Plant Identification with Computer Vision and Improvement Through Location Data

Master's thesis in Statistics

Vegard Josvanger

Supervisors

Hans Arnfinn Karlsen

Department of Mathematics

University of Bergen

December 2023

Preface

This is the Master's thesis of my Master of Statistics and Data Science degree at the University of Bergen. The work on this thesis was done from August 2023 to December 2023.

I would like to thank my supervisor Hans Karlsen at the Department of Mathematics at UiB for his help and guidance, as well as my partner Katarina Sandbakk for her patience.

Vegard Josvanger
December 2023
Oslo

Sammendrag

Det å kunne riktig identifisere arter er et veldig viktig verktøy i kampen for å bevare artsmangfoldet i verden, men dette er kunnskap som er enten forbeholdt eksperter, eller så er det veldig tidskrevende for ufagkyndige. I nyere tid har fremskritt i teknologi gjort kameraer med god kvalitet blitt tilgjengelig for de aller fleste gjennom mobiltelefon samtidig så har nyutvikling innenfor GPUer og maskinlæring gjort bildegjenkjennings teknologi tilgjengelig for de fleste gjennom forskjellige apper. Gjennom slike apper kan lekfolk også bidra til å bevare artsmangfoldet.

Dette prosjektet beskriver prosessen med å lage et plantegjenkjenningsmodel og undersøker mulighetene for å forbedre modellen ved å inkludere lokasjonsdata i modellen.

Samlet bilder og lokasjons data fra nettsiden artsobservasjoner.no samt andre dataset. Trente forskjellige "computer vision" modeller på bilder av de 100 mest fotograferte artene hos artsobservasjoner. ResNet-101 modellen oppnådde en trefsikkerhet på 70.948%. Ved bruk av relativ frekvens vekting, vekting basert på planter i området og en statistisk modell knyttet til blant annet landskapstypen, høyde over havet og koordinat økte modellen sin treffsikkerhet med 3.964% opp til 74.912%.

Abstract

Recent development in technology has led to both ubiquitous camera-phones and high quality image recognition software. One of the areas this is useful is species-conservation where accurate plant identification through cellphones is an invaluable tool.

Introduced Relative Frequency Weighting, weighting based on the surrounding plants and statistics based on coordinates, landscape and altitude of the image location in order to improve the plant species identification accuracy.

This increased the accuracy of the deep learning computer vision model between 3.5 and 4.0%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

25% of all species are threatened by extinction[1]. In order to combat this species knowledge is essential. Acquiring this knowledge both difficult and time consuming. Thankfully new technology has made this task a lot easier. This is among others, due to new technology within image recognition and machine learning. This has made it possible to identify species quickly without using time-consuming field guides or identification keys[2]. Proper mapping and identification of species is crucial in halting and reversing biodiversity loss[1] and this has made it possible for the common man to contribute to preserving the local biodiversity.

In this project I will both create a program for plant recognition with collected data and see if it is possible to improve the plant image recognition through the use of location data. This is done with minimal input from biologists which is a big oversight, but since this project is focusing on larger areas and common species it will hopefully not become to detrimental to the project.

In this project I will collect data from various online sources both easily available and datasets collected ourselves through web scraping. Will the process and collate the data to be used in an image recognition model. Will then try to improve the model buy including location data through various statistical methods.

Chapter 2

Theory

In this chapter most of the mathematical theory behind the statistical methods used in this thesis is presented. In addition to this, several computer vision models are presented as well as the some theory behind how they are trained/fitted to the data.

2.1 Classifications methods

2.1.1 Multinomial Logistic Regression

When there is no natural ordering in a categorical response variable Y, we can use multinomial logistic regression to predict the categories. One category is chosen as the reference category π_1 and the logit function for the remaining categories is given by:

$$\text{logit}(\pi_j) = \log\left(\frac{\pi_j}{\pi_1}\right) = \beta_{0j} + \beta_{1j}x_{1j} + \ldots + \beta_{Nj}x_{Nj} = \mathbf{x}_j^\intercal \boldsymbol{\beta}_j \quad \text{for j} = 2,\ldots, \text{J} \tag{2.1}$$

Where π_j is the probability of category j and N is the number of covariates in the model [3], we use the equations (2.1) to estimate the parameters β_i as b. The probability is then given by:

$$\hat{\pi}_j = \hat{\pi}_1 \exp\left(\mathbf{x}_j^\intercal \boldsymbol{b}_j\right) \quad \text{for j} = 2,\ldots, \text{J} \tag{2.2}$$

And since $\sum_{j=1}^{J} \hat{\pi}_j = 1$, $\hat{\pi}_1$ is given by:

$$\hat{\pi}_1 = \frac{1}{1 + \sum_{i=2}^{J} \exp\left(\mathbf{x}_i^\intercal \boldsymbol{b}_j\right)} \tag{2.3}$$

We can write the probability equation as:

$$\hat{\pi}_j = \frac{\exp\left(\mathbf{x}_i^\intercal \boldsymbol{b}_j\right)}{1 + \sum_{j=2}^{J} \exp\left(\mathbf{x}_i^\intercal \boldsymbol{b}_j\right)} \quad \text{for j} = 2,\ldots,\text{ J} \tag{2.4}$$

Then, using the multinomial logistic model as a classification method, the class is assigned according to its highest probability.

2.1.2 Support vector machine(SVM)

A Support vector machine(SVM) is a supervised learning model which can be used for classification [4]. It works by having a p-dimensional hyperplane and it assigns categories depending on which side of the plane the data point is located. The hyperplane is defined by the equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots \beta_p X_p = 0 \tag{2.5}$$

If $\boldsymbol{\beta}\mathbf{X} > 0$ the point $\mathbf{X}$ is on one side of the plane and and if $\boldsymbol{\beta}\mathbf{X} < 0$ the point is on the other. This is used in classification where we label the observations $y_1 = 1$ and $y_2 = -1$. A hyperplane is then created between them so that the perpendicular distance between the points on each side and the hyperplane(margin) is as large as possible. This hyperplane is created by solving the maximization problem:

$$\max_{\boldsymbol{\beta}, M} M \tag{2.6}$$

$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.7}$$

$$y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots \beta_p X_p) > M \quad \forall \quad i = 1, \ldots, n \tag{2.8}$$

Where M is the margin. The observations which lie on the margin are called support vectors. It is not always possible to separate the training set classification by a single hyperplane. To account for this we can introduce a soft margin where we allow the observations to cross the margins and even be on the wrong side of the hyperplane. This transforms the optimizatiomn problem to:

$$\max_{\boldsymbol{\beta}, \epsilon_1, \ldots, \epsilon_n, M} M \tag{2.9}$$

$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.10}$$

$$y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots \beta_p X_p) \geq M(1 - \epsilon_i), \tag{2.11}$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \leq C \tag{2.12}$$

Where C is a non-negative tuning parameter and $\epsilon_1, \ldots, \epsilon_n$ are the slack parameters. The slack parameters are what allows the observations to be on the wrong side of the margin and also possibly the separating hyperplane.

If the classes are not separated by a linear border/hyperplane, we may handle this non linearity by expanding the feature space through the inclusion of terms such as quadratic, cubic or other kernel functions[4]. Kernel functions are functions dependent on the inner product of the observations. The inner product is given by:

$$\langle\, x_i, x_{i'} \,\rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{2.13}$$

The classification function then becomes :

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_{i'}) \tag{2.14}$$

Where K is the kernel function. The Kernel function can be the inner product (2.13) by itself, which is the linear kernel. The polynomial:

$$K(x, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \tag{2.15}$$

is the polynomial kernel when $d > 1$ and the radial kernel is given by:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{2.16}$$

Where γ is a positive constant and the Radial kernel can be shown to be represented as a function of an inner product [4]. In order to use this classification technique when there are more than two categories we have two options: one vs one or one vs all.
One vs One:
Creates $\binom{n}{2}$ classifiers, one for every pair of classes and count the number of classifications in each class.
One vs All:
Create n SVM classifiers, where each classifier classifies an observation between the n-th class and the remaining $n-1$ classes. The class with the highest value, and therefore largest confidence, gets the classification.

2.1.3 k-Nearest Neighbors

k-Nearest Neighbors(k-NN) is a non parametric classification algorithm which aims to classify the observation based on the K nearest training observations. Where the estimated probability of class j given sample X_0 is:

$$P(Y = j|X_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{2.17}$$

Where $\mathcal{N}_0$ are the K closest points. Choosing a small K gives an overly flexible classifier with a low training error and often a high test error while high K gives a classifier that is not flexible enough. Therefore the K value needs to be tuned appropriately[4].

SMOTE

k-NN on an imbalanced data set will always prioritize the majority class of the samples and therefore have a poorer performance on the minority class. This can be solved by either undersampling the majority class or oversampling the minority class. One of the ways to oversample is duplicating data-points in the minority class. This has the downside of being prone to overfitting[5]. Another way is to synthesize new points from the existing points. One such method is called Synthetic Minority Oversampling Technique(SMOTE)[5]. SMOTE works as follows:

1. Sample a point and find its k nearest neighbours(default 5) in the original minority class.
2. Calculate the difference between the sampled point and the nearest neighbours.
3. Multiply the difference with a random number between 0 and 1.
4. Add the multiplied difference to our initial sample point and add a synthetic sample.
5. Repeat until desired amount of synthetic points are created.

SMOTE can be beneficial for handling unbalanced data sets, but is often outperformed by undersampling the majority classes[6].

2.1.4 Kernel Density Estimation

Kernel Density Estimation(KDE) [7] is a generalisation of Histogram Density Estimation. In the Histogram Density Estimation the density is estimated with the formula:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} w\left(\frac{x - x_i}{h}\right) \tag{2.18}$$

Where the h is the bucket width and the weights w are given by $w(x) = \frac{1}{2} I(|x| < 1)$. In the kernel density estimator the uniform weights are replaced by a general weight function:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h\left(x - x_i\right) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.19}$$

The general weight function has the property $\int_{-\infty}^{\infty} K(x)dx = 1$. In this thesis we use the Gaussian kernel:

$$K(x, x_i) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(x - x_i)^2}{2}\right) \tag{2.20}$$

Here the bandwidth value h decides the smoothness of the estimation. A low bandwidth gives more local features and variation while a large bandwidth produces a smoother estimate with less variation.

This can be used in classification by estimating the probability distribution over an area for several classes and classifying a new point according tho the larges probability given by the KDEs.

2.2 Computer vision models

2.2.1 VGG

Visual Geometry Group(VGG) is a neural network based on AlexNet. Simonyan and Zisserman 8 found that by replacing the convolutional layers in AlexNet(11x11) with smaller ones (3x3) the VGG network is able to go deeper and preform better on the ImageNet Challenge. The VGG-16 architecture can be seen in figure 2.1. It consists of 13 layers of either 3x3 convolution layers with Rectified Linear Unit(ReLU) activation and 3 fully connected layers also with ReLU activation. There are also max-pooling layers between every second or third layer. Another model, the VGG-19, has three additional 3x3 convolution with ReLU layers.

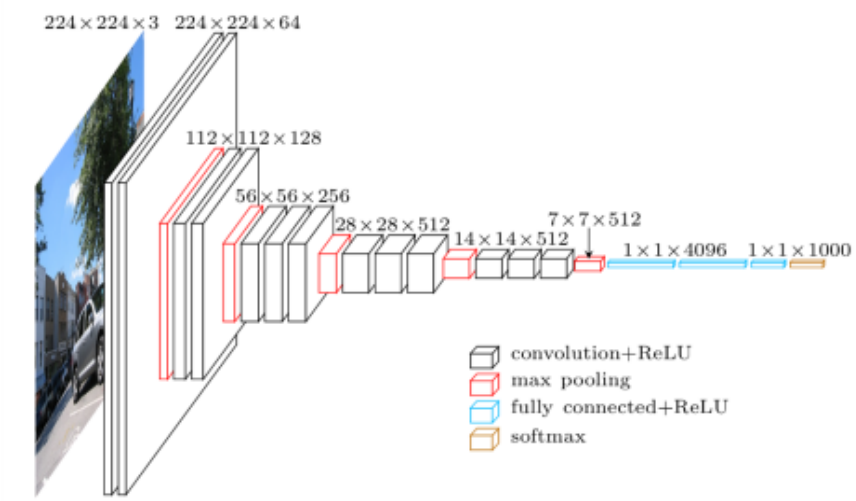


Figure 2.1: VGG-16 architecture. Here two layers of 3x3 convolutions with ReLU activation are followed by a 2x2 max-pooling layer with stride 2. This is repeated once. Then repeated twice is three 3x3 convolution layers with ReLUs followed by a max-pooling layer. Finally the model haves three fully connected dense layers with ReLU activation and a final softmax classification layer. Figure form "Very deep convolutional networks for large-scale image recognition"[8]

2.2.2 Residual Neural Network

Residual Neural Network also known as ResNet are deep learning networks with shortcut identity mappings[9].

This type of network was created to address the higher training and test error which appeared when increasing the network depth past a certain point. The deeper models should be able to preform as well as the shallower models by learning the identity function on their additional layers. In practice this is hard to do and resulted in a higher error overall. To solve this He et al. 9 proposed a skip connection(residual connection) between layers which preserves the identity connection and allows the layers to fit a small perturbation rather than the identity function. Such a connection can be seen in Figure 2.2. Experiments points toward that learned residual functions in general have small responses and are easier to optimize. Making it possible to gain accuracy from the increased network depth.

2.2.3 Vision Transformer(ViT)

Since the Vision Transformer(ViT) is adapted form the deep learning architecture Transformer, it is useful to briefly discuss the encoder of the Transformers architecture.

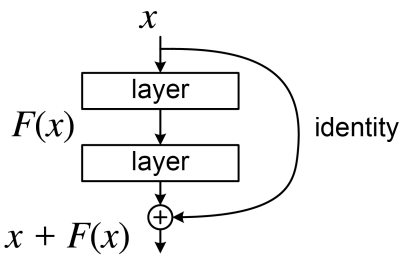


Figure 2.2: A ResNet block where the input x is passed to the function $F(\cdot)$ and the output is added to x through the residual(identity) connection. Figure from Wikipedia released under the CC BY-SA 4.0 license [10]

Transformer

Transformers were introduced in 2017 in the article "Attention is All You Need " by Vaswani et al. 11 and has gained a lot of popularity especially within Natural Language Processing(NLP), where it is used in tasks such as translation and text generation. The Transformer architecture can be seen in figure 2.3

Encoder

The input is first transformed to vectors in an embedding space. This can for example be a way of grouping words with similar meanings. The embedding vectors are then given positional information by a positional encoder. The input is then passed through N blocks containing a residual(skipped connection) Multi-Head Attention layer and a residual Feed Forward Layer. The Multi-Head Attention layer creates vectors containing correlation between the different parts of the input. This is done in parallel for all parts of the input and the output and the result is called an attention vector. The attention vector is an average of 8 different vectors with different attention(correlation) measures [11]. The attention vectors are then passed through a feed forward layer and after the N blocks the encoder output is passed to the decoder.

Vision Transformer

A Vision Transformer works in a way similar to that of a Transformer. First the input image is divided into fixed size patches, given a positional embedding and passed through a linear projection layer. The linear projection layer is sometimes implemented with a Convolutional Neural Network(CNN)-module as in PyTorch's Vision Transformers. After being passed through the linear projection layer the patches are passed through N transformer encoder

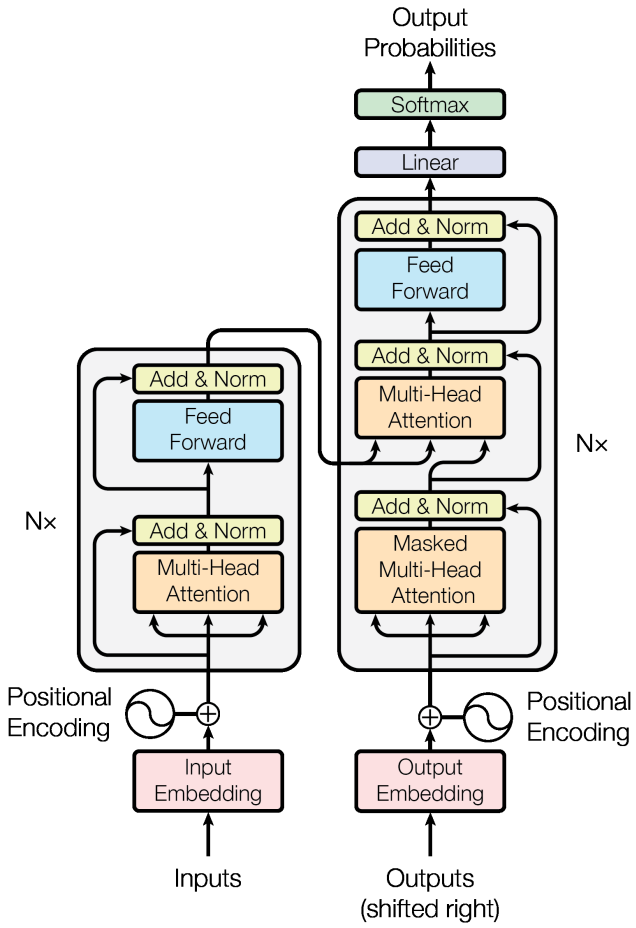


Figure 2.3: Transformer architecture. Figure from "Attention is all you need"[11]

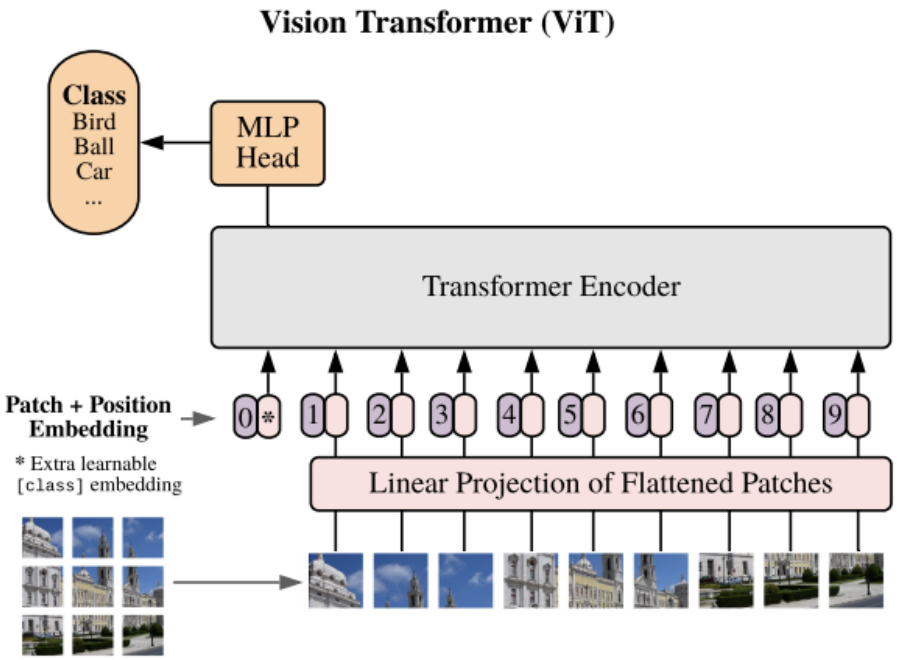


Figure 2.4: Visual Transformer Architecture. The input image is split into patches and passed to a Transformer Encoder. The Transformer Encoder has the same structure as the input encoder in figure 2.3. Figure from "An image is worth 16x16 words: Transformers for image recognition at scale" [12]

blocks, which works the same way as with the NLP Transformer by generating attention vectors for the image patches. The vectors are then passed to Multilayer Perceptron(MLP) through which a final classification is performed[12]. The Visual Transformer architecture can be seen in Figure 2.4.

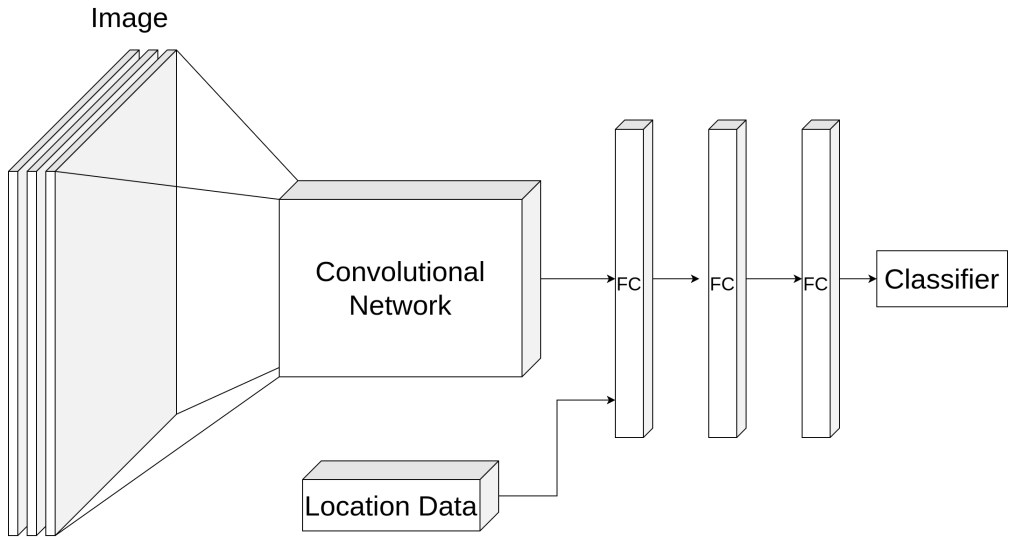


Figure 2.5: Sketch of neural network architecture where image data is combined with location data. Sketch created with the web application diagrams.net.

2.2.4 Combining image and location data

Combining image data and a location data in a neural network can be done by adding location information to the first fully connected layer after the convolutional layers of our model as seen in figure 2.5. A similar setup was used by Ghafoorian et al. 13, and documented in the article "Location Sensitive Deep Convolutional Neural Networks for Segmentation of White Matter Hyperintensities", where they found that CNNs that incorporated location information outperformed the CNNs without location information in some medical image analysis tasks.

