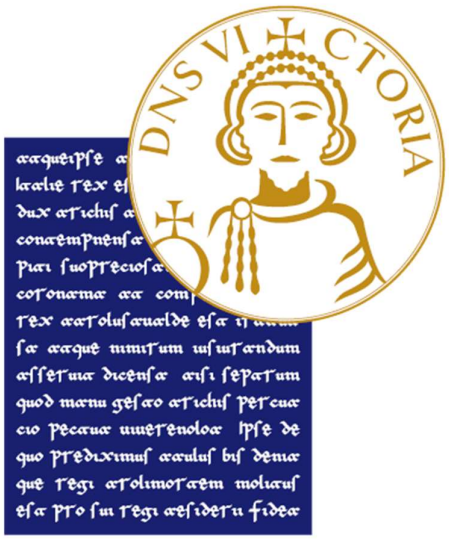
UNIVERSITÀ DEGLI STUDI DEL SANNIO



Dipartimento di Ingegneria

Corso di Laurea in Ingegneria Informatica

Pre-trained models reuse: a case-study in precision farming

Relatore

Prof. Damian Andrew Tamburri

Candidato

Loris Umberto Pennucci
Matr. 863003783

Correlatrice

Federica Pepe

Anno Accademico 2024-2025

Table of Contents

1. Introduction

Artificial Intelligence and Machine Learning have become a cornerstone of modern innovation, enabling solutions across diverse domains through the use of pre-trained models. These models, developed on large and often heterogeneous datasets, encapsulate knowledge that can be transferred and adapted to new tasks. To give a sense of scale, Hugging Face alone hosts over 2 million models [1], many of which trained on dozens to hundreds of datasets ranging from text corpora with billions of tokens to multimodal collections of satellite imagery, sensor data, and images. Foundation model suites such as IBM Granite, OpenAI's GPT family, Google's Gemini, and Meta's LLaMA report training on massive corpora that span language, vision, code, speech, and geospatial data. This breadth of training data illustrates why reusability is possible: the models capture generalizable patterns that can be adapted and reused in specialized contexts.

Model reusability refers to the ability to utilize such pre-trained models beyond their original scope, reducing the need for extensive retraining and accelerating the deployment of AI systems. Reusability is not only a matter of technical feasibility but also of efficiency, scalability, and sustainability, as it allows organizations to maximize the value of existing computational resources while minimizing costs and time.

Meanwhile, precision farming has emerged as an innovative approach to agriculture, integrating advanced technologies to optimize crop management, resource allocation, and environmental sustainability. By harnessing data from sensors, drones, satellites, and other digital tools, precision farming allows farmers to make informed decisions that improve yields, reduce waste, and mitigate ecological impact. AI can play a pivotal role in this domain, offering predictive insights, automated monitoring, and adaptive strategies tailored to specific agricultural contexts.

The intersection of these two concepts presents both opportunities and challenges: while utilizing pre-trained models could provide a strong and cheap option to process the gathered data, evaluating their reusability requires careful consideration of criteria and metrics such as domain relevance, data compatibility, performance robustness, and adaptability to local conditions.

This thesis aims to propose a method of evaluating the reusability of pre-trained AI models in the context of precision farming, and specifically the Farm-Tech project; while we are considering this specific field, we aim to develop a method that may generalize to other fields.

More formally, the research question we aim to answer in this work is:

- What criteria and metrics can be considered to evaluate the reusability of pre-trained models in precision farming, and how can these be applied within the Farm-Tech project?

To answer this, this thesis will aim to establish a method for assessing reusability and apply it to a list of selected pre-trained models, systematically analyzing their potential deployment Farm-Tech.

Evaluating the reusability of pre-trained models in precision farming (and more in general for any field outside of the original model's scope) is important to ensure that technological investments translate into practical solutions. Without a structured method, models applied inconsistently may lead to inefficiencies or misaligned outcomes when transferred to new contexts; by instead establishing clear criteria and metrics, developers can systematically assess whether a model's knowledge base, performance, and adaptability align with the unique demands of their area of interest, effectively providing a valuable resource in the considered context.

The work is organized as follows: chapter 2 describes the methodology used to collect the models considered in this work, presenting a high-level analysis of the results obtained from the collection process in chapter 3; chapter 4 presents a brief overview of each model, describing its underlying architecture and data used during the training process, as well as pointing its relevance to FARM-TECH; chapter 5 presents our proposed classification of reusability for a model, applying it to the models seen in this work.

1.1 Research Context

This research is conducted in the context of the Farm-Tech project. The project reflects an initiative, developed in conjunction by Smart Shaped and Trace Technologies, aimed at developing an innovative smart platform to support sustainable agricultural production in Mediterranean environments. Its core mission is to provide a decision support system able to integrate different data sources (such as IoT sensors, satellite and drone imagery) into predictive models capable of guiding farmers and stakeholders toward more efficient, resilient, and environmentally sustainable practices.

At its core, Farm-Tech leverages GeoNode [2] as the baseline framework, enhanced with advanced data pipelines, machine learning components, and visualization tools; Farm-Tech is conceived to operate as both a technical solution and a collaborative ecosystem, enabling knowledge sharing and integration across several entities, such as research institutions, companies, and end users.

The architecture of Farm-Tech is modular, designed to handle diverse agricultural datasets. Its main modules include:

- **Data Storage and Management**, handled by MinIO [3] for large geospatial files, such as drones and satellite imagery.
- **Data Visualization**, utilizing GeoServer [4] to transform raw geospatial data into interactive map layers and a front-end, built in ReactJS, to provide an intuitive presentation to end users.
- **Big Data Processing** through the use of Apache Spark [5], allowing distributed processing of the gathered data.
- **Machine Learning and Predictive Analytics**, performed by the chaM3leon framework [6], a framework developed by Smart Shaped designed to support machine learning applications. Being a modular framework itself, more information can be found on its GitHub repository [6].
- **Back-end Services**, implemented through Django [7] to coordinate storage, analytics, and visualization layers, supported by a PostgreSQL database [8]. In this module security is also implemented through the use of KeyCloak [9].

This modular design makes Farm-Tech a valid candidate for evaluating the reuse of pre-trained models. The machine learning component, in particular, provides a flexible environment for integrating external models.

2. Research Methodology

To provide a structured foundation for our methodology, we align our approach with the CRISP-DM framework, a widely recognized model for guiding data-driven projects. CRISP-DM is composed by six iterative phases, each of which can be mapped to the structure of this thesis; in Figure 1, we present a graphical representation of the approach used. While CRISP-DM is traditionally iterative, in Figure 1 we resent it sequentially; the cycle can be repeated from the Data Understanding to the Deployment phase for each new model (or set of models) under consideration.

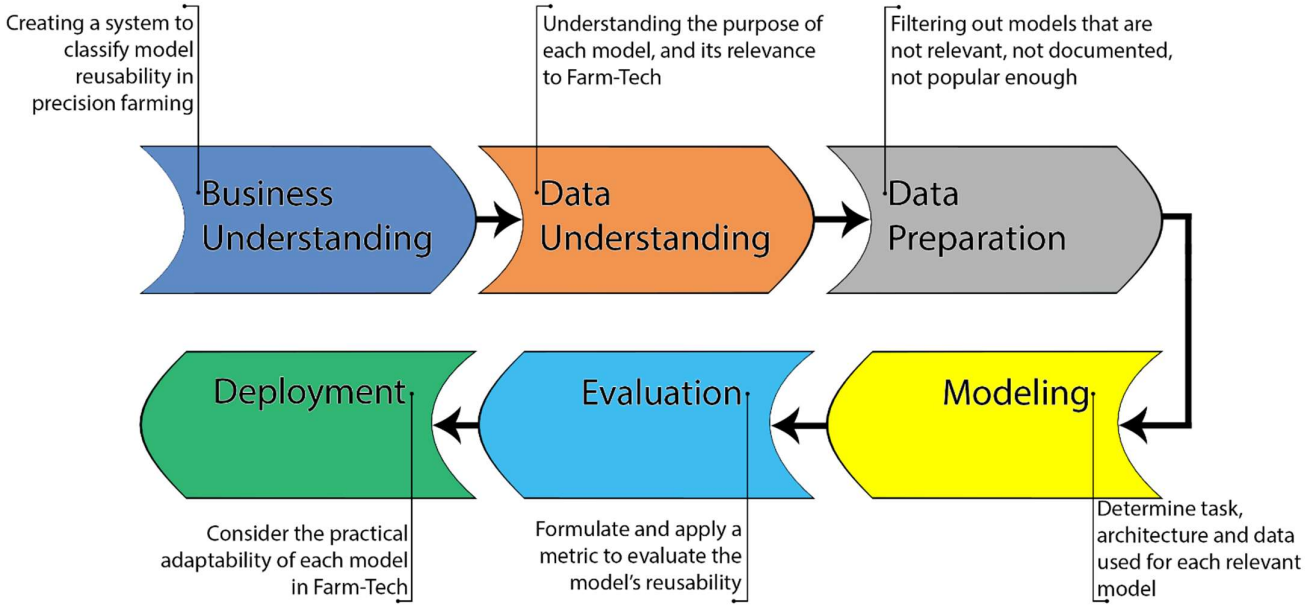


Figure 1: CRISP-DM approach used in this work.

Below is a breakdown of each phase:

- **Business Understanding**: our goal is to assess the reusability of pre-trained AI models for precision farming and produce a practical evaluation method applicable to Farm-Tech; to this purpose, we aim to propose a method of classifying the reusability of models and apply it to a list of models.
- **Data Understanding**: starting from a list of models to evaluate, we can determine their purpose and relevance to Farm-Tech based on the information provided in their model cards and/or additional documentation provided. The method used to compile our list is explained later in this chapter.
- **Data Preparation**: after retrieving each model's metadata, we will perform a manual review to remove false positives and undocumented entries, practically done through a dual independent human review to reduce subjectivity.
- **Modeling**: based on the understanding of the models gathered from the previous phases, we can group the relevant models by task and analyze in-depth their architecture and data used during training to better interpret their relevance to Farm-Tech.

- **Evaluation**: combining all the information gathered for the relevant models, we can formulate a metric to evaluate the reusability of each model in regards to precision farming and Farm-Tech in the specific.
- **Deployment**: considered in the final reflections on how these models could be practically adopted within the Farm-Tech platform.

The phases of Data Understanding and Preparation are discussed in this and the following chapter; the Modeling phase is seen for each model throughout chapter 4; chapter 5 and 6 cover, respectively, the Evaluation and Deployment phases.

The models were gathered from Hugging Face, one of the most widely used open-source platforms for sharing and reusing machine learning models. Similarly to GitHub's readme, each model presents a model card, which typically includes information regarding the model, making the hub a valuable resource for evaluating both technical relevance and community adoption. Since these model pages are written by the model's authors, the quality and precision of the information reported can vary widely. In Figure 2, the preview of a well-documented model is presented.

Using Hugging Face's endpoint API [10], we collected a list of pre-trained models, retaining for each both its ID and download count as a way to document their popularity; the models were selected if either their repository name or readme contained any of a predefined list of terms, which can be found in the appendix of this paper; additionally, only models with more than 100 downloads were considered.

This last requirement was put in place for several reasons: 1) given the large amounts of models that have less than 100 downloads, we decided to exclude them in order to not expand too much the scope of this work; 2) a higher number of downloads suggests that the model has attracted interest from the community, making it more relevant for empirical analysis; 3) the number of downloads also makes it more likely for these models to have been integrated into real-world projects, reducing the risk of including toy or experimental projects; 4) is assumed that, in case a model contains any kind of malicious software (as evidenced by [11]), a large amount of downloads would have exposed such a security risk.

