LEARNING-BASED CROP MANAGEMENT OPTIMIZATION USING MULTI-STREAM CONVOLUTIONAL NEURAL NETWORKS

BY

ALEXANDRE BARBOSA

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Professor Naira Hovakimyan, Chair
Professor Nicolas Martin
Professor Srinivasa Salapaka
Professor Petros Voulgaris
Professor Humphrey Shi

ABSTRACT

Improving crop management is an essential step towards solving the food security challenge. Despite the advances in precision agriculture, new methods are needed to create decision-support systems to help farmers increase productivity while accounting for environmental impacts and financial risks. This dissertation presents a class of learning-based optimization algorithms for spatial allocation of crop inputs, and a new framework for online coverage path planning with potential use in tasks such as planting and harvesting. The proposed algorithms use Multi-stream Convolutional Neural Networks (MSCNN) to learn relevant spatial features from the environment and use them to optimize the available control inputs. In the crop inputs optimization problem, an MSCNN combines five input variables as in a regression problem to better predict yield. The predictive model is then used as the base of a gradient-ascent algorithm to maximize a custom objective function. To leverage the applicability of this algorithm, a risk-aware version of this method is also proposed. The predictive uncertainty is measured and used as a constraint to comply with different levels of risk-aversion. Experiments with real crop fields demonstrate that this method significantly reduces the yield prediction errors when compared to the state of the art algorithms. Results from the optimization algorithm show an increase in the expected net revenue of up to 6.8% when compared with the status quo management while providing safety bounds.

In the coverage path planning framework, an MSCNN agent learns a control policy from demonstrations of paths obtained offline through heuristic algorithms, by using imitation learning. The resulting control policy is further improved through policy-gradient reinforcement learning. Simulations show that the improved control policy outperforms the offline algorithms used during the imitation learning phase, and that the proposed framework can be easily adapted to different cost functions.

"It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no telling where you might be swept off to."

J.R.R. Tolkien

In dedication to my wife Dianna with whom I share an unexpected journey.

ACKNOWLEDGMENTS

Firstly, I want to express my sincere and profound gratitude to my adviser Prof. Naira Hovakimyan. Her guidance, trust, and academic excellence inspired me to leave the ordinary and work hard to create impact. I will carry her lessons through my future carrier, and I will always be thankful and honored for being her student.

I would like to thank my co-adviser Prof. Nicolas Martin for always been supportive and for teaching me the necessary tools for collaborative work in crop sciences. I am also grateful to the other members of my Ph.D. committee, Prof. Srinivasa Salapaka, Prof. Petros Voulgaris, and Prof. Humphrey Shi, for their insightful feedback on my research.

I want to particularly thank my friend and lab-mate Thiago Marinho, who encouraged me to pursue my Ph.D. degree, and altruistically helped me to succeed. I also want to extend my gratitude to the other members of Prof. Hovakimyan's research group for creating such a good and productive environment for research. I am especially grateful for some of my lab-mates and friends with whom I interacted the most: Gabriel, Andrew, Arun, Venanzio, Hamid, Hyung-Jin, Zhuohuan, Javier, Donglei, and Mitchell. They all made my life in grad school much more enjoyable and were always there for a productive discussion. Also, I want to thank the postdocs in our group: Aditya, Pan, and Hunmin, for their incentive and help in my work.

Last, but by no means least, I want to deeply thank my parents Denise

and Conrado, for their unconditional love and for their efforts on providing me all the tools and moral principles necessary to achieve my goals; and my sisters, Patricia and Fabiola for being whole models to me. Most of all, I am heartfelt grateful to my beloved wife Dianna, to whom I dedicate this dissertation. This journey would not have been possible without her love, patience, and friendship.

TABLE OF CONTENTS

NOTATIONS, SYMBOLS, AND ACRONYMS

$\mathbb{R}^n$	n-dimensional Euclidean space.
$\mathbb{R}^{n\times m}$	Set of all $n \times m$ real matrices.
$\mathbb{N}$	Set of natural numbers.
$\lvert \cdot \rvert$	Absolute value.
∇	Gradient operator.
$\mathbb{E}$	Expectation operator.
$\mathcal{N}$	Gaussian distribution.
$\mathbb{1}_A$	Indicator function of the event set A.
ANN	Artificial Neural Network.
CPP	Coverage Path Planning.
DDS	Decision-Support System.
GA	Genetic Algorithm.
GP	Gaussian Process.
MSCNN	Multi Stream Convolutional Neural Network.
NLL	Negative Log-Likelihood.

OFPE	On-Farm Precision Experiment.
PA	Precision Agriculture.
RA-CNN	Risk Averse Convolutional Neural Network.
RF	Random Forest.
RL	Reinforcement Learning.
RMSE	Root Mean Squared Error.
SVM	Support Vector Machine.
VaR	Value at Risk

CHAPTER 1

INTRODUCTION

Improving crop management is an essential step towards solving the food security problem [1]. Traditional farm management practices have led to several environmental and production problems, including excessive fertilization of crops and soil compaction. The first generates a surplus nutrient flow that pollutes the water system [2] and often reduces productivity, while compaction has negative effects in the soil's biotic activity [3]. Although Precision Agriculture (PA) has been consolidated as a management strategy to tackle some of these problems [4, 5], the improvement of decision tools is yet to reach its full potential [6]. New methods are needed to take full advantage of new field technologies (e.g., variable-rate applicators, RTK localization systems, and yield monitors) and create Decision-Support Systems (DSS) to help farmers increase production while accounting for environmental impacts [7]. Also, the diffusion of new PA technologies has been limited by a set of multidimensional factors ranging from socio-economic to technological [8]. Previous studies [9, 10] have demonstrated that farmer's financial risk perception is one of the top reasons many PA solutions are not adopted, even when they present significant expected profitability advantages. Then, a technology that provides safety bounds when changing the status quo is more likely to be adopted, even that it demonstrates a slightly reduced profitability when compared to other emergent solutions.