2.2.5 Hyper parameters

Objective function/loss function

In order to compute the best parameters for a Deep Learning model we need to choose a loss function to minimize. The best parameters are the ones which solve this optimization problem. The neural network output can be made summable to 1, hence it can be interpreted as a probability distribution. This is very useful for multi-class classification tasks and can be done by applying the softmax function to the output[14]:

$$\sigma(x)_i = \frac{\exp x_i}{\sum_{j=1}^{C} \exp x_j} \tag{2.21}$$

Where $\sigma(x)_i$ is the softmax output for class i. Which is the exponential of output i divided by the sum of the exponential of all the outputs[15].

To compute the difference between the softmax output and the correct classification we use the Cross Entropy function:

$$l(x, y) = \frac{1}{N} \sum_{n=1}^{N} l_n, \quad l_n = -\log \left(\frac{\exp x_{n,y_n}}{\sum_{c=1}^{C} \exp x_{n,c}} \right) \tag{2.22}$$

Where l_n is the negative log value of the softmax function for the output x_{n,y_n} in the correct target class y_n. $l(x, y)$ is the mean loss over a batch of size N[16] and is the loss function we want to minimize.

Stochastic Gradient Decent

The Gradient Decent Algorithm is an iterative algorithm for finding the local minimum of a differentiable function:

$$\theta_t = \theta_{t-1} - \gamma \nabla F(\theta_{t-1}) \tag{2.23}$$

Where θ is the parameters/weights of the model, γ is the learning rate and F is the objective function we are trying to minimize. In this case the objective function is the Cross Entropy function. In order to use this algorithm on a machine learning model, it is necessary to use all training data in calculating the gradient $\nabla F(\theta_{t-1})$. This is very inefficient and computationally expensive. The Stochastic Gradient Decent(SGD) algorithm approximate the gradient decent by using the expected gradient based on a small set of samples. m'

samples are drawn uniformly form the training set and the gradient estimate is then given by [15]:

$$g_t = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} F(x_i, y_i, \theta_{t-1}) \tag{2.24}$$

In order to prevent overfitting we introduce weight decay:

$$g_t = g_{t-1} + \lambda \theta_{t-1} \tag{2.25}$$

Where λ is the weight decay coefficient. This introduces a penalty to the weights θ which should prevent them from becoming to large and start overfitting.

Momentum is introduced to accelerate learning where, among others, there is high curvature, small consistent gradients or noisy gradients. It mirrors the physical analog where an object with mass and velocity continues in its general direction and takes time to stop or change its direction. This gradient is given this property by accumulating exponentially decaying moving averages [15]:

$$b_t = \mu b_{t-1} + (1 - \tau) g_t \tag{2.26}$$

Where b_t is the momentum from the previous time step and τ is the dampening factor. Another way of increasing the rate of convergence is to use the Nesterov momentum:

$$g_t = g_t + \mu b_t \tag{2.27}$$

The Nesterov momentum tries to estimate the current momentum at t instead of the momentum at $t-1$. This is done by adding a correction factor to the momentum calculated at $t-1$. The Nestrov momentum does not necessarily increase convergence, but might under certain conditions[15].

Finally we have the updated model parameters:

$$\theta_t = \theta_{t-1} - \gamma g_t \tag{2.28}$$

Where γ is the learning rate, indicating how far in the g_t direction to move. The learning rate is often set to decrease in training after a certain time or because of lack of progress.

2.2.6 Training

The goal in training a model is for the model to be able to recognize the general patterns of the objects it is being trained on and to recognize these patterns in new unseen data. This ability is called generalization. When training the model the data is split into training and test data. The model is trained on the training data to minimize the error on them, training

error, and the model is then tested on the test data and the generalization error, or test error, is used to evaluate the model and choose the optimal model parameters.

The two main challenges when training a model are underfitting and overfitting. Underfitting is when the model is not trained enough to pick out the general pattern in the training data. This can be mediated by either training the model for longer or by switching to a model with a higher capacity. A model with a higher capacity generally means a more complicated model with more parameters, which can fit the model better. Overfitting on the other hand, is when the model is trained past the point where it learns the general pattern, to where it also learns the specific pattern in the training data. This is characterized by a widening of the generalization gap where the training error decreases and the generalization error increases, as seen in figure 2.6. We can control overfitting by choosing the parameters for the model at the point where the generalization gap is at its lowest. In addition to this we can introduce weight decay, as in equation (2.25) and dropout. Dropout is a practice where we randomly remove non-output units for the model[15]. The effect of this is analogous to bagging, where multiple models are trained and evaluated on every test input and their collective output is used as the final prediction. This is very time consuming for large models so dropout is a reasonable alternative.

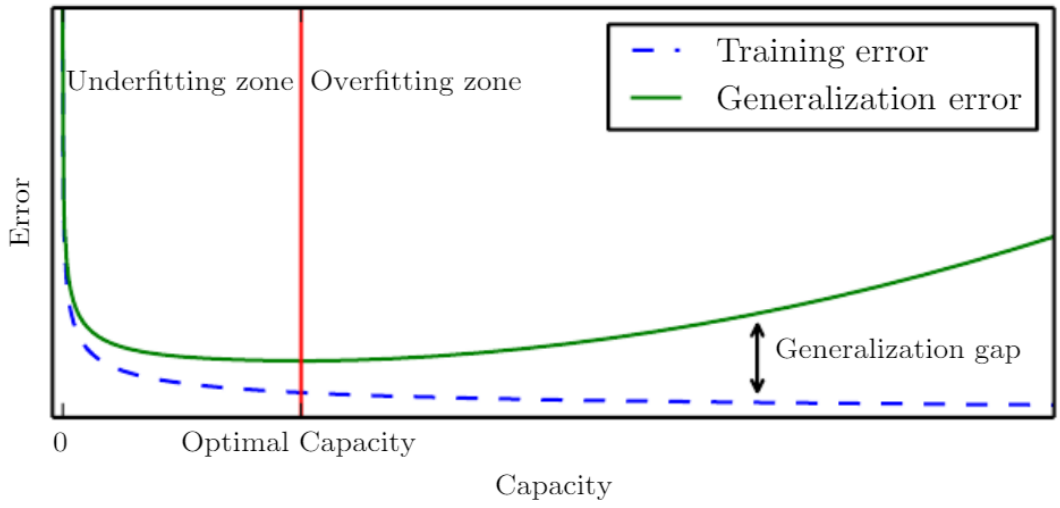


Figure 2.6: Illustration of the Generalization gap during training. Where the Generalization gap widens as the capacity of the model increases. Figure from the book Deep Learning by Goodfellow [15]

2.2.7 Metrics

When evaluating model performance it is important to choose the most suitable performance metric.
Accuracy:

$$\frac{\text{True Predictions}}{\text{Total Predictions}} \tag{2.29}$$

is often a suitable metric for model evaluation, but when dealing with an unbalanced data-set accuracy is often not the optimal metric. E.g if the model is designed to find outliers such as images with disease or bot-detection in social networks the model can achieve high accuracy by predicting false for every instance without predicting any positives.

When the predictions correctness is the primary interest precision is a good metric.
Precision:

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.30}$$

When detecting the True Positives, it is more important than their correctness, we can use recall.
Recall

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.31}$$

Precision and Recall can be opposed to each other, where high Precision can lead to low Recall and vice versa. The performance of the model, when using Precision and Recall, can be expressed through a Precision-Recall curve or the F-score[15]:

$$\frac{2 \cdot \text{ Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.32}$$

Chapter 3

Literature

3.1 Earlier research

In the article "Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review", Waldchen and Mäder present a literature review of 120 articles published between 2005 and 2015. Here, they compare the different studies on publicly available datasets, their methods and accuracies. In the majority of the articles studied, leaves were used for identification. A leaf is a "usually green, flattened, lateral structure attached to a stem and functioning as a principal organ of photosynthesis and transpiration in most plants"[17]. Leaves consists of a blade and a small stalk, called petiole, which goes through the blade branching off veins along the way before it connects to the stem. Furthermore Waldchen and Mäder separate leaves into simple and compound leaves depending on whether or not the leaf is undivided or divided into two or more leaflets. In classical manual classification the leaf base, tip and edge of the leaf blade are used to identify the species[17], but automatic methods mostly use the whole leaf and ignore local features.

Most of the studies pertained to scans or pseudo-scans of leaves. Only 25 of the studies used photos taken in a natural environment, as opposed to the images collected through scanning or photographs against a plain background. Among the most used publicly available datasets are the Flavia and LeafSnap Dataset. These datasets will be presented further in Chapter 4.

For most classification based on leaf or flower analysis several features are extracted from the scans. Depending on the study the features could be features connected to shape, texture and color. Shape features include area, diameter, aspect ration and image moments.

The texture features are among others given by Scale-invariant feature transform(SIFT), Fourier Transforms and co-occurrence matrices and the color features are for example color moments, Color Scale-invariant feature transform(CSIFT) and color histograms.

To classify the leaves and flowers based on the scans/pseudo-scans the methods SVM, k-NN and Neural Networks are used. Achieving an accuracy between 25.3% and 97.8% on the Flavia dataset with SVMs and k-NN, using various extracted features. While the Neural Networks achieved between 87% and 96%[17]. On the LeafSnap Dataset k-NN method reached 58.51% and SVM reached 72.64% [18].

In recent years (after 2015), Deep Learning methods have outperformed the more classical classification methods on the benchmark datasets [19]. Using computer vision models the accuracy on the Flavia set increased to 99.7% [20] and the LeafSnap accuracy increased to 97.6% [18]. The computer vision models also removed the need for manual feature extraction, which was necessary in classical classification. The neural networks automatically determines the features which are suitable for classification. This removes a process which is labor intensive and requires expert knowledge[21]. Better results, ease of use and the increase in computational power through the use of GPUs seems to have led to Deep Learning models to be the dominant classification tool used on plant images, as well as images in general.

3.2 Free Automated Plant Applications

Along with increasing accuracy from machine learning, widely available mobile phones with cameras and the fact that plant identification is both useful and interesting has led to the development of several free and commercial plant identification applications. Some of these applications were tested by Hart et al. and the results are presented in the article "Assessing the accuracy of free automated plant identification applications". Among the tested applications were the "PlantNet" and "iNaturalist Seek" applications which reached an accuracy of 95% and 93% respectively on 857 professionally identified images of 277 different plant species.

3.2.1 PlantNet

The PlantNet application(stylized as Pl@ntNet) is a part of the The Pl@ntNet Project which is a cooperation between the french research institutes CIRAD, INRAE, INRIA and IRD. They have released an application for identifying plant species. The PlantNet model takes an input consisting of the plant in question and a tag specifying what part of the plant it is an image of(leaf, flower fruit, bark, habit or other), as seen in Figure 3.1. The model then

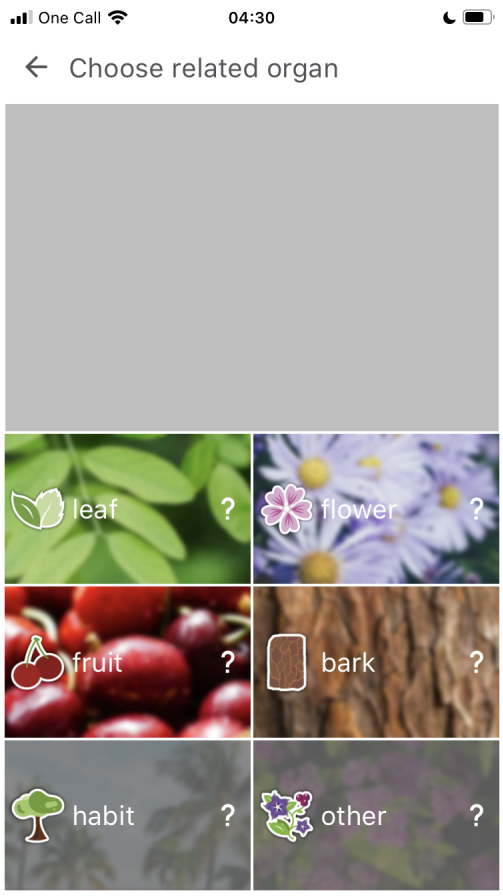


Figure 3.1: Screenshot taken form the PlantNet app where a picture is given a tag for easier discrimination.

classifies the image between 10 000 different species(2017[22]) and a rejection class indication that the object is not a plant. In 2017 the CNN model used was based on the inception model. This model also checks the proposed species against plants observed in the region of the mobile device location, eg West Europe, North Africa, North America etc.

3.2.2 iNaturalist Seek

iNaturalist Seek is an application developed by iNaturalist for plant and animal identification. iNaturalist is an American nonprofit social network of naturalists, citizen scientists, and biologists. They observe and map species all over the world. iNaturalist Seek is an application for species identification, while iNaturalist is both a website and an application where species identification and registration can be done with automatic methods and with help from other users. The species includes plants, animals and fungi, as well as other organisms. iNaturalist species recognition model uses the Xception architecture as a basis and the models predictions are weighted by the iNaturalist geomodel.

iNaturalist Geomodel

Until recently (21.09.23) iNaturalist used a Seen-Nearby Geomodel where the relative number of observations in the surrounding area is used to weight their Computer Vision model. The surrounding area is given by a 3x3 grid. Where each grid is a 1-degree lat long square. This model has since be replaced with an Expected-Nearby Geomodel where the species geographical range is estimated from sparse observations [23]. This switch improved the the top-1 suggestion accuracy 4%. Where the Seen-Nearby Geomodel improved the Top-1 accuracy from 75% to 83% and the Expected-Nearby Geomodel increased Top-1 accuracy to 87%.

Figure 3.2: Image from https://www.inaturalist.org/blog/84677-introducing-the-inaturalist-geomodel demonstrating iNaturalist Geomodel[24]. Where the Sceloporus consobrinus(southern prairie lizard) and Sceloporus occidentalis(western fence lizard) is weighted according to the expected species nearby.

3.3 Bias in collected Data

Most of the data used in this project is collected by citizen scientists and not professionals within the relevant fields such as biology and data science. This is reflected in the data

through the reliability of the identification and the over-representation of easily recognizable and charismatic species [25]. This also causes species which are difficult to identify to be under-represented in the Data.

Chapter 4

Data Sources and Data Collection

In this chapter the Data used in this thesis is presented, as well as the data collection methods and processing pipeline used on the collected data.

4.1 Data Sources

4.1.1 Species Opservations

Most of the data used for this thesis is collected from the website *Artsobservasjoner.no*. *Artsobservasjoner*(Species observations) is a database where citizens can report any wildlife observation anywhere in Norway. The database is owned and maintained by Artsdatabanken(Species Databank) which is a Norwegian public organization for biological diversity with the mission of providing independent, updated and easily available information about the Norwegian flora, fauna and habitat. The database contains both flora and fauna observations with observation dates and locations of the observations. Some of the observations are also presented with pictures. The veracity of some of the observations are confirmed by the umbrella organization SABIMA (Council for biodiversity) and their members, such as the Norwegian Botanical Association and Norwegian Ornithological Society, but most the observations are not verified. As of May 2020 over 23 million observations have been registered with over 3 million of them being vascular plants observations. A small overview of registrations between 31.07.23 and 08.11.23 in southern Norway can be seen in Figure 4.1.

Two images connected to the same observation of a Sitka spruce(Picea sitchensis) can be seen in Figure 4.2 and location for the observation can be seen in Figure 4.3. Those kinds of observations with pictures is the data collected from *Artsobservasjoner*.

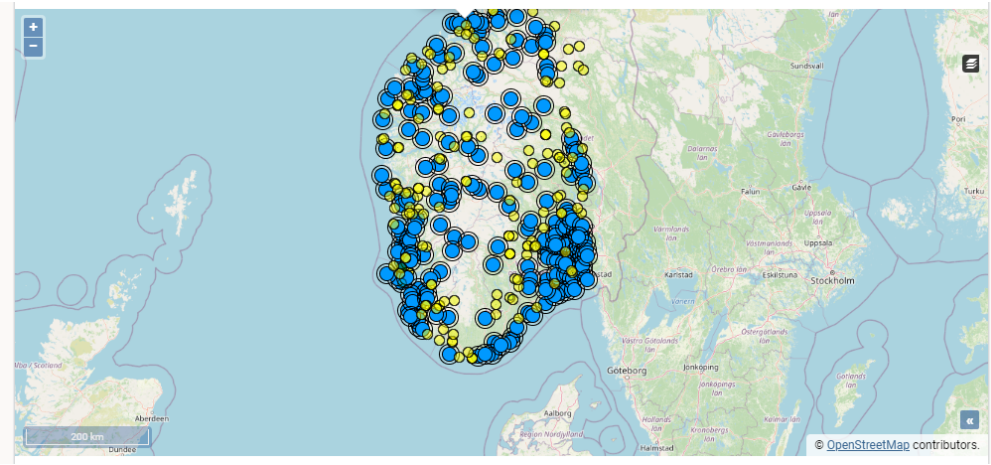


Figure 4.1: Map showing the species observations in Norway from 31.07.23 to 08.11.23. Where yellow circles indicate single observations while blue circles indicates clusters.

Sitka spruce A

Sitka spruce B

Figure 4.2: Two images of a Sitka spruce(Picea sitchensis) taken and registered as an observation at *artsobservasjoner.no* by Snorre Henriksen. Images released under the creative common license.

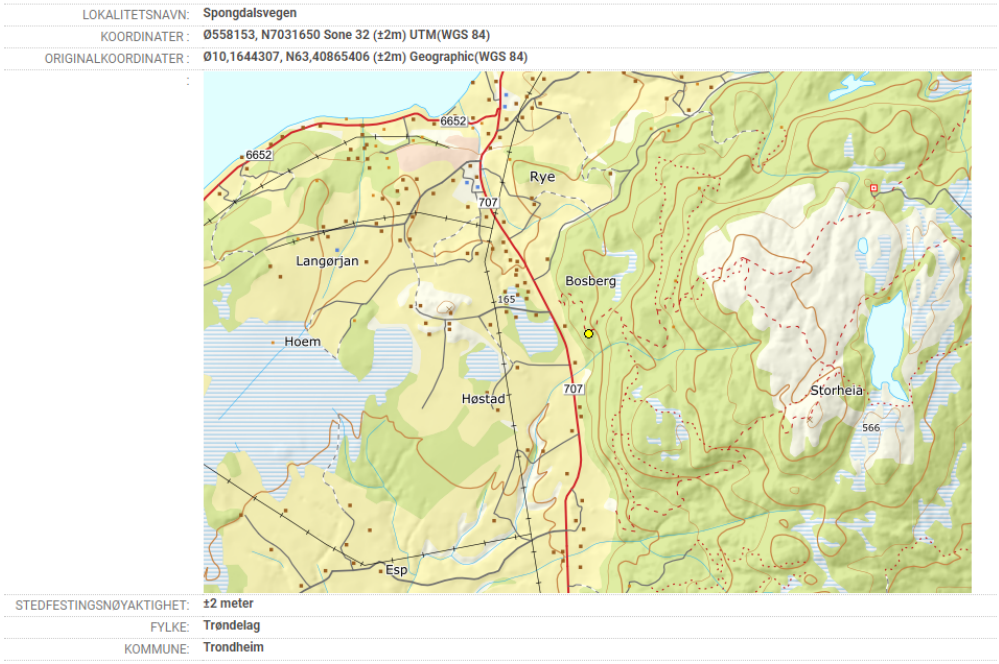


Figure 4.3: Screenshot from *artsobservasjoner.no* . Location registration corresponding to the two Sitka spruce images. Here the location is Spongdalsvegen, Trondheim municipality, Trøndelag county, with UTM-32 coordinates: E558153,N7031650 (±2m),

Elevation data

Based on the locations of the observations we are able to get the elevation of that location from Norwegian Mapping Authority(Statens kartverk). This is done by querying their API. The data is collected by Flying Laser Scanning with 1 meter resolution. Where the elevation data is unavailable from the Laser Scanning the remaining points are estimated using interpolation, usually giving an accuracy between 0.1 and 3 meters, but may be as much as 10 certain places[26].

AR-50

AR-50 is Database/Land resource map of mainland Norway which classifies the areas according to suitability for agriculture and natural plant production [27]. It is collected and maintained by the Norwegian institute for bio-economy(NIBIO) The map classifies each area with the following values:

- ARTYPE - Area type: Buildings, Agriculture, Forrest, Field, Swamp, Glacier, Fresh water or Ocean.

- ARJORDBR - Agricultural Land: Fully cultivated, Infield grazing or Not relevant.
- ARTRESLAG - Forrest type: Conifer forest, Deciduous forest, Mixed forest or Not relevant.
- ARSKOGBON - Forest bonity: High bonity, Medium bonity, Low bonity, Unporductive woodland or Not relevant
- ARVEGET - Field type: Fresh vegetation, Medium fresh vegetation, Lichen covered, Patchy vegetation, No vegetation or Not relevant.

Landscape Data

The Norwegian Species Map Service(NSMS),(Norwegian:*Artsdatabanken*) also has several maps describing landscapes based on their NiN(Nature in Norway, Norwegian: *Natur i Norge*)-system. NSMS separates the Norwegian landscape into three groups with three major categories and within them there are again several sub-categories. The main groups and categories are:

Marine Landscape
- Marine hill and mountain landscape
- Marine valley landscape
- Marine plains

Coastal landscape:
- Coastal hill and mountain landscape
- Fjord landscape
- Coastal plain landscape

Inland landscape:
- Inland hill and mountain landscape
- Inland valley landscape
- Inland plain landscape

The polygons classifying the area as either Marine, Costal or Inland Landscape can be seen in Figure 4.4 and the categories within Inland Landscape can be seen in Figure 4.5.

These landscape types describe larger connections in nature, where nature-systems and landforms and man-made objects are included as elements of the landscape. The goal is to describe landscape variations in the simplest way[28].