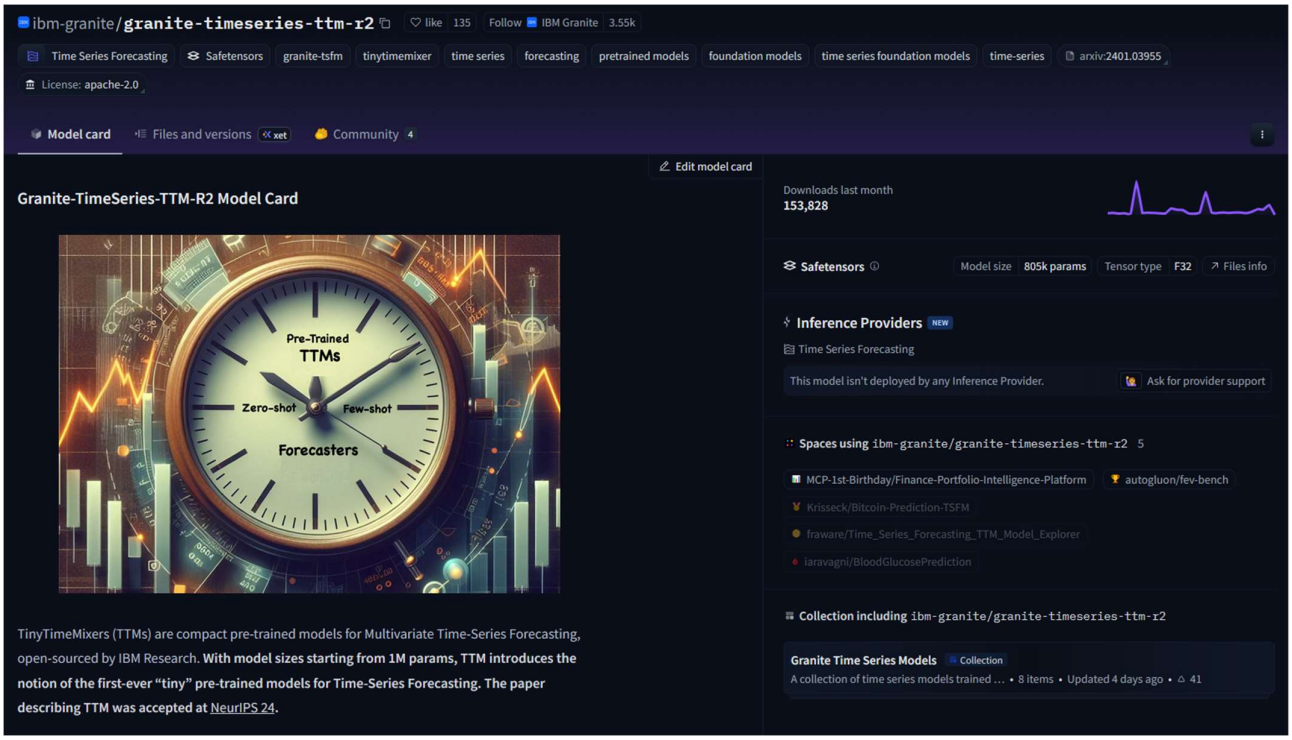


Figure 2: example of a model card; being developed by a major corporation, this model is well-documented, including license, intended task and, lower in the card, both the training datasets and the paper in which the model was introduced.

It is important to note that, given the selection criteria, some of the models retrieved may be false positives, that is they may contain some of the searched terms but not be relevant to precision farming: to solve this issue, we manually sorted through the list, marking false positives as "non relevant"; a model is deemed as not relevant based on its purpose specified in its readme/documentation provided, as well as the data (if shared) utilized during the training process; models with an empty readme page, or with no kind of documentation on the model itself, were also discarded, including models whose readme was non-informative, such as containing a generic LLM-generated pages.

In order to minimize subjectivity during the manual review, two separate researchers went through the list independently marking models as non-relevant or not documented; for all discrepancies found at the end of this process, they discussed together their reasoning to find the most appropriate classification.

3. Search Results

A total of 100 models were collected; the full list is presented into an online spreadsheet for ease of readability [12]. Out of these models, 85 were discarded, of which 60 because deemed not relevant as, during the manual revision process, we found their purpose didn't align with precision farming nor Farm-Tech specifically; the remaining 25 models were discarded because they provided either no or non-sufficient documentation: on the spreadsheet, the former are marked as "discarded (not relevant)" and the latter as "discarded (no doc)". For the remaining 15 models, marked as "relevant", we analyzed their readmes and any additional information provided to determine what their intended task is; below is a breakdown of all the tasks we found.

- **Time series Analysis**: models trained to perform forecasting, classification and anomaly detection on multivariate time series, a sequence of observations collected over time that records multiple variables simultaneously, capturing how they evolve together. These models can be useful in smart farming as they're able to analyze diverse sensor data to predict resource needs and detect irregularities in crop or soil conditions, enabling targeted interventions that maximize yield and minimize waste.
- **Image Classification**: models trained to classify an image, useful for determining whether some crops are infected with some kind of disease.
- **Drone Image Processing**: models trained to analyze drone-captured images, providing either a natural-text description of the image contents or estimate biomass present in the imagery.
- **Chatbot Assistant**: a chat-assistant capable of answering questions related to farming, such as crop planting, pest control, and irrigation. While not fundamental for precision farming, it can be a simple and useful feature for end users.

Figure 3 presents a graph showing the number of relevant models per task.

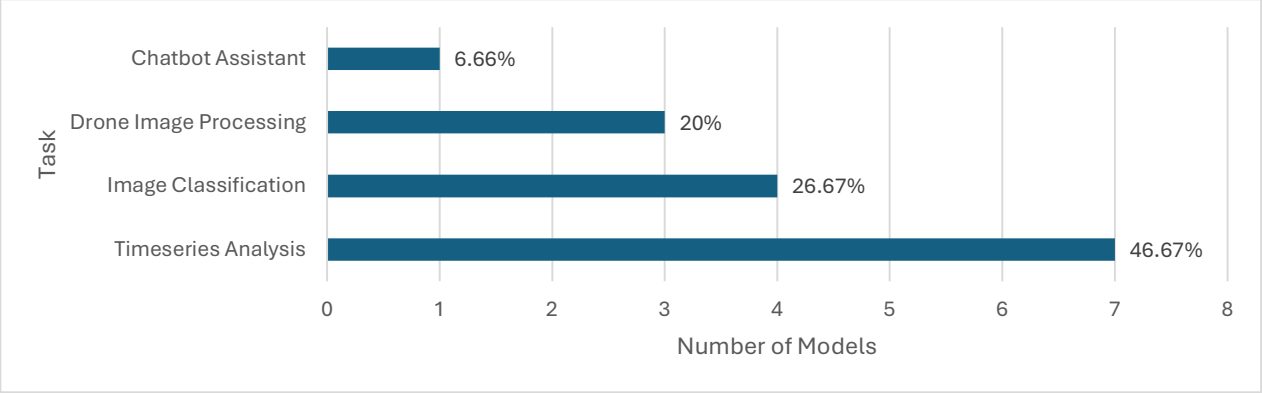


Figure 3: number of relevant models per task, reporting also the respective percentages of relevant models

Almost half of the models were trained to perform time series analysis. Table 1 reports all of the models deemed as relevant; the "Dataset" column represents if the training dataset was shared somewhere in the model page/documentation, with "provided" meaning that a direct link was provided, "mentioned" that either the dataset's name or its content was briefly described; "Bias" represents if any kind of bias was discussed, "mentioned" indicating that it was at least pointed out; "License", represents the license under which the model was released, in the following subchapter we analyze this point more in detail.

Model	Download Count	Dataset	Bias	License	Task
TinyTimeMixers v2 [13]	361.669	Provided	No	Apache 2.0	Time series Analysis
TinyTimeMixers v1 [14]	217.922	Provided	No	Apache 2.0	Time series Analysis
TSPulse [15]	55.288	Provided	No	Apache 2.0	Time series Analysis
PatchTST [16]	7.464	Provided	No	Apache 2.0	Time series Analysis
Disease Detection [17]	3.689	Mentioned	Mentioned	MIT*	Image Classification
PatchTSMixer [18]	730	Provided	No	Apache 2.0	Time series Analysis
FlowState [19]	689	Provided	No	Apache 2.0	Time series Analysis
AdaptLLM Remote Sensing (Qwen) [20]	495	Provided	No	None*	Drone Image Processing
Fruits and Vegetables Detector [21]	426	Provided	No	Apache 2.0	Image Classification
Leaf Detection and Classification [22]	375	Mentioned	Mentioned	None*	Image Classification
CropSeekLLM [23]	277	Provided	Mentioned	DARJYO License v1.0*	Chatbot Assistant
Classification of Leaf Diseases [24]	247	Provided	No	None*	Image Classification
AdaptLLM Remote Sensing (LLama) [25]	186	Provided	No	None*	Drone Image Processing
Geospatial Biomass [26]	135	Provided	No	Apache 2.0	Drone Image Processing
Timeseries Anomaly Detection [27]	103	Provided	No	None	Time series Analysis

*Table 1: list of all the models deemed relevant, reporting their name, download count, dataset, bias, license and task. Licenses marked with a * will be described in detail later*

The underlined models are all part of IBM Granite, a family of foundation models developed by IBM; being foundation models, they were trained on a large corpus of data not strictly relevant to farming, as such they will most likely require a process of fine-tuning before being used in a real-world environment.

Additionally, it should be pointed out that the number of downloads of a model should be seen strictly as an indicator of its popularity, as the number of downloads does not determine the reusability of a model.

3.1 Licenses and Bias

An important factor affecting a model's reusability is the license under which it is released, as it will determine whether developers can integrate the model into commercial products; without a clear license, using the model could expose individuals or organizations to legal risks.

Most of the relevant models (~53.34%) are released under the Apache 2.0 license, allowing these models to be reused in commercial products; the remaining models present either no licenses or contradicting information on their license (marked with an asterisk in the above table), they are listed below:

- Disease Detection [17]: this model is a finetuned version of Google's Vision Transformer (ViT) [28], released under license Apache 2.0, while the model itself presents a MIT license; being both permissive licenses, this model can be used in commercial environments.
- AdaptLLM Remote Sensing [20] [25]: while on their respective pages they report no license, both of them link to the GitHub repository hosting the code used to train the models, which presents the Beijing Institute for General Artificial Intelligence BIGAI License Agreement, stating "The code and/or data is strictly for non-commercial academic research only. · Any commercial use of the code or data requires prior contact and negotiation of licensing fees with the original authors or [BIGAI]". Before using these models, it may be wise to contact the original authors to ask for more information.
- Leaf Detection and Classification [22]: this model was fine-tuned from YOLOv8 [29], a real-time computer vision model developed by Ultralytics that performs object detection, classification, and segmentation; while the model's page doesn't report any license, Ultralytics has released YOLOv8 under the AGPL-3.0 license.
- CropSeekLLM [23]: this model was fine-tuned from DeepSeek-r1, released under MIT, while the model lists "DARJYO 1.0" as a license; on the company's website [30], it states that the license bars the use of this model for commercial purposes.
- Classification of Leaf Diseases [24]: this model lists no license, but it is a fine-tuned version of LLaVA 1.5 [31], released under the llama2 license; llama2 allows commercial uses but requires a license from META if the amount of monthly active users of the product is more than 700 million.

In conclusion, only CropSeekLLM [23] outright prohibits its use in commercial settings, [20], [25] and [22] require the purchase of a license, with [24] requiring one only if we experience a large number of users.

In terms of bias, most of the models (80%) do not discuss the argument; the few that do ([17], [22] and [23]) only provide some brief information on it, such as [17] stating that "the model may exhibit bias toward the crops and diseases present in the training dataset, leading to lower performance on unrepresented diseases or crop varieties". Since not much information is shared on any observed bias in all of the models, testing may be necessary to ensure their correct functioning in new environments.