To optimize crop fertilization rates, descriptive and predictive models of

yield response to crop input management are necessary. As the decision on the rates of fertilizer and seeds is taken at the beginning of the growing season, a model for forecasting yield based on data available at such time is a key element for improving crop management. Many models mapping environmental and management variables to crop yield have been proposed [11]. They can be separated into statistical models [12, 13], and analytic crop models [14]. While analytic crop models are dynamic system simulations based on variables that may not be measurable by farmers, statistical models are constrained by the representativeness of the collected data. On-Farm Precision Experimentation (OFPE) is often used to improve statistical models [15,16] by generating site-specific representative data. However, even at a field scale, the spatial structure of environmental and management variables may affect the yield through events such as nutrient and water transportation [17]. Therefore, the spatial structure of environmental and treatment variables plays an important role when trying to create a predictive model for yield. Moreover, the interaction between different explanatory variables may depend on such spatial structures in a nonlinear way, and an efficient way to extract relevant spatial features from the data is needed.

The intensive traffic of tractors is one of the main causes of soil compaction [18]. Optimizing the path used in area coverage tasks such as spraying and harvesting is necessary to reduce the number of passes over the same place, and hence reduce soil compaction. Besides, the level of compaction also depends on the path's slope, being high slopes usually associated with high levels of soil compaction and erosion. Coverage algorithms are classified as online or offline, as proposed by [19]. Opposed to online, the offline algorithms assume full prior knowledge and stationarity of the environment, which may be an unfeasible assumption for crop fields. The optimality of

paths obtained offline is often very sensitive to even slight changes in the environment, which in case of crop fields may be a result of water accumulation and other obstacles in the field. Nevertheless, they commonly achieve better performance over online algorithms at the price of taking much longer processing time. Online algorithms are, in general, based on offline heuristic approximations. They use hand-crafted inference from the environment 's spatial structure to define a strategy [20], which may create limitations under certain scenarios. Ideally, the performance of an offline solution should be achieved by an online algorithm in similar environments.

Solutions for both problems rely on efficiently extracting features from the spatial structure of the data/environment. A similar spatial feature extraction problem is also present in image recognition software, where Convolutional Neural Networks (CNN) have demonstrated significantly higher performance over other methods [21]. Convolutional layers can be trained to encode relevant visual (and here we can also use the term "spatial") features of varying complexity when combined with fully connected layers. When input variables are not correlated, Multi-Stream Convolutional Neural Network (MS-CNN) architectures have demonstrated potential benefits over more traditional CNN. To the best of author's knowledge, MS-CNN have not been used to learn relevant spatial features from different and weakly correlated crop input variables as in a regression problem, and neither the structure of an environment to design coverage algorithms. In this context, this dissertation explores the use of MS-CNN as a base of DSS for crop input management.

Many studies have emerged in PA aiming to create data-driven DSS for choosing crop inputs based on yield prediction models [22]. The quality of such DSS heavily depends on the model's accuracy on predicting yield, and on the system's ability to use this model to find the optimal input management

to achieve the desired result. For instance, simple methods such as quadratic [23] and piecewise linear [24] regression provide straightforward optimization strategies, but present low accuracy when predicting yield. On the other hand, Machine Learning (ML) approaches have higher prediction accuracy, but pose a more challenging optimization problem due to their complex form. Also, to increase the chances of diffusion of the DSS, the yield model, as well as the optimization algorithm, must account for the uncertainty in both data and predictions. Then, to take full advantage of the performance of CNN in practical applications, we need a proper framework for uncertainty quantification and optimization.

This dissertation proposes two learning-based optimization algorithms using MS-CNN architectures for improving fertilizer rates and the path used in coverage tasks. First, a yield prediction model is designed to extract and combine spatial features present in different environmental and manageable crop input variables that are available at the beginning of the growing season. Then, this model composes the objective function of a gradient-ascent optimization algorithm, that aims to find the optimal spatial distribution of manageable input variables (i.e., fertilizer and seed maps). The final step in this framework consists of an online CPP algorithm to optimally apply the fertilizer and seeds in the field. This algorithm is based on an MS-CNN agent that finds relevant features in the environment, mimics offline CPP algorithms, and improves its performance through reinforcement learning [25].

Following the trade-off between prediction accuracy and optimization complexity, the CNN yield prediction architectures pose a very challenging optimization problem since they have a much larger input space than other methods. Increasing the input space without collecting more data makes the optimization algorithm more likely to find a solution outside the domain

of the data used to train the model, which may lead to overconfident solutions that are unlikely to work in practice. Hence, this dissertation also revisits the proposed solution to incorporate uncertainty and leverage the applicability of the MS-CNN model into DSS. The vast majority of uncertainty quantification research focuses on Bayesian formalism [26], where a posteriori distribution over the network's parameters is obtained from the data. This approach is, however, often harder to implement and slower to train when compared to non-Bayesian neural networks. A simpler and more scalable solution for uncertainty quantification was proposed in [27], where a deep ensemble of neural networks performed as well as Bayesian networks in different datasets. The proposed approach requires few modifications to non-Bayesian neural networks and is suitable for parallel computing, making it attractive for our problem. Hence, the MS-CNN architecture is modified under the deep ensemble framework to provide uncertainty estimation of predictions. The objective function of the optimization algorithm is also reformulated to incorporate risk constraints and provide safety bounds on the algorithm's output.

