Single-hop Edges (E)**: A simple transformation from one input node to one output node.
- **Two-hop Edges (TH-E)**: A path that goes through an intermediate predicted node.
- **Ensemble Hyper-Edges (EH)**: Combines all single-hop predictions for one node to help predict other nodes.
- **Aggregation Hyper-Edges (AH)**: Concatenates all input nodes to predict a single output node.
- **Cycle Hyper-Edges (CH)**: Uses all inputs and the outputs of AH hyper-edges to make a final prediction for a node.

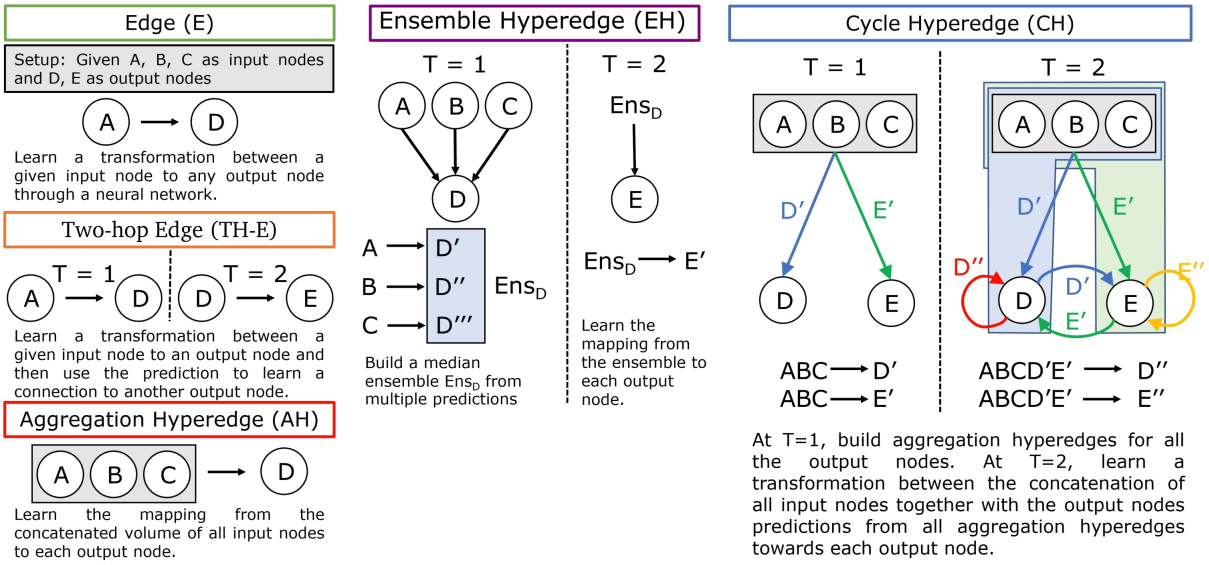


Figure 5.3: Types of edges and Hyper-Edges in the Hyper-Graph. In Chapter 4, we only used edges (E and $TH-E$), while here we introduce hyper-edges (AH, EH and CH) to capture complex relations between layers.

Semi-Supervised Iterative Training and Ensembles

We re-use the same iterative semi-supervised training algorithm described in the previous chapter in Section 4.1. A key novel contribution is our method for **learning the hyper-edge ensembles**. Instead of just averaging the candidate predictions, we introduce several learnable models, shown in Figure 5.4. These range from a simple linear model (**S-L**$_{FW}$) to more complex neural networks that can dynamically weigh each candidate prediction at the pixel level (**S-NN**$_{DPW}$) or learn a direct mapping to the final output (**S-NN**$_{D}$). These learned ensembles act as powerful teachers, guiding the semi-supervised learning process.

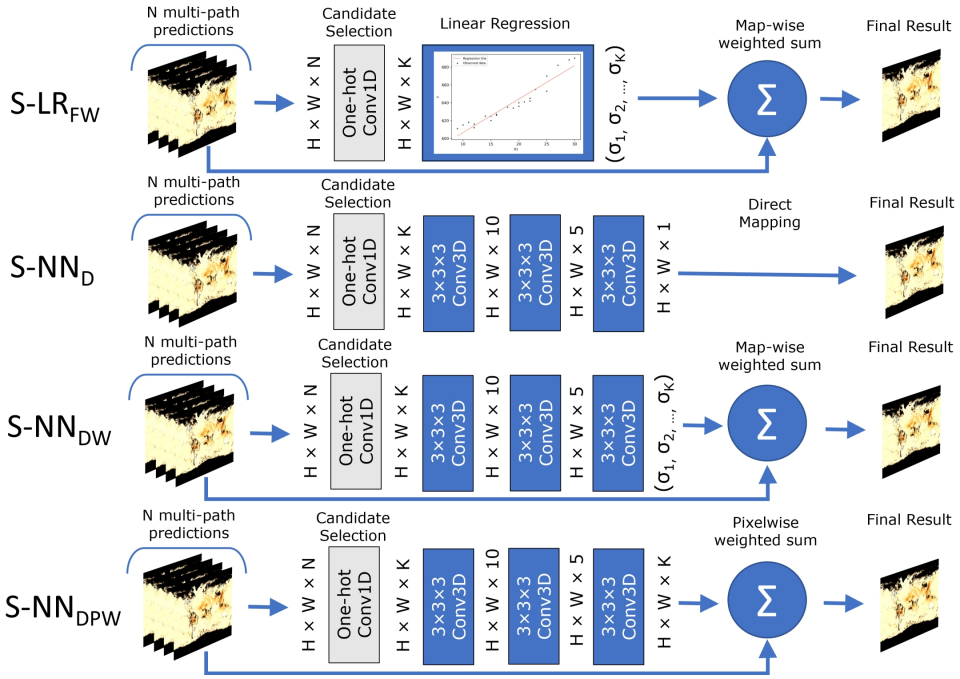


Figure 5.4: Ensemble Architectures. We introduce four types of ensembles, all with an initial learnable candidate selection model, which keeps only the relevant candidates before combining them.