4. Relevant Models

This chapter will analyze all the models deemed relevant. Out of 100 models originally retrieved, 15 have been deemed as relevant: 6 of these models are general-purpose, time-series forecasting models which, while not trained specifically on relevant data, can be fine-tuned to better fit the intended purposes. All of the general-purpose models are part of the IBM Granite suite [32], while the remaining 9 models were trained to perform image analysis.

4.1 IBM Granite - General Overview

IBM Granite is a family of enterprise-grade AI foundation models developed by IBM, designed to deliver high performance, safety, and efficiency across a wide range of business tasks. These multimodal models support language, vision, speech, and time-series forecasting. Within the Granite suite, these last time-series-related models can be of particular importance to Farm-Tech, like we've mentioned earlier in tasks such as monitoring a set of IoT sensors, predicting crop yield or detecting malfunctioning sensors.

4.1.1 Granite TinyTimeMixers

Granite TinyTimeMixers (TTM) [13] [14] are lightweight, efficient time-series forecasting models within the IBM Granite suite. They are designed to handle large-scale, multivariate time-series data, making them suitable for applications such as sensor data analysis and predictive maintenance in smart farming.

TTM is based on the lightweight TSMixer architecture [33], which allows a rapid training and fine-tuning for TTM, thanks to the replacement of heavier self-attention blocks in Transformers with MLPMixer blocks. TSMixer will not be analyzed any further in this work, please refer to the original paper for more information. TTM is pre-trained using multiple public datasets, totaling in around a billion of samples; all the datasets used can be found on the HuggingFace page.

TSMixer is further enhanced in the TTM training process by adding the following features.

- **Adaptive patching (AP)**: given the different sampling rates and context lengths present in the various datasets, patching (that is, the process of grouping consecutive time steps into tokens) cannot be performed at a specific patch length; as such, the patching length and number patches changes depending on the backbones layer.
- **Diverse resolution sampling (DRS)**: high-resolution datasets (that is, datasets where sampling is performed very frequently) will account for a larger fraction of the total samples compared to lower-resolution ones, which can lead to a bias towards finer resolution data; the solution is to resample these datasets at lower resolutions, for example, by averaging k samples into a single one, or outright retaining only the k-th sample; the following example is presented in the paper: "from a 4-second resolution dataset, we derive multiple datasets at minutely (k=15) and hourly resolutions (k=900) [...] increasing the number of datasets for each resolution which greatly improves the model performance".
- **Resolution prefix tuning (RPT)**: the purpose of this technique is to explicitly embed resolution information in the first patch, facilitating resolution-conditioned modeling while training on diverse datasets. * multi-level modeling strategy: TTMs are first pre-trained in a channel-independent way, and then fine-tuned with channel mixing to incorporate correlations across targets and exogenous channels in the target domain.

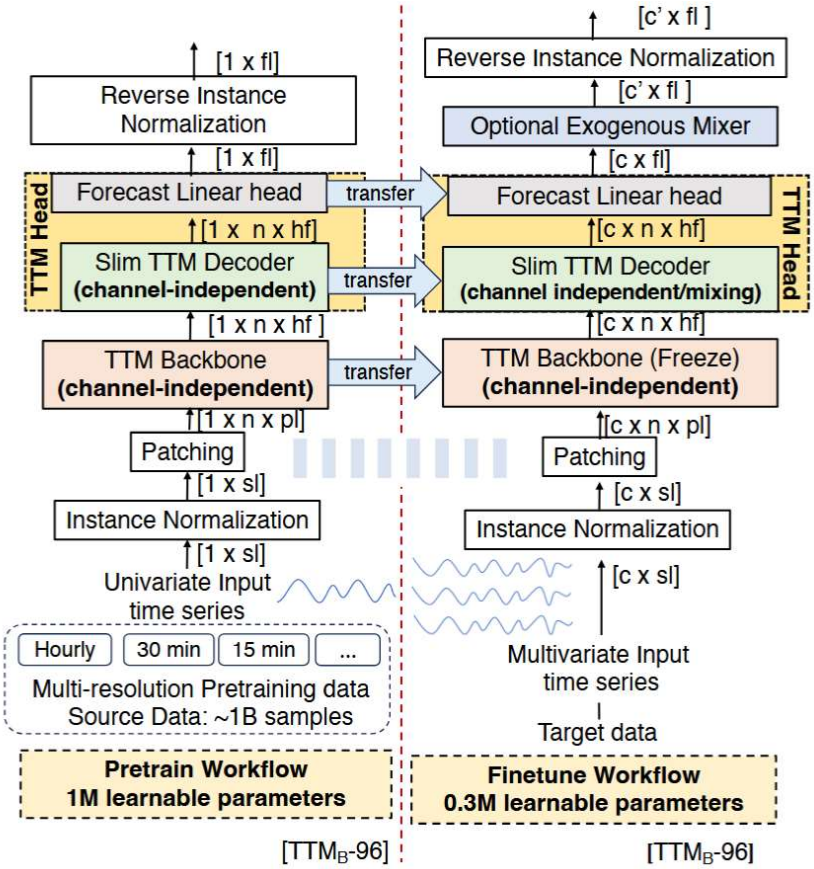


Figure 4: TTM's architecture overview

As can be seen in Figure 4, TTM operates in two stages, pre-training and fine-tuning: both workflows share the normalization and patching blocks; each time series is normalized per instance, with the reverse process performed at the end of the workflows before computing loss; the data is then patched (this process is different from AP) into n non-overlapping windows of length pl, which are then fed to the TTM Backbone. Following is an overview of the TTM backbone.

While the backbone will not be seen in detail in this work, this is where the TSMixer blocks are used, with AP interfacing each block with the next. At the start of the backbone RPT is performed, which explicitly learns and incorporates a new patch embedding as a learnable prefix into the input data based on the input resolution. For more information on the backbone's structure, please refer to the original paper [34].

In the pretrain workflow, the backbone utilizes the processed data to perform multi-resolution training on a single channel of the timeseries; the backbone's output will then be used by the decoder and forecast head to produce the forecast. The functioning of the Decoder and Forecast head will not be discussed in this work, please refer to the original paper for more information [34].

In the fine-tuning workflow, the pre-trained model will be working with data from the target domain, which has no overlap with the pre-training datasets. Forecasting can be performed in three ways:

- zero-shot: the pre-trained model is directly used to perform forecasting.
- few-shot: a tiny portion (5-10%) of the training part of the target dataset is used to update the pre-trained weights in the TTM Head.
- full-shot: the entire training part is used to update the pre-trained weights.

While the backbone will still perform channel-independent computing, the following decoder is able to be fine-tuned utilizing channel mixing or channel independence for multivariate or univariate target data, respectively. Again, for more information on the exact functioning of both, please refer to the original paper [34].

Granite TTMs are available in two main revisions, named as r1 and r2. While architecturally identical, TTM r2 models have been pre-trained on larger datasets (~700M samples) than r1 models (~250M); IBM reports that with a larger training set, performances have increased by over 15% on average, although they still advise to experiment with both r1 and r2 models to pick the best-suited for the intended application. Additionally, both r1 and r2 models come in different sizes of context and prediction length: on the HuggingFace repo, each model can be found on separate branches with the naming convention *cl-pl-rn*, where *cl* is the context length, *pl* the prediction length and *rn* is either r1 or r2.

While general-purpose, Granite TinyTimeMixers provide a valid solution for time-series forecasting in smart farming, and the vast documentation and guides provided by IBM would greatly simplify and speed up the process of introducing these models in a live application.

4.1.2 Granite TSPulse

Also part of the IBM Granite suite are the TSPulse models [15], ultra-compact pre-trained models developed for tasks such as classification, anomaly detection, imputation, and similarity search in multivariate time-series. Their main strength comes from their very limited size, totaling at 1 million parameters, compared to other multivariate time-series analysis models such as Google's TimesFM (200M parameters), Amazon's Chronos models (20M for the smallest) and even Lag-LLaMA (2.49M), allowing these models to perform GPU-free inference.

Like TTMs, TSPulse is also built on top of the TSMixer architecture, while also introducing several enhancements:

- **Dual-space masked reconstruction strategy**: masked inputs are simultaneously reconstructed in both time and frequency domains, leveraging the intuition that certain patterns are easier to detect in the time domain while others are more salient in the frequency domain. By learning to mix, attend and reconstruct across both spaces, TSPulse is able to build richer and robust representations.
- **Dual-embedding disentanglement**: TSPulse generates two types of embeddings during pre-training, that is detailed embeddings for fine-grained analysis, and high-level semantic embeddings for broader task understanding. By capturing both levels of information during pre-training, TSPulse enables faster and more robust generalization across tasks.
- **TSLens**: a neural component which enables effective attention during fine-tuning by learning to focus on the most informative regions within and across the dual embeddings, allowing the model to prioritize task and data-relevant signals.
- **Hybrid masking pre-train strategy**: by combining point-wise and block-wise masking with varying ratios per sample, the model is able to reconstruct irregular masking structures, mimicking realistic scenarios.

The model's architecture is quite complex, as such it will be considered at a high level; for more information, please refer to the original paper [35].

Below is a sequence of all the steps performed by the model:

1. TSPulse masks the input sequence, that is it hides portions of the input which the model will then have to reconstruct. Masking can be done using two strategies: **block** and **hybrid masking**. In block masking, entire input patches are randomly replaced with learnable tokens, while hybrid masking is able to mask individual tokens inside of a patch, while still capable of masking full patches. The masked timeseries is then normalized, but do note that the mean and variance are computed from the unmasked values; the normalized and masked values are fed directly into Torch's Fast Fourier Transform (FFT) to produce the frequency-domain representation of the timeseries, ensuring that data is consistently hidden both in time and frequency domains.
2. Both time and frequency representation are patched into N non-overlapping patches of length pl; these are then projected into an embedding space and then concatenated, alongside R learnable register tokens (as suggested by [36]) shared across all channels, to be used as input for the TSPulse Backbone. Note that the patching length pl used in this process is the same as the patching used during the masking stage.
3. The data processed earlier is fed to the Backbone which, as said earlier, is itself based upon the TSMixer Backbone; its goal is to transform the input into semantically rich, task-robust

representations. Since the input already integrates time and frequency information, TSMixer is able to mix these views, learning dual-space representations that capture temporal and spectral correlation. For more information, refer to the original paper [35], as well as TSMixer's paper [33].

4. The backbone output is passed through a lightweight mini-decoder, described as a mirror of the backbone but 10-20% of its size; its purpose is to refine and adapt the learned representations during fine-tuning.

5. The decoder's output is disentangled in two separate embedding types: (1) long embeddings, corresponding to the time and frequency domains, and are used for detailed reconstruction; (2) short embeddings, which correspond to the remaining register embeddings, used for semantic reconstruction tasks such as short-horizon forecasting and global frequency signature prediction.

6. Long embeddings are used by the **reconstruction heads**: the patch embeddings related to the time domain are passed through a linear layer called the Time Head, and are inverse-normalized to obtain the reconstruction of the timeseries; the frequency-domain patches, similarly, are projected through the FTT Head to reconstruct the input frequency spectrum, although, other than reconstructing the input frequency-domain timeseries, the patches are also passed through Torch's inverse Fourier transform (torch.ifft) to reconstruct an alternative time-domain timeseries.
   Short Embeddings are instead used by the **semantic heads**: one head outputs a predicted softmax distribution over the log-magnitude frequency spectrum to capture global spectral semantics; a second head predicts a short-horizon forecast to model high-level temporal dynamics.