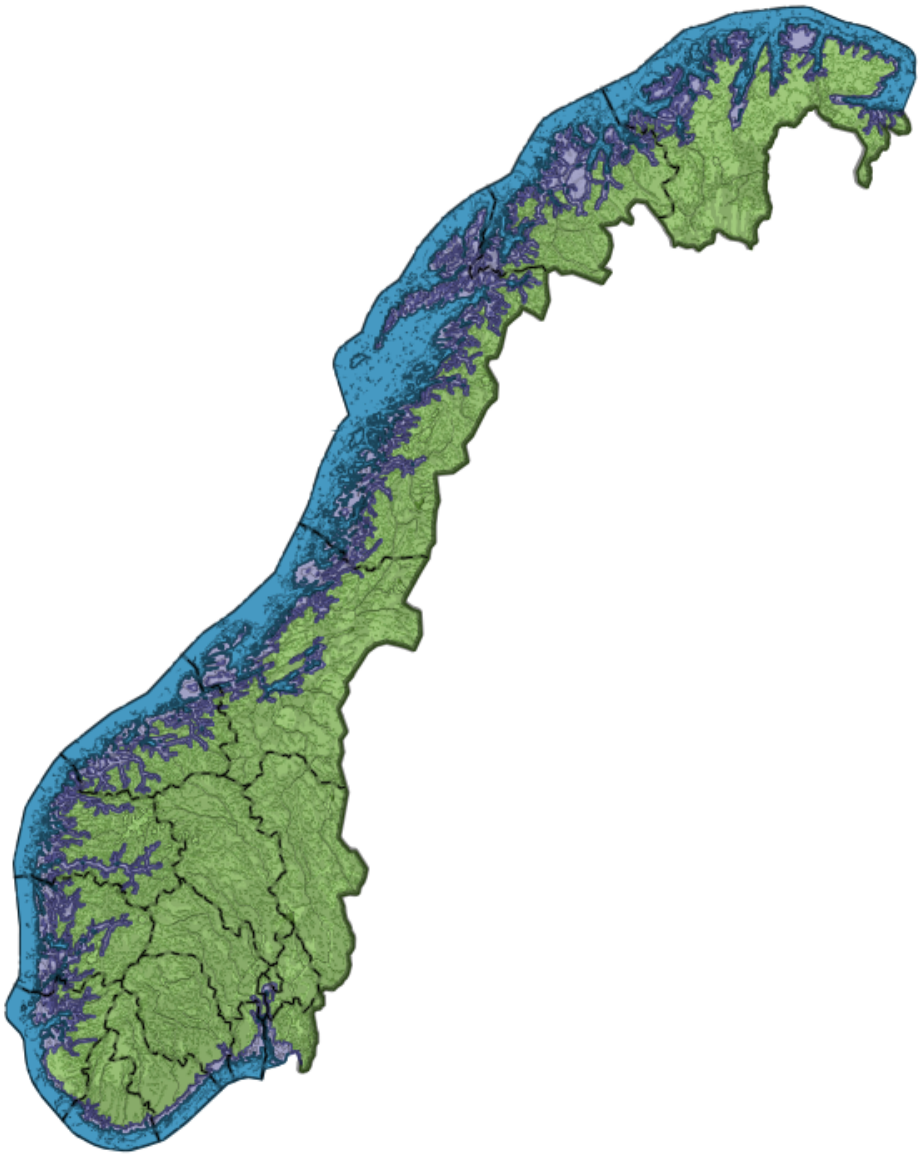


Figure 4.4: Overview of polygons classifying the area as Marine Landscape(Blue), Costal Landscape(purple) and Inland Landscape(Green). Image from The Norway's Species Map Service web-page

4.1.2 LeafSnap

LeafSnap is a data set consisting of leaf images from 185 tree species from north-eastern part of the United States. These images and accompanying segmented images were used in a study from 2012 by Kumar et al[29] to create a system for automatic plant species

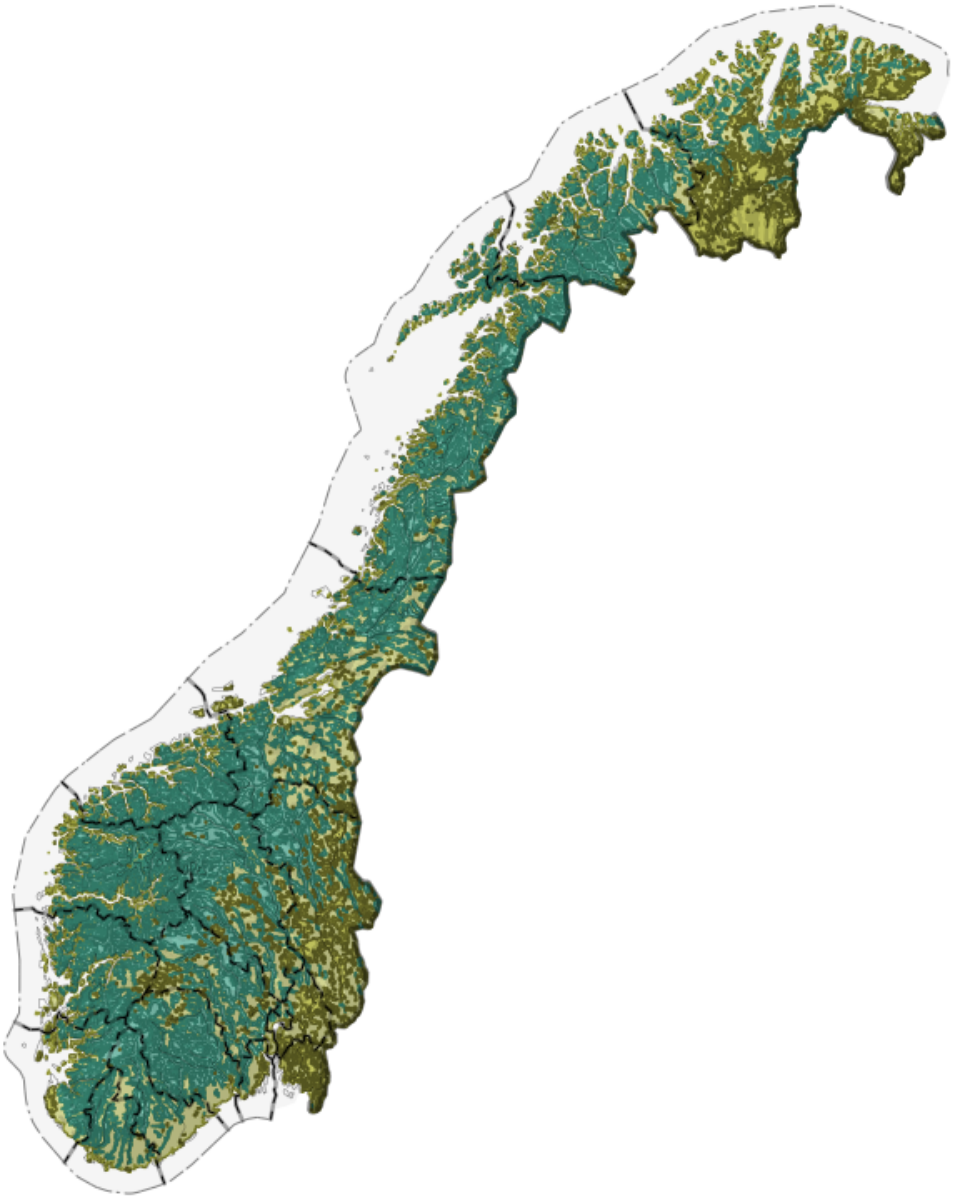


Figure 4.5: Overview of polygons within Inland Landscape characterizing the area further as Hill and Mountain, Valley or plain landscape. Image from The Norway's Species Map Service web-page

identification by extracting morphological features and classifying the images using a combination of the Support Vector Machine(SVM) Algorithm and the K-Nearest Neighbors(k-NN) Algorithm[29]. The data set contains 23,147 "high-quality" laboratory images(scans) of pressed leaves from the Smithsonian collection and 7719 field images taken with cellphones or other mobile devices[17]. The Dataset used in this thesis uses augmented training images(flipped, cropped and/or mirrored) in order to increase the generality of the Model.

4.1.3 Flavia Dataset

The Flavia Dataset contains 1907 images of leaves from 32 different plant species with between 50 and 77 images of each plants leaves. The images are taken with scanners or digital cameras and are collected at Nanjing University and The Nanjing Botanical Garden Memorial Sun Yat-Sen in Jiangsu, China. The images each depict a single leaf on a plain white background[17].

4.2 Data Collection

In order to collect the observation data from *artsobservasjoner.no*, a python pacakage called selenium was used. Selenium WebDriver is a tool for web browsing automation through which web-pages can be navigated and information can be gathered automatically from them. Selenium Python is an API, which allows the user access the Selenium WebDriver. This tool allows the user to collect data much more efficiently.

4.2.1 Location Data

When collecting the location data the WebDriver opens the webpage *https://www.artsobservasjoner.no/ViewSighting/SearchSighting*, selecting the Vascular plants as the *species group* of interest and then selecting the date range from 2000 to 2023, as seen in figure 4.6. Vascular plants(Tracheophyta) are plants in which water and nutrients are transported through a vascular system[30].

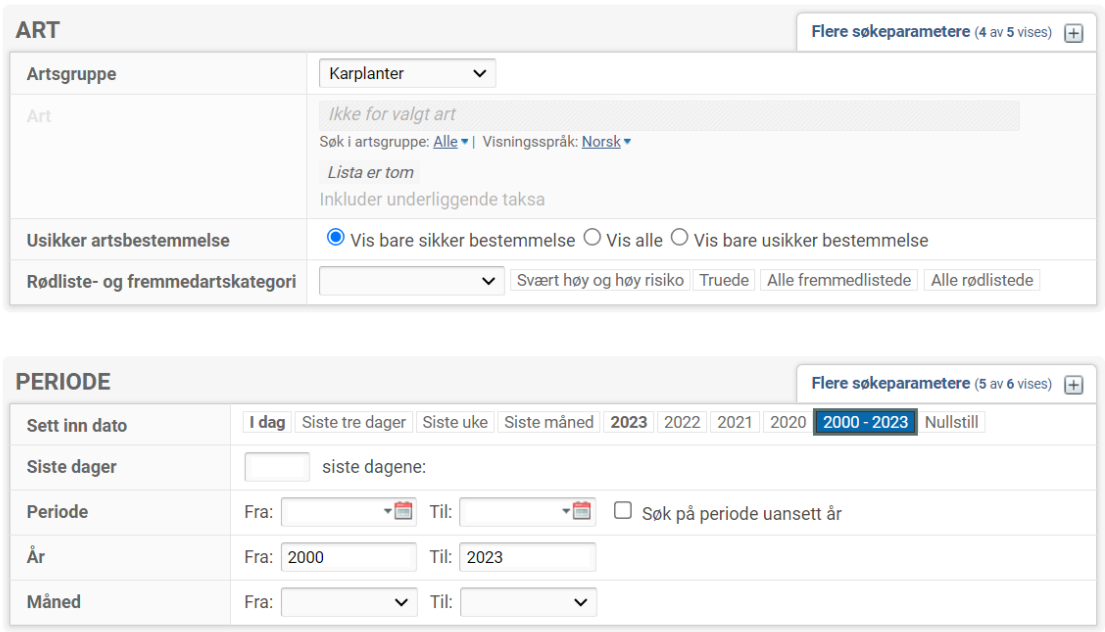


Figure 4.6: Web-interface at *artsobservasjoner.no*. Here selecting the species group Vascular plants (*Karplanter*) for the time period 2000-2023.

Then choosing the output as exporting data(*Eksportere data*), the webdriver maneuvers to a new page. Where the vascular plants observations can be downloaded as Excel-files 4.7. Each Excel-file contains 2000 observations. Using selenium most observations were downloaded as Excel-files before being concatenated to one file. In the image 4.7 there is a total of 3205434 logged observations available. The total number used here is a bit lower due to some of the downloads failing and the screenshot in 4.7 was taken in the middle of November 2023, while the data collection was done early September 2023.

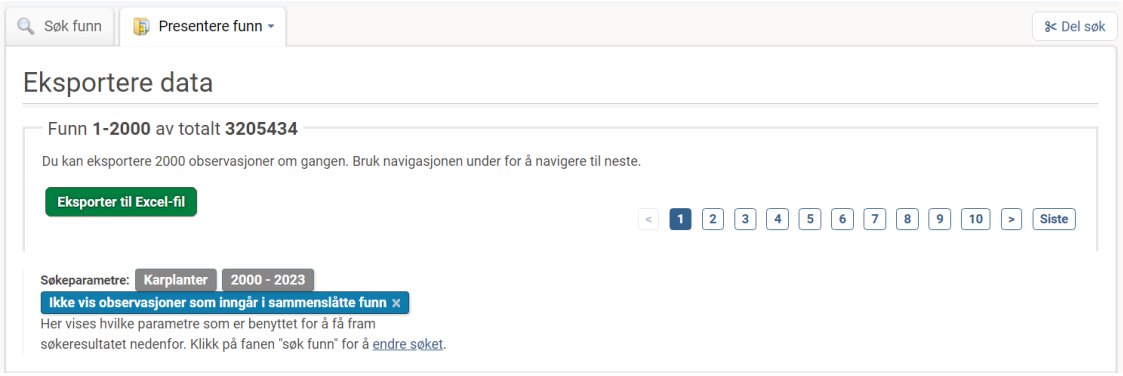


Figure 4.7: Exporting data from *artsobservasjoner.no* to Excel-files for observations of Vascular plants (Karplanter) in the time period 2000-2023.

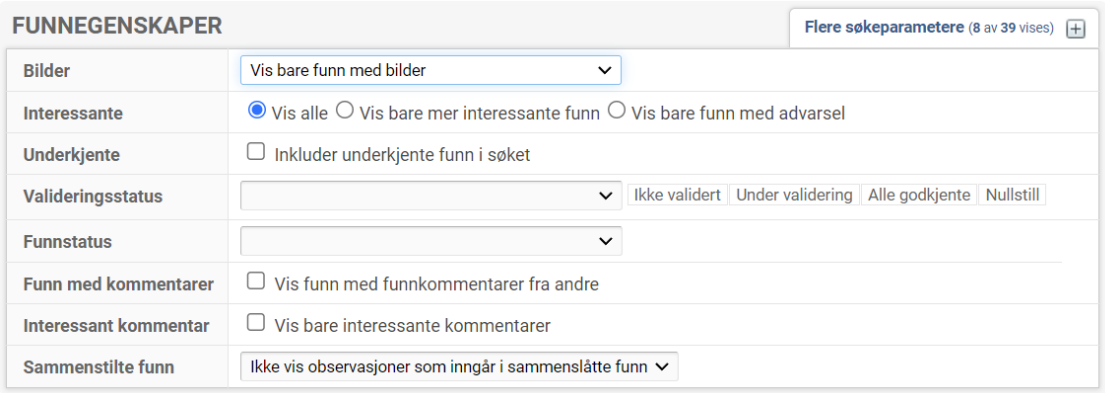


Figure 4.8: Additional search parameters when collecting observations where images have been taken.

This data was used to create three different datasets. The first dataset consists of the locations of the top 185 most registered plant species. This dataset was used in an experiment where LeafSnap species were mapped to the top 185 in the Species-Observation Dataset. This dataset will be referred to as Species-Observation-185 Dataset. The second dataset, the Species-Observation-100 Dataset, is made out of the location data for the 100 most photographed species. The third dataset is a merger of the image data described in the next section and the location data collected, as well as the data collected from AR-50(4.2.4), Elevation(4.2.3) and Landscape data(4.2.5). This dataset will be referred to as the Combined Species-Observation Dataset or Combined Dataset.

4.2.2 Image Data

The process of collecting the logged observation with uploaded images was initially quite similar to the process of collecting the observation data. Choosing Vascular plants(*Karplanter*), as for the location data in section 4.2.1, and the time interval from 2000 to 2023. In addition to this, we include the condition that we only get returned observations registered with a picture. The settings for this search can be seen in figure 4.8.

The Excel-files were collected as described in section 4.2.1 before selecting the 100 most commonly observed plants which were registered with a picture. Each of these observations have an Id which identifies the observation. The pictures connected to the observation can be found at the webpage *https://www.artsobservasjoner.no/Image/*{Id}. Where {Id} is replaced with a specific observation-Id. When accessing the website we use selenium to find the images and download them. Before creating a new data-file where the row connected to the observation is connected to the image-Id and the local computer path to the downloaded image. For observations with more than one image the observation-row is duplicated. Images

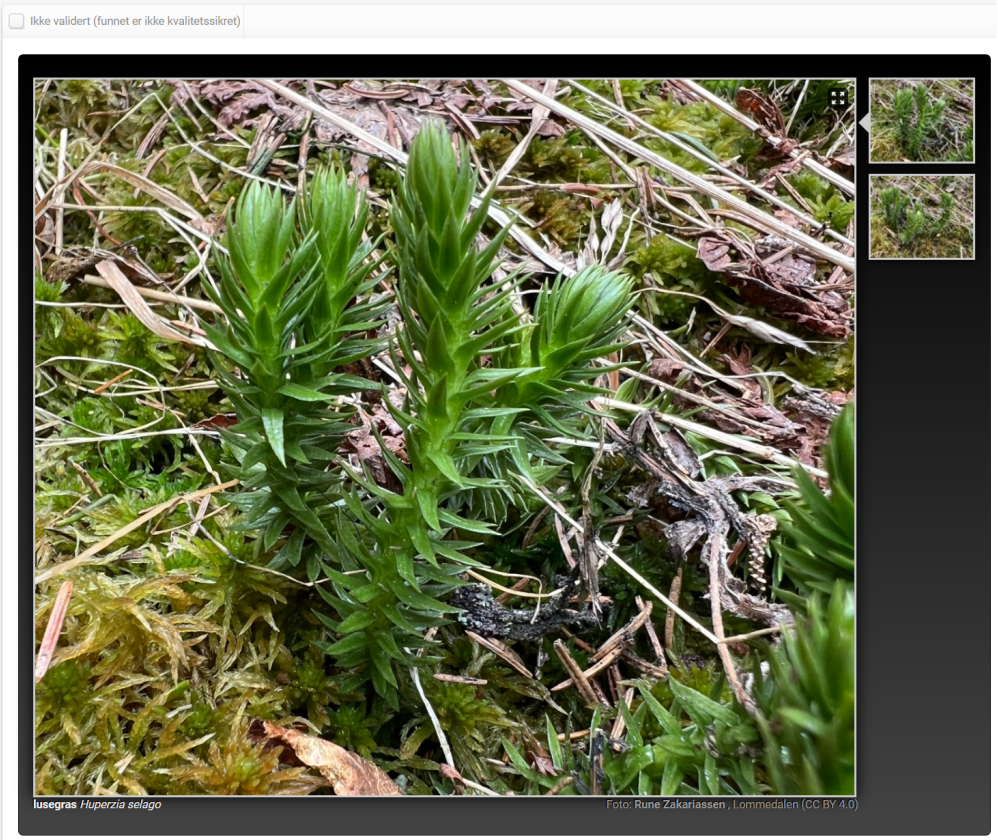


Figure 4.9: Screenshot form the webpage *https://www.artsobservasjoner.no/Image/2444828*. Here we see two images connected to the same observation. This serves as only as an example and the observation was arbitrarily chosen from observations with two or more images under creative commons licence. Image taken by Rune Zakariassen and published under creative commons licence. Website accessed 23.11.23

were collected for 200 observations for each of the 100 most popularly photographed plants. An example of two images connected to an observation can be seen in the screenshot in figure 4.9.

4.2.3 Elevation data

Height data was added to the observations through the Norwegian Mapping Authority's API, where the data was queried in chunks of 50 points at a time. For the first iteration the returned heights are from airborne laser scanning with a resolution of 1 meter. This is form the Dataset *Høydedata-laser* at *kartkatalog.geonorge.no/*. In the next iteration all the points whose height were returned as *None*, were queried against a different source giving an interpolated height. The second source(*N50 Kartdata*) has an error between 0.1-3m for the majority of the points, but the error also be in the tens for some cases. After trying both sources for the elevation data, the remaining *None* values were filled with the average value.

4.2.4 Area Resources(AR-50)

The AR-50 data-file was downloaded from NIBIOs website as a geodatabase(gdb). The file was opened with geopandas. Then the coordinates were taken from the species observations and iterated through the many polygons in the AR-50 geodatabase and added the data from AR-50 to the Species-Observations-Combined Dataset. In addition to geopandas the Python packages pandas and shapley were used for this.

4.2.5 Landscape Data

As with the AR-50 Dataset the Landscape Datasets consists of a series of polygons. This time the data was found at the website of the Species Databank(*artsdatabanken*). Where the classification of different landscapes can be downloaded as a geojson files. The geojson files were opened with geopandas, and the package shapley was used to check every location point against the polygons in the geojsons. Most observation points were classified through this and the remaining points were classified according to the prediction of the k-NN algorithm.

4.3 Computer Code Used for Data Collection and Processing

The code for collecting and processing the Data can be found in the GitHub project:
https://github.com/vegardwho/MasterThesis
In the jupyter Notebooks:

- Webscraping the Norwegian Species Observation Service(*Artsobservasjoner)* webscraping_species_observations.ipynb
- Processing and merging the collected data: geo_pipeline_species_observation.ipynb

Chapter 5

Experiments

In this chapter the experiments performed for this thesis are described. This includes fitting and training SVMs and computer vision models, and the use of additional location information to improve their accuracy.

5.1 Location as Auxiliary Information

Initial experiments were based on the possibility of using existing computer vision models as is and supplying additional information using location data. This was either done by weighting the model predictions with the location model predictions.

5.1.1 SVM on LeafSnap with Height and Latitude

Gupta and Florescu [31] created a SVM classifier on the Flavia dataset. The classifier produced a 98.604% accuracy. The classification is based on a collection of extracted features such as mean and standard deviation of the RGB values, aspect ratio, area, circumference, circularity, rectangularity and image moments. These are among the features outlined by Waldchen and Mäder [17] in their literature review. I was able to confirm through personal correspondence with Aayush Gupta, that these features were selected from various different studies in order to get the best results. Four examples of leaf scans from the Flavia Dataset, with rectangular and ellipsoid features drawn on the them, can be seen in Figure 5.1.

The same procedure was applied for feature extraction on the LeafSnap Dataset to determine how well the SVM method scales and how well it works on a different dataset. In addition to this I tried to add sampled location data from the Species-Observation Dataset where a species from Flavia is mapped to a species in Species-Observation Dataset.