5.2 Experimental analysis on the Dronescapes dataset

Dataset and Experimental Setup

We introduce Dronescapes, a new large-scale UAV video dataset featuring diverse rural, urban, and seaside scenes, as shown in Figure 5.5. It includes odometry, 3D information, and sparse manual annotations for semantic segmentation, making it a challenging real-world benchmark.

For our experiments, we set up an iterative learning procedure where we start with a small set of manually labeled frames. In subsequent iterations, we use our hyper-graph to generate pseudolabels for new, unlabeled video frames, and retrain the model on this expanded dataset. We focus on predicting three tasks: semantic segmentation, depth, and surface normals.

Figure 5.5: Sample frames from each of the 10 scenes from the Dronescapes dataset, showing a large variation in landscapes and class distributions.

Results and Findings

Our experiments on Dronescapes demonstrate several key advantages of our method. In Table 5.1 we show that our new, more complex **hyper-edges (AH, CH) consistently outperform simpler single-hop and two-hop edges**.

	Type	Train Unlabeled (iter 2)			Train Unlabeled (iter 3)		
		(1)	(2)	(3)	(1)	(2)	(3)
Edges	E: rgb	42.85	5.04	10.37	32.79	21.66	12.40
	E: hsv	41.70	4.69	10.54	33.51	19.90	12.48
	E: softedges	32.47	6.26	11.56	27.28	18.61	13.53
	E: softseg	30.71	5.97	11.14	24.68	22.70	12.76
	TH-E: sseg	-	6.25	11.39	-	19.00	12.93
	TH-E: depth	29.24	-	12.22	24.11	-	13.79
	TH-E: norm	30.56	6.17	-	26.35	21.15	-
Hyper-Edges	AH	41.80	5.33	10.37	33.63	23.96	12.24
	CH	44.63	4.93	10.32	36.92	20.36	12.23

Table 5.1: Evaluation of edges and Hyper-Edges for multiple tasks: 1 - semantic segmentation (sseg); 2 - depth estimation (depth); 3 - surface normals (norm).

In Table 5.2, we show that our learned parametric ensembles significantly improve performance over previous methods that use non-parametric averaging on the NEO dataset.

Method	IoU(↑)			
	Barsana	Comana	Norway	Mean
NGC [20] (Mean)	41.53	40.75	27.38	36.55
NGC (Mean) + HE	42.61	42.17	27.96	37.58
CShift [15] (Mean)	43.91	42.13	29.68	38.57
CShift (Mean) + HE	44.71	43.88	30.09	39.56
LR (Ours)	46.51	**45.59**	**30.17**	**40.76**
NN (v1) (Ours)	45.53	42.92	28.37	38.94
NN (v2) (Ours)	45.48	43.25	26.36	38.36
NN (v3) (Ours)	**48.21**	44.85	28.94	40.67

Table 5.2: Learned ensembles compared to existing methods.

In Table 5.3, we show that the iterative semi-supervised process leads to substantial gains in both accuracy and temporal consistency across all tasks on the Dronescapes dataset.

Type	Semantic		Depth		Normals	
	IoU (↑)	Cons. (↑)	L1 (↓)	Cons. (↑)	L1 (↓)	Cons. (↑)
rgb-sup.	25.04	88.85	-	-	-	-
rgb-iter1	32.79	94.04	21.66	5.89	12.40	98.32
rgb-iter2	37.26	95.72	17.34	7.06	11.93	98.87
rgb-iter3	**40.31**	**98.13**	**16.64**	**30.26**	**11.71**	**99.30**

Table 5.3: Iterative learning consistently improves both accuracy (IoU, L1) and temporal consistency (Cons.) for the main *rgb* → *task* edge on the test scenes.

Finally, we show that our model can take the output of a powerful pre-trained expert (Mask2Former) and further refine its predictions, improving both accuracy and consistency on novel, unseen scenes (Figure 5.6).

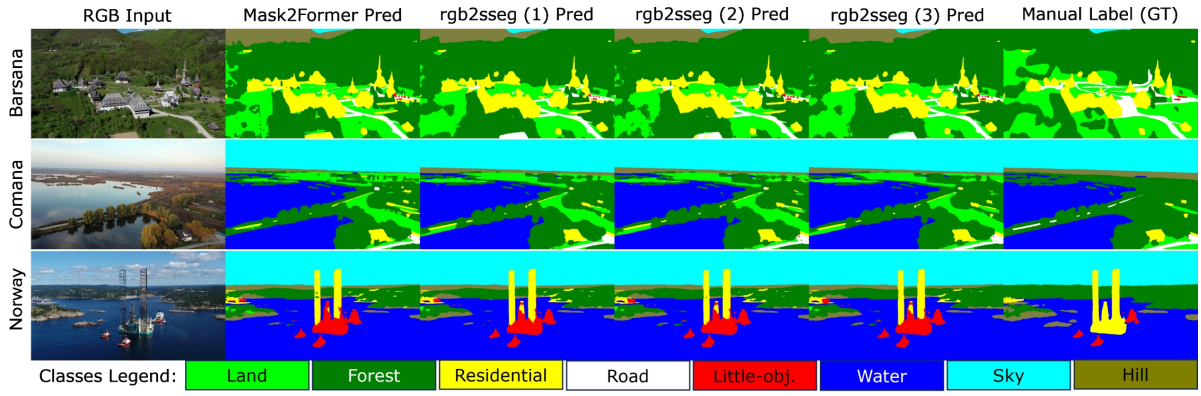


Figure 5.6: Qualitative results on test images. Our method improves upon the labels from a strong baseline (Mask2Former), especially on out-of-distribution scenes like Norway.