Three variants of pre-trained TSPulse models are released, each specialized in a different task: **tspulse-hybrid-allhead-512-p8-r1**, recommended for anomaly detection; **tspulse-hybrid-dualhead-512-p8-r1**, recommended for imputation and search; **tspulse-block-dualhead-512-p16-r1**, recommended for classification.

The terms "hybrid" and "block" represent the type of masking used during pre-training: "allhead" and "dualhead" specify which heads were enabled during pre-training, with "dualhead" meaning that only time and prob heads were enabled (prob head, while not explicitly stated, most likely refers to the first semantic head as the paper names its output as Y^{f}_{prob}); 512 is the base context length of each model; the number following "p" represents the timeseries patches' length during training.

The released models were trained on a collection of datasets which span from weather tracking to electricity demands; the full list of datasets used can be found both on the model's HuggingFace page [15] and the paper's appendix [35].

In conclusion, thanks to their compact architecture and strong benchmark performance, TSPulse models represent a valid choice for anomaly detection tasks in smart farming. Their small parameter count enables efficient, resource-light deployment, while maintaining a high degree of efficiency compared to state-of-the-art models.

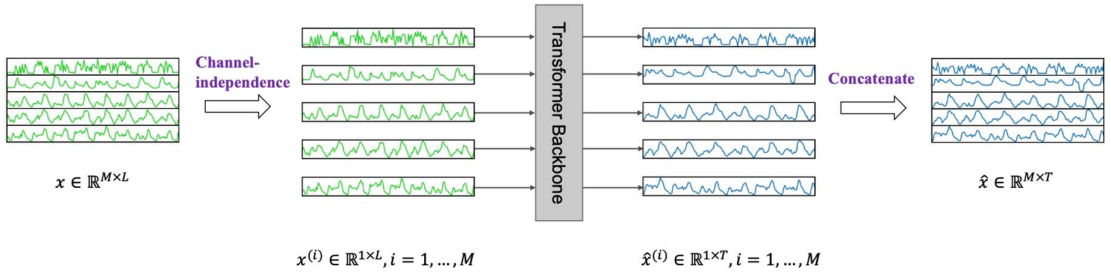
4.1.3 Granite PatchTST

Like TTMs, PatchTST [16] is a transformer-based model for tasks related to multivariate time-series, such as forecasting, regression and classification.

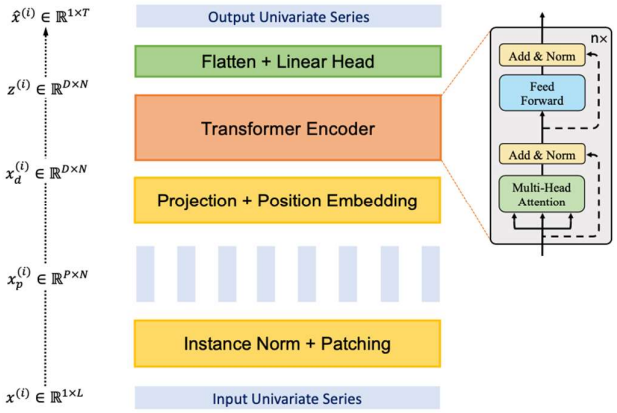
PatchTST makes use of two techniques: **patching**, like the previous models, aggregates several time steps into subseries-level short contiguous segments; **Channel-independence**, a multivariate time series is a multi-channel signal, which can be aggregated into a single data point to serve as input token for the Transformer; channel-independence, instead, means that each input token contains information from a single channel.

The purpose of these two ideas are mainly three: 1) a reduction on time and space complexity, since patching will reduce the amount of input tokens fed to the Transformer, this will be seen more in detail later; 2) increased locality and capture of comprehensive semantic information that would be missed in point-level evaluation, and 3) the separate per-channel attention would increase adaptability across heterogeneous series, reducing overfitting while still allowing cross-series weight sharing for transferability.

Below is a representation of the model's architecture.



(a) PatchTST Model Overview



(b) Transformer Backbone

Figure 5: a) contains a high-level overview of PatchTST; b) is a close-up of each layer of the Transformer Backbone

Where x is the matrix containing data of all M channels, with lookback window of length L. The first step is applying channel-independence on x, breaking it down in M vectors $x^{(\mathrm{i})}$, where $i = 1, \ldots, M$. At this point, for each i the Transformer backbone will produce a forecast of *T* future values, that is to say $\hat{x}^{(i)} = (\hat{x}_{L+1}^{(i)}, \ldots \hat{x}_{L+T}^{(i)})$.

Figure 5 (b) shows in detail the transformer's behavior: each $x^{(i)}$ is divided into patches, which can be either overlapped or not. Given P as patch length and S as stride (that is, the non-overlapping region between two consecutive patches), the patching process will generate a sequence of $x_p^{(i)} \in \mathbb{R}^{P \times N}$ patches, where $N = \lfloor \frac{L-P}{S} \rfloor + 2$ is the number of patches.

This is a significant improvement, since through the patching process we reduce the amount of input tokens to the transformer, from L to $N \approx L/S$: since the vanilla transformer's complexity is $O(N^2)$, the memory usage and computational complexity of the attention map are quadratically reduced by a factor of S.

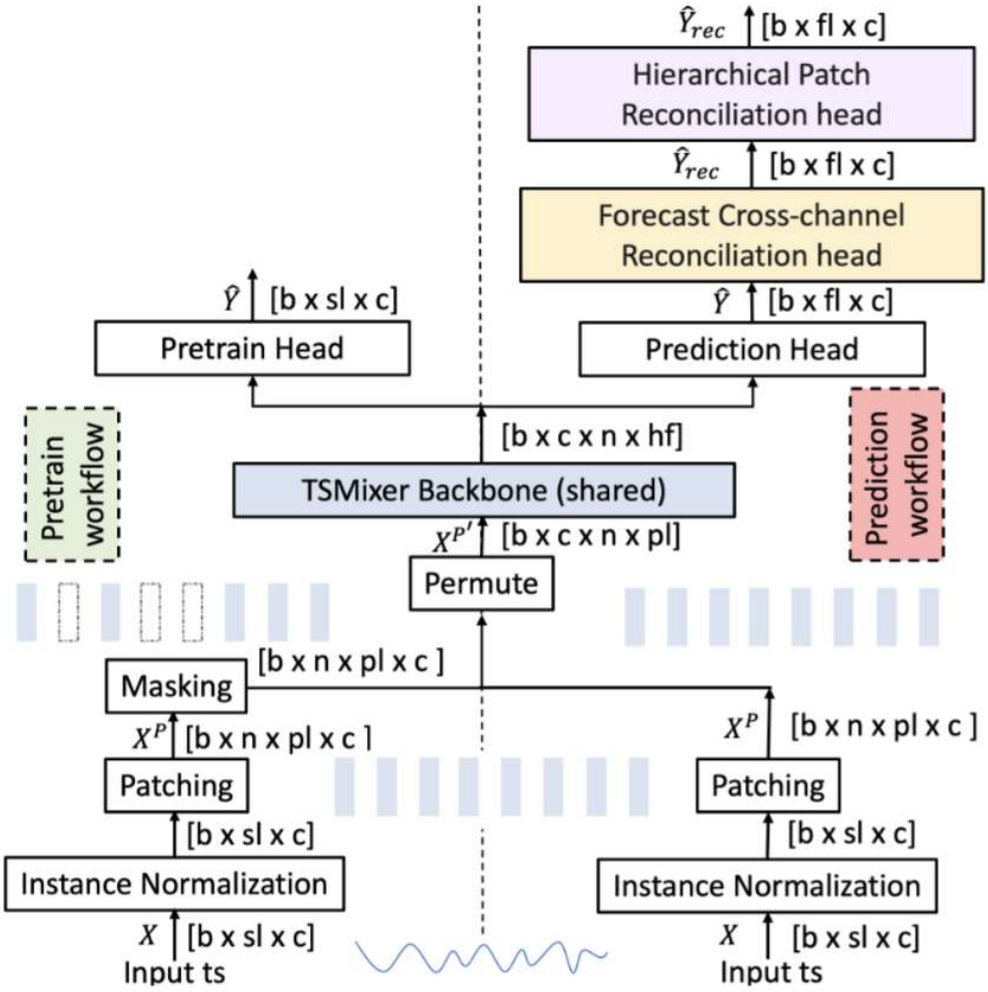
The model released was trained on a dataset monitoring 2 electrical transformers over the span of 2 years, tracking information such as the current load and temperatures; while the data used is not relevant, IBM has provided a guided demo on how to train a model from scratch on a custom dataset.

While providing the same functionalities of the previous models, being trained on unrelated data strongly hinders the reusability of the released mode; as such, for use in a precision farming context will most likely require the team to train a new model from scratch.

4.1.4 Granite PatchTSMixer

PatchTSMixer [18] is yet another model designed for multivariate time-series analysis. As the name suggests, this model still makes use of the patching process of PatchTST, while differing from it sue to the swap of the Transformer backbone with TSMixer [33], allowing it to be much lighter than it's Transformer-based counterpart: through empirical testing on 7 different datasets, IBM has found that, while TSMixer outperforms PatchTST by just 1-2%, it shows a reduction in training time and memory usage by a factor of 2-3X.

Below is an overview of the architecture.



Where $X_{c \times L}$ is a multivariate time series of length L and c channels, $sl \leq L$ the input sequence length, fl the forecast length, b the batch size, n and pl respectively the number of parches and a patch's length, and M the DNN model; the forecasting task is defined as predicting the future values:

$$\hat{Y}_{fl \times c} = \mathrm{M}(X_{sl \times c})$$

The actual future values are named as $Y_{fl \times c}$.

Training can be performed in two ways: supervised (following "prediction" workflow) and self-supervised ("pretrain" workflow).

- **supervised training**: the input sequence is normalized, patched and processed through a permutation process. Then, the result enters the TSMixer backbone, responsible for the training process. The backbone's output embedding is then converted into the base forecast $\hat{Y}$ by the prediction head; at this point, the model is trained to minimise the loss between $\hat{Y}$ and Y. The extra online forecast reconciliation heads, if activated, can tune the base forecasts and produce more accurate forecasts by leveraging cross-channel and patch-aggregation information. For more details on their functioning, refer to IBM's original paper.
- **self-supervised training**: while the normalization and patching processes are nearly identical, a masking process randomly masks a fraction of the input patches. The model is then trained to recover these missing patches. Afterwards, the pretrained model is finetuned through the "prediction" workflow.

The patching process is identical to PatchTST, although the self-supervised training, in contrast to supervised, needs patches to be strictly non-overlapping.

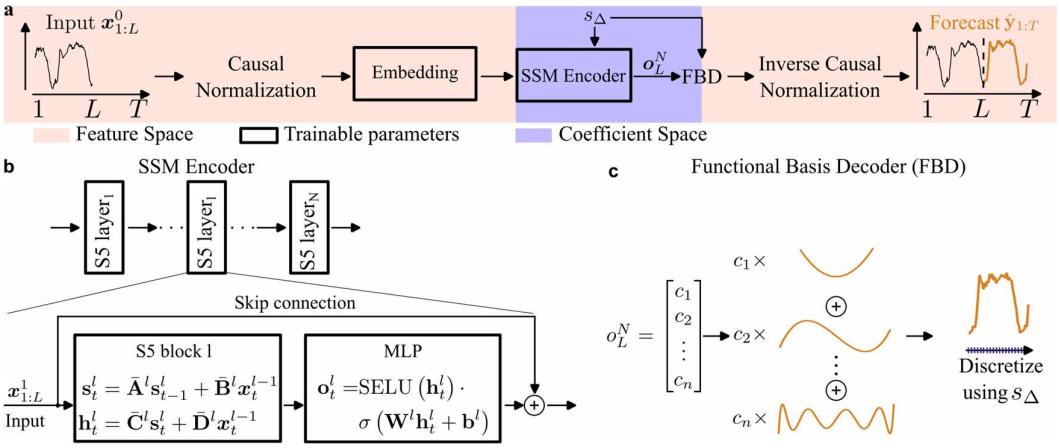
The TSMixer backbone will take in the several patches and produce an output embedding, which will then be used by either the prediction or pretrain head to forecast the future values (or missing patches); it's inner working will not be discussed in this work, please refer to IBM's paper for more information [37].