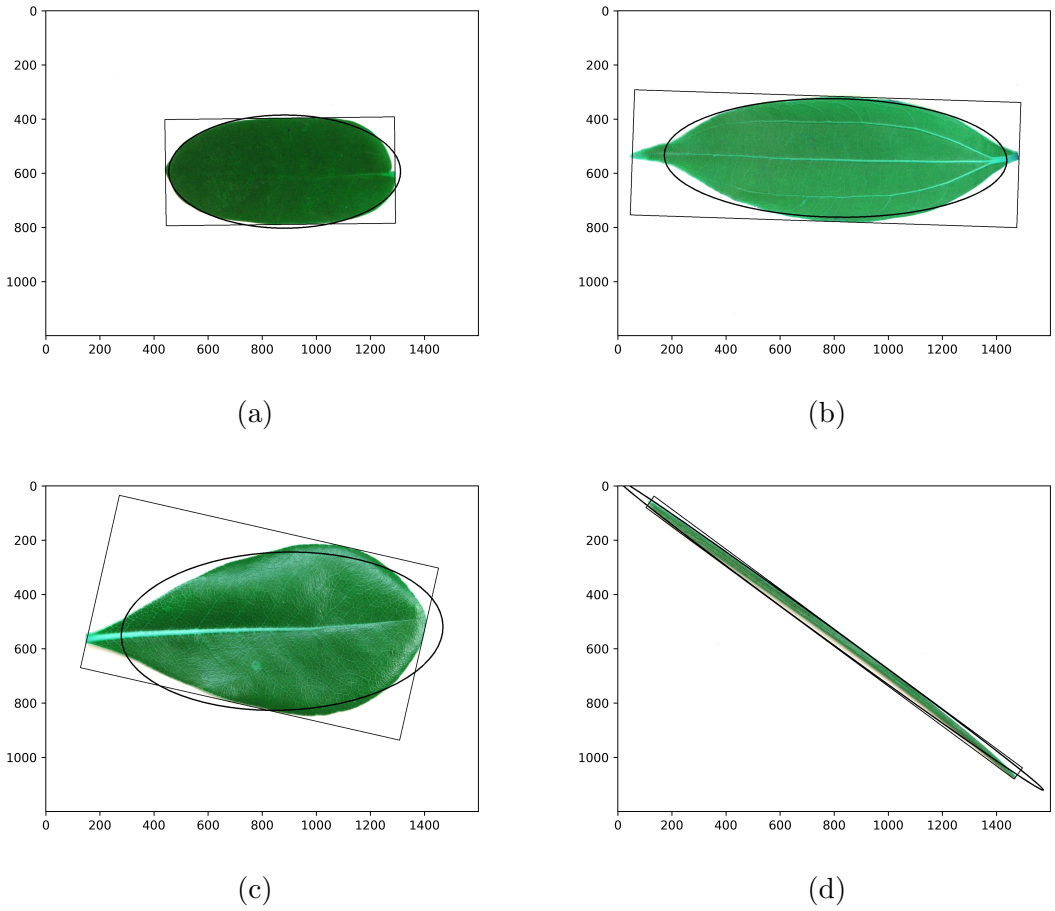


Figure 5.1: Four leafs from the Flavia Dataset, with the features rectangle and ellipsoid drawn around them.

5.1.2 LeafSnap with CNNs and Semi-Simulated Location Data

A CNN classifier created by Vishwajith [32] on the LeafSnap data-set increased the accuracy from 70.8% to 86.2% for Top-1, and 96.8% to 98.4% for Top-5, when compared with the [29, 32]. It should also be possible to gain 91.86% top 1 accuracy with a pretrained VGG-19 model[33].

An exploratory experiment was done using the LeafSnap images and location data from *artsobservasjner.no*, where a Convolutional Neural Network(CNN) is trained on the LeafSnap Dataset and the CNN-output is augmented based on location data. Here the 185 species from LeafSnap-data are mapped to the 185 most common species in the Species-Observation-185 Dataset. Each image is given a location sampled from the Observation-Data corresponding to the species the LeafSnap-species is mapped to. The pretrained models, Resnet-18, 34, 50 and 101, are trained on the LeafSnap-images and their output is augmented either

by multiplication or addition with KDE, KNN and an *In Area* value. Here the unbalanced data-points in the Species-Observation-185 Dataset is handled with SMOTE. KDE and KNN are explained in the Theory chapter, while the *In Area with Floor* values gives 1 to species observed within a given radius and 0.1 to species not observed. This is further explained in section (5.2.1) *In Area* values are added both through addition and multiplication.

5.2 Species-Observations with Locations

On the Species-Observation-Combined Dataset we first trained different computer vision models on the collected images. These images are transformed into a resolution of 225x225 pixels before being put through the Computer Vision models. This is quite a low resolution, but gives us more errors to work with when trying to improve the models accuracies. It is also a manageable size for the my laptop to work with. After training we choose one or two models to see if their accuracy can be increased by using location based methods. The species concentration for the top 100 observed photographed plant species(Species-Observation-100 Dataset) can be seen in Figure 5.2. The dataset contains 844603 samples of 100 different plant species and a histogram showing the number of observation per species can be seen in Figure 5.3.

The different distributions for the species *Vaccinium myrtillus/European Blueberry* and *Acer pseudoplatanus/Sycamore* can be seen inn Figure 5.4 and an image of both can be seen in Figure 5.5. From the Figure it is easy to see that these two plants have different distributions. The Sycamore is a non native species imported in the 17-hundreds and is now considered an invasive species[34]. Since it is imported it makes sense that the plant is found along the coast. The European blueberry grows all over Norway, as can be seen in Figure 5.4. It is this difference seen here that we wish to exploit in order to increase the precision of the models.

5.2.1 Location Based methods

The location based method used in order to improve the predictions are:

- k-Nearest Neighbours (k-NN).
- Kernel Density Estimation (KDE).
- Weighing plant species within a given radius.
- Multinomial logistic model.
- Relative frequency weighting.

Here is a quick overview of how they are implemented

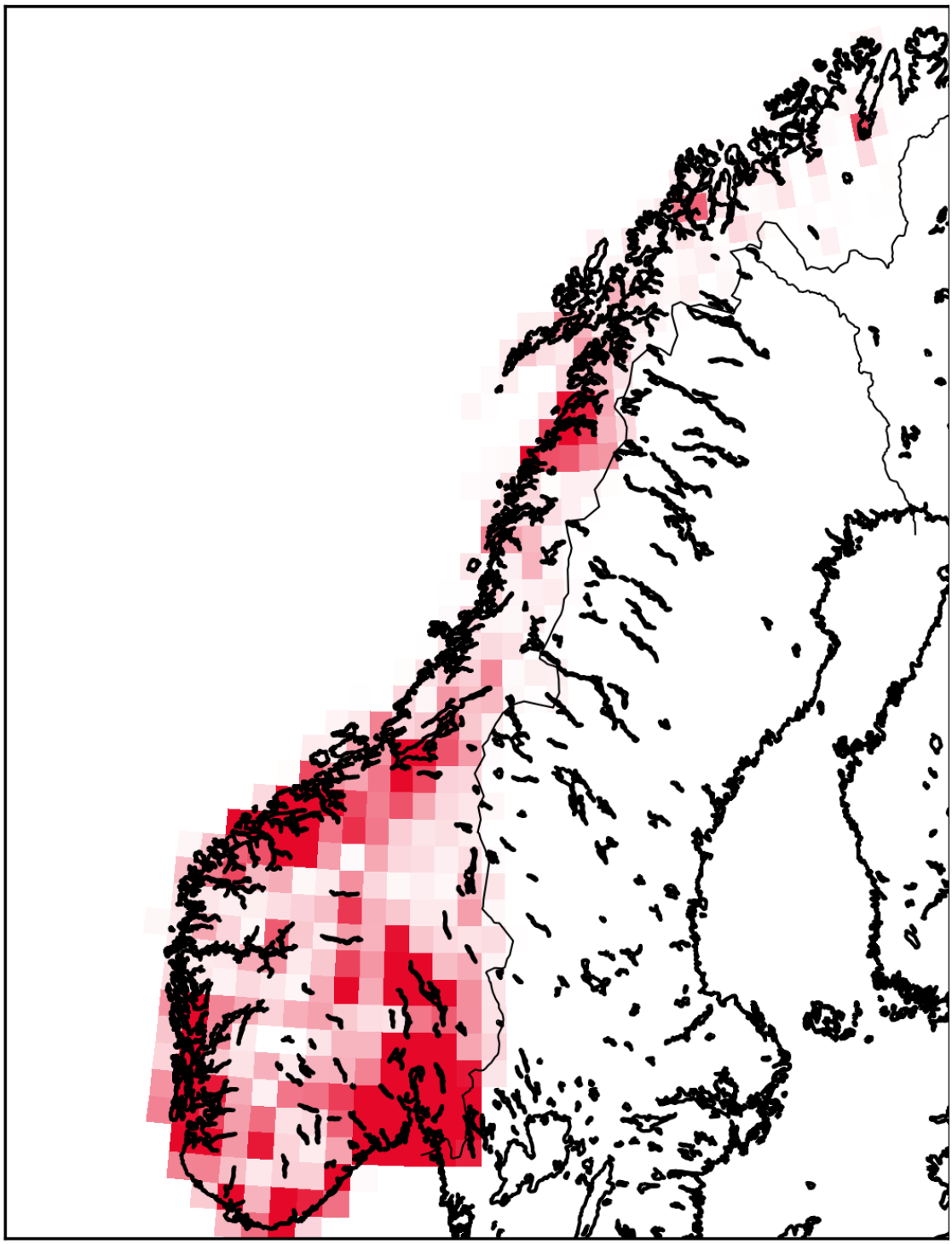


Figure 5.2: Heatmap showing the distribution and concentration of the Species-Observation-100 Dataset

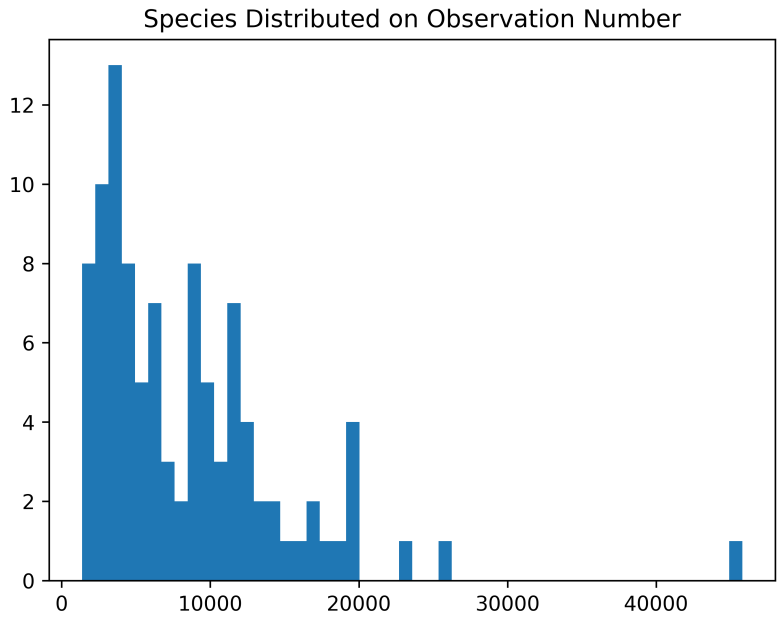


Figure 5.3: Histogram showing the number of observation per species in the Species-Observation-100 Dataset.

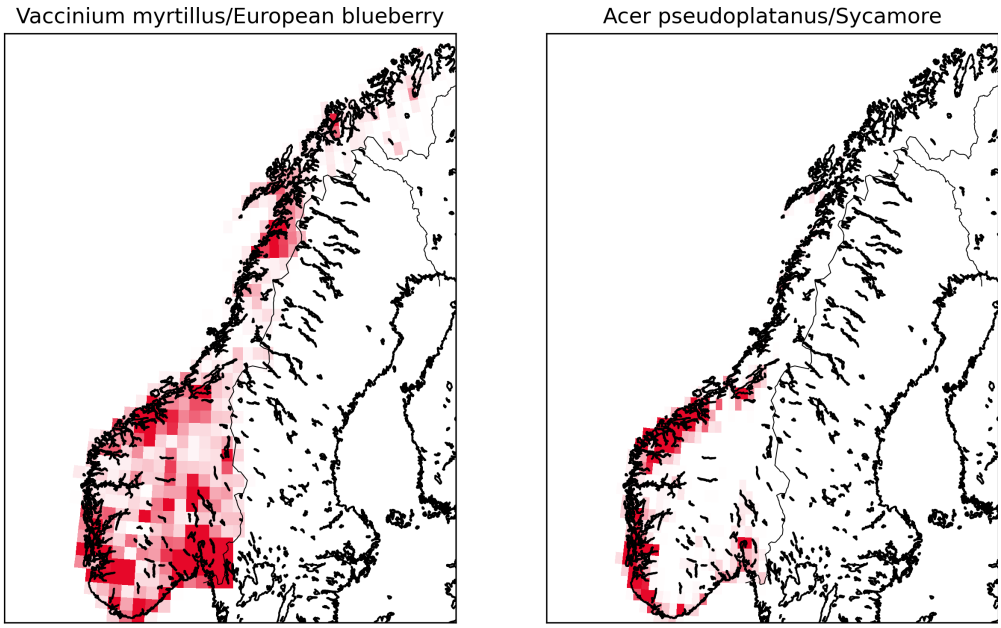


Figure 5.4: Heatmap showing the different distributions of European Blueberry and the Sycamore tree.

(a) European Blueberry (b) Sycamore

Figure 5.5: Image of a Blueberry and a Sycamore

k-NN

k-NN method is outlined in subsection 2.1.3. Where the k nearest points votes on what the point is and the probability is given by this. Here it is implemented on the entire Combined Dataset using the locations from the Species-Observation-100 Dataset to give probabilities for the location in question. Every row in the Combined Dataset is given a array of 100 probabilities for every k value we want to look at.

KDE

The KDE estimation for a point is calculated by taking a square around the point and using the points within that square in the Species-Observation-100 Dataset. This produced an array of length 100 with a KDE value for every plant species at the given point in the Combined Dataset. Again every row got an array for every combination of square and bandwidth size.

Weighting plants with *In Area*

The *In Area* value comes from looking at the surrounding area of a point, from the Combined Dataset, and weighting all plants found within a given radius in the Species-Observation-100 Dataset as 1. The plants not within the radius are given the value 0. This creates an array of length 100 for each point and for every radius we are interested in. The *In Area* weighting is done in two different ways. The first way is to add extra weight for the plants observed in the area. This will be referred to as *In Area*. The other way is to set all plants observed in

the area to 1 and all other values to a value c and multiply the proposed probabilities, from the computer vision model, with the tensor containing the 1s and cs and then normalizing the product. This will be referred to as *In Area with Floor*

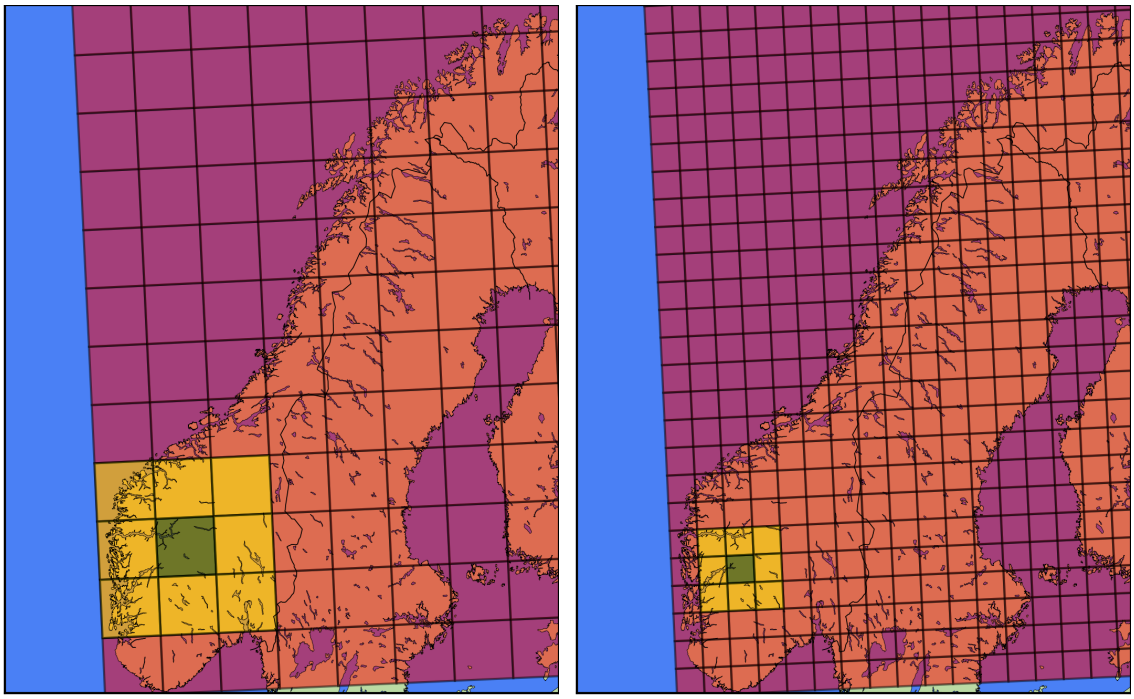
Multinomial Logistic Model

This model uses all relevant categorical values connected to the location in question. This includes the landscape and sub landscape classification for the Landscape Data. From AR-50 Dataset the values for Area type, Agricultural Land, Forrest type, Forest bonity and Field type. The height was also slit into 5 categories corresponding to height intervals, did the same for the latitude spans. In addition to this I also added the county for the given observation and used the *LogisticRegression* from the python package *sklearn*.

Relative Frequency Weighting.

Weighting the models' proposed probabilities by the Relative Frequency within the current area. This was done in two slightly different manners. For both the map is partitioned into squares and the relative frequency of the plants observed in each square is determined. This was done for squares of varying sizes by taking the eastern and western most points and separating the area between them into 8, 10 , 15, 20, 30, 40, 50 ..., 100 intervals. Then taking intervals of same lengths from the southern most point until the northern most point is within an interval. These squares are partitions of the map consisting of areas of sizes between 28606 km^2 and 143 km^2.

For the first method the probabilities for an observation in a square are multiplied by the plant species relative frequency in that square and the surrounding eight. The second method we give all values of zero the lowest positive number found in the combined nine squares. These two methods will be referred to as *Relative Frequency Weighting without Floor* and *Relative Frequency Weighting with Floor*.The partition and squares of interest for 131km x 131km and 62km x 62km can be seen in Figure 5.6. This method is based on *iNaturalist*s former geomodel [23].



(a) 131km x 131km (b) 62km x 62km

Figure 5.6: Partitioned area for 10 intervals east to west b) giving squares of 17161km^2, and area for 20 intervals a) giving squares of 3844km^2. With the area of interest and its surrounding squares highlighted.

5.3 Machine Learning Training

The code for training the machine learning models utilized in this thesis can be found in the GitHub project:
https://github.com/vegardwho/MasterThesis
In the jupyter Notebooks:

- SVM on Flavia and LeafSnap data: SVM_flavia_leafsnap.ipynb
- Creating the geo samples for the Leafsnap experiment with semi-simulated data: geo_functions_leafsnap.ipynb
- Training and Testing the computer vision models on LeafSnap data and location data : train_test_geo_leafsnap.ipynb
- Training the Computer vision models on species observation data: geo_on_species_observation.ipynb
- Create plots used in the thesis: species_observation_create_plots.ipynb

Most of the code used for training the computer vision models is based on the Deep-Leafsnap github code of Vishwajith [32] at https://github.com/sujithv28/Deep-Leafsnap. It has however been changed significantly. The code in SVM_flavia_leafsnap.ipynb is mostly an extention of Gupta and Florescu [31] work at: https://github.com/AayushG159/Plant-Leaf-Identification.

Chapter 6

Results

In this chapter the results from the experiments are presented.

6.1 Preliminary experiments:

These are the results of the first experiments on the already existing Datasets Flavia and LeafSnap, where the species in those sets are mapped to species in the Species-Observations Dataset and locations are sampled from the mapped species in the Species-Observations Dataset.

6.1.1 SVM on LeafSnap with Height and Latitude

By using SVM Gupta and Florescu [31] were able to classify the leaves in the Flavia-dataset with an accuracy of 98.604%. As seen in Table 6.1.

Table 6.1: Results for SVM on Flavia-dataset

# Plants	Accuracy/Recall[%]	Precision[%]	F-Score[%]
32	98.604	98.604	98.604

Using the same code I was able to perform the same feature extraction from the LeafSnap Dataset. Four examples of LeafSnap scans with the features rectangularity and ellipsoid drawn on them can be seen in Figure 6.1. In this experiment we discard samples form the

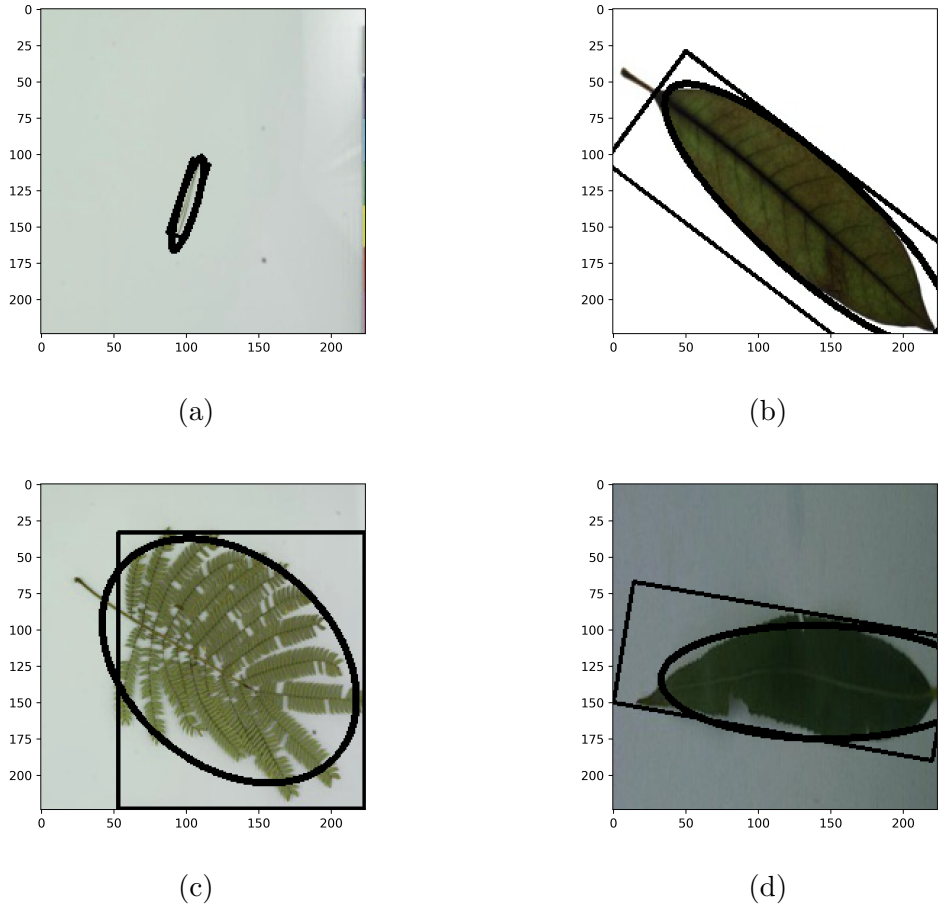


Figure 6.1: Four leafs from the LeafSnap Dataset, with the features rectangle and ellipsoid drawn around them.