5.3 Experimental analysis on the NEO dataset

Dataset and Experimental Setup

To demonstrate the generality of our approach, we also apply it to the problem of Earth Observation using the NASA NEO dataset. This dataset contains 22 years of monthly satellite observations across various layers like Land Surface Temperature, Aerosol Optical Depth, and Leaf Area Index. The data is challenging due to its sparsity and the presence of missing measurements over long periods, making it an ideal case for semi-supervised learning. We use 12 layers as inputs to predict 7 different output layers.

We observe similar results on this dataset compared to the Dronescapes one, such as improved results using the learned ensembles as well as improved results using iterative semi-supervised learning, as presented in Figure 5.7. The iterative learning also leads to a significant increase in temporal consistency, meaning the predictions for a given location are more stable and reliable over time.

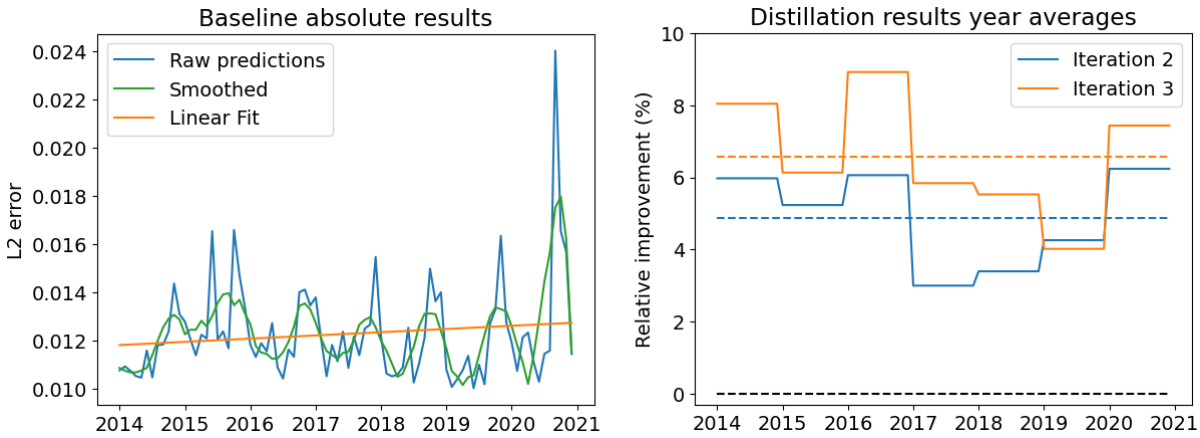


Figure 5.7: Prediction errors over seven years. **Left**: The baseline error shows a gradual increase over time, indicating a data distribution shift. **Right**: Our semi-supervised iterations significantly reduce this error, demonstrating the model's ability to adapt.

5.4 Conclusions

In this chapter, we introduced a novel multi-task, semi-supervised Hyper-Graph model. Its main strengths are the use of complex hyper-edges to capture higher-order relationships between tasks and the introduction of learnable ensembles that act as teachers to guide learning from unlabeled data.

We validated our approach on two very different and challenging real-world problems: aerial scene understanding with our new Dronescapes dataset, and long-term earth observation with the NASA NEO dataset. In both cases, our method showed significant improvements in accuracy, generalization to new data, and temporal consistency, even when starting with very limited labeled data. This demonstrates the power and flexibility of the hyper-graph framework for semi-supervised learning.

Chapter 6

Probabilistic Hyper-Graphs using Masked Autoencoders, ensembles and efficient distillation

This chapter summarizes the recent advances of my work on neural hyper-graphs, ensembles, and distillation. It builds upon the previous chapters by introducing a novel probabilistic method for defining hyper-graphs using random masking in a multi-modal, multi-task setting. The Dronescapes dataset, introduced previously, is extended with new UAV scenes and intermediate modalities derived from pre-trained experts.

The real world is inherently multi-modal. For an outdoor scene from the Dronescapes dataset, we can infer distinct views such as semantic segmentation, depth, or motion. Our system combines these views to produce a coherent, unified understanding. The early work of this thesis modeled these relationships using graphs (Chapter 4) and hyper-graphs (Chapter 5) with a fixed structure. We address this limitation with a neural network that learns the interdependencies between views from data. This approach models the distribution of all possible hyper-graph configurations, with each inference pass sampling a specific configuration.

We adapt the masked autoencoding (MAE) proxy task by defining fixed sets of input and output views and using task-specific losses (e.g., cross-entropy for classification, L2 for regression). This unifies the pre-training and fine-tuning steps into a single loop. Furthermore, it allows the construction of inference-time ensembles through random pathways. We also show that this knowledge can be distilled into very small models (under 1M parameters) with minimal performance loss. In this work, we improve performance by masking entire views rather than patches, thus creating probabilistic hyper-nodes at modality level, as in the previous chapter. We call our method Probabilistic Hyper-Graphs, or PHG-MAE. We also show that this knowledge can be distilled into very small models (under 1M parameters) with minimal performance loss.