1.1 Related Work

Yield Prediction Models

Many spatial econometrics models were developed to account for data's spatial structure [28]. For example, the Generalized Least Squares (GLS) method [29] combines a geostatistical semivariogram with linear regressions. This approach, however, uses a fixed kernel to model the influence of neighbor data in a particular sample, based only on the distance between them, rather than on the spatial structure of the data. Another example can be

found in [30], which used spatial econometrics to estimate yield response to fertilizer rates across the studied fields.

Most Machine Learning algorithms applied to yield prediction don't consider information from the spatial distribution of the input variables. Instead, they assume only local information from the variables as each sample in the field is independent of the others. For instance, methods such as Artificial Neural Networks (ANN) [13], Support Vector Machines (SVM) [31, 32] and Random Forest (RF) [33] have previously been used for predicting yield with reasonable accuracy. In one of the few works considering the spatial data, [34] proposed a deep learning framework for real-time yield forecasting. Authors use a dimensionality reduction based on histograms over remote sensing images. Then a Deep Gaussian Process is implemented to integrate spatiotemporal information present in the histograms. This work, although interesting, uses temporal information for predicting yield, making it not suitable for optimizing fertilization rates. Also, the model is designed to work on a larger field-scale than the one proposed in this dissertation.

In agriculture, applications of CNNs [35] usually focus on disease [36, 37] and plant [38] classification, and on image-based estimation, such as soybean leaf defoliation level [39]. Also, [40] used a CNN to estimate yield based on multispectral imagery collected during the growing season, not providing actionable pre-season information. Some of the concepts of MS-CNN were explored in different problems. A CNN was proposed in [41] for flower grading, where three images are necessary to fully describe a flower, and authors concatenate the result of independent convolutions from each image in the first layer before performing subsequent convolutions. Different architectures are also present in the literature related to human action identification in video data. A multi-stream architecture is proposed in [42], in which in-

puts are combined late in the network, demonstrating better performance than architectures using stacked frames as inputs [43]. A 3D CNN was proposed by [44], which uses 3D filters convolved with adjacent input frames, demonstrating results as good as the state of the art techniques.

Optimization of Fertilizer Rates

The concept of management zones is one of the most common approaches to optimize fertilization rates. In this framework, the field is divided into different zones according to the variability observed in the soil's properties and historical yield. Then, crop simulations or regression methods are used to define the optimal rate for each zone. Two good examples of this approach are given by [45] and [46]. Although simple, this method is limited by a small number of management zones.

Neural Networks (NN) have previously been used to optimize levels of fertilizer. In [47], authors used a NN to create a yield model from independent sampled data and choose rates from a finite and small set of possible fertilizer rates. Although reasonable, this approach ignores the effect of the fertilizer applied to other points in the field, works with a limited set of rates, and doesn't consider the potential advantages of a variable rate applicator.

Recently, [48] proposed a Bayesian numerical optimization framework based on Gaussian Processes (GP) models. The authors account for uncertainty in their recommendations by using Monte-Carlo sampling over simulated data and provide a comprehensive optimization framework. However, the spatial distribution of input variables is not considered, which can be a limiting factor in prediction accuracy. Moreover, the uncertainty is obtained a posteriori, and predictions are not risk-aware, providing only expected values but not their variance.

Coverage Path Planning

Coverage path planning (CPP) is the task of finding an efficient path that passes overall desired points of an environment's area [49]. This task is present in applications such as automated harvesting [50], inspection of mechanical structures [51], painter robots [52], vacuum cleaners [53], among many others [54]. Due to the NP-hardness of CPP [55], coverage algorithms are heuristic and often achieve a sub-optimal solution for the problem. Some examples of offline algorithms are cellular decomposition [56], spanning trees [57], genetic algorithms [58], and the wavefront algorithm [59]. Online algorithms are usually approximations of the offline heuristics. For example, a topological coverage algorithm is proposed by [60], which uses landmarks in the environment to determine reachable sub-regions to be covered by a back and forth motion. This solution has limited efficiency when the number of connections between reachable sub-areas is high.

Some studies have already explored the concept of learning-based path planning. In [61], authors propose an imitation learning framework for planning a path from point A to B. They use an oracle of optimal solutions based on full-state representation to train an agent that imitates the optimal actions using partial observations of the environment. They condense the state's representation based on a set of heuristics. This approach is not directly applicable to the problem of CPP since the last has a much bigger input space, and the heuristic representations are not easily derived. An inverse reinforcement learning was proposed by [62] to represent the cost model underlying the desired driving behaviors. The method is demonstrated to infer suitable cost functions for long-term planning on traversing problems.

An end-to-end RL approach to CPP is presented in [63]. In this work,

authors use a CNN as a feature extractor and decision network for a policy gradient algorithm. They demonstrate full coverage of maps with the minimal cost associated with the energy required by a tetromino robot. The findings suggest promising use of RL on CPP problems but are not scalable since the size of the map is limited by the size of CNN's inputs. Moreover, the technique is not directly applicable to the problem in this dissertation. In our case, the cost of overlapping a previously covered node is much greater than the cost of energy, which brings the optimization problem to a much longer time horizon.