LeafSnap Dataset with low quality, where features such as area and rectangularity were not possible to determine.

For each selection of plants species, going from 10 to 150 different plants in increments of 10s, 5 SVMs were created. Each of the SVMs is fitted with randomly sampled plant species with 60 sampled images for each of them. The parameters for each SVM were determined by optimizing the result on the training data using a grid search over Radial basis function kernels and Linear Kernels. The average and max values for the SVMs classifications can be seen in Table 6.2.

We see that the result on the Flavia set has a 26.3% percent higher accuracy than the SVMs on the LeafSnap set when we compare the accuracy of the Flavila SVM, which has 32 species, with the LeafSnap SVM which has 30 species.

Table 6.2: SVM on the LeafSnap Dataset where the result is the average of 5 different SVMs, with 5 different subsets of plants. Every plant with 60 sampled images.

	Accuracy/Recall		Precision		F-score	
# Plants	Avg	Max	Avg	Max	Avg	Max
10	0.800	0.822	0.808	0.829	0.798	0.820
20	0.759	0.803	0.766	0.808	0.756	0.800
30	0.723	0.769	0.733	0.778	0.721	0.768
40	0.658	0.699	0.678	0.724	0.658	0.701
50	0.653	0.663	0.667	0.679	0.651	0.660
60	0.654	0.696	0.671	0.711	0.652	0.696
70	0.604	0.625	0.623	0.645	0.603	0.625
80	0.594	0.607	0.611	0.620	0.592	0.603
90	0.595	0.610	0.615	0.631	0.595	0.610
100	0.573	0.594	0.591	0.614	0.569	0.592
110	0.568	0.589	0.587	0.602	0.567	0.586
120	0.556	0.569	0.573	0.585	0.554	0.568
130	0.547	0.559	0.565	0.581	0.545	0.558
140	0.537	0.549	0.555	0.571	0.533	0.546
150	0.540	0.550	0.560	0.570	0.538	0.549

The process was repeated for 10, 20 and 30 plant species, with 10 different SVMs for each, but including height samples and northern coordinates. The height and coordinates are sampled from the Species-Observation Dataset, where the plant species from LeafSnap has been mapped to a species in the Species-Observation Dataset. The height and northern coordinates were included both separately and combined. The result for coordinates is presented in Table 6.3, for heights in Table 6.4 and combined in Table 6.5.

Both by them selves, as seen in Table 6.3 and 6.4, and together, seen in Table 6.5.

Table 6.3: SVM results for leafs from the LeafSnap Dataset with sampled Northern coordinates.

	Accuracy/Recall		Precision		F-score	
# Plants	Avg	Max	Avg	Max	Avg	Max
10	0.785	0.856	0.798	0.868	0.785	0.858
20	0.732	0.781	0.750	0.796	0.733	0.780
30	0.676	0.717	0.693	0.730	0.674	0.711

Table 6.4: SVM results for leafs from the LeafSnap Dataset with sampled observation heights.

	Accuracy/Recall		Precision		F-score	
# Plants	Avg	Max	Avg	Max	Avg	Max
10	0.785	0.856	0.798	0.868	0.785	0.858
20	0.732	0.781	0.750	0.796	0.733	0.780
30	0.676	0.717	0.693	0.730	0.674	0.711

Table 6.5: SVM results for leafs from the LeafSnap Dataset with sampled observation heights and northern coordinates.

	Accuracy/Recall		Precision		F-score	
# Plants	Avg	Max	Avg	Max	Avg	Max
10	0.752	0.811	0.768	0.832	0.752	0.816
20	0.693	0.756	0.715	0.772	0.694	0.759
30	0.628	0.674	0.648	0.699	0.627	0.673

Neither the height of the plant observation nor the northern coordinate has a positive impact on the accuracy in the model. While some of the Max Accuracies for the SVMs with added height or Northern coordinate are higher than the original, it only happened when there were 10 different species. The rest of the max accuracies were lower for all the combined height and coordinate SVMs and the height SVMs and the coordinate SVMs for 20 and 30 species.

Table 6.6: Result for the ResNet models trained on the LeafSnap Dataset.

Model	Best Epoch	Test Top-1[%]	Test Top-5[%]	Train Top-1[%]	Train Top-5[%]
ResNet-18	42	91.334	98.885	98.207	99.985
ResNet-34	31	91.993	99.003	98.923	99.995
ResNet-50	27	91.841	99.122	99.129	99.999
ResNet-101	35	91.486	99.037	99.008	99.997
ResNet-152	40	89.713	98.699	99.113	99.997

6.1.2 LeafSnap with CNNs and Semi-Simulated Location Data

For species recognition on the LeafSnap Dataset several models were trained. Among these models were the ResNet-18, 34, 50, 101 and 152.

These models were pretrained on the ImageNet-Dataset, before being further trained on the training-set partition of the LeafSnap dataset with images down-scaled to a resolution of 64x64. Training started with a learning rate of 10^{-2} for the first 15 epochs, 10^{-3} for the next 15 and another 15 with 10^{-4}, bringing it to a total of 45 epochs. The Final result can be seen in the Table 6.6, while the training progression can be seen in Figure 6.2 and 6.3. Where the progression of the Average Top-1 Test accuracy and the progression of the Average Top-1 accuracy training error are plotted respectively. Nearly all ResNet models were able to get a 91% accuracy, except for the ResNet-152 model.

The ResNet-18 model was selected for testing of the location based methods. The ResNet-18 model was trained again and were able to achieve a test accuracy of 93.380% on the LeafSnap Dataset. In the following sections I will illustrate how the location data is added to the ResNet-18 models predictions.

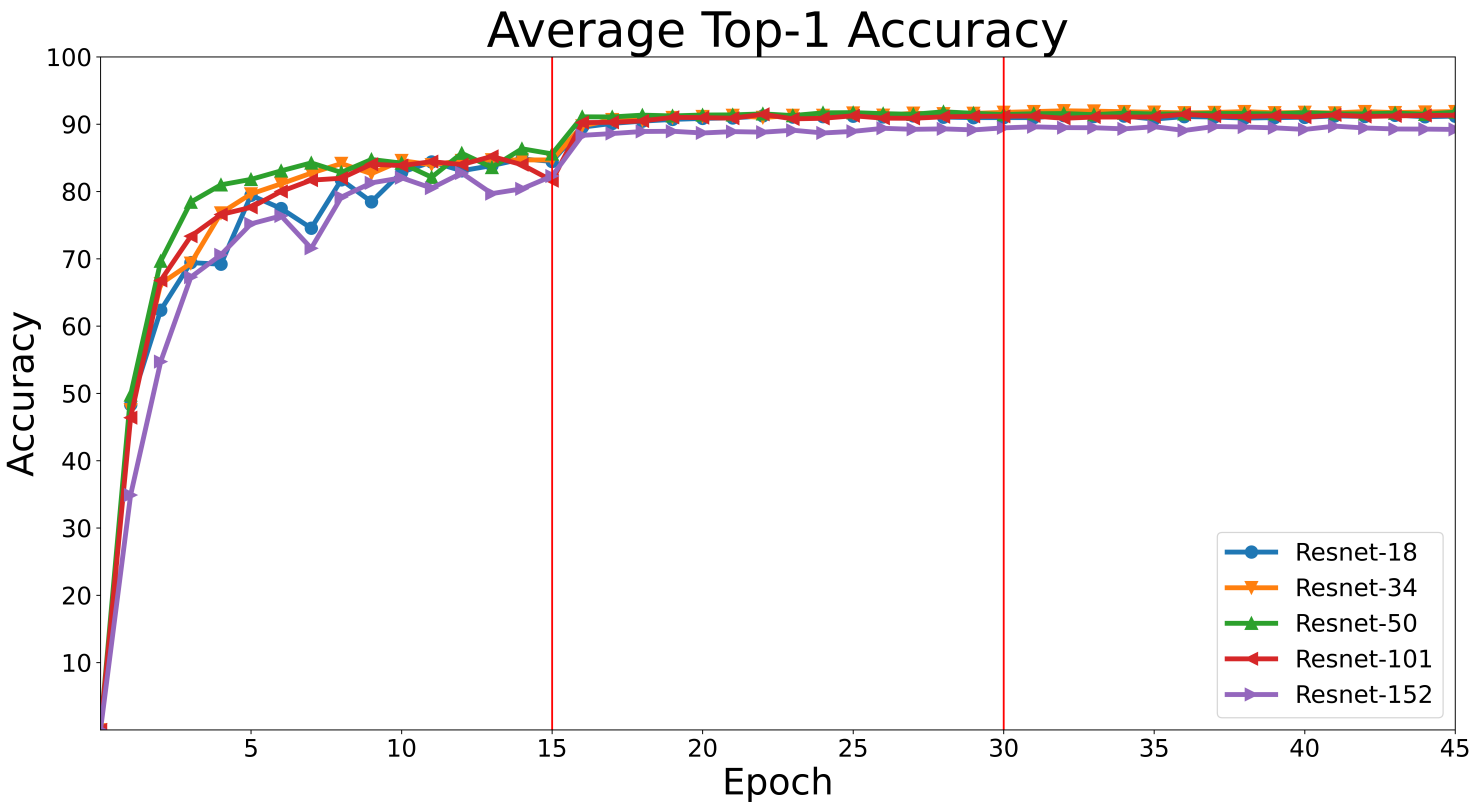


Figure 6.2: Average Top-1 Validation Accuracy for the pretrained models at each epoch of training. At the vertical lines the the learning rate goes from 10^{-2} to 10^{-3} and from 10^{-3} to 10^{-4} respectively.

Augmenting with location information:

When checking if the it was possible to improve the prediction by adding a location based weight to them, the location based weights are added as following to the original prediction:

$$y' = y + s \cdot w \tag{6.1}$$

Where y is the original prediction, w is the weight based on the location data and s is the scaling factor. When multiplying with a location value such as the *In Area with Floor* value the output is given by:

$$y' = \frac{y \cdot w}{\sum y_i \cdot w_i} \tag{6.2}$$

Where the original model output y is multiplied element wise with the location vector w and is then normalized.

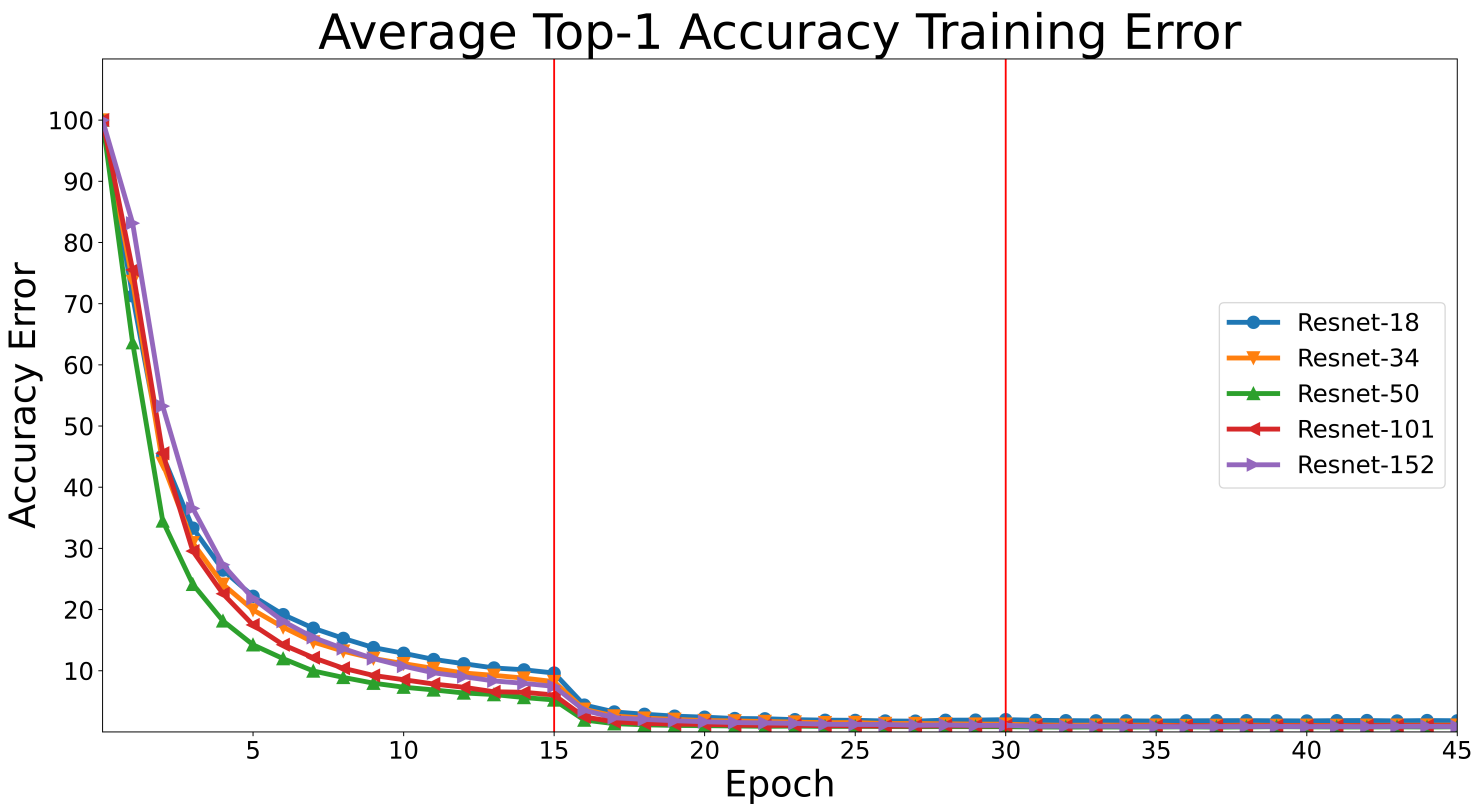
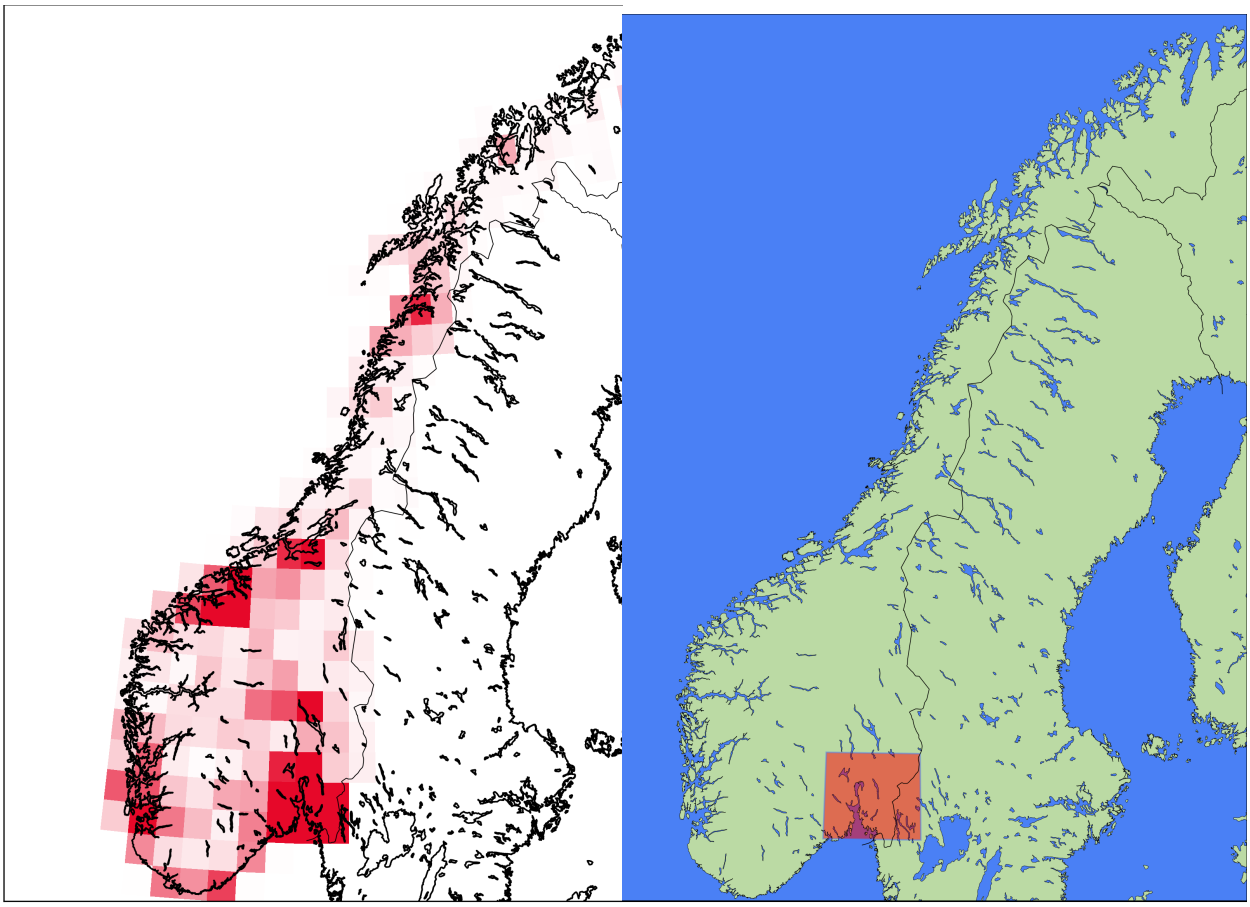


Figure 6.3: Average Top-1 Training Error for the pretrained models at each epoch of training. At the vertical lines the the learning rate goes from 10^{-2} to 10^{-3} and from 10^{-3} to 10^{-4} respectively.

In order to maximize the likelihood for methods such as Kernel Density Estimation(KDE) and k-NNs to give a useful result, an area around Oslo was selected for testing, because this area had the highest density of observations. The distribution of the 185 most observed plants can be seen in Figure 6.4a and the selected area can be seen in Figure 6.4b.

In the following sections I will present the results of adding the KDE, k-NN and *In Area With Floor* values to the



(a) Heatmap. (b) Area of interest

Figure 6.4: (a) Heatmap of distribution the distribution of the 185 most registered plant species and (b) the area selected for testing of location based methods on the LeafSnap Dataset.

Adding Kernel Density Estimation(KDE) Values

This experiment, increasing accuracy by adding KDE values, was done three times with different mapping between the LeafSnap Dataset and the Species-Observation-185 Dataset for each experiment. For each mapping, location-points were sampled from the Species-Observation-185 Dataset to represent the leaves in the LeafSnap Dataset. This was done inside the selected area (seen in Figure 6.4b) and for every sampled location-point the KDE values, for every species, were calculated for a square around the sampled location-point. This was done for squares with varying side lengths and the KDE values were calculated with a Gaussian kernel (2.20) with different bandwidths.
The KDE values were added to the CNN outputs with a factor 0.1. The factor was determined by maximizing the accuracy of the augmented ResNet-18 model, while still being larger or equal to 0.1. The KDE augmented outputs for the ResNet CNN is presented in

the Table 6.7. There the subscript a indicates the augmented output and the subscript max indicates the result of the original non-augmented network at the epoch with the highest Top-1 accuracy. I also tried to multiply the probabilities given by the ResNet-18 model, but the results were significantly lower than the original non-augmented model.

Table 6.7: The average result with the Kernel Density Estimator(KDE) output added to the ResNet-18 output with the KDE weighted with 0.1 at epoch 27. The KDE parameters are given on the left followed by the augmented Top-1 and Top-5 accuracy. Side Length is the side lengths of the square surrounding the observation and the Bandwidth is the bandwith for the Gaussian KDE. The non-augmented CNN has a max Top-1 accuracy of 93.380% at epoch 27 and Top-5 accuracy of 99.378%

Side Length	Bandwidth	Top-1_a[%] max	Top-5_a[%] at epoch$_a$	Top-1_a-Top-1_{max}[%]	Top-5_a-Top-5_{max}[%]
1000	500	11.600	13.661	-81.780	-85.717
1000	1000	32.084	34.733	-61.296	-64.645
1000	1500	53.435	57.030	-39.945	-42.348
1000	2000	70.936	75.394	-22.444	-23.984
1000	3000	89.925	95.262	-3.455	-4.116
1000	4000	93.319	98.930	-0.062	-0.448
2000	500	12.666	14.738	-80.714	-84.640
2000	1000	33.412	36.145	-59.969	-63.237
2000	1500	54.729	58.430	-38.652	-40.948
2000	2000	71.806	76.303	-21.574	-23.075
2000	3000	90.201	95.589	-3.179	-3.790
2000	4000	93.324	98.902	-0.056	-0.476
3000	500	13.861	15.989	-79.519	-83.389
3000	1000	34.683	37.511	-58.697	-61.867
3000	1500	55.916	59.651	-37.465	-39.727
3000	2000	72.764	77.339	-20.617	-22.039
3000	3000	90.565	95.953	-2.815	-3.426
3000	4000	93.336	98.880	-0.045	-0.498
4000	500	14.774	16.930	-78.607	-82.448
4000	1000	36.270	39.154	-57.111	-60.224
4000	1500	57.058	60.805	-36.322	-38.573
4000	2000	73.581	78.118	-19.799	-21.261
4000	3000	90.828	96.205	-2.552	-3.174
4000	4000	93.336	98.874	-0.045	-0.504

Table 6.8: Average result for the k-NN augmented ResNet-18. Where the k-NN output weighted with 0.7 before being added to the CNN-output before normalizing the sum. Subscript a indicates the post augmented results. The non-augmented CNN has a max $prec1$ at epoch 27 with 93.380% and $prec5$ at 99.378%

k	Top-1_a[%]	Top-5_a[%]	Top-1_a-Top-1_{max}[%]	Top-5_a-Top-5_{max}[%]
25	93.470	97.866	0.090	-1.512
50	93.470	97.866	0.090	-1.512
75	93.470	97.866	0.090	-1.512
100	93.442	97.726	0.062	-1.652
200	93.442	97.726	0.062	-1.652
500	93.442	97.726	0.062	-1.652
1000	93.442	97.726	0.062	-1.652
1500	93.442	97.726	0.062	-1.652
2000	93.442	97.726	0.062	-1.652
2500	93.442	97.726	0.062	-1.652
3000	93.442	97.726	0.062	-1.652

Adding k-NN Values

This experiment was, as for the KDE experiment, performed three times with different mappings between the Datasets. It is the average of the three experiments which is presented here.

k-NN values were calculated for each of the data samples and added to the prediction from the Computer vision models. The added k-NN values were multiplied by 0.7, which was found by manual search between 0 and 1. The result of k-NN values added to the models with values from k =25 to k = 3000 can be seen in Table 6.8 for the ResNet-18 modiel. I also tried to multiply the k-NN probabilities with the outputted probabilities from the model, but again the result was less accurate than the original model.