To address the varying difficulty of predicting different tasks, we introduce derived intermediate modalities. These act as a bridge between low-level inputs (like RGB) and high-level outputs

(like segmentation), simulating a form of curriculum learning. This, combined with our probabilistic approach, allows the model to learn optimal input-output dependencies directly from data.

A key part of this research was the ability to rapidly iterate and add new views from pre-trained experts. To facilitate this research, we have developed an open-source data pipeline for automatic dataset generation from arbitrary videos by generating multiple representations using pre-trained experts[1]. While our work focuses on outdoor UAV scenes, the methodology is applicable to other domains like autonomous driving. This tool was used to extend the Dronescapes dataset with new UAV videos augmented with expert knowledge, which in turn yielded better performance on the downstream tasks.

In short, our main contributions are:

1. Probabilistic Hyper-Graphs using Masked Autoencoders (PHG-MAE): an extension of the standard MAE algorithm that enables Random Masking Ensembles at inference time and unifies previous Hyper-Graph methods under a single neural model.
2. Merging of pre-training and task-specific fine-tuning under a single training loop by defining inputs and outputs differently from the standard MAE loop. Furthermore, we only mask an entire view, not at patch level as previously done.
3. Inclusion of derived intermediate modalities from pre-trained experts on diverse datasets to leverage their knowledge and smooth the learning difficulty from low-level inputs to complex high-level tasks.
4. Efficient training and distillation showcasing competitive performance with small (4.4M) and very small (400k) CNN networks on the Dronescapes test set and unseen videos, enabling research on commodity hardware.
5. An open source data-pipeline that enables efficient extraction of new views from pre-trained experts on videos, simplifying the process of training multi-task vision models on large video datasets.

Datasets We extend the Dronescapes dataset, introduced in Chapter 5, with additional modalities and new videos. This increases the total annotated frames from 23K to 80K. We use our data pipeline to augment the dataset with new views from pre-trained experts, such as semantic segmentation, depth estimation, optical flow, and derived views like camera normals and binary segmentation maps. Moreover, our distillation dataset includes a total of 148K frames.

Multi-modal Multi-task learning (MTL) We employ a multi-task learning setup where input views are referred to as *modalities* and output views as *tasks*. Our experiments use RGB as the primary input modality and aim to predict three output tasks: semantic segmentation, depth estimation, and camera normals estimation.

Ensemble learning and inference time ensembles Ensemble learning improves prediction performance and consistency. In previous chapters, we created ensembles using different pathways in a neural graph. Here, we generate ensembles by leveraging the randomness in

[1] https://gitlab.com/video-representations-extractor/video-representations-extractor

our model. We perform multiple inference passes for the same input, each time with a different random mask applied to the input modalities. Unlike previous methods that mask patches, we mask entire views, which corresponds to sampling different hyper-edges. The resulting predictions are then aggregated through simple averaging.

Semi-supervised learning In this chapter, we also perform a single iteration of semi-supervised learning via distillation, though the primary focus is on the probabilistic hyper-graph model itself.

6.1 PHG-MAE: Probabilistic Hyper-Graphs using Masked Autoencoders model

The main limitation of the hyper-graphs defined in previous chapters was their fixed structure, which made them difficult to modify without extensive retraining. Motivated by this, we formulated the hyper-graph concept within a single neural network. As a theoretical contribution, we show that a multi-layer neural graph is equivalent to a single, larger neural network. By appropriately masking the inputs and internal weights of this single network, we can replicate the behavior of multiple smaller, independent networks that form the edges of a graph. We demonstrate that this holds for convolutional neural networks, as illustrated in the figures below, and the principle generalizes to deep networks and other layer types. This equivalence allows us to represent an entire hyper-graph within one model.

We modify the standard MAE algorithm to perform both pre-training and task-specific prediction simultaneously. We achieve this with two changes:

- Task-specific loss function. We use appropriate loss functions for each output task, such as cross-entropy for semantic segmentation and L2 loss for regression tasks like depth estimation.
- Define inputs and outputs. We define two disjoint sets of views. Input views (e.g., RGB) are easy to acquire and are **always seen** by the model. Output views (e.g., semantic segmentation) are hard to acquire and are **always masked**, forcing the model to predict them. This combines the auto-encoder paradigm with standard supervised learning.

Random Masking Ensembles

Instead of hand-crafting these masks, we let the model learn the interdependencies between views using the principles of Masked Autoencoders (MAE). We treat each full image view as a single token to be masked.

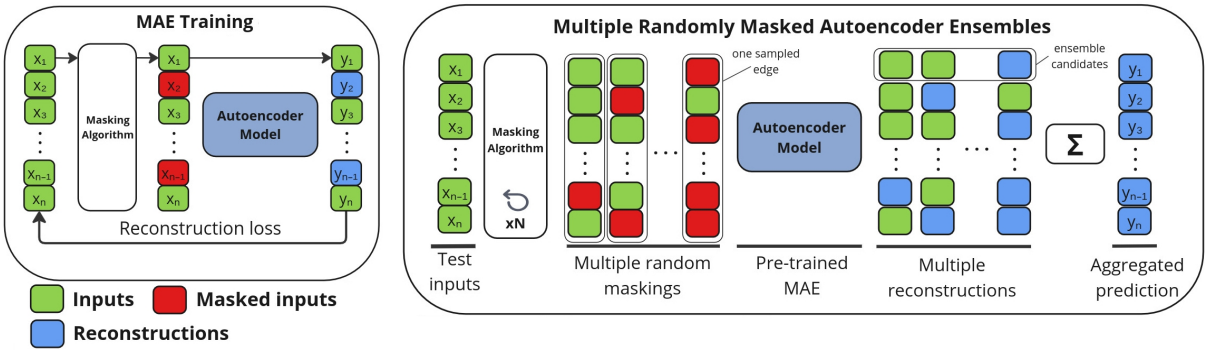


Figure 6.1: Left: Training a standard MAE model. Right: Test-time Random Masking Ensembles using the trained model.