1.2 Dissertation Outline

This dissertation has six chapters for which a brief overview is given below:

- Chapter 2 details the CNN architectures used to create the yield prediction model based on environmental and manageable variables. Four different CNN architectures combining the inputs at different stages in the network are compared. Data from nine cornfields across the US are used to test and compare the proposed architectures. Such fields are part of an OFPE with randomized nitrogen and seed rates prescription. The models are trained in a supervised fashion, tested, and compared with a linear model, a fully connected neural network, a random forest regression, and a support vector machine. The results presented in this chapter are published in [64].

- In Chapter 3, inspired by the backpropagation algorithm, the gradient of the MS-CNN is used to derive the gradient of the expected net revenue with respect to the manageable input variables, taking network

weights as constants. This gradient is further used in a gradient ascent optimization algorithm for finding the best spatial distribution of manageable input variables (i.e., fertilizer and seed maps). The proposed method is compared with an evolutionary algorithm, and experiments are conducted to demonstrate the profitability potential of this solution over traditional farm management. This chapter also presents a sensitivity index to account for input variable importance on yield prediction. The results shown in this chapter are published in [65].

- Chapter 4 presents the incorporation of uncertainty quantification into our previously proposed CNN yield prediction model and a new risk-averse optimization algorithm. The CNN architecture is redesigned under the Deep Ensemble framework, so the predictive model outputs a probability distribution instead of a single value. Then, the objective function of the previous optimization algorithm is reformulated to reduce the uncertainty in the solutions, and satisfy risk-constraints that can be easily adjusted to match different levels of the farmers' risk aversion. This chapter also demonstrates how uncertainty maps can be created to give insights to farmers regarding areas of their fields that present higher variability that is not explained by the collected data.

- Chapter 5 proposes a novel learning framework for online coverage path planning. We train an agent to find relevant features in the environment, mimic offline algorithms, and improve its performance through reinforcement learning [25]. There are three steps in this framework. First, the software generates grid-based maps containing different shapes and obstacles and solves them for coverage using offline techniques that minimize the path's cost. We use Boustrophedon cel-

lular decomposition [66] for solving planar maps and a novel heuristic Dijkstra's algorithm for solving non-planar maps (i.e., a grid projection of 3D surfaces). Next, a CNN agent is trained in a supervised fashion, using examples from paths obtained by the offline algorithms (i.e., imitation learning [67]). Each example comprises an action along the path and its respective observation of the state. When trained, the agent maps an observed state from the environment to a probability distribution over possible actions in the grid. The agent's path is then constructed greedily by following the actions with the highest probability of being optimal. Finally, the agent's policy is refined using a reinforcement learning algorithm.

- Concluding this dissertation, Chapter 6 makes some final remarks and presents possible future directions.

CHAPTER 2

MODEL OF YIELD RESPONSE TO NUTRIENT MANAGEMENT

In this chapter, we propose a Convolutional Neural Network (CNN) for learning relevant spatial structures from different fields' attributes and combine them to create a model to predict yield. Four different CNN architectures combining the inputs at different stages in the network are compared using data from nine corn fields across the US. Such fields are part of an OFPE with randomized fertilizer and seed rates prescription. The models are trained in a supervised fashion, tested, and compared to a linear model, a fully connected neural network, a random forest regression, and a support vector machine. The models proposed in this chapter will serve as the base for further optimization algorithms.

2.1 Dataset Construction

New field technologies for site-specific management have become available to farmers in recent decades. They include but are not limited to yield monitoring, remote sensing imaging, and variable rate input application. Data from these machines are easily visualized by the farmer and bring insightful information for site-specific crop management. Such technology makes it possible to design large-scale OFPE, generating large amounts of data to model yield response to the spatial distribution of input variables.

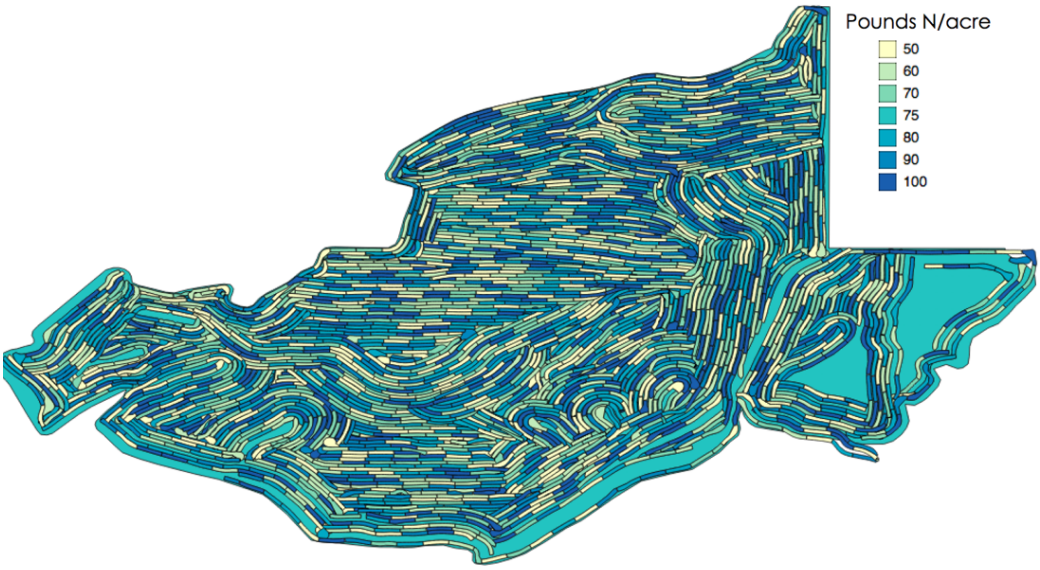


Figure 2.1: OFPE design in a field with irregular boundaries.