Adding *In Area with Floor* Weight Values

This experiment was also performed three times with different mapping between the datasets and the average of these experiment is presented here. The *In Area with Floor* values are given for several different radii between 10m and 2000m, where the plant species observed within the radii are given value 1 and the species not observed are given the value 0.1. These values were used to augment the model by adding the values to the predictions and by multiplying the *In Area with Floor* value with the predictions.

Table 6.9: Average result for the *In Area with Floor* sum augmented ResNet-18. Where the plants within the area are weighted with 1 and the others 0.1, before being multiplied by 0.75 and added to the CNN-output. Subscript a indicates the augmented results. The non-augmented CNN has a max Top-1 accuracy at epoch 27 with 93.380% and Top-5 at 99.378%

Sample Radius[m]	Top-1_a[%]	Top-5_a[%]	Top-1_a-Top-1_{max}[%]	Top-5_a-Top-5_{max}[%]
10	93.380	99.365	0.000	-0.011
20	93.380	99.367	0.000	-0.011
30	93.380	99.378	0.000	0.000
40	93.386	99.378	0.006	0.000
50	93.392	99.378	0.011	0.000
100	93.380	99.378	0.000	0.000
200	93.380	99.378	0.000	0.000
300	93.380	99.378	0.000	0.000
400	93.380	99.378	0.000	0.000
500	93.380	99.378	0.000	0.000
1000	93.380	99.378	0.000	0.000
2000	93.380	99.378	0.000	0.000

Adding the *In Area with Floor* values with a scaling factor of 0.75 to the ResNet-18 predictions did technically increase the Top-1 accuracy, but not enough to say for certain. The results can be seen in Table 6.9. The scaling Factor was found by manual search between 0 and 1, and selected the value which gave the highest increase in Top-1 accuracy.

Multiplying the *In Area* values with the probabilities from the CNN model gave a slight increase in Top-1 accuracy, as seen in Table 6.10.

Table 6.10: Average result for the *In Area with Floor* product augmented ResNet-18. Where the plants within the area are weighted with 1, before being added to the CNN-output. Subscript a indicates the augmented results. The non-augmented CNN has a max Top-1 accuracy at epoch 27 with 93.380% and Top-5 at 99.378%

Sample Radius[m]	Top-1_a[%]	Top-5_a[%]	Top-1_a-Top-1_{max}[%]	Top-5_a-Top-5_{max}[%]
10	93.386	99.378	0.006	0.000
20	93.392	99.378	0.011	0.000
30	93.380	99.378	0.000	0.000
40	93.398	99.378	0.017	0.000
50	93.391	99.378	0.011	0.000
100	93.380	99.378	0.000	0.000
200	93.380	99.378	0.000	0.000
300	93.380	99.378	0.000	0.000
400	93.380	99.378	0.000	0.000
500	93.380	99.378	0.000	0.000
1000	93.380	99.378	0.000	0.000
2000	93.380	99.378	0.000	0.000

6.2 Species-Observations with Locations

The Species-Observation-Combined Dataset consists of 100 plant species with 200 images and a location for each species. For the purpose of this study the Dataset was split into a Test and Training set with 4000 and 16000 images and location-points in each respectively. A further Validation set was also collected. The Validation Set contained 4843 images with location data. 18 species with 50 images, 33 with 49 images, 28 with 48 images, 16 with 47 and 5 species with 46 images. The models were loaded with pretrained weights from the ImageNet-Dataset. Then the models are trained on the plant images, with learning rate 10^{-2} from 0 to epoch 10, 10^{-3} from 10 to 15 and 10^{-4} to 17 for the ResNet models. The training was continued for the Visual Transformer(ViT) model and VGG-19 with learning rate 10^{-5} for another two epochs. The models training progression can be seen in Figures 6.2 and 6.6 for the Top-1 and Top-5 Accuracy on the test set, and in Figure 6.7 the models the training errors can be seen. The final result for each model is presented in Table 6.11, where the ResNet-152 achieved the highest Top-1 accuracy of 70.132%.

i

Table 6.11: Result of the Computer Vision Models on the Combined Species-Observation Dataset

Model	Best Epoch	Test		Training	
		Top-1	Top-5	Top-1	Top-5
VGG-19	17	49.413	78.955	94.238	94.238
ViT	14	63.759	87.103	99.950	99.950
Resnet-18	17	59.810	83.354	100.000	100.000
Resnet-34	17	61.360	85.329	99.994	99.994
Resnet-50	12	69.283	89.253	98.869	98.869
Resnet-101	15	70.018	90.227	99.650	99.65
Resnet-152	16	70.132	89.928	99.706	99.706

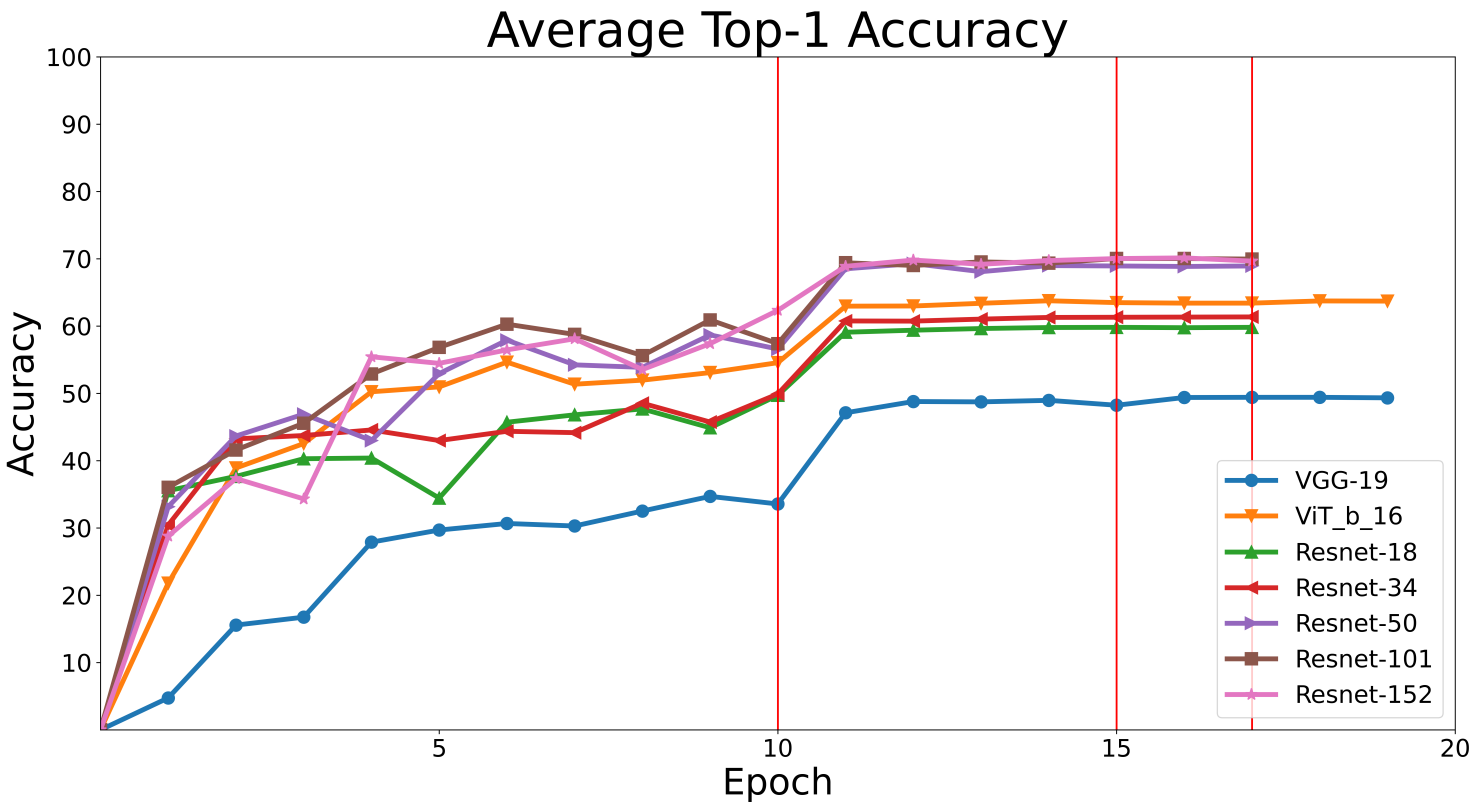


Figure 6.5: Average Top-1 Validation Accuracy for each epoch of training. At the vertical lines the the learning rate goes from 10^{-2} to 10^{-3}, from 10^{-3} to 10^{-4} and from 10^{-4} to 10^{-5} respectively.

Added location information:

Added values based on the location information to improve the Accuracy of the models. The in *In Area* and Logistic model, k-NN and KDE values were added to the models prediciton the same way as in equation (6.1). The Relative Frequency Weight(RFW) or *In Area with Floor* values are added to the models predictions the same way as in equation (6.2).

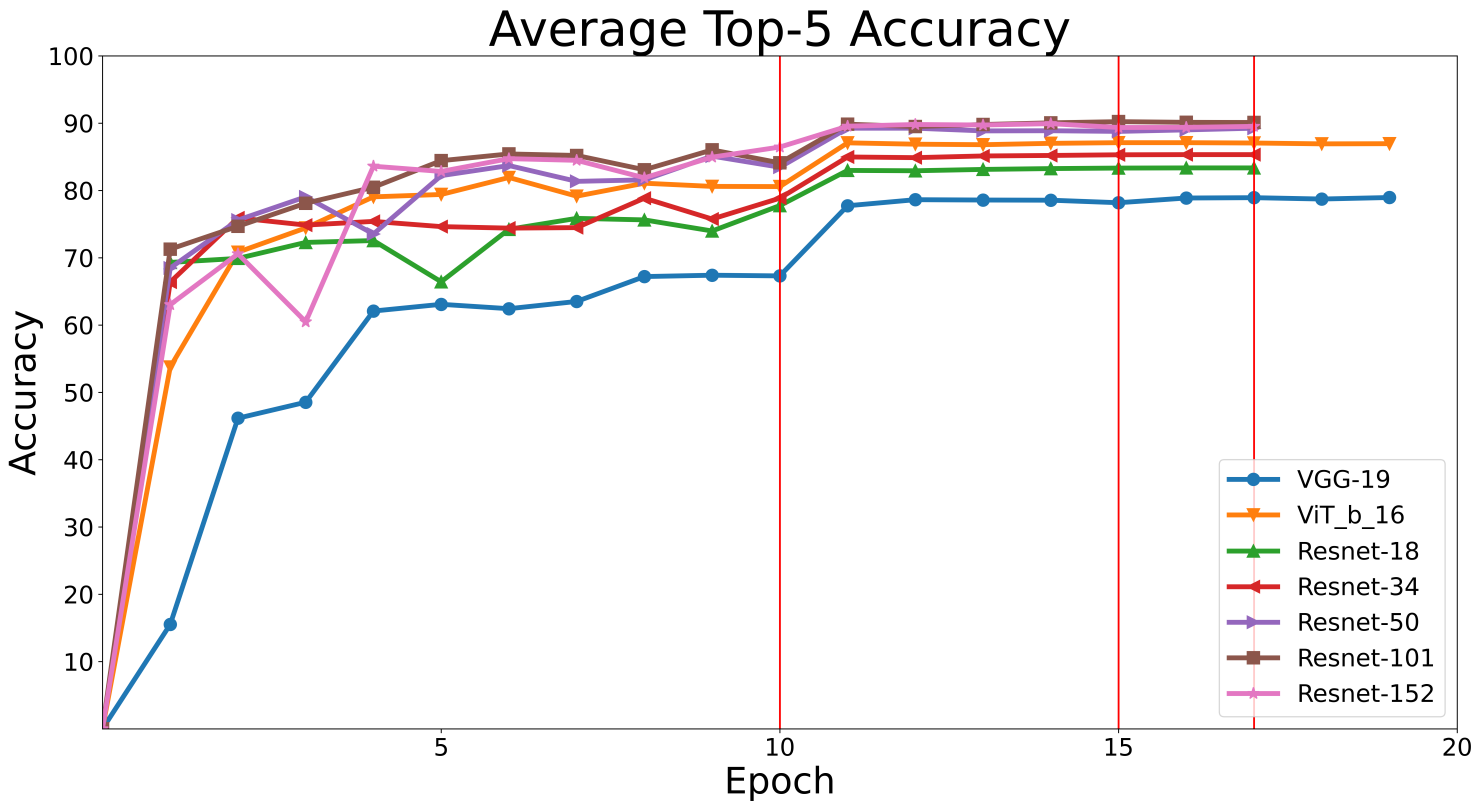


Figure 6.6: Average Top-5 Validation Accuracy for each epoch of training. At the vertical lines the the learning rate goes from 10^{-2} to 10^{-3}, from 10^{-3} to 10^{-4} and from 10^{-4} to 10^{-5} respectively.

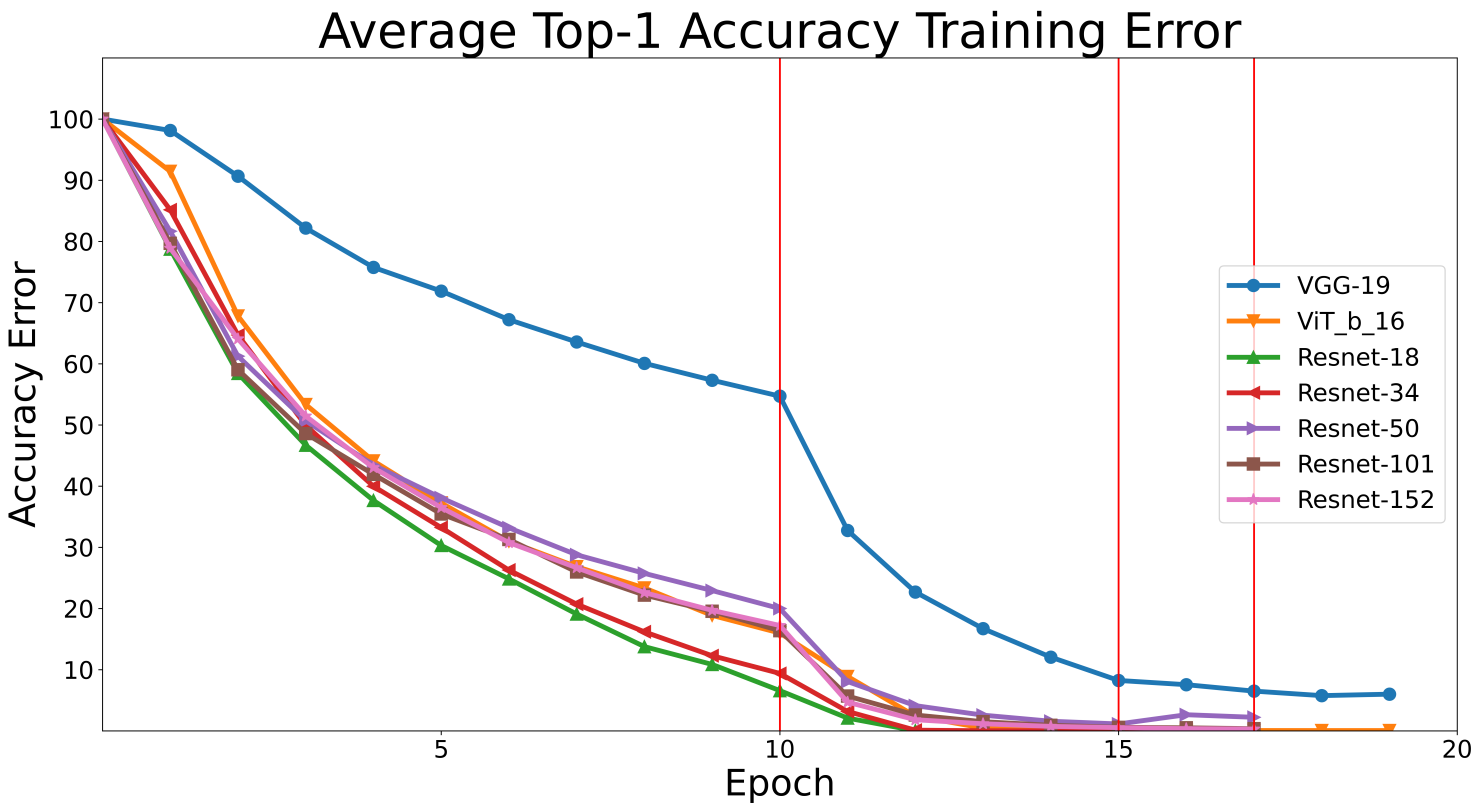


Figure 6.7: Average Top-1 training error on each epoch of training. At the vertical lines the the learning rate goes from 10^{-2} to 10^{-3}, from 10^{-3} to 10^{-4} and from 10^{-4} to 10^{-5} respectively.

6.2.1 New Models

When processing the data again to create the files for relative frequency weighting with floor(see section 6.3.4), an overlooked datapoint was processed along with the other data. This caused the split between training and test data to be different and the previous test and training files were overwritten. So new models were trained for the ResNet-101 and ViT model for further test of location based augmentation.

6.3 ResNet-101 on Species Observation data

The new ResNet-101 models accuracy for the Top-1, Top-5 and Top-10 predictions can be seen in Table 6.12.

Table 6.12: Average accuracy for the original ResNet-101 model in the Top 1, 5 and 10 predictions

Top n	Accuracy[%]
1	70.425
5	90.875
10	95.025

6.3.1 ResNet-101: KDE and k-NN

I was not able to increase accuracy for the Top 1, 5 or 10 predictions by using Kernel Density Estimation or k-NN methods.

6.3.2 ResNet-101: Observed *In Area*

Weighting the predictions by whether or not the species has been observed in the surrounding area, the scaling factor 0.6 was found manually through trial and error by iterating through a number range and choosing the factor which gives the highest accuracy. The results can be seen in Table 6.13.

We see that weighting plant species found in the area with radius 1000 m around the observation location weighted with 0.55 gives a better prediction of 1.375%. Meaning around 55 extra correct predictions. While the correct predictions in Top 5 and 10 are reduced by

Table 6.13: Prediction accuracy when weighting plants found in surrounding area with a factor of 0.55

Radius[m]	scaling factor	Top n	Accuracy[%]	Difference[%]
200	0.6	1	71.325	0.9
		5	87.225	-3.65
		10	92.15	-2.875
500	0.6	1	71.7	1.275
		5	87.15	-3.725
		10	91.05	-3.975
1000	0.6	1	71.775	1.350
		5	88.125	-2.75
		10	91.175	-3.85
1500	0.6	1	71.7	1.275
		5	88.5	-2.375
		10	91.625	-3.4

2.7% and 3.8% respectively.

The model probabilities were multiplied with the *In Area with Floor* values. This was done for several different radii and floor levels at 0.9, 0.8 and 0.7. The best result was with radius 500m and floor at 0.8 which gave a Top-1 improvement of 2%. The results of the *In Area with Floor* augmentation can be seen in Table 6.14.

Table 6.14: Prediction accuracy when multiplied with the *In Area with Floor* vector for the areas of different radii. Where the Species observed in the are given the value 1 and the other values are given a floor value.

Radius[m]	Floor value	Top n	Accuracy[%]	Difference[%]
500	0.1	1	72.45	2.025
		5	92.175	1.3
		10	95.625	0.6
1000	0.1	1	72.3	1.875
		5	92.15	1.275
		10	95.675	0.65
1500	0.1	1	72.075	1.65
		5	92.15	1.275
		10	95.675	0.65

6.3.3 ResNet-101: Multinomial logistic model

The multinomial logistic model was used to predict the species based on location values. The model has a weighted accuracy of 20%, 11% recall, f-score of 13% and the correct prediction is 86% of the time in the top 50 predicted values. As seen in Table 6.15 .

Table 6.15: Percentage of target species being in Top-n predictions by the Multinomial logistic model.

n	Predictions in Top n
90	99.725
80	98.250
70	95.599
60	92.098
50	86.572
40	79.070
30	69.742
20	58.515
10	40.810

Two different ways of weighting the predictions were used. The first way is to add the logistic models prediction probabilities to the ResNet model predictions. The result of this is an increase in accuracy of 1.625% for the Top-1 on the test data-set, as seen in Table 6.16. The scale factor was found by manual search to optimize the increase in Top-1 accuracy.

The other method, which worked significantly better, was to weight all plants within the Top-40 predicted species equally and add that to the prediction. This increased the Top-1 prediction accuracy by 2.100% to 72.525%, while decreasing the Top-5 and Top-10 accuracy, as seen in Table 6.17. Both weighting the top 40 most probable predictions and the scale factor of 0.4 were found by performing a manual grid search. I will refer to these weights as the Top-40 logistic values.

Table 6.16: Accuracy of the Augmented model when adding the weight of the probability predictions by the multinomial logistic model with a scaling factor.

Scale Factor	Top n	Accuracy[%]	Difference[%]
1.5	1	71.1	0.675
	5	85.75	-5.125
	10	89.525	-5.5

Table 6.17: Result of adding a scale factor for all species within the Top-40 most likely species according to the multinomial logistic model.

Scale Factor	Top n	Accuracy[%]	Difference[%]
0.6	1	71.6	1.175
	5	87.9	-2.975
	10	90.125	-4.9

6.3.4 ResNet-101: Relative frequency weighting

The Relative frequency weighting is performed as described in 5.2.1. The result can be seen in Table 6.18. Where the size of the squares referes to the partition size explained in section 5.2.1 and not the size of the 3x3 grid within which the relative frequency is calculated.