Like it's Transformer counterpart, the released model was trained on an electrical dataset; as such, use in precision farming will require training a new model from scratch.

4.1.5 Granite FlowState

FlowState [19] is a Time Series foundation model designed to perform zero-shot forecasting independently of the data's sampling rate. To achieve this, FlowState makes use of an encoder-decoder architecture, combining a state space model encoder with a functional basis decoder. A state space model (SSM) [38] is a kind of model trained to perform predictions on a system based on its internal state evolution over time; in particular, FlowState utilizes several state space layers, namely S5 layers [39], which, thanks to their architecture, allow FlowState to perform sampling rate-independent forecasting.

Below is an overview of the model's architecture.



The model's inner workings are quite complex, in the following section a high-level overview will be presented; for more information, consult IBM's paper [40].

a) is a complete overview of the model's architecture: the input time series of length L is normalized and then fed to the embedding module; the embedding is then fed to the encoder without any patching, in contrast to previous models; the SSM encodes the data into a coefficient space, that is to say the input series is processed into coefficients of a continuous basis function, allowing the Functional Basis Decoder (FBD) to produce a continuous output function; lastly, the decoder's output is inverse-normalized, using the inverse process of the input normalization; the result of this last process form the forecast of the model.

b) is a close-in of the SSM encoder: it is composed by a series of S5 layers, each consisting of an S5 block followed by an MLP; each S5 layer is connected by a skip connection to the later layers, and through the matrixes $\bar{A}^l$, $\bar{B}^l$, $\bar{C}^l$ and $\bar{D}^l$ (which represent, respectively, the state transition, input, output and skip connections); the final output of the SSM encoder is fed into the Functional Basis Decoder.

c) the FBD will interpret the SSM's final output as coefficients of a functional basis, and will generate a continuous output function as forecast. The advantage of the output being continuous lays in the fact that we will be able to sample it with any desired sample rate. Again, for more information on how the FBD works, consult the original paper [40].

Note that, in the image above, the value s_Δ is a parameter called scale factor, which is used both in the encoder and decoder to adjust for unseen sampling rates. On FlowState's model page, the values in Table 2 are suggested for different sampling rates.

Sampling Rate	Recommended Scale Factor
15min	0.25
30min	0.5
hourly	1.0
daily	3.43 if data has a weekly cycle, else 0.0656
weekly	0.46
monthly	2

Table 2: recommended scale factor per sampling rate, as reported on the model card.

IBM compared FlowState in zero-shot forecasting to several state-of-the-art models, such as NXAI's TiRex, Amazon's Chronos and Google's TimesFM, using the Gift-Eval benchmark [41]: two variants of different sizes (2.6M and 9.1M params) were trained on the benchmark data, and, as of July 31, 2025, both variants outperformed other state-of-the-art models at a fraction of their size. Below is an extract of the testing done by IBM, utilizing MASE (mean absolute scaled error) as a metric:

Model	#Param	MASE
FlowState-9.1M	9.1M	0.728
TiRex	35M	0.733
FlowState-2.6M	2.6M	0.733
TimesFM-2.0	500M	0.764
Chronos-bolt-b	205M	0.832

At the time of writing, the model published on Hugging Face has been trained on a subset of data from both Gift-Eval Pretrain [41] and Chronus Pretraining Data Corpus [42]; additionally, the current model does not support more than one input channel, rendering this model unable to perform forecasting on multivariate time series.

4.1.6 Granite Geospatial Biomass

Granite Geospatial Biomass (GGB) [26] has been developed for the purpose of predicting the total amount of aboveground biomass utilizing satellite imagery. The model has been trained using Terratorch [43], a toolkit intended for simplifying fine-tuning, evaluation and deployment of Geospatial Foundation models; TerraTorch's architecture will not be discussed in this work, please refer to their paper for more information.

GGB was fine-tuned from a model similar to Prithvi [44], a model created to test the proposed training framework for Geospatial Foundation models, which will not be discussed in detail in this work; GGB switches the proposed Vision Transformer (ViT) for the Swin-B Transformer backbone [45] as an encoder, as it enables the model to process at higher effective resolutions while also improving computational efficiency thanks to the use of windowed self-attention. Swin-B will also not be discussed in this work, as such please consult the original paper for more information [45]; more information regarding the fine-tuning process performed by IBM can be found here [46].

Pretraining was performed using SimMIM [47], a self-supervised learning strategy based on masking a part of the input data which will then be reconstructed by the model.

Two datasets were used for the training process, Harmonized Landsat-Sentinel 2 (HLS) [48] and Global Ecosystem Dynamics Investigation (GEDI) [49], both provided by NASA. Both training and testing require a cloud-free snapshot of an area: starting from HLS data gathered during the leaf-on season for each hemisphere, pixels not contaminated with clouds were selected; the mean value of each cloud-free pixel is computed during the leaf-on season for each spectral band, which is then assembled into a composite image representative for that area; the corresponding GEDI data obtained during the same leaf-on season are interpolated to the HLS grid (CRS:4326) such that the measured biomass points are aligned with HLS data.

GGB can be very useful in relation to precision farming and Farm-Tech, first and foremost in enabling continuous monitoring of fields and yield estimation; while originally trained on data retrieved from forests, IBM has provided a notebook on how to perform few-shot forecasting on an unseen biome: perhaps by performing few-shot fine-tuning on crop imagery the model could generalize fairly well, although testing is needed.

4.2 Disease Detection

This model [17] has been developed to detect diseases in crops based on a picture of its leaves. It is a Vision Transformer model, finetuned from Google's "vit-tiny-patch16-224" [28]; do note that this model's Hugging Face repository is a reupload of the model from the Timm repository [50], which contains a large number of models and utilities for image processing; this is because Google did not release the original model on HuggingFace.

While the exact dataset has not been shared, the model's page describes it as a "diverse dataset of plant images, including different disease categories affecting crops such as corn, potato, rice, and wheat [...] includes images captured under various lighting conditions, from both controlled and uncontrolled environments and angles"; the exact number of pictures is not shared, but a table of all the classes present in the dataset is:

Crop	Class
Corn	Common Rust
	Gray Leaf Spot
	Healthy
	Leaf Blight
Potato	Early Blight
	Healthy
	Late Blight
Rice	Brown Spot
	Healthy
	Leaf Blast
Wheat	Brown Rust
	Healthy
	Yellow Rust

It should be noted that in the above table the class "rice_hispa" is not present, but later in the model's readme a confusion matrix is presented, displaying such a class.

While it's not clear if the confusion matrix above is the one being referenced, testing was performed on a validation set consisting of 20% of the original dataset; if the confusion matrix is the result of this testing, then the original dataset was composed of ≈23k images.

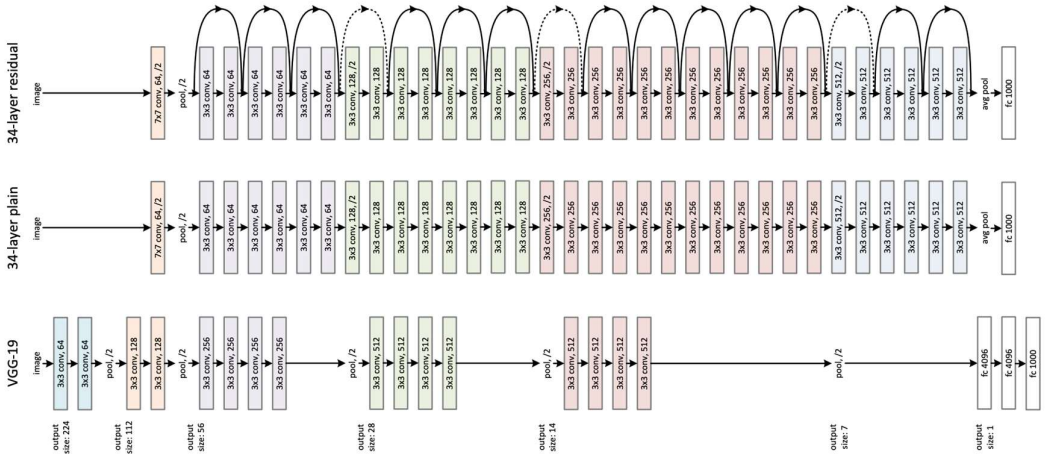
The author reports that the model may not generalise to unseen crops; as such, fine-tuning on relevant crops (that is, crops expected to be processed) is strongly suggested.

Being able to classify illnesses based on just the leaves and not the crop itself, this model can be very useful in precision farming, although finetuning may be necessary to include other kinds of crops.

4.3 Fruits and Vegetables Detector

This model [21] is a fine-tuned version of ResNet-50 [51], a deep neural network developed by Microsoft for the purpose of image classification; 50 represents the depth of the network.

The main takeaway from ResNet is the use of Shortcut Connections, which allow the input of a layer to be passed directly to the next layer's input, effectively acting as an identity mapping; in the paper, it is demonstrated that these shortcut connections allow a deep network to not degrade any worse than a shallower counterpart. More information can be found in the original paper [51]. Below is an overview of its architecture compared to other DNNs, where the shortcut connections are connecting each layer's input to the next.



Going back to Fruits and Vegetables detectors, it is not well documented: the model was trained on a dataset containing 3825 images of 36 different fruits, a link to the dataset is provided [52]; testing on the evaluation set found it achieves an accuracy of 0.97; other than this, the training hyperparameters were provided, they can be found on the model's HuggingFace Repo. Additionally, this model exclusively performs fruit detection, as such it is unclear whether it is able to detect disease; such a matter should be taken into consideration before implementation into a live field.

Being released under Apache 2.0, as well as its father being under the same license, this model can be freely used in commercial practices, although, possibly lacking the ability to detect diseases, [17] may be better suited for this purpose.

4.4 Leaf Detection and Classification

This model [22] has been trained using YOLOv8 [29] to detect and classify plant leaves: YOLOv8 is a computer vision model architecture developed by Ultralytics, which allows to train models at performing detection and classification of images and real-time video feeds.

While Ultralytics has provided no proper documentation on YOLO's internal architecture nor inner workings, it has uploaded a vast number of tutorials and guided examples on how to train a model with it, which would speed up the process of building a system ourselves if needed.

Going back to the model, its card does not share much information about the training process: the dataset utilized was not shared, but it is stated that it's composed by "hundreds of images of 46 different plants, including both disease-infected and healthy leaves"; training was ran for 50 epochs, and, while the terms "testing" nor "evaluation" appear on the card, it is claimed that "The model has achieved a precision (mAP@0.5) of 0.946 on the object detection task".