The traditional approach for OFPE is the design of a "checkerboard" containing rectangle cells with the same width as the variable-rate applicator. Each cell is an experimental unit in which different treatments are applied. This approach works well on rectangular fields, but when the boundaries have irregular shapes, it may result in a lower coverage of the field. This dissertation uses data from the Data Intense Farm Management (DIFM) project [68], recorded during the 2017 season. The database contains georeferenced management and environmental data from OFPE conducted on fields averaging 40 ha and 200 experimental units represented by 85 m long and 18 m wide polygons. As some fields have irregular shapes, we use a different method for designing the experiments. We sequentially create each experimental unit along the path previously used by the farmer. This path is usually available from harvesting logs and ensures that most of the field will be covered by experimental cells. Figure 2.1 depicts an OFPE design for a large and irregular field.

Nine fields were selected, from which six are rain-fed fields in Illinois (Fields

2, 4, 7, 8, and 9) and Ohio (Field 5), and three are irrigated fields in Nebraska (Fields 3 and 6) and Kansas (Field 1). Nitrogen and seed rates were randomly assigned from four different levels to each experimental unit in a field, except for field 1, where the nitrogen rate is constant. The explanatory variables chosen for this study are nitrogen fertilizer and seed rates prescription maps, elevation map, and soil's shallow electroconductivity (EC). Soil's EC measurements are proxies of chemical and physical properties, including texture, bulk density, soil organic carbon, water content, salinity, and cation exchange capacity [69]. In general, in fields as the ones we are evaluating, it is expected to observe a positive association between EC and soil's capacity to supply water and nutrients to crops. Information about EC is also valuable, given that it is more spatially detailed than that of soil samples [70]. Additionally, a single cloud-free satellite image was used to characterize the bare soil variability of each field. Only the surface reflectance of the red band from a 3m spatial resolution Planet Labs [71] PSSE4 multispectral image was used. The image was taken after soil tillage, approximately one week before planting, when the field was clear from any vegetation and the soil was exposed. The bare soil reflectance has been shown to correlate with soil attributes such as organic matter content and particle size distribution [72], and may work as a proxy to explain yield and nitrogen mineralization potential. Yield data is used as the response variable and was collected by yield monitors during harvesting. Notice that all explanatory variables are available at the beginning of the season, so one can use the model for optimizing nitrogen and seed rates before applying them.

As each field had its data recorded using different equipment and software, a way to put each variable from all fields into the same support is first needed. The smallest unit of analysis is defined as a square with a side of five meters

and is represented by a single element in a large raster spanning the field. Variables stored as polygon data (e.g., prescription maps) were sampled and converted to each element in the raster by using the mean method, while the ones stored as points (e.g., elevation map and soil's electroconductivity) were first interpolated using kriging [29] and rasterized to the desired support (5 $\times$ 5m cell). The satellite picture was resampled to the desired support since it is already a raster.

The key assumption in this chapter is that the yield at a single unit of analysis depends on the spatial structure of observed explanatory variables around it. It is fair to assume, however, that the influence of neighbor data over a unit is limited to a certain range. So, by finding this range, the model's complexity can be reduced without losing valuable data, which makes the model easier to train and more data-efficient. Therefore, a variogram (Figure 2.2) containing the averaged variability from all explanatory variables is constructed to compute the range that better describes the data. For the selected fields, the computed range is approximately 100m, indicating that no significant higher variability is obtained when considering a greater distance between two points.

Then, a sample used as input for the proposed models described in the next section is defined as a set of 21 $\times$ 21 elements rasters (one for each of the five explanatory variables), spanning a square of 100 $\times$ 100m around the unit where the yield value is being predicted (Fig. 2.3). Then, to construct a dataset, rasters (with dimension equal to 21 $\times$ 21) are cropped from the original raster variables around each non zero cell in the yield map. Such procedure results in a set of samples containing five rasters each and the value of yield at their respective center cell.

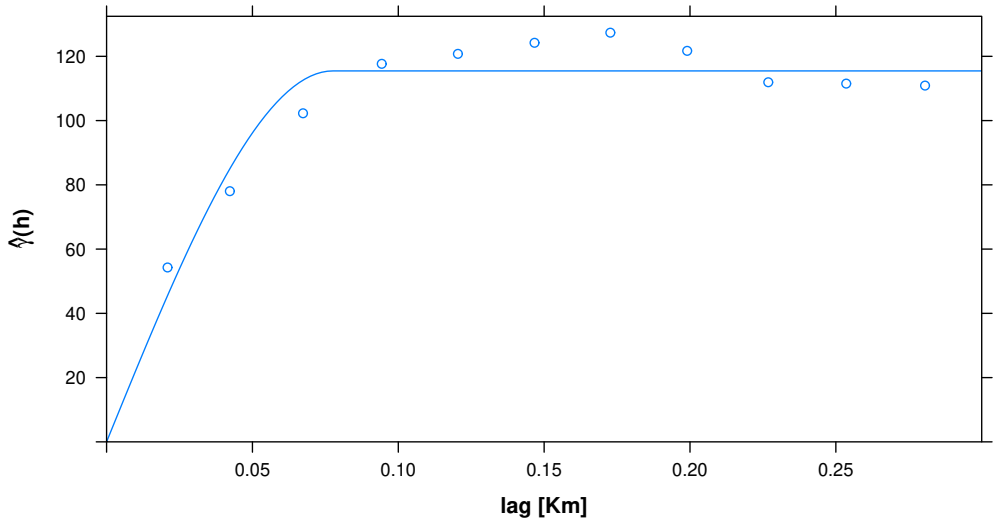


Figure 2.2: Averaged variogram from fields' attributes.