As shown in Figure 6.1, during training, we randomly mask some views and task the network with reconstructing them from the visible ones. At test time, we can query the trained model multiple times for the same input, each time with a different random mask. The collected predictions are then aggregated to form an ensemble.

Proposition A forward pass through a Masked Autoencoder is equivalent to a forward pass of a single edge in a Hyper-Graph. Through random masking we are sampling from the distribution of all hyper-edges.

Intermediate modalities for ensemble diversity

If inputs are always seen and outputs are always masked, there is no randomness to generate ensembles at test time. To solve this, we introduce a set of intermediate modalities. These are derived from pre-trained experts using our data pipeline, as shown in Figure 6.2. These modalities, which include expert predictions for depth and segmentation as well as simpler derived views like binary maps for "vegetation" or "sky", are **sometimes masked and sometimes seen** during training. At inference, randomly masking these intermediate modalities allows us to generate a diverse set of predictions for ensembling. These views also serve as a bridge between low-level inputs and complex high-level tasks, aiding the learning process.

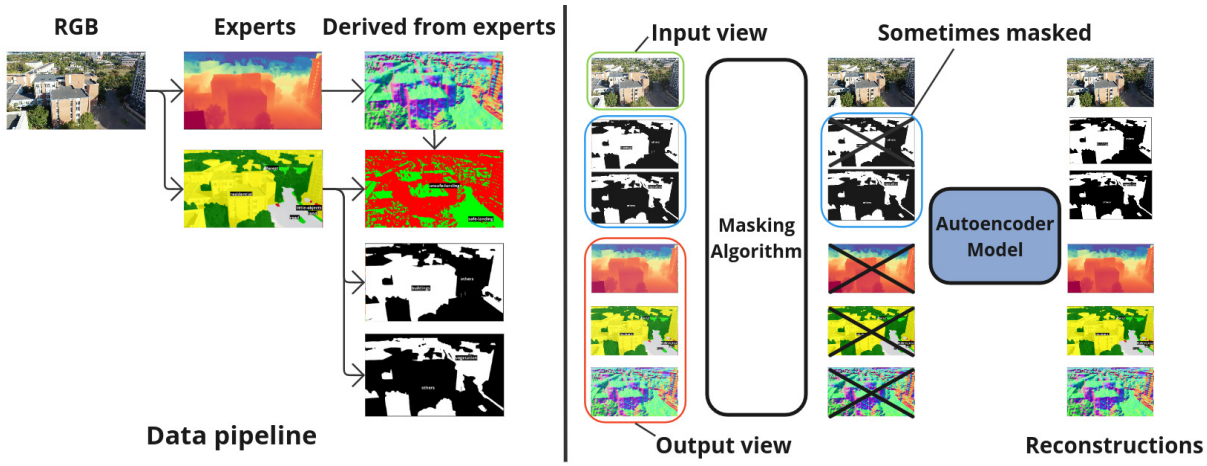


Figure 6.2: The Data-Pipeline and PHG-MAE model on real data. Left: Deriving modalities from pre-trained experts using RGB only. Right: Integration of modalities in the model.

6.2 Experimental analysis

Dataset description

All experiments are evaluated on the *Dronescapes* dataset benchmark. We extend the original dataset with 8 new UAV video scenes and numerous modalities extracted with our data pipeline. This results in the *Dronescapes-Ext* and *Dronescapes-*-M* variants, which significantly increase the training data from 23K to 80K frames. Table 6.1 summarizes the dataset variations. Our models are trained efficiently on consumer-grade hardware, with our largest model being trained in just under a week, while our largest distillation model in about two days.

Name	Data Points (GT labels)	I/O Views	UAV scenes	Description
Dronescapes-Train	12K (233)	5/3	7	Original train set
Dronescapes-Semisup	11K (207)	5/3	7	Original semi supervised set
Dronescapes-Test	5.6K* (116)	5/3	3	Original test set
Dronescapes-Pseudo	23K (233)	5/3	7	Pseudolabels from Dronescapes-Semisup
Dronescapes-Train-M	12K (233)	14/3	7	All new experts and intermediate modalities
Dronescapes-Ext	80K (440)	1/3	15	First two sets combined plus data-pipeline on new videos (no new modalities)
Dronescapes-Ext-M	80K (440)	14/3	15	All new experts and intermediate modalities on the extended dataset
Dronescapes-Ext2-Pseudo	148K (440)	1/1	23	Pseudolabels on Dronescapes-Ext plus 8 new UAV videos

Table 6.1: Dronescapes dataset variations and stats. Numbers in parentheses represent the semantic human annotated data.

Results on Multi Task Learning

We compare our PHG-MAE model against baselines from previous chapters on the three main tasks of the Dronescapes dataset. As shown in Table 6.2, our models, particularly when trained on the extended dataset with additional modalities, achieve state-of-the-art performance, significantly outperforming prior methods like NGC and SafeUAV-MTL. Qualitative results in Figure 6.3 confirm that our model generalizes well to unseen test scenes.