Table 6.18: ResNet-101: Accuracy of the **Relative Frequency Weighting without Floor**(left) and Accuracy of the **Relative Frequency Weighting with Floor**(right)

Square size	Top n	Accuracy[%]	Difference[%]
584km^2	1	72.025	1.600
	5	91.500	0.625
	10	95.250	0.225
403km^2	1	72.150	1.725
	5	91.450	0.575
	10	94.925	-0.1
294km^2	1	71.925	1.500
	5	91.125	0.250
	10	94.675	-0.350
225km^2	1	71.575	1.150
	5	90.750	-0.125
	10	94.400	-0.625
177km^2	1	71.950	1.525
	5	90.800	-0.075
	10	94.425	-0.6
143km^2	1	71.500	1.075
	5	90.350	-0.525
	10	93.750	-1.275

Square size	Top n	Accuracy[%]	Difference[%]
584km^2	1	71.850	1.425
	5	91.400	0.525
	10	95.275	0.25
403km^2	1	72.025	1.600
	5	91.625	0.750
	10	95.300	0.275
294km^2	1	71.925	1.500
	5	91.650	0.775
	10	95.350	0.325
225km^2	1	71.900	1.475
	5	91.450	0.575
	10	95.200	0.175
177km^2	1	72.025	1.600
	5	91.625	0.750
	10	95.325	0.300
143km^2	1	72.175	1.750
	5	91.700	0.825
	10	95.350	0.325

6.3.5 ResNet-101: Combining multinomial logistic model, *In Area* weight and Relative Frequency Weight

I combined location based methods which increased accuracy on the Test Dataset the most. In the following section, unless otherwise is stated, the values Relative Frequency Weight with Floor(RFWwF) will be for the square size of 143km^2 and the *In Area with Floor*(IAwF) radius will be 500m.

ResNet-101 Augmented Model 1:
The ResNet predictions were multiplied with the RFWwF and the IAwF values and normalized between multiplications. The Augmented model 1 gave a increase in Top-1 accuracy of 2.925% and further results can be seen in Table 6.19.

ResNet-101 Augmented Model 2:
The ResNet models predictions were multiplied by the RFWwF and IAwF values and normalized between multiplications. Then added the Top-40 logistic values and *In Area* values for the plants within a 500m radius. These were given their full Scaling Factor of 0.6 and 0.6 respectively. The Augmented model 2 gave a decrease in Top-1 accuracy of -2.275% and further results can be seen in Table 6.19.

ResNet-101 Augmented Model 3:
The ResNet models predictions were multiplied by the RFWwF and IAwF values and normalized between multiplications. Then the Top-40 logistic values were added and *In Area* values for the plants within 500m radius with half weights, 0.2 and 0.2 respectively. The Augmented model 3 gave an increase in Top-1 accuracy of 3.100% and further results can be seen in Table 6.19.

Table 6.19: Result for the Augmented ResNet-101 model 1, 2 and 3 on the Test set. Accuracy for the Top-1, Top-5 and Top-10 predictions and the difference between the Augmented and original model.

ResNet Augmented Model 1		
Top n	Accuracy[%]	Difference[%]
1	73.350	2.925
5	92.500	1.625
10	95.775	0.750
ResNet Augmented Model 2		
Top n	Accuracy[%]	Difference[%]
1	68.150	-2.275
5	81.725	-9.15
10	85.850	-9.175
ResNet Augmented Model 3		
Top n	Accuracy[%]	Difference[%]
1	73.525	3.100
5	87.400	-3.475
10	89.775	-5.250

ResNet-101 Augmented Model 4:
The ResNet models predictions were multiplied by the RFWwF values and normalized. Then added the Top-40 logistic values with half value 0.3. The Augmented model 4 decreased the Top-1 in accuracy by 3.175% and further results can be seen in Table 6.20.

ResNet-101 Augmented Model 5:
The ResNet models predictions were multiplied by the RFWwF and IAwF values and normalizing in between. Then added the Top-40 logistic values with full value of 0.6. The Augmented model 5 gave an increase in Top-1 accuracy of 3.175% and further results can be seen in Table 6.20.

ResNet-101 Augmented Model 6:
The ResNet models prediction multiplied by the RFWwF and IAwF values with radius 1500m and normalized between multiplications. Then added the Top-40 logistic values with full value of 0.6. The Augmented model 5 gave an increase in Top-1 accuracy of 3.000% and further results can be seen in Table 6.20.

Table 6.20: Result for the Augmented ResNet-101 model 4, 5 and 6 on the Test set. Accuracy for the Top-1, Top-5 and Top-10 predictions and the difference between the Augmented and original model.

ResNet Augmented Model 4		
Top n	Accuracy[%]	Difference[%]
1	73.5	3.075
5	90.425	-0.45
10	92.025	-3.0
ResNet Augmented Model 5		
Top n	Accuracy[%]	Difference[%]
1	73.6	3.175
5	89.75	-1.125
10	91.325	-3.7
ResNet Augmented Model 6		
Top n	Accuracy[%]	Difference[%]
1	73.425	3.000
5	89.375	-1.500
10	91.200	-3.825

6.3.6 ResNet-101: Validation set

The ResNet-101 model was run on the Validation Dataset. The Dataset contains 4843 images with location data. 18 species with 50 images, 33 with 49 images, 28 with 48 images, 16 with 47 and 5 species with 46 images. The original ResNet-101 model on the Validation set performed with a Top-1 accuracy of 70.948%, as seen in Table 6.21.

Table 6.21: ResNet-101: Original model classification accuracy on the Validation set

Top n	Accuracy[%]
1	70.948
5	88.003
10	93.083

The results of the Augmented models the Validation Dataset, can be seen in Table 6.22. Where the Augmented model 5(multiplied by RFWwF and IAwF and added 0.4 to Top-40 logistic probabilities) achieved the highest increase in Top-1 accuracy of nearly 4%. In addition to the models described in the previous section achieved The Relative Frequency Weighting with Floor for squares of size 143km^2 model an increase in Top-1 accuracy of 2.065%.

Table 6.22: ResNet-101: Average accuracy of the Original and Augmented models on the Validation set.

Augmented ResNet model 1 on the Validation set		
Top n	Accuracy[%]	Difference[%]
1	74.024	3.076
5	90.316	2.313
10	94.239	1.156
Augmented ResNet model 3 on Validation set		
Top n	Accuracy[%]	Difference[%]
1	74.582	3.634
5	88.788	0.785
10	92.422	-0.661
Augmented ResNet model 4 on the Validation set		
Top n	Accuracy[%]	Difference[%]
1	74.417	3.469
5	91.039	3.036
10	94.652	1.569
Augmented ResNet model 5 on Validation set		
Top n	Accuracy[%]	Difference[%]
1	74.912	3.964
5	90.977	2.974
10	94.26	1.177
Augmented ResNet model 6 on Validation set		
Top n	Accuracy[%]	Difference[%]
1	74.644	3.696
5	90.688	2.685
10	93.744	0.661

6.4 ViT on Species-Observation-Combined data

The Vision Transformer model achieved a Top-1 accuracy of 53.325% on the Test Dataset. This can be seen in Table 6.23, where

Table 6.23: ViT: results on the Test Dataset

Top n	Accuracy[%]
1	53.325
5	78.275
10	85.850

The following methods were tried to increase the accuracy of the Visual Transformer.

6.4.1 ViT: KDE and k-NN

I were not able to increase accuracy for the Top 1, 5 or 10 predictions by using kernel density estimation or k-NN methods for the ViT model either.

6.4.2 ViT: Observed *In Area*

Weighting the predictions by whether or not the species has been observed in the surrounding area, the scaling factor 0.4 was found manually through trial and error by iterating through a number range and choosing the factor which gives the highest accuracy. This increased the Accuracy 1.65% when weighting plants within a 1 km radius. The result of this and for other radii can be seen in Table 6.24. Scaling factor is found by manually optimizing the increase in Accuracy on the test dataset.

Table 6.24: Prediction accuracy of the augmented model when adding a weight to plants found in surrounding area with a factor of 0.4.

Radius[m]	Scaling Factor	Top n	Accuracy[%]	Difference[%]
200	0.4	1	54.65	1.325
		5	76.125	-2.15
		10	83.525	-2.325
500	0.4	1	55.2	1.875
		5	76.175	-2.1
		10	82.95	-2.9
1000	0.4	1	55.1	1.775
		5	77.225	-1.05
		10	83.375	-2.475
1500	0.4	1	55.125	1.8
		5	77.925	-0.35
		10	83.95	-1.9

Table 6.25: Prediction accuracy when multiplied with the *In Area with Floor* values for the areas of different radii. Where the Species observed in the are given the value 1 and the other values are given a floor value.

Radius[m]	Floor value	Top n	Accuracy[%]	Difference[%]
500	0.1	1	55.825	2.5
		5	80.85	2.575
		10	87.725	1.875
1000	0.1	1	56.0	2.675
		5	80.75	2.475
		10	87.975	2.125
1500	0.1	1	55.85	2.525
		5	80.625	2.35
		10	87.55	1.7

6.4.3 ViT: Multinomial logistic model

As with the ResNet model two different methods of weighting the ViT model with the multinomial logistic model values were tried. One by adding the probability from the logistic model multiplied by a scaling factor and one by adding a constant to the top 40 most probable predictions. When adding a scaled probability to the ViT model predictions the accuracy increased with 0.525% on the test set. As seen in Table 6.26

Table 6.26: Accuracy of the augmented model when adding a weight of the predictions by the multinomial logistic model with a scaling factor.

Scale Factor	Top n	Accuracy[%]	Difference[%]
1.6	1	54.375	1.050
	5	74.25	-4.025
	10	80.85	-5.00

Table 6.27: Accuracy of the augmented model when adding 0.4 to the top 40 most probable plants given by the logistic model.

Table 6.28: Accuracy for the augmented

Scale Factor	Top n	Accuracy[%]	Difference[%]
0.4	1	54.875	1.55
	5	77.425	-0.85
	10	82.95	-2.9

Again when weighting the 40 most probable species according to the logistic model with a scaling factor, the accuracy increased by more than when adding the scaled probabilities, this time increasing the accuracy with 1.575% to 65.316%. The Top-5 and 10 accuracy also decreases as before, as seen in Table 6.28.

6.4.4 ViT: Relative Frequency Weighting

The relative frequency weighting of the predictions can be seen in Table 6.29. This was found that when weighting without a floor. The partition with squares of size 294km^2 or 225km^2 gave the greatest increase in accuracy in the species predictions with 2.725%. For RFWwF squares with of same size gave the highest increase in accuracy with 2.650%, which is slightly less than than the RFW.

Table 6.29: ViT: Accuracy of the **Relative Frequency Weighting without Floor**(left) and Accuracy of the **Relative Frequency Weighting with Floor**(right)

Square	Top	Accuracy[%]	Difference[%]
403km^2	1	55.800	2.475
	5	80.325	2.050
	10	87.225	1.375
294km^2	1	55.975	2.650
	5	80.375	2.100
	10	87.225	1.375
225km^2	1	55.725	2.400
	5	80.0	1.725
	10	86.9	1.05
177km^2	1	56.050	2.725
	5	79.975	1.700
	10	87.175	1.325
143km^2	1	55.800	2.475
	5	79.375	1.100
	10	86.700	0.850

Square	Top	Accuracy[%]	Difference[%]
403km^2	1	55.650	2.325
	5	80.3	2.025
	10	87.375	1.525
294km^2	1	55.775	2.450
	5	80.550	2.275
	10	87.525	1.675
225km^2	1	55.450	2.125
	5	80.475	2.200
	10	87.375	1.525
177km^2	1	55.975	2.650
	5	80.550	2.275
	10	87.575	1.725
143km^2	1	55.950	2.625
	5	80.300	2.025
	10	87.525	1.675

6.4.5 ViT: Combining *In Area*, Logistic model and Relative Frequency Weighting

The location based methods which increased accuracy on the Test Dataset the most were combined. With some exceptions. Mainly using the Relative Frequency Weight with Floor(RFWwF) value instead of the Relative Frequency Weight without Floor(RFW). This is based on the belief that RFWwF will handle new observations in an area better. It also performed better for the ResNet-101 model and the difference for the ViT model is quite small at only 0.008%. The augmented model with the seemingly most robust location augmentation was also included, using the largest radius for the *In Area with Floor*(IAwF) for the multiplication augmentation on the proposed probabilities by the original ViT model. Unless otherwise is stated the radius for IAwF is 1000m.

ViT Augmented Model 1:
The ViT predictions with the RFWwF and the (IAwF) values were multiplied with a normalizing step between multiplications. With a RFWwF square size of 177km^2 and the *In Area with Floor* values were 1 for plants within a 1000m radius and 0.1 value for plants not within the radius. The Augmented model 1 gave an increase in Top-1 accuracy of 3.475% and further results can be seen in Table 6.30.

ViT Augmented Model 2:
The ResNet models predictions were multiplied by the RFWwF and IAwF values and normalized between multiplications. Then the Top-40 logistic values and *In Area* values for the plants within a 1km radius were added. These were given their full Scaling Factor of 0.4 and 0.4 respectively. The Augmented model 2 gave a increase in Top-1 accuracy of 3.625% and further results can be seen in Table 6.30.

ViT Augmented Model 3:
The ResNet models predictions were multiplied by the RFWwF and IAwF values and normalized between multiplications. Then the Top-40 logistic and the *In Area* values with half weight were added, 0.4 and 0.4 respectively. The Augmented model 3 gave an increase in accuracy of 1.2% and further results can be seen in Table 6.30.

Table 6.30: Result for the Augmented ViT model 1, 2 and 3 on the Test Dataset. Accuracy for the Top-1, Top-5 and Top-10 predictions and the difference between the Augmented and original model.

ViT Augmented Model 1		
Top n	Accuracy[%]	Difference[%]
1	56.8	3.475
5	81.925	3.65
10	88.925	3.075
ViT Augmented Model 2		
Top n	Accuracy[%]	Difference[%]
1	56.95	3.625
5	74.925	-3.35
10	80.575	-5.275
ViT Augmented Model 3		
Top n	Accuracy[%]	Difference[%]
1	57.1	3.775
5	76.975	-1.3
10	82.125	-3.725

ViT Augmented Model 4:
The ResNet models predictions were multiplied with the RFWwF values and the product normalized. Then the Top-40 logistic values with half value 0.2 were added. The Augmented model 4 gave a decrease in Top-1 accuracy of 3.775% and further results can be seen in Table 6.31.

ViT Augmented Model 5:
The ResNet models predictions were multiplied by RFWwF and the IAwF values and normalized between multiplications. Then the Top-40 logistic values with full value of 0.4 were added. The Augmented model 5 gave am increase in accuracy of 3.875% and further results can be seen in Table 6.31.

ViT Augmented Model 6:
The ResNet models predictions are multiplied with the RFWwF and the IAwF values for 1500 meter radius and normalized the product in between. Then the Top-40 logistic values with full value of 0.4 are added. The Augmented model 6 gave am increase in accuracy of 4.025 % and further results can be seen in Table 6.31.

ViT Augmented model 7:
ResNet models predictions were multiplied with the IAwF values for 1000 meter radius and normalized the product. Then the Top-40 logistic values with full value of 0.4 were added. The Augmented model 6 gave am increase in accuracy of 3.075 % and further results can be seen in Table 6.31.

Table 6.31: Result for the Augmented ViT model 4, 5, 6 and 7 on the Test set. Presenting the accuracy for the Top-1, Top-5 and Top-10 predictions and the difference between the Augmented and original model.

ViT Augmented Model 4		
Top n	Accuracy[%]	Difference[%]
1	57.100	3.775
5	80.275	2.000
10	85.050	-0.800
ViT Augmented Model 5		
Top n	Accuracy[%]	Difference[%]
1	57.200	3.875
5	79.525	1.25
10	84.275	-1.575
ViT Augmented Model 6		
Top n	Accuracy[%]	Difference[%]
1	57.350	4.025
5	79.650	1.375
10	84.200	-1.650
ViT Augmented Model 7		
Top n	Accuracy[%]	Difference[%]
1	56.400	3.075
5	79.350	1.075
10	84.825	-1.025

6.4.6 ViT: Validation Set

The ViT model was run on the Validation Dataset. The Dataset contains 4843 images with location data. 18 species with 50 images, 33 with 49 images, 28 with 48 images, 16 with 47 and 5 species with 46 images. The result of the ViT model on the Validation Dataset can be seen in Table 6.32

Table 6.32: ViT: results on the Test Dataset

Top n	Accuracy[%]
1	53.325
5	78.275
10	85.850

The results of the Augmented ViT models on the Combined Validation Dataset, can be seen in Table 6.33. Where the ViT Augmented Model 7(multiplied by IAwF with 1000m radius and added 0.4 to Top-40 logistic probabilities) achieved the higest additional Top-1 accuracy of 3.696% followed by the ViT Augmented model 5(multiplied by RFWwF and IAwF and added 0.4 to Top-40 logistic probabilities) at 3.551% additional Top-1 accuracy.

6.5 Neural Networks with Location Data

Several attempts at combining the computer vision models from Section 6.2 were done. This was done in the same way as described in Section 2.2.4, but were not able to achieve a higher accuracy for any of the models. The architecture that was used was freezing the weights of the ResNet or ViT model and take that models output as well as the additional location-data into 4 to 6 fully connected layers. Tried several different versions of combining the data. One way was using the same data used in the multinomial logistic model and adding the categorical data as one-hot arrays to the original output of the ResNet and/or ViT. Another way of combining the location and image data in the model was to include the numerical values Latitude, Longitude and Height as the additional location data, but none performed better than the original model.

Table 6.33: Result for the Augmented ViT model 1, 2, 3, 4, 5, 6 and 7 on the Validation Dataset. Presents the accuracy for the Top-1, Top-5 and Top-10 predictions and the difference between the Augmented and Original model.

ViT Augmented Model 1		
Top n	Accuracy[%]	Difference[%]
1	59.075	2.581
5	82.284	3.779
10	88.416	2.416
ViT Augmented Model 2		
Top n	Accuracy[%]	Difference[%]
1	59.839	3.345
5	79.434	0.929
10	85.154	-0.846
ViT Augmented Model 3		
Top n	Accuracy[%]	Difference[%]
1	59.632	3.138
5	80.446	1.941
10	86.124	0.124
ViT Augmented Model 4		
Top n	Accuracy[%]	Difference[%]
1	59.509	3.015
5	83.316	4.811
10	89.036	3.036
ViT Augmented Model 5		
Top n	Accuracy[%]	Difference[%]
1	60.045	3.551
5	83.275	4.770
10	88.994	2.994
ViT Augmented Model 6		
Top n	Accuracy[%]	Difference[%]
1	59.88	3.386
5	83.213	4.708
10	88.871	2.871
ViT Augmented Model 7		
Top n	Accuracy[%]	Difference[%]
1	60.190	3.696
5	82.841	4.336
10	89.180	3.180

Chapter 7

Discussion

Will here discuss the result from the different experiments as well as general challenges when working with crowd-sourced data and different data sources.

7.1 Preliminary experiments:

7.1.1 SVM

While the Support Vector Machine achieved an accuracy of 98.604% on the Flavia Dataset the SVM model performed significantly worse on the sampled subsets of the LeafSnap Dataset, where the accuracy was at its best 76.9% and on average 72.3%. There are many possible reasons for this. The main reason is most probably that the features are selected in order to fit the Flavia Dataset. The Flavia Dataset consists of high quality scans of leafs on a plain white background, where every scan has the dimensions 1600x1200 and every plant species, except one, has a simple leaf(not divided). While the scans from the LeafSnap Dataset have a varying dimensions, but generally around 800x600, which is half the resolution of the Flavia Set. This difference is clear when comparing the images in the Figure 5.1 and 6.1 for four images of leaves form the Flavia and LeafSnap Dataset respectively.

The LeafSnap Dataset also contains a many compound leaves which the features for the SVM are not intended to be able to discriminate. In addition to compound leaves the LeafSnap also contains three different kinds of spruce(Norway spruce, Oriental spruce and Blue Spruce) and their needle-like leaves(seen in Figure 6.1(a)) are also not intended for the model.

As seen in Table 6.2, the SVM decreases in accuracy by 1.9% per 10 additional plants introduced to the model. Have not been able to find SVMs with handcrafted features,such as the ones used here, being used in literature on the LeafSnap Dataset. SVMs have however been used as a final classification step after extracting the features using a convolutional neural networks [35]. Through this Turkoglu et al. 35 achieved an accuracy of 94.38% on the LeafSnap Dataset. The SVM method for classification of the Flavia Dataset has also in recent years(2017) been outperformed by CNNs[36].

As seen in the Tables **??**, the inclusion of height and/or Latitude features from the sampled plants, in the LeafSnap Dataset, does not increase accuracy. It does rather the opposite where every SVM with height and/or Latitude features has a lower accuracy when used on 20 or 30 species. This points toward the need for more data processing before these location based values can provide any information about their connected species.

7.1.2 Leaf Recognition with CNN and Semi-Simulated Location Data

All of the trained ResNet models achieved a Top-1 accuracy of about 90% and Top-5 at about 99%. Selected the simplest model ResNet-18 due to the principle of parsimony and chose the model parameters from the epoch with accuracy on the test set, to counteract over-training. From further training the ResNet-18 model reached a Top-1 accuracy of 93.38% and Top-5 accuracy of 99.378%.

One quite big oversight here was not a buffer around the square of interest as seen in Figure 6.4b. This means that points close to the border will loose the influence from points outside the area which could lead to worse results when using location based methods. This oversight was rectified into account later when the Relative Frequency Weight was calculated. The RFW used the surrounding squares when calculating the RFW and by that avoiding inaccuracies due to the location point being close to the the square border.

Adding Kernel Density Estimation(KDE)

I did not manage to achieve any increase in accuracy by adding Kernel Density Estimation(KDE) values to the original ResNet-18 predictions. The parameters for the KDE which gave the lowest decrease in accuracy, was when the bandwidths were larger than the sides of the surrounding square. Means that the there is little if any variation in in the KDE value across the square surrounding the location of the observation. This would end up giving approximately the same value for every species, but not exactly since the accuracy is slightly lower.