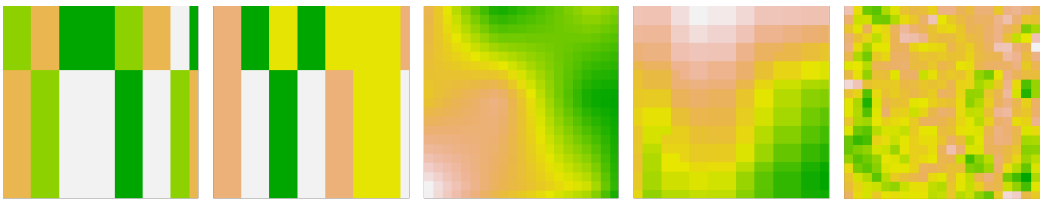


Figure 2.3: Rasters representing five different input variables at the same point: nitrogen rate (a), seed rate (b), elevation map (c), soil's electroconductivity (d), and satellite image (e).

When considering the yield data collected closer than 50m from the field's boundaries, the rasters centered at such cells cover an area where data is not available. Trying to sample such data would assign null values to these cells, resulting in a non-representative sample. Depending on the size of the field, a considerable number of data points in the yield map lies within 50m from the border, and eliminating such points from the dataset would considerably reduce the amount of data. We address this problem by creating a buffer around the rasters of each explanatory variable (Fig. 2.4) and padding them accordingly to each variable. The seed and nitrogen maps are padded with

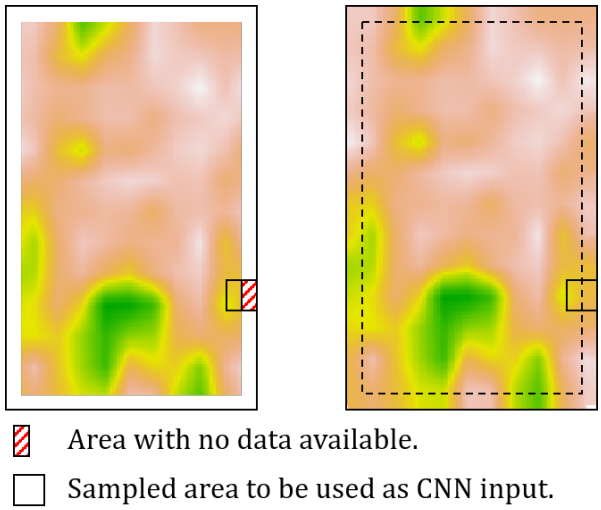


Figure 2.4: Example of a buffer created to assign values to areas with no data on an EC map.

zero since neither of them was applied at the buffer's area. The elevation map and soils' shallow electroconductivity values were extrapolated using kriging, while the satellite picture is taken already considering the buffer. Finally, the data is standardized variable-wise to make the model's training process easier, and samples in which yield data is more than three standard deviations from the mean were removed from the dataset.

2.2 CNN Model

This dissertation uses five different fields' attributes to create a model for yield prediction. Two of them are manageable variables (i.e., nitrogen fertilizer and seed rates), and three are environmental (i.e., elevation map, soil's electroconductivity, and satellite image). Figure 2.5 depicts the maps of input variables and the resulting yield in field 7. Defining how these different sources of information are combined through the network is a crucial step towards obtaining efficient modeling.

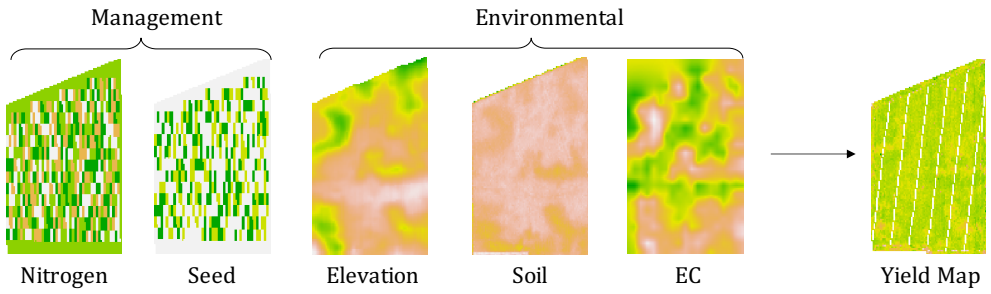


Figure 2.5: Example of input maps and yield response.