Model	Training Dataset	Parameters	Semantic ↑ Segmentation	Depth ↓ Estimation	Camera Normals Estimation ↓
PHG-MAE-MTL	Dronescapes-Ext	4.4M	52.04	18.84	12.67
PHG-MAE	Dronescapes-Ext-M	4.4M	49.09 $\pm$ 3.8	19.57 $\pm$ 2.2	13.68 $\pm$ 1.7
PHG-MAE	Dronescapes-Train-M	4.4M	42.84 $\pm$ 4.1	18.23 $\pm$ 1.5	12.54 $\pm$ 1.5
PHG-MAE-MTL	Dronescapes-Train	1.1M	39.23	19.31	13.18
PHG-MAE-MTL	Dronescapes-Train	4.4M	39.1	20.55	13.48
NGC-HE(mean) [22]	Dronescapes-Pseudo	32M	37.58	21.81	12.40
NGC(mean) [20]	Dronescapes-Pseudo	32M	36.55	20.08	12.97
SafeUAV-MTL [21]	Dronescapes-Train	1.1M	32.79	21.66	12.40

Table 6.2: Multi task learning comparison.

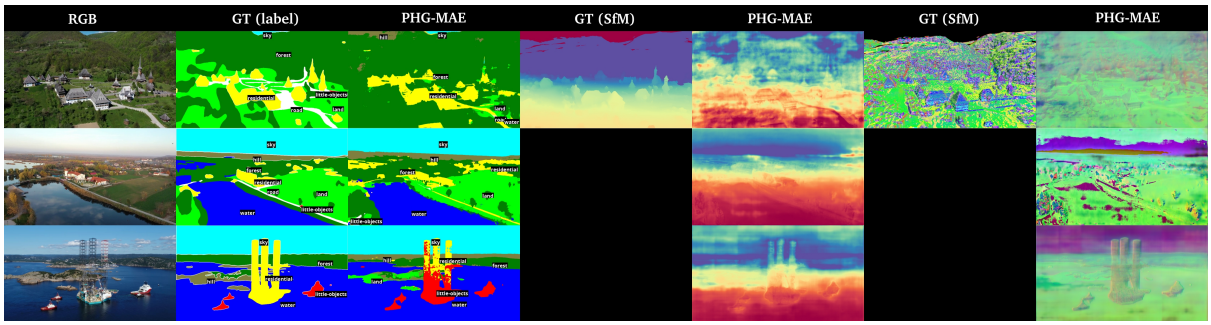


Figure 6.3: Multi Task Learning (MTL) qualitative results of our best model from Table 6.2. The Semantic Segmentation ground truth labels are human annotated, while for Depth and Camera Normals, it is based on a Structure from Motion reconstruction. Only one scene is available on the original Dronescapes test set for these two tasks.

Results on Semantic Segmentation

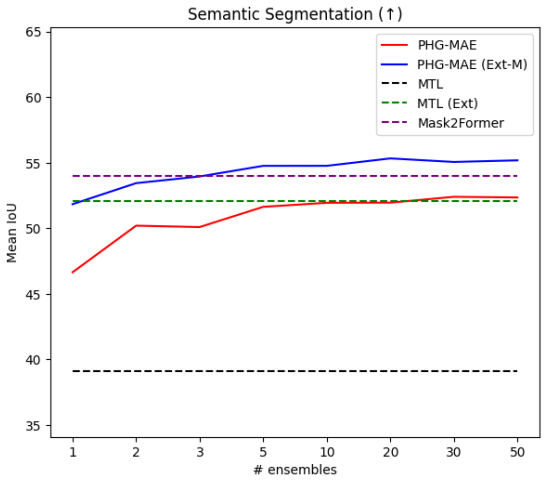
Focusing on semantic segmentation, which has reliable human-annotated ground truth, our best model trained on the extended dataset achieves a Mean IoU of 52.04. As shown in Table 6.3, this result outperforms previous works and is competitive with Mask2Former, a much larger transformer-based model.

Model	Training Dataset	Parameters	Mean IoU ↑
Mask2Former[17]	Mapillary[23]	216M	53.97
PHG-MAE-MTL	Dronescapes-Ext	4.4M	52.04
PHG-MAE	Dronescapes-Ext-M	4.4M	51.83±3.3
PHG-MAE	Dronescapes-Train-M	4.4M	46.64±5.1
NGC-HE-LR[22]	Dronescapes-Pseudo	32M	40.76
SafeUAV[21]	Dronescapes-Pseudo	1.1M	40.31
PHG-MAE-MTL	Dronescapes-Train	1.1M	39.23
PHG-MAE-MTL	Dronescapes-Train	4.4M	39.1
NGC-mean[20]	Dronescapes-Pseudo	32M	36.55
SafeUAV[21]	Dronescapes-Train	1.1M	32.79

Table 6.3: Semantic segmentation evaluation on Dronescapes test set.

Results on Ensemble Learning

Our Random Masking Ensembles provide a substantial boost in performance. As shown in Figure 6.4, aggregating predictions from multiple random masks at test time improves the semantic segmentation score of our best model from 51.83 to 55.32 Mean IoU. This ensembled result surpasses the performance of the 216M parameter Mask2Former expert. The ensembles also produce more stable and qualitatively better predictions on unseen scenes, as shown in Figure 6.5.



Model	Mean IoU ↑ (no ensembles)	Mean IoU ↑ (ensembles)
PHG-MAE (Ext-M)	51.83±3.3	$\mathbf{55.32 \pm 0.54}$
Mask2Former	**53.97**	n/a
MTL (Ext)	52.04	n/a
PHG-MAE	46.64±5.1	52.26±0.54
MTL	39.1	n/a

Figure 6.4: Multi Task Learning (MTL) results with ensembles against baseline. Left: Plot with various intermediate results for a handful of ensemble candidates. Right: Best single prediction (no ensemble) and the best ensembled prediction (50 candidates).