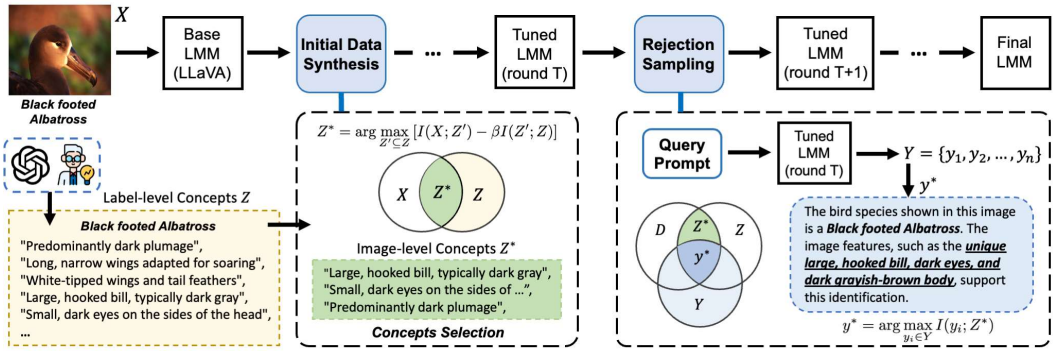
While the model card reports no license, YOLOv8 is released either under AGPL 3.0 or YOLOv8 Commercial Licensing [53]; since it's not clear if this model can be used in commercial settings, it may be wise to contact the authors to request more information.

Given the little information available in regards to this model, and also given that it performs the same task as [17], it may be better to use the latter model in a live application.

4.5 Classification of Leaf Diseases

This model [24] was trained for the purpose of evaluating the SelfSynthX framework [54], a framework developed to fine-tune large multimodal models (LMMs) to improve their ability of performing fine-grained visual reasoning and increase explanation quality. We report a brief overview of SelfSynthX, for more information please consult the original paper [54].

Below is an overview of the framework's architecture:



Given an image, X represents it's true contents and c it's class label, each c is associated with a set of visual concepts Z, which can either be defined by domain experts or by an LLM; the base LMM we wish to fine-tune (LLaVa in the paper) will now be prompted to describe the image, generating a set of descriptions D; the model is then tuned to maximise the mutual information between X, D and Z, and to select a set of $Z^* \subseteq Z$ which maximises the relevance of D to X. Once Z^* is found, it's possible to generate an explainable answer for a classification on each image by prompting the base LMM, such as "What is this bird's species? Explain your reasoning."; the pairs image and query-answer are later used to fine-tune the LMM. The question reported above was asked to the evaluation model trained on a dataset of birds

After the above steps, the tuned model goes through the Rejection Sampling process: for each image and query, several candidate answers are generated, scoring each by how well it aligns with the image-specific Z^*; the highest scoring candidate is accepted for successive fine-tuning only if it also correctly predicts the image's class label.

As stated earlier, the model [24] was trained to evaluate SelfSynthX: as such, it is a fine-tuned version of LLaVa trained on the Plant Diseases Dataset [55]; other than this model, more models were trained on different datasets to evaluate the framework on different domains, unrelated to precision farming.

4.6 Timeseries Anomaly Detection

This model [27] is an example implementation provided by the Keras team, designed to perform anomaly detection in time series data using a reconstruction-based autoencoder architecture. Like previous discussed models that perform anomaly detection, this model finds uses in smart farming to detect sensor data irregularities due to malfunction or environmental stress.

Keras is a high-level deep learning API built on top of frameworks like TensorFlow, designed to simplify the creation, training, and deployment of neural networks.

The model is built around a convolutional autoencoder, which learns to reconstruct time series signals. During training, the autoencoder is exposed to normal patterns of behavior; anomalies are then detected when the reconstruction error exceeds a defined threshold.

The model was trained on the Numenta Anomaly Benchmark (NAB) dataset [56], which provides synthetic time series data with labeled anomalous periods. While useful for demonstrating anomaly detection, NAB is not specific to agriculture, as such fine-tuning, or outright training a model from scratch, would be necessary for deployment in Farm-Tech.

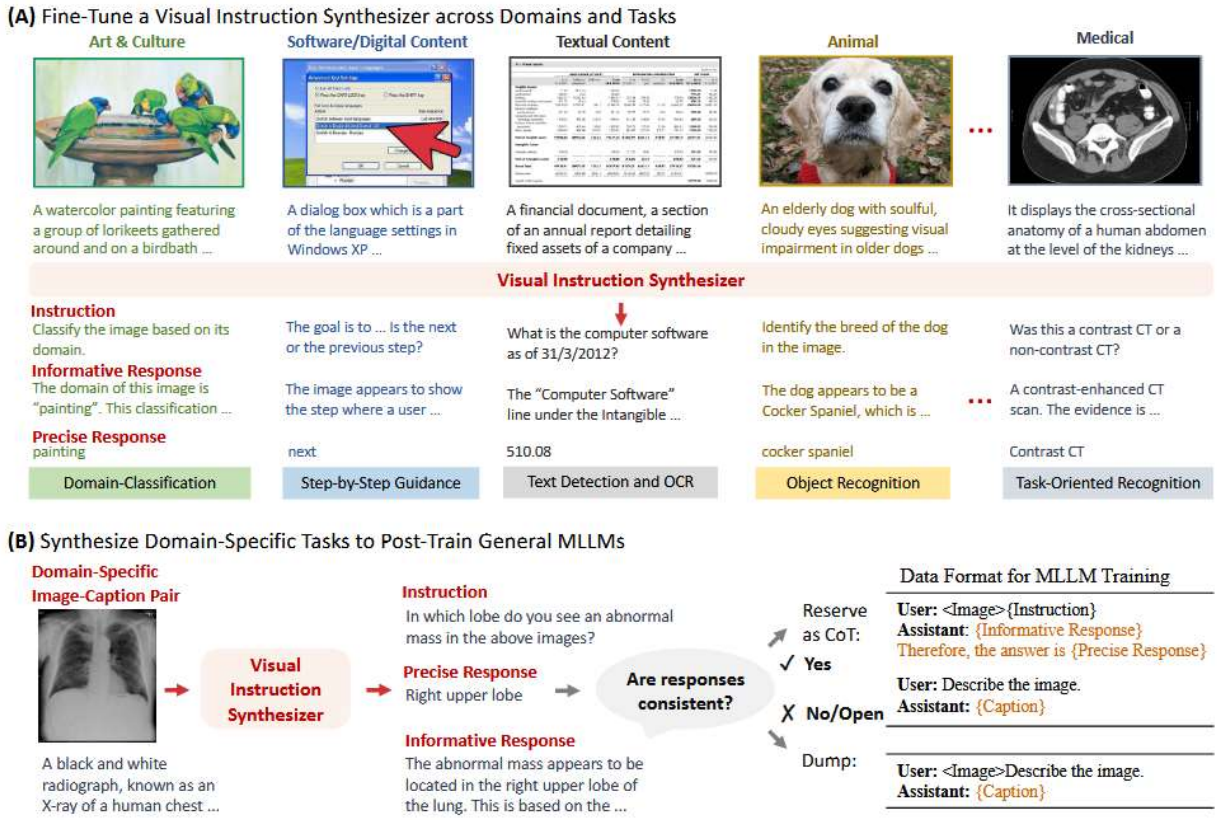
No license is reported on the model card, although the GitHub repository containing the code used for training the model reports an Apache 2.0 license.

Concluding, this model provides the same functionality of the IBM Granite models, with the deficit of having been trained on a small dataset unrelated to farming; additionally, the data used contains only univariate time series, further hindering the applicability of this model in a live application. As such, it is believed that the reusability of this model is not on-par with other, similar models; instead of training a new model from scratch utilizing the Keras API and relevant data, it may be more efficient to instead fine-tune other models seen earlier in this work.

4.7 AdaptLLM Remote Sensing

These models [20] [25] have been trained in the context of this paper [57], which aims at adapting general multimodal large language models (MLLMs) to specific domains: the domains considered in the paper are biomedicine, food, and remote sensing, while the models analyzed are Qwen2-VL-2B, LLaVA-v1.6-8B, and Llama-3.2-VL-11B; the two models retrieved for our research are the models adapted from Llama and Qwen2 for the remote sensing domain, the LLaVa was not retrieved as it had less than 100 downloads.

Below is an overview of the training process:



The process to fine-tune the original MLLMs is quite simple: A) a base model is fed a corpus of image-caption pairs plus a few triplets composed by an instruction (a natural-language task prompt derived from the image and caption), informative reasoning (chain-of-thought style steps supporting the answer) and precise answer (the concise final response), and fine-tuned to generate more of these triplets given an image and caption pair, as such it is now called Synthesizer; B) the synthesizer now goes through a language model which classifies each triplet as consistent or not, that is whether the precise and informative responses align, dumping the inconsistent triplets; lastly, the consistent triplets generated by the synthesizer are used to post-train the MLLM model on the domain of interest. Do note that this last step showcased in this work has been simplified, more information can be found in the original paper [57].

To experiment the proposed fine-tuning, a synthesizer based on LLaVa-v1.6-8B was used to generate triplets relevant to three domains, that is biomedicine, food and remote sensing; Qwen2, LLaVa and Llama were then trained on the synthesizer data and then compared to several baselines (different for each domain), resulting in the models trained with the proposed method consistently outperforming the baselines.

The models retrieved in this work are the Qwen and LLaVa based trained on remote sensing; since both models perform the same task, testing may be necessary to determine which one is more appropriate for the implementation we're aiming for.

Performing drone-imagery analysis can be a useful task for precision farming, providing a natural-text description of images to end users, for example estimating the status of the crop. It is important to remember the licenses: neither of the model cards report any license, but they both redirect to a GitHub repository [58] which lists the "Beijing Institute for General Artificial Intelligence BIGAI License Agreement", which requires to pay a fee to the original authors; as such, contacting the authors before implementing these models is in order.

4.8 CropSeek LLM

This model [23] is a fine-tuned language model designed for agricultural applications, with a focus on crop optimization. It is based on the DeepSeek R1 model [59] and fine-tuned using a dataset [60] containing a small corpus of question-answer pairs, each linked to the crop being considered; Figure 6 contains a preview of the dataset. The model leverages LoRA (Low-Rank Adaptation) techniques [61] to efficiently adapt the base model to domain-specific tasks.

Crop string · classes 4 values	Question string · lengths 35 76	Answer string · lengths 43 114
Green Peppers	What fertilizer is best for green peppers?	Apply balanced NPK fertilizer (5:10:10) and supplement with calcium and magnesium.
Green Peppers	How do I maximize green pepper yield?	Pruning lower leaves, staking, and using mulch help boost production.
Green Peppers	What temperature range is best for green pepper growth?	Green peppers thrive between 20°C and 30°C.
Green Peppers	How can I extend the shelf life of harvested green peppers?	Store at 7–10°C with high humidity to prevent shriveling.
Jam Tomatoes	What is the best variety of jam tomatoes for South Coast, Durban?	Roma VF and Heinz 1370 are well-suited for warm coastal conditions.
Jam Tomatoes	How should I prepare soil for tomato farming?	Ensure well-drained loamy soil with a pH of 5.5–6.8, rich in compost.
Jam Tomatoes	What is the best way to prevent tomato blight?	Rotate crops, avoid overhead watering, and use copper-

Figure 6: Sample of the dataset used for fine-tuning

The training dataset is very limited in size for the intended task, totaling at 40 entries; only four types of crops are present: Green Peppers, Jam Tomatoes, Cabbage and Marigold Flowers. Given the small size of the corpus, as well as the small variety of crops considered, further fine-tuning is most likely needed before use in a live application.

Within Farm-Tech, CropSeek could serve as a conversational assistant for farmers and agronomists. Its ability to provide recommendations on planting seasons, soil conditions, pest control, and irrigation strategies aligns well with Farm-Tech's mission of supporting sustainable agricultural practices. Integrated into Farm-Tech's platform, it could enhance decision support systems by offering real-time guidance to end users. Although, careful consideration has to be put into the data used during training.

The model is released under the DARJYO License v1.0 [30], which explicitly restricts commercial use. This limitation poses a significant barrier to integration within Farm-Tech: given this, as well as the aforementioned limited fine-tuning data, it may be more efficient to fine-tune a version of DeepSeek (or any other LLM) with data relevant to the expected crops and climates in which Farm-Tech will be deployed.