Four CNN architectures combining the inputs in different ways are proposed and tested in this work. The first combines the inputs as in the most commonly used convolutional neural networks, by stacking (we call it ST) them as a multi-channel image (Fig. 2.6). Sixteen 3×3 filters are used in the first convolutional layer with stride one, followed by a 2×2 max-pooling layer with stride two. Then, outputs are flattened and fed to two sequential fully connected ReLU layers with 512 neurons each. Finally, outputs are connected to an output neuron with a linear activation function. The second proposed architecture is a multi-stream Early-Fusion (EF) network. In this network, each input is connected to an independent convolutional layer (defining a multi-stream network) with eight 3×3 filters each with stride one, followed by a 2×2 max-pooling layer with stride two (Fig. 2.7). The outputs of the max-pooling layers are then flattened, concatenated, and fed to a fully connected ReLU layer with sixteen neurons, followed by an output neuron with a linear activation function. The third architecture is also a multi-stream network named Late Fusion (LF). We add a fully connected ReLu layer with sixteen neurons to each stream after the max-pooling layer, followed by a single ReLu neuron (Fig. 2.8). Then, the five resulting neurons are concatenated and fed to the last two layers, as in the EF architecture. The last proposed architecture is a 3D CNN (named 3D for short) very similar

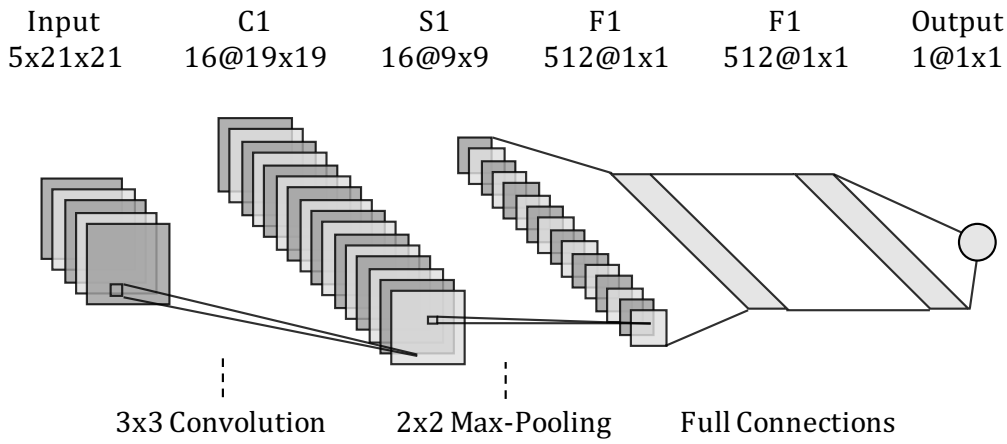


Figure 2.6: CNN-ST architecture.

to ST, except the convolutional layer has sixteen $3 \times 3 \times 1$ filters convolved also in the channel dimension with stride one (Fig. 2.9).

To create a baseline, commonly used models for yield estimation are compared with the proposed CNN architectures. Such models are the multiple linear regression (MLR), a fully connected neural network (FC), a random forest (RF) regression, and a support vector machine (SVM). Fully connected neural networks have received significant attention over the last decade [73–75] for being a powerful tool when modeling both linear and nonlinear relations between explanatory and response variables. A broad comparative study is presented by [76], in which random forest demonstrates higher performance over machine learning competitors when predicting yield. [77] explores SVM as a comprehensible methodology for accurate yield prediction. Even so, all these models do not consider the spatial structure of the data, while having as inputs the values of each variable at the exact cell where yield is estimated.

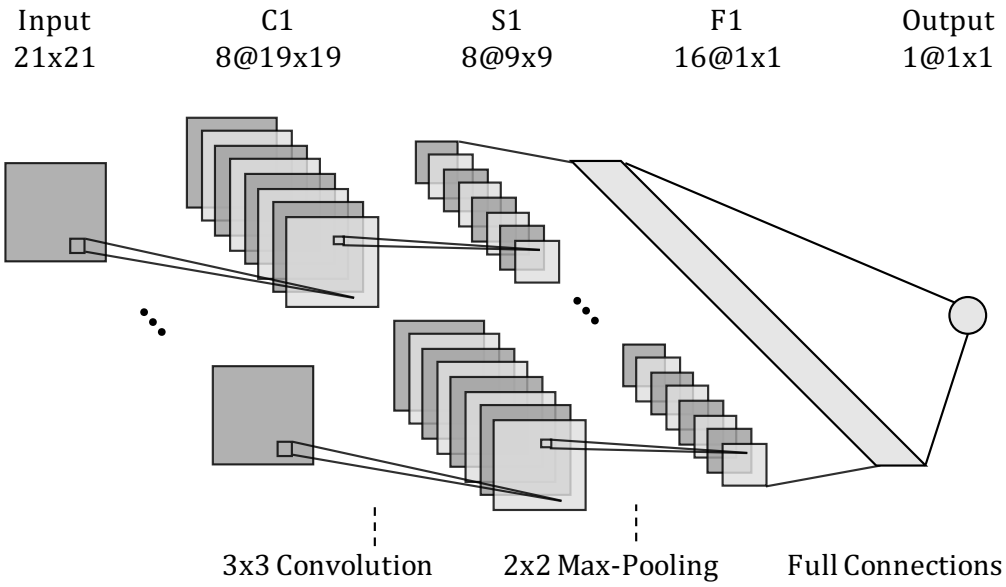


Figure 2.7: CNN-EF architecture.