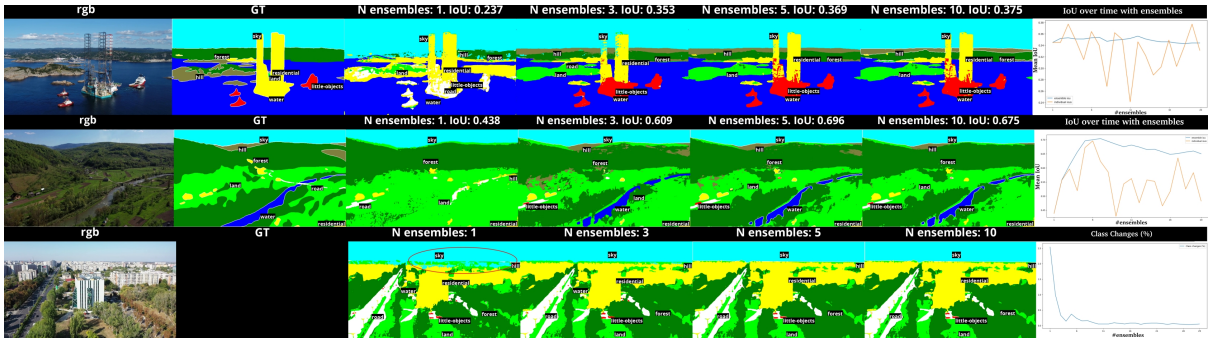


Figure 6.5: Random Masking Ensembles on unseen testing scenes.

Results on Distillation

To create a practical model for deployment, we distill the knowledge from our complex, multi-modal ensembled model into a simple, lightweight CNN that requires only RGB input.

As detailed in Table 6.4, this process is highly effective. A distilled model with only 430k parameters achieves a Mean IoU of 54.94, retaining nearly all the performance of its large teacher while being orders of magnitude faster and easier to deploy, as seen in Table 6.4.

Model	Parameters	# Input Modalities	Data Pipeline	Random Ensembles	Additional Data	Mean IoU ↑	Runtime (s) ↓
PHG-MAE	4.4M	12	✓	✓	✗	55.32±0.54	78.9
PHG-MAE-Distil	4.4M	1	✗	✗	✓	55.05	0.064
PHG-MAE-Distil	430k	1	✗	✗	✓	54.94	0.054
PHG-MAE-Distil	4.4M	1	✗	✗	✗	54.37	0.064
PHG-MAE-Distil	1.1M	1	✗	✗	✓	54.3	0.058
Mask2Former [7]	217M	1	✗	✗	✗	53.97	0.79
PHG-MAE-Distil	150k	1	✗	✗	✓	53.32	0.052
PHG-MAE-Distil	430k	1	✗	✗	✗	52.44	0.054
PHG-MAE-MTL	4.4M	1	✗	✗	✗	52.04	0.064
PHG-MAE	4.4M	12	✓	✗	✗	51.83±3.3	74.4

Table 6.4: Single Task Learning Distillation results on top of PHG-MAE ensembles.

Temporal consistency We also evaluate temporal consistency using a metric based on optical flow alignment between consecutive frames. Our distilled models are not only accurate but also produce highly consistent predictions over time, outperforming both the ensembled teacher and the large Mask2Former baseline. This makes them well-suited for real-world robotics applications where temporal stability is crucial.

6.3 Conclusions

We introduce a new test-time ensemble algorithm that works on any Masked Autoencoders model, which is a very popular pre-training mechanism today for large models. We tested our algorithm on small CNN-based models with 1.1M and 4.4M parameters on the Dronescapes dataset with models trained on commodity hardware. We show that the test-time ensembles outperform the classical Multi Task Learning prediction paradigm by producing higher quality and more consistent semantic segmentation maps even with the simple average ensembling method. This also enables further research, such as better aggregation methods, like using a secondary neural network for direct aggregation or searching and filtering the candidates.

Chapter 7

Conclusions, final thoughts and future work

This final chapter presents an overview of the thesis, reflecting on its core ideas and outlining interesting avenues for future research.

The work sits at the intersection of several fields, including *deep learning* with *neural networks*, *multi-task learning*, *prediction consensus via ensemble learning*, *model distillation*, *graphs*, and *semi-supervised learning*. We applied these concepts to robotics, specifically for *aerial scene understanding* with UAVs and Earth observation data.

The thesis chapters are designed to build on one another. In Chapter 3, we explored ensemble learning and distillation to improve depth estimation. In Chapters 4 and 5, we used graphs and neural networks to model the multi-modal world, introducing a new semi-supervised learning algorithm. Finally, in Chapter 6, we distilled the complex graph model into a single neural network using random masking, connecting our work with state-of-the-art pre-training methods.

A key takeaway from this research is the importance of sharing our work. By making ideas, code, experiments, and results openly available, we help others build upon our findings and accelerate scientific progress. This process of openly sharing and allowing others to *distill* previous work is crucial for the community.

Future Work

The research presented here opens up several promising directions for future exploration.

Scene understanding While this thesis focused on aerial scene understanding, the field is much broader. Future work could integrate modern 3D reconstruction techniques like NeRF [10] and 3D Gaussian Splatting [4]. Another interesting path is to combine data-driven methods with approaches that model physical laws, such as Physics Informed Neural Networks [8] or World Models [34], to achieve a more holistic understanding of a scene.