5. Models reusability

In this chapter, we propose a scoring of each model's reusability in regards to farmtech. An important concept that can help us defining a way to measure the reusability of a model is Transfer Learning.

Transfer learning [62] is a technique that adapts a model pre-trained on one task or dataset to a different but related task, leveraging previously learned representations to reduce training time, data requirements, and computational cost. In the context we're examining, transfer learning enables reusing models trained on general time-series or vision datasets by fine-tuning them on domain-specific sensor data or crop imagery.

For a pre-trained model to be a valid candidate for reuse, the following conditions should be evaluated:

- **Task compatibility**: the source task and the target task must share relevant structure (for example, image classification to detect plant disease, or multivariate forecasting to predict sensor readings). Greater similarity increases the likelihood of successful transfer.
- **Availability of fine-tuning data**: sufficient labeled or semi-labeled target data must be available for fine-tuning. Few-shot or zero-shot scenarios are possible for some foundation models, but practical deployment typically requires at least a representative dataset for calibration.
- **Licensing and legal constraints**: the model's license must permit the intended use (commercial in our case), license incompatibilities or restrictive clauses will block reuse regardless of technical suitability.
- **Domain shift and bias**: differences between the pre-training data distribution and the target domain (climate, crop varieties, sensor types) must be considered, as significant domain shift may require more extensive fine-tuning or domain adaptation techniques.
- **Operational constraints**: inference latency, memory footprint, and hardware availability in Farm-Tech's deployment environment must be compatible with the model's requirements.

The conditions outlined above provide a theoretical foundation for assessing whether a pre-trained model can be successfully reused in precision farming. In practice, we apply these conditions in our proposed evaluation method, listed in the next page.

To evaluate each model, we define a set of categories representing an aspect of the model with a score from 1 to 3; following are the proposed categories, alongside the meaning for each value:

- **Project alignment (PA)**, represents how relevant the model is to the project, both in purpose and training dataset: 1 represents a general affinity to Farm-Tech's field of application; 2 an immediate correlation to a task relevant to Farm-Tech; 3 satisfies 2's condition plus, during training, data immediately related to farming was used.
- **Quality of documentation (QD)**, represents how much the documentation provided is useful in the implementation of the model in a production-ready environment: 1 the documentation is enough for the sole scope of running the model; 2 an overview of the architecture is provided; 3 satisfies 2's condition plus a significant number of guides and examples are provided.
- **Ease of evaluation (EE)**, represents how simple it is to evaluate a model's efficiency in the novel field: 1 requires significant testing; 2 means that specific data has to be gathered or generated; 3 means that even utilizing general-purpose data allows us to effectively evaluate the model.
- **License (LC)**, the license under which the model was released: 1 means that the model was released under a restrictive license; 2 means that the model either lists no license or is ambiguous under which license it uses; 3 means that the model is released under a permissive license

To provide a final score for each model, a simple average of all categories is computed: while we are utilizing a simple average, another valid approach is to compute a weighted sum of each score depending on which category we value more; for instance, given the importance of the license status, the total score may have been computed as $Total\ Score = PA * 0.2 + QD * 0.2 + EE * 0.2 + LC * 0.4$. Additionally, depending on the context in which this method is used, the maximum score may be chosen higher (or lower) than our proposed 3, depending on the precision for which a team is aiming for.

In Table 3 (next page), we report each model's score for each category, explaining the reasoning for each score in the next paragraph

Model	PA	QD	EE	LC	Total Score
TTM [13] [14]	2	3	2	3	2.5
TSPulse [15]	2	3	2	3	2.5
PatchTST [16]	2	3	2	3	2.5
PatchTSMixer [18]	2	3	2	3	2.5
FlowState [19]	1*	3	2	3	2.2
Geospatial Biomass [26]	2	3	2	3	2.5
Disease Detection [17]	**3**	**2**	**3**	**3**	**2.7**
Fruits and Vegetables Detector [21]	2*	2	3	3	2.5
Leaf Detection and Classification [22]	3	2	3	2	2.5
Classification of Leaf Diseases [24]	3	2	1	2	2
Timeseries Anomaly Detection [27]	1*	2	2	2	2
AdaptLLM Remote Sensing [20] [25]	3	2	1	2	2
CropSeek LLM [23]	2*	3*	1	1	1.7

Table 3: evaluation of each model; the asterisks (*) represent an exception to the scoring rules defined above; they will be explained in the following section. Do note that TTM v1 and v2 as well as the two AdaptLLM models were combined into a single entry.

As can be seen in Table 3, the Disease Detection model [17] emerges as the strongest candidate overall, achieving the highest total score of 2.7: finding diseases in crops is a very useful task in precision farming, and having been trained directly on a large corpus of common crop imagery grants a score of 3 in PA. The scoring of 2 in QD is because, while a single usage example is presented in the model card, it is a fine-tuning of Google's ViT [28], of which the architecture is known. EE has a score of 3 since we can easily compile a test dataset of plant imagery to evaluate the model; LC also gets a score of 3 as, while the model presents contradicting license information, both are permissive licenses. This combination of high project alignment, ease of evaluation, and permissive licensing makes [6] the most promising model for immediate integration into Farm-Tech.

All IBM Granite models have an identical scoring: timeseries forecasting, anomaly detection and pattern classification are all functions that are highly relevant to a smart farming application, but, being general purpose models, they were trained on non-relevant data, thus getting a 2 in the PA category. FlowState [19] has a score of 1 in this category since, as of time of writing, it is limited to work with timeseries composed by a single channel, reducing its utility in a live application. Having a large documentation backing them, these models all get the maximum score in the QD category, while the scoring in the EE category is set at 2 since, in order to properly evaluate them, proper data has to be produced. Being released under Apache 2.0, LC gets the maximum score.

Fruits and Vegetables Detector[21]: this model received a score of 2 in PA as it is unable to detect disease in crop, instead only classifying a picture as a type of crop, limiting its actual uses in a live application. All the other scores are the same as [17] since the model's architecture is known, is easy to evaluate and is released under a permissive license.

Leaf Detection and Classification [22]: same as[21] , with the difference of LC having a score of 2 as yolo-based models require a commercial license.

Classification of Leaf Diseases [24]: same as [17], with the difference of EE getting a 1 since, using an LLM to describe images, it may require further testing to ensure the model doesn't hallucinate. Additionally, LC gets a score of 2 since LLaVa may require a commercial license depending on the number of monthly users.

Timeseries Anomaly Detection [27]: performing only anomaly detection on time series, and trained on non-relevant, univariate data, this model gets a 1 in PA; 2 in QD because it was trained as part of a framework so you know it's background; EE gets also a score of 2 for the same reason of the IBM models; since it reports no license, this model gets a score of 2 in LC.

AdaptLLM Remote Sensing [20] [25]: performing natural-language descriptions of drone imagery can be useful to end users, as such it gets 3 on PA. In QD it gets a scoring of 2 since not many use-cases are provided. Like [24], EE gets a score of 1 as it may require significant testing to ensure the correct functioning of the LLM. LC gets a score of 2 since it is not clear the license under which these models are released.

CropSeekLLM [23]: given the limited fine-tuning data used, PA gets a score of 2; additionally, being a chatbot doesn't require much documentation, therefore getting a score of 3 in QD. Like for [20], [24] and [25], ensuring correct behavior may require significant testing as well as domain-specific experts, as such it scores a 1 in EE. Lastly, given the restrictive license, this model gets a score of 1 in LC.

6. Limitations

Given the search method used to compile the models list, it is possible that other models more in-line with precision farming and farm-tech itself were not retrieved; as such, this work could be expanded by several methods, such as including more search terms, diversifying the source from which the models were pulled, or removing the requirement of needing more than 100 downloads in order to be considered.

Another limitation of this work is determined by the lack of relevant testing data on Smart Shaped's side that may have been used to experiment with the models, as a way to obtain a more accurate evaluation of their performances in Farm-Tech; the models most affected by this are the time series analysis models, since it is very likely possible that their performance may depend on the overall structure of the data used.

A noteworthy observation to make is that nearly half of the relevant models (46.6%) belong to the IBM Granite family. This concentration suggests a potential bias in the retrieval process, as the methodology may have favored models from a single provider, thereby reducing the diversity of approaches considered. Such bias could have been caused by the exclusive use of HuggingFace as the sole source of models considered; expanding the number of sources will provide a wider range of variety, possibly increasing the number of relevant models. Of course, this can be determined only through additional research.

7. Conclusions

The work presented in this thesis set out to explore the reusability of pre-trained models in the context Farm-Tech: by systematically collecting, filtering, and analyzing models from HuggingFace, we established an evaluation methodology to verify their applicability to precision farming.

Several insights emerged from this study:

- **Model relevance and diversity**: out of 100 models initially retrieved, only 15 were deemed relevant to this field. This highlights both the scarcity of domain-specific models and the importance of careful filtering when reusing general-purpose models. The predominance of time series analysis models (46.6%) underscores the centrality of sensor data forecasting in smart farming.
- **Licensing constraints**: While most models were released under permissive licenses such as Apache 2.0, a significant subset presented unclear or restrictive licensing terms. This factor directly impacts reusability, as legal barriers can prevent integration into commercial platforms. CropSeekLLM [23], for example, explicitly prohibits commercial use, while others require negotiation with the original authors.
- **Bias and documentation gaps**: The majority of models did not provide sufficient discussion of bias or limitations in their training data. This omission poses risks when transferring models to new agricultural contexts, where crop varieties, diseases, and environmental conditions may differ substantially from those represented in the original datasets. Documentation quality also varied, with many models discarded due to insufficient information.
- **IBM Granite models as a foundation**: Nearly half of the relevant models belonged to IBM's Granite suite. Although trained on general-purpose datasets, their robust architectures and extensive documentation make them valid candidates for fine-tuning in precision farming. Their modularity and efficiency align well with Farm-Tech's architecture, particularly for tasks involving sensor data forecasting and anomaly detection.

Considering these, we can conclude that model reusability in precision farming, and likely in any other field, is feasible but requires careful considerations: licensing clarity, data compatibility, and bias awareness are important factors to monitor. The proposed evaluation methodology provides an approach that can be generalized beyond agriculture to other domains where pre-trained models are repurposed.

In conclusion, this thesis demonstrates that pre-trained models hold significant promise for advancing precision farming, but their successful reuse requires an in-depth evaluation that balances technical performance with legal and contextual considerations. By adopting such a structured approach, Farm-Tech and similar initiatives can accelerate innovation while ensuring sustainability, scalability, and trust in AI-driven agriculture.

References

[1] Hugging Face [Online]. Available: https://huggingface.co/docs/hub/index. [Accessed 26 November 2025].

[2] OSGeo, "Geonode," [Online]. Available: https://geonode.org. [Accessed 26 November 2025].

[3] MinIO Inc., "MinIO," [Online]. Available: https://www.min.io/. [Accessed 26 November 2026].

[4] OSGeo, "GeoServer," [Online]. Available: https://geoserver.org/. [Accessed 26 November 2025].

[5] Apache Software Foundation, "Apache Spark," [Online]. Available: https://spark.apache.org/. [Accessed 26 November 2025].

[6] Smart Shaped s.r.l., "chaM3Leon," [Online]. Available: https://github.com/Smart-Shaped/chaM3Leon. [Accessed 26 November 2025].

[7] Django Software Foundation, "Django," [Online]. Available: https://www.djangoproject.com/. [Accessed 26 November 2025].

[8] PostgreSQL Global Development Group, "PostgreSQL," [Online]. Available: https://www.postgresql.org/. [Accessed 26 November 2025].

[9] KeyCloak Project, "KeyCloak," [Online]. Available: https://www.keycloak.org/. [Accessed 26 November 2025].