Adding k-NN Values

By adding the k-NN values, scaled by 0.7, to the probabilities given by the ResNet-18 model, we were able to increase the Top-1 accuracy on average 0.09%. This is only a slight increase, but it shows that it might be possible to increase the accuracy. The accuracy increase goes down slightly when the number of neighbours taken into account is raised.

SMOTE was used to handle the class inequality in the Species-Observation-185 Dataset. This seemed like an interesting solution to the problem of unbalanced classes, but later proved to be difficult to utilize on the Combined Dataset. Used the K-means SMOTE algorithm and that algorithm begins by begins by applying a K-means clustering on the data before oversampling within the clusters to balance the classes[37]. On the relatively concentrated Species-Observation-185 Dataset, which only has data-points with location around Oslo, the K-means SMOTE algorithm was possible, but on the Combined Dataset, which has points all over Norway, this was more troublesome. On the Combined Dataset the algorithm found, for some classes, no cluster with sufficient samples to perform the SMOTE Sampling. Could probably work around this by "lowering the cluster_balance_threshold or increasing the number of clusters"(python RuntimeError message), but had at that point found an article which stated that that SMOTE was often outperformed by undersampling the majority classes[6] and with barely positive results on the LeafSnap images with locations from Species-Observation-185 Dataset and negative results for the undersampling on the Combined set, I decided to move away from k-NN as a possible method for increasing accuracy. Did therefor not try undersampling for the k-NN method on the Species-Observation-185 Dataset.

Adding *In Area with Floor* Weight Values

When the *In Area with Floor* values were added to the predictions, the Top-1 accuracy increased only slightly, by 0.011. In addition to this it did this while achieving the same Top-5 accuracy as before. In comparison with the k-NN augmented results, where the Top-5 decreased for every k value, adding the *In Area with Floor* values to the predictions seems safer, in that it might perform the same or better as when *In Area with Floor* is added compared to the original model.

The *In Area with Floor* values multiplied with the original model gave also only a very slight increase in Top-1 accuracy of 0.017% and the *In Area with Floor* augmentation did not decrease the Top-5 accuracy at any point.

Knowing what we know now from the Species-observation experiments, it seems possible that the concentration of location-points in the "Oslo area" (Figure 6.4b) is to high for the

In Area with Floor to produce any positive results. Either that or the fact that the ResNet-18s Top-1 accuracy is much higher for the LeafSnap test set. The ResNet-18(LeafSnap) classifies about 392 images incorrectly while the ResNet-101 and ViT models classifies 1406 and 2260 images incorrectly, respectively. Some quick math shows that the fraction of the wrongly classified images, by the original ViT, the augmented ViT model classifies correctly is about ten times higher than the fraction the augmented ResNet-18(LeafSnap) classifies correctly. This points towards a concentration to high for the *In Area with Floor* values to do any good. Given more time it could be interesting to map the areas where multiplying the models predictions with the *In Area with Floor* values increases accuracies and where it does not.

7.2 Species-Observation with Location

Both the ResNet-101 and the ViT model classified the majority of the images correctly. With the ResNet-101 achieving a significantly higher accuracy. Especially on the retrained models which were used in the augmented models. The fact that the ViT models accuracy decreased around 10% when retraining suggest that it it hit a local minima during training and were unable to escape. Given more time could the model could probably have reached a higher accuracy through retraining or possibly through simulated annealing.

For the location augmented models, the augmented Model 1 and 5 were the only models which consistently increased not only the Top-1 accuracy but also the Top-5 and Top-10 accuracy. It looks like the RFWwF and IAwF values increases the accuracy across the board, while the logistic Top-40 only consistently increased the Top-1 accuracy.

The Relative frequency weighting here is based on the geomodel to iNaturalis. It is probably more useful when looking at at larger areas such as continents in comparison with small countries.

Multinomial Logistic model

Other things which could have made the multinomial logistic model perform better is the inclusion of more data. One datapoint which was not used was the date of the observations which probably be useful since for exampel flowers are not photographed during the winter.

Other then this a thorough analysis of the logistic model should be done to remove covariants which does not provide any information. This is not possible to do with the python package sklearn for logistic regression and

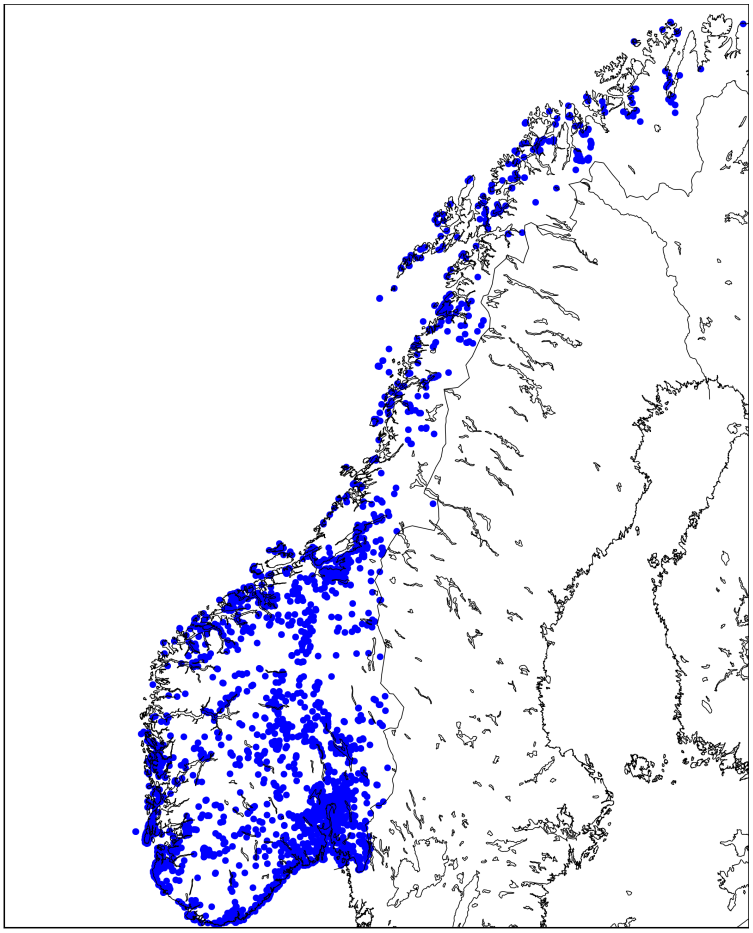


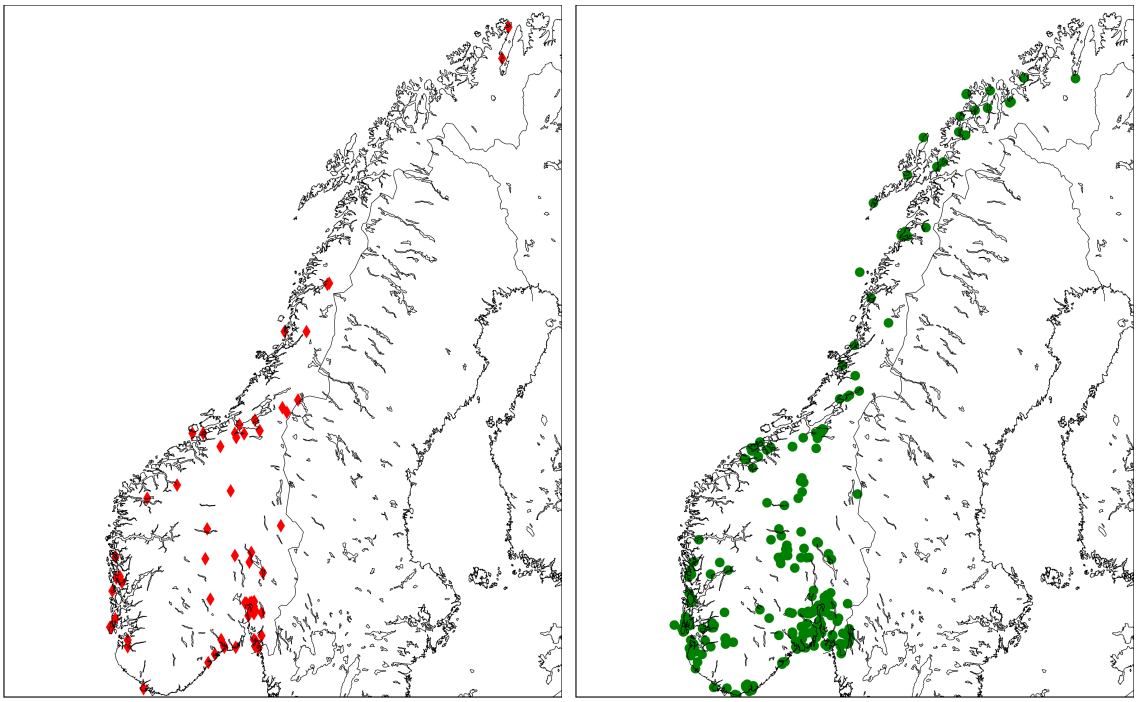
Figure 7.1: Distribution of the Combined Validation Set.

7.3 Further steps

As seen in Figure 7.1 the Combined Validation Dataset has data-points all over the country and there is no clear connection in Figures 7.2 and 7.3 where points are classified correctly because of the location augmentation and where they are wrongly classified because of the location model. This could be investigated further to see if high density areas could have a smaller radius for the IAwF values and/or smaller square for the RFWwF values.

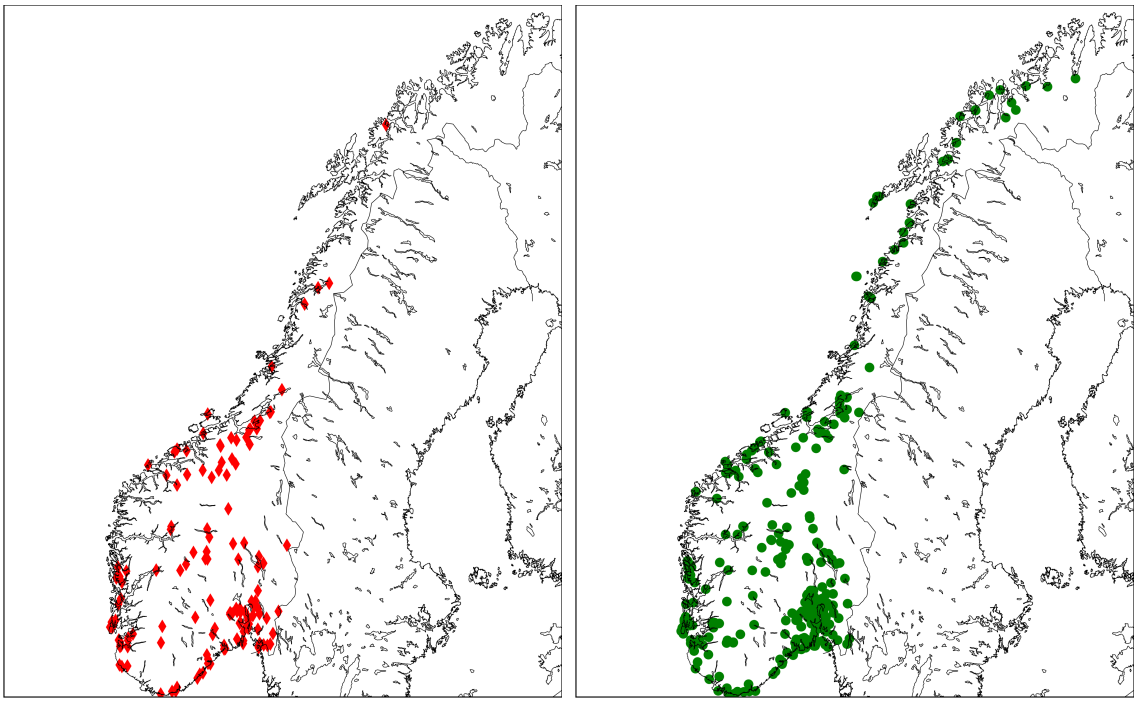
7.4 General Challenges

One general problem when working with a lot of different data and datasources is reliability. Since most of the data Collected for the thesis is crowd-sourced(species-observations)with



(a) ResNet: Wrong with Augmented Model (b) ResNet: Correct with Augmented model

Figure 7.2: a) Distribution of the 156 additional observations which were wrongly classified as a result of the ResNet-101 Augmentation model. b) Distribution of the 328 additional observations which were correctly classified as a result of the ResNet-101 Augmentation model 5.



(a) ViT: Wrong with Augmented Model (b) ViT: Correct with Augmented model

Figure 7.3: a) Distribution of the 156 observations which were wrongly classified as a result of the ViT Augmentation model. b) Distribution of the 328 observations which were correctly classified as a result of the ViT Augmentation model 5.

(a) Elm 1 (b) Elm 2

Figure 7.4: Two different observations of Elm. Both Images published under the creative commons license.

little professional involvement, there is a fair amount of uncertainty connected to the data. This applies to both the identification of species in the Dataset and the logging of their location. For some of the location there is an accuracy estimate, but since this was only present in around half the locations I decided not to try to use that data. Another source of uncertainty is the different people have different equipment and have different focus. Because of this and few guidelines as of how the registration images should be taken, there can be a large variation within the images taken of the same species. As seen in Figure 7.4. This is both good for machine learning since a large variety of images should make the model more robust when trained upon, but worse when you would like to identify the image since.

Another difficulty when working with crowd-sourced data is the fact that ther is a bias towards pretty and/or recognizable plants, as briefly discussed in section 3.3.

Another challenge in this project was knowing what data to use and what data to discard. Did for example not use any research connected to the plantClef, which is a yearly plant identification challenge with over 15k plant species and about 1 million images. There were also other datasets which could be usefull. The main one is data from the Global Biodiversity Information Facility(GBIF) "is an international network and data infrastructure funded by

the world's governments and aimed at providing anyone, anywhere, open access to data about all types of life on Earth"[38]. They provide access to an API through which it seems possible to get more information. Even though most of the data registered at GBIF is provided by the Norwegian Species Observation Service. GBIF also recives data from PlantNet and iNaturalist.

7.4.1 Other remarks

Many of the decisions for how to train were models were inherited form the the Git-Hub code for Deep LeafSnap[32]. This is the reason that Nesterov-momentum was used during training, even though it does not improve the rate of convergence for stochastic gradient decent [15]. This was first discovered after training most, if not all of the models used in this thesis.

KDE calculation was very inefficient and therefor not possible to do for a greater number of area and bandwidth combinations. Could for example have been done with the same method as with the relative frequency where we calculate for the whole area at once using geographical squares surrounding the points of interest, but the disappointing results of the KDE method shifted focus elsewhere. Specifically towards weighting plants found in the area without considering the distance between the plants. This was because the KDEs which gave the best results had a bandwidth larger than the area in question, meaning the estimation more or less only tells us if the plant is in the area or not. Which is what the in Area modulation does with the original model.

Other inefficiencies were the iteration through the polygons provided by the AR-50 and Landscape Datasets. Since we for every point check against every polygon until it finds the polygon the point is within, the run-time for this is O(n). This could be brought closer to log(n) by sorting the polygons by their northern most coordinate first, and then by their eastern most coordinate. While not necessary for the Speceis-Observation-Combined Dataset, it would make it easier to give the about 800000 points in the Species-Observation-100 Dataset auxiliary information such as landscape type, bonity and above or below the tree line. This could again give a better statistical basis for the multinomial logistic model.

Chapter 8

Conclusion

When it comes to plant identification using computer vision, while powerful classical methods such as Support Vector Machines, k-NNs and Random Forests based on handcrafted extracted features have been overtaken by newer Deep Learning methods.

Deep Learning methods such as Residual Neural Networks performs well on images and scans taken in a laboratory setting, but the accuracy is significantly reduced when plant images are taken in nature. Some of the gap in accuracy between laboratory and nature setting can be bridged through the use of additional information such as Location.

Were able to increase the accuracy of two different deep learning computer vision models between 3.5 and 4.0 %, through a combination of Relative Frequency Weighting, weighting based on the surrounding plants and statistics based on coordinates, landscape and altitude of the observation.

Bibliography

[1] Scott Loarie. inaturalist as a tool for conservation impact and the role of machine learning, 2023. URL https://www.youtube.com/watch?v=nsxayWTucco.

[2] Adam G. Hart, Hayley Bosley, Chloe Hooper, Jessica Perry, Joel Sellors-Moore, Oliver Moore, and Anne E. Goodenough. Assessing the accuracy of free automated plant identification applications. *People and nature (Hoboken, N.J.)*, 5(3):929–937, 2023. ISSN 2575-8314.

[3] Annette J. Dobson. An introduction to generalized linear models, 2008.

[4] Gareth James. An introduction to statistical learning : With applications in python., 2023.

[5] N V Chawla, K W Bowyer, L O Hall, and W P Kegelmeyer. Smote: Synthetic minority over-sampling technique. *The Journal of artificial intelligence research*, 16:321–357, 2011. ISSN 1076-9757.

[6] Rok Blagus and Lara Lusa. Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):106–106, 2013. ISSN 1471-2105.

[7] Maria L. Rizzo. Statistical computing with r, 2007.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] LunarLullaby. File:resblock.png, 2015. URL https://commons.wikimedia.org/w/index.php?curid=131458370.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[13] Mohsen Ghafoorian, Nico Karssemeijer, Tom Heskes, Inge W M van Uden, Clara I Sanchez, Geert Litjens, Frank-Erik de Leeuw, Bram van Ginneken, Elena Marchiori, and Bram Platel. Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities. *Scientific reports*, 7(1):5110–12, 2017. ISSN 2045-2322.

[14] Rajalingappaa Shanmugamani. Deep learning for computer vision : expert techniques to train advanced neural networks using tensorflow and keras, 2018.

[15] Ian Goodfellow. Deep learning, 2016.

[16] PyTorch Contributors. CROSSENTROPYLOSS documentation, 2023. URL `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`.

[17] Jana Waldchen and Patrick Mäder. Plant species identification using computer vision techniques: A systematic literature review. *Archives of computational methods in engineering*, 25(2):507–543, 2018. ISSN 1134-3060.

[18] Pornntiwa Pawara, Emmanuel Okafor, Olarik Surinta, Lambertus Schomaker, and Marco A Wiering. Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition. In *International Conference on Pattern Recognition Applications and Methods*, 2017. URL `https://api.semanticscholar.org/CorpusID:215763094`.

[19] Jana Wäldchen, Michael Rzanny, Marco Seeland, and Patrick Mäder. Automated plant species identification-trends and future directions. *PLoS computational biology*, 14(4): e1005993–e1005993, 2018. ISSN 1553-7358.

[20] Yu Sun, Yuan Liu, Guan Wang, and Haiyan Zhang. Deep learning for plant identification in natural environment. *Computational Intelligence and Neuroscience*, 2017:7361042–6, 2017. ISSN 1687-5265.

[21] Jana Wäldchen and Patrick Mäder. Machine learning for image based species identification. *Methods in ecology and evolution*, 9(11):2216–2225, 2018. ISSN 2041-210X.

[22] Antoine Affouard, Hervé Goëau, Pierre Bonnet, Jean-Christophe Lombardo, and Alexis Joly. Pl@ntNet app in the era of deep learning. In *ICLR: International Conference on Learning Representations*, Toulon, France, April 2017. URL `https://hal.science/hal-01629195`.

[23] Elijah Cole, Grant Van Horn, Christian Lange, Alexander Shepard, Patrick Leary, Pietro Perona, Scott Loarie, and Oisin Mac Aodha. Spatial implicit neural representations for global-scale species mapping, 2023.

[24] iNaturalist. File:fig:naturalis_expected_nearby.png, 2023. URL https://www.inaturalist.org/geo_model/43188/explain.

[25] Wouter Koch, Laurens Hogeweg, Erlend B Nilsen, and Anders G Finstad. Maximizing citizen scientists' contribution to automated species recognition. *Scientific reports*, 12 (1):7648–7648, 2022. ISSN 2045-2322.

[26] Åpent API for høyde- og dybdedata fra Kartverket 1.0.1 . https://ws.geonorge.no/hoydedata/v1/, 2023. Accessed: 2023-09-12.

[27] Eva S. Flo Heggem. Ar50 – arealressurskart i målestokk 1:50 000. et heldekkende arealressurskart for jord- og skogbruk, 2019.

[28] Trond Simensen, Lars Erikstad, and Rune Halvorsen. Nin landskapstyper - en kort innføring. URL https://www.artsdatabanken.no/Files/29394/NiN_Landskapstyper_-_en_kort_innf_ring.pdf.

[29] Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida C. Lopez, and João V. B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 502–516. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 9783642337086.

[30] The Editors of Encyclopaedia Britannica. Taxonomy. URL https://www.britannica.com/plant/tracheophyte.

[31] Aayush Gupta and Oana Florescu. Plant-Leaf-Identification, 08 2020. URL https://github.com/AayushG159/Plant-Leaf-Identification.

[32] Sujith Vishwajith. Deep-Leafsnap, 05 2017. URL https://github.com/sujithv28/Deep-Leafsnap.

[33] Emmanuel Adetiba, Oluwaseun T. Ajayi, Jules R. Kala, Joke A. Badejo, Sunday Ajala, and Abdultaofeek Abayomi. Leafsnapnet: An experimentally evolved deep learning model for recognition of plant species based on leafsnap image dataset. *Journal of Computer Science*, 17(3):349–363, Apr 2021. doi: 10.3844/jcssp.2021.349.363. URL https://thescipub.com/abstract/jcssp.2021.349.363.

[34] Dag Øystein Hjermann and John Magne Grindeland. platanlønn i store norske leksikon på snl.no., 2023. URL https://snl.no/platanl%C3%B8nn.

[35] Muammer Turkoglu, Muzaffer Aslan, Ali Arı, Zeynep Mine Alçin, and Davut Hanbay. A multi-division convolutional neural network-based plant identification system. *PeerJ. Computer science*, 7:e572–e572, 2021. ISSN 2376-5992.

[36] Syed Umaid Ahmed, Junaid Shuja, and Muhammad Atif Tahir. Leaf classification on flavia dataset: A detailed review. *Sustainable computing informatics and systems*, 40: 100907, 2023. ISSN 2210-5379.

[37] Felix Last, Georgios Douzas, and Fernando Bacao. Oversampling for imbalanced learning based on k-means and smote. *arXiv.org*, 2017. ISSN 2331-8422.

[38] GBIF. What is gbif?, 2023. URL `https://www.gbif.org/what-is-gbif`.