Neural networks We primarily used the SafeUAV [21] architecture due to its efficiency. An obvious next step is to explore the integration of more recent architectures, such as Transformers [32], which have become standard in many fields. Another direction is to use techniques like AutoML [29] to automatically discover network architectures optimized for a specific set of tasks.

Deep learning and reasoning Current deep learning models are often trained on large, redundant datasets. Future work could explore dataset selection techniques [1] to identify the most informative data points for training. Furthermore, there is a growing interest in moving beyond simple pattern recognition towards reasoning. This involves using methods like chain of thought [33] or program synthesis [2] to break down complex problems into smaller, solvable parts.

Multi-modal multi-task learning through ensembles and graphs This topic was central to the thesis. Our iterative process of learning graph edges, applying ensembles, and then distilling could be integrated into a single, end-to-end training loop. We could also expand our model to use more complex hyper-edge types (see Section 5.1) and incorporate these learned ensembles into the single-network approach developed in Chapter 6.

Unsupervised and semi-supervised learning This thesis was largely based on supervised learning. However, recent breakthroughs have shown the power of a two-stage approach: first, pre-training a model on large, unlabeled datasets using unsupervised algorithms, and then fine-tuning it on specific, labeled tasks. Adopting this paradigm would be a valuable extension of our work.

Integration and deployment in real-world systems A personally interesting direction is the integration of these deep learning models into real-world robotic systems. In this context, machine learning serves as a perception component within a larger system that must also *act* on its environment. This blend of traditional code and learned models is sometimes referred to as Software 2.0 [18], and it represents a critical step in bringing research into practical application.

These are just a few ideas for what could come next. The key is to pursue work that inspires curiosity and enjoyment. Thank you for reading.

Bibliography

[1] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

[2] Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. Hysynth: Context-free llm approximation for guiding program synthesis. *Advances in Neural Information Processing Systems*, 37:15612–15645, 2024.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[6] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

[7] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

[8] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3): 88, 2022.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[10] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

[11] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019.

[12] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11543–11552, 2020.

[13] Google. Google earth, 2018. URL https://www.google.com/earth/. Available at https://www.google.com/earth/, version 7.3.0.

[14] Carsten Griwodz, Simone Gasparini, Lilian Calvet, Pierre Gurdjos, Fabien Castan, Benoit Maujean, Gregoire De Lillo, and Yann Lanthony. Alicevision Meshroom: An open-source 3D reconstruction pipeline. In *Proceedings of the 12th ACM Multimedia Systems Conference - MMSys '21*. ACM Press, 2021. doi: 10.1145/3458305.3478443.

[15] Emanuela Haller, Elena Burceanu, and Marius Leordeanu. Self-supervised learning in multi-task graphs through iterative consensus shift. *arXiv preprint arXiv:2103.14417*, 2021.

[16] Emanuela Haller, Elena Burceanu, and Marius Leordeanu. Self-supervised learning in multi-task graphs through iterative consensus shift. *BMVC*, 2021.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[18] Andrej Karpathy. Software 2.0. https://web.archive.org/web/20250323195948/https://karpathy.medium.com/software-2-0-a64152b37c35, 2025. [Online; accessed 04-April-2025].

[19] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[20] Marius Leordeanu, Mihai Cristian Pîrvu, Dragos Costea, Alina E Marcu, Emil Slusanschi, and Rahul Sukthankar. Semi-supervised learning for multi-task scene understanding by neural graph consensus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1882–1892, 2021.

[21] Alina Marcu, Dragos Costea, Vlad Licaret, Mihai Pîrvu, Emil Slusanschi, and Marius Leordeanu. Safeuav: Learning to estimate depth and safe landing areas for uavs from synthetic data. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[22] Alina Marcu, Mihai Pirvu, Dragos Costea, Emanuela Haller, Emil Slusanschi, Ahmed Nabil Belbachir, Rahul Sukthankar, and Marius Leordeanu. Self-supervised hypergraphs for learning multiple world interpretations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 983–992, 2023.

[23] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.

[24] Yassine Ouali, Céline Hudelot, and Myriam Tami. Semi-supervised semantic segmentation with cross-consistency training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12674–12684, 2020.

[25] Mihai Pirvu, Victor Robu, Vlad Licaret, Dragos Costea, Alina Marcu, Emil Slusanschi, Rahul Sukthankar, and Marius Leordeanu. Depth distillation: unsupervised metric depth estimation for uavs by finding consensus between kinematics, optical flow and deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3215–3223, 2021.

[26] Mihai Pirvu, Alina Marcu, Maria Alexandra Dobrescu, Ahmed Nabil Belbachir, and Marius Leordeanu. Multi-task hypergraphs for semi-supervised learning using earth observations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3404–3414, 2023.

[27] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[29] Imrus Salehin, Md Shamiul Islam, Pritom Saha, SM Noman, Azra Tuni, Md Mehedi Hasan, and Md Abu Baten. Automl: A systematic review on automated machine learning with neural architecture search. *Journal of Information and Intelligence*, 2(1):52–81, 2024.

[30] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[31] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[33] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[34] Zheng Zhu, Xiaofeng Wang, Wangbo Zhao, Chen Min, Nianchen Deng, Min Dou, Yuqi Wang, Botian Shi, Kai Wang, Chi Zhang, et al. Is sora a world simulator? a comprehensive survey on general world models and beyond. *arXiv preprint arXiv:2405.03520*, 2024.