[10] Hugging Face, "Hugging Face Hub API Endopoints," 2023. [Online]. Available: ttps://huggingface.co/. [Accessed 26 November 2025].

[11] Ding, Ziqi, et al. “A Rusty Link in the AI Supply Chain: Detecting Evil Configurations in Model Repositories.” Version 1, arXiv:2505.01067, arXiv, 2 May 2025. *arXiv.org*, https://doi.org/10.48550/arXiv.2505.01067.

[12] Full List of Pre-trained models, 2025. [Online]. Available: https://docs.google.com/spreadsheets/d/17MHOh_VFrFM3qAwXw3VAOcdX4I0mnw9C517qEQdEdTg/edit?usp=sharing.

[13] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam, "Granite TimeSeries TTM R2," [Online]. Available: https://huggingface.co/ibm-granite/granite-timeseries-ttm-r2. [Accessed 26 November 2025].

[14] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam, "Granite TimeSeries TTM R1," [Online]. Available: https://huggingface.co/ibm-granite/granite-timeseries-ttm-r1. [Accessed 26 November 2025].

[15] Ekambaram, Vijay, et al. “TSPulse: Dual Space Tiny Pre-Trained Models for Rapid Time-Series Analysis.” arXiv:2505.13033, arXiv, 25 June 2025. *arXiv.org*, https://doi.org/10.48550/arXiv.2505.13033.

[16] Nie, Yuqi, et al. “A Time Series Is Worth 64 Words: Long-Term Forecasting with Transformers.” arXiv:2211.14730, arXiv, 5 Mar. 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2211.14730.

[17] Wambugu Kinyua. Smart farming disease detection transformer, 2024.

[18] Ekambaram, Vijay, et al. “PatchTSMixer model pre-trained on ETTh1 dataset” [Online]. Available: https://huggingface.co/ibm-granite/granite-timeseries-patchtsmixer. [Accessed 26 November 2025]

[19] Graf, Lars, et al. “FlowState: Sampling Rate Invariant Time Series Forecasting.” arXiv:2508.05287, arXiv, 19 Aug. 2025. *arXiv.org*, https://doi.org/10.48550/arXiv.2508.05287.

[20] Cheng, Daixuan, et al. “Adapting Large Language Models via Reading Comprehension.” 2023, The Twelfth International Conference on Learning Representations. *openreview.net*, https://openreview.net/forum?id=y886UXPEZ0.

[21] J. Macedo, " fruits-and-vegetables-detector-36," [Online]. Available: https://huggingface.co/jazzmacedo/fruits-and-vegetables-detector-36. [Accessed 26 November 2025].

[22] persadian~Darshani Persadh, DARJYO. Cropseek-llm: A fine-tuned language model for agricultural applications. https://huggingface.co/persadian/CropSeek-LLM, 2023.

[23] persadian~Darshani Persadh, DARJYO, "CropSeek-LLM," [Online]. Available: https://huggingface.co/persadian/CropSeek-LLM. [Accessed 26 November 2025].

[24] Yucheng Shi, Quanzheng Li, Jin Sun, Xiang Li, and Ninghao Liu. Enhancing cognition and explainability of multimodal foundation models with self-synthesized data. In The Thirteenth International Conference on Learning Representations, 2025.

[25] Cheng, Daixuan, et al. "Adapting Multimodal Large Language Models to Domains via Post-Training," [Online]. Available: https://huggingface.co/AdaptLLM/remote-sensing-Llama-3.2-11B-Vision-Instruct. [Accessed 26 November 2025].

[26] Muszynski, Michal, et al. “Fine-Tuning of Geospatial Foundation Models for Aboveground Biomass Estimation.” arXiv:2406.19888, arXiv, 28 June 2024. *arXiv.org*, https://doi.org/10.48550/arXiv.2406.19888.

[27] P. Vijay, " Timeseries anomaly detection using an Autoencoder," [Online]. Available: https://huggingface.co/keras-io/timeseries-anomaly-detection. [Accessed 26 November 2025].

[28] Wu, Bichen, et al. “Visual Transformers: Token-Based Image Representation and Processing for Computer Vision.” arXiv, 2020. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2006.03677.

[29] Ultralytics, "YOLOv8," [Online]. Available: https://yolov8.com/. [Accessed 26 November 2025].

[30] DARJYO, "CropSeek-LLM," [Online]. Available: https://cropseek.darjyo.com/features.html. [Accessed 26 November 2025].

[31] University of Wisconsin-Madison, Microsoft Research, and Columbia University. "LLaVA v1.5 7B," [Online]. Available: https://huggingface.co/llava-hf/llava-1.5-7b-hf. [Accessed 26 November 2025].

[32] IBM, "IBM Granite" [Online]. Available: https://www.ibm.com/granite. [Accessed 26 November 2025].

[33] Chen, Si-An, et al. “TSMixer: An All-MLP Architecture for Time Series Forecasting.” arXiv:2303.06053, arXiv, 11 Sept. 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2303.06053.

[34] Ekambaram, Vijay, et al. “Tiny Time Mixers (TTMs): Fast Pre-Trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series.” arXiv:2401.03955, arXiv, 7 Nov. 2024. *arXiv.org*, https://doi.org/10.48550/arXiv.2401.03955.

[35] Ekambaram, Vijay, et al. “TSPulse: Dual Space Tiny Pre-Trained Models for Rapid Time-Series Analysis.” arXiv:2505.13033, arXiv, 25 June 2025. *arXiv.org*, https://doi.org/10.48550/arXiv.2505.13033.

[36] Darcet, Timothée, et al. “Vision Transformers Need Registers.” arXiv:2309.16588, arXiv, 12 Apr. 2024. *arXiv.org*, https://doi.org/10.48550/arXiv.2309.16588.

[37] Ekambaram, Vijay, et al. “TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting.” *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* [Long Beach CA USA], 2023, pp. 459–69. *DOI.org (Crossref)*, https://doi.org/10.1145/3580305.3599533.

[38] IBM, "What is a state space model?," [Online]. Available: https://www.ibm.com/think/topics/state-space-model. [Accessed 26 November 2025]

[39] Smith, Jimmy T. H., et al. “Simplified State Space Layers for Sequence Modeling.” arXiv, 2022. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2208.04933.

[40] Graf, Lars, et al. “FlowState: Sampling Rate Invariant Time Series Forecasting.” arXiv, 2025. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2508.05287.

[41] Salesforce AI Research, "GIFT-Eval Pre-training Datasets," [Online]. Available: https://huggingface.co/datasets/Salesforce/GiftEvalPretrain. [Accessed 26 November 2025].

[42] AutoGluon, " Chronos datasets," [Online]. Available: https://huggingface.co/datasets/autogluon/chronos_datasets. [Accessed 26 November 2025]

[43] Gomes, Carlos, et al. “TerraTorch: The Geospatial Foundation Models Toolkit.” arXiv, 2025. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2503.20563.

[44] Jakubik, Johannes, et al. “Foundation Models for Generalist Geospatial Artificial Intelligence.” arXiv, 2023. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2310.18660.

[45] Liu, Ze, et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows.” arXiv, 2021. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2103.14030.

[46] Muszynski, Michal, et al. “Fine-Tuning of Geospatial Foundation Models for Aboveground Biomass Estimation.” arXiv, 2024. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2406.19888.

[47] Xie, Zhenda, et al. “SimMIM: A Simple Framework for Masked Image Modeling.” arXiv, 2021. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2111.09886.

[48] Masek, J., Ju, J., Roger, J.-C., Skakun, S., Vermote, E., Claverie, M., Dungan, J., Yin, Z., Freitag, B., & Justice, C. (2021). *HLS Sentinel-2 Multi-spectral Instrument Surface Reflectance Daily Global 30m v2.0* [Data set]. NASA Land Processes Distributed Active Archive Center. https://doi.org/10.5067/HLS/HLSS30.002. [Accessed 26 November 2025]

[49] Dubayah, R.O., J. Armston, J.R. Kellner, L. Duncanson, S.P. Healey, P.L. Patterson, S. Hancock, H. Tang, M.A. Hofton, J.B. Blair, and S.B. Luthcke. 2021. GEDI L4A Footprint Level Aboveground Biomass Density, Version 1. ORNL DAAC, Oak Ridge, Tennessee, USA. https://doi.org/10.3334/ORNLDAAC/1907.

[50] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. [Accessed 26 November 2025]

[51] He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” arXiv, 2015. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.1512.03385.

[52] Kritik Seth, "Fruits and Vegetables Image Recognition Dataset," [Online]. Available: https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition. [Accessed 26 November 2025].

[53] Ultralytics, "YOLOv8 Commercial License," [Online]. Available: https://roboflow.com/model-licenses/yolov8. [Accessed 26 November 2025]

[54] Shi, Yucheng, et al. “Enhancing Cognition and Explainability of Multimodal Foundation Models with Self-Synthesized Data.” arXiv, 2025. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2502.14044.

[55] Samir Bhattarai, "New Plant Diseases Dataset," [Online]. Available: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset. [Accessed 26 November 2025]

[56] BoltzmannBrain, "Numenta Anomaly Benchmark (NAB)," [Online]. Available: https://www.kaggle.com/datasets/boltzmannbrain/nab. [Accessed 26 November 2025]

[57] Cheng, Daixuan, et al. “On Domain-Adaptive Post-Training for Multimodal Large Language Models.” arXiv, 2024. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2411.19930.

[58] Beijing Institute for General Artificial Intelligence , "QA-Synthesizer," [Online]. Available: https://github.com/bigai-ai/QA-Synthesizer. [Accessed 26 November 2025]

[59] DeepSeek-AI, et al. “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.” arXiv, 2025. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2501.12948.

[60] Darshani Persadh, Research, Innovation & Development, DARJYO, "sawotiQ29 Crop Optimization Dataset" [Online]. Available: https://huggingface.co/datasets/DARJYO/sawotiQ29_crop_optimization. [Accessed 26 November 2025]

[61] Hu, Edward J., et al. “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, 2021. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2106.09685.

[62] IBM, " What is transfer learning?," [Online]. Available: https://www.ibm.com/think/topics/transfer-learning. [Accessed 26 November 2025].

Appendix

A Models Retrieval Script

The Python script used to compile the list of models is available at the following link https://github.com/lPen88/tesi/blob/main/fetch_script.py

It was executed on September 23, 2025; as such, any discrepancies in the retrieved model list during future reproductions may result from changes in download counts over time.

The terms used to index the models to be retrieved are the following:

- "agriculture", "crop", "biomass", "yield", "nitrogen", "phosphorus", "soil", "plant", "drone", "remote sensing", "time series", "field", "tomato", "heatmap", "organism", "vegetable"

Some specific terms used, such as “tomato”, were selected as Smart Shaped specifically mentioned them in an example use-case of the cham3leon framework [6]; as such, we included them in order to fully represent the showcased capabilities of the project.

B Non-Relevant Models

The full list of models can be found on the online spreadsheet linked earlier in chapter 3 [12], below we point out some highlights that can be derived from the list.

- A large portion of the discarded models (34.12%) are all part of the OpenMed Named-entity recognition (NER), a suite of open-source transformer models, developed by OpenMed for the purpose of identifying biomedical entities contained in clinical texts, research papers and other healthcare-related documents. 29 of these models were retrieved as their model cards contained the word “organism”.
- 8 models were developed by Plan de Tecnologías del Lenguaje (PlanTL), a government-owned Spanish company, with different intended purposes, such as removal of personal information from documents, text classification and more. They were retrieved as their model cards contained the abbreviation “PlanTL”.
- 5 models were trained to perform modeling of genomic sequences of plants; after an interview with Smart Shaped, we determined this to not be relevant to Farm-Tech.