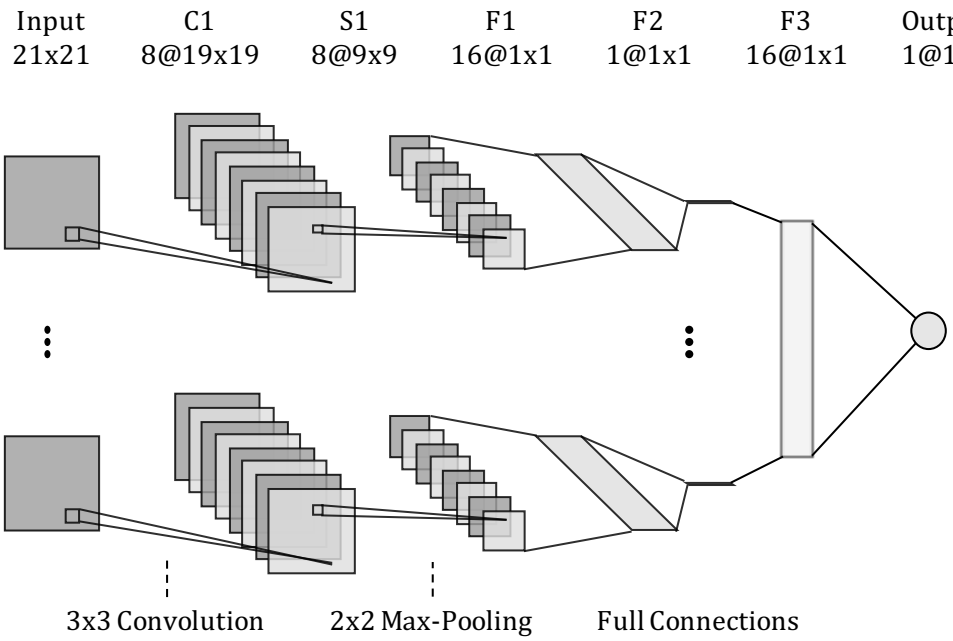


Figure 2.8: CNN-LF architecture.

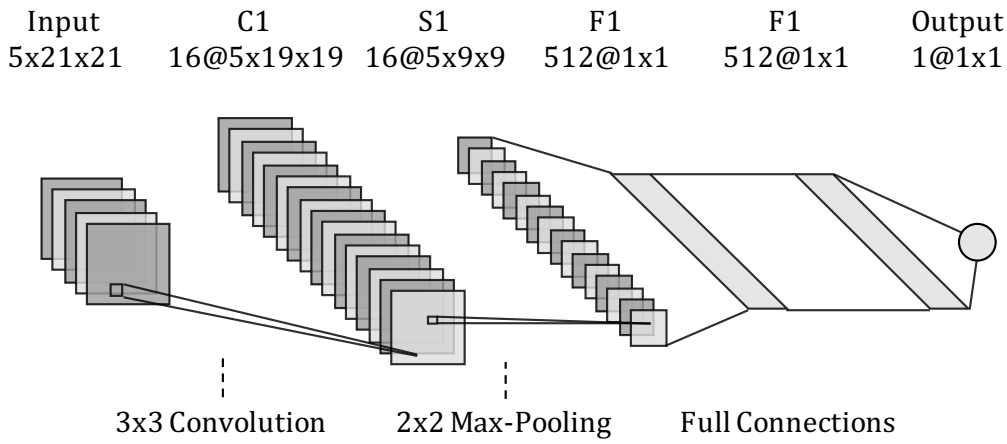


Figure 2.9: CNN-3D architecture.

2.3 Experiments and results

Yield response to nutrient management also depends on other environmental factors and management practices with whole field impact (e.g., solar radiation, planting date, seed genetics, among others). These factors vary from one field to another while being constant within the same field. Using site-specific data is a way to reduce the input space and increase the representativeness of the data aiming better recommendations to each farmer individually. So, we train and test a different model for each of the nine fields (described in Section 2) rather than creating a single model to work in all fields. However, data within a single field is often spatially autocorrelated and may lead to model over-fitting depending on the way the test set is chosen. For instance, a random partition of the data results in training and test sets with very similar samples, which overcasts the generalization power of the model. So, we spatially partition the data to account for this problem, as proposed by [78].

Data from each field was spatially partitioned in five stripes perpendicularly to the longest dimension of the field to maximize the distance between

samples from two different partitions. Three stripes (60% of the data) were used for training the model, one (20% of the data) for validation, and one (20% of the data) for tests. We performed a grid search to define hyper-parameters, including the number of filters in the convolutional layers, the number of neurons in the fully connected layers, and also the dropout regularization probability for all models. Each architecture was trained over the same database using Adam optimizer [79], and the validation set was used online as an early stop criterion (allowing at most eight consecutive iterations of increase in the validation loss) to avoid over-fitting the data. The loss function is given by the Mean Squared Error (MSE) between yield predictions and true yield values at every raster's position in the batch.

Cross-validation was used to evaluate the model using five folds according to the spatial partitioning. Table 2.1 shows the averaged Root Mean Squared Error (RMSE) over the five test sets for each model in each field. Notice that since yield data is standardized, the RMSE value represents a fraction of the standard deviation from the original yield data (shown in the most right column) in each field. The best results are formatted in bold.

The LF model is the one with the lowest RMSE value for eight of the nine tested fields, being field eight the only in which the SVM had a lower value. We show further in this section that this field is the one where the variability is the least explained by the spatial structure of the data. The random forest regression shows the second-best results in four of the fields.

The better performance of LF over other CNN may be explained by the way each architecture combines the input variables in the network. The ST model linearly combines the inputs element-wise at the very first convolutional layer, suppressing any possible nonlinear interaction between different attributes in the field. To make this statement clear, consider a n $\times$ n filter convolved with

Field	RMSE								Yield [Kg/ha]	
	MLR	FC	LF	EF	3D	ST	SVM	RF	Stdv.	Mean
1	1.53	0.97	**0.66**	0.73	0.75	0.69	0.86	0.92	3240	12500
2	1.29	0.88	**0.83**	0.86	0.87	0.88	0.86	0.90	2290	10700
3	1.90	0.63	**0.58**	0.58	0.59	0.60	0.86	0.59	1230	14400
4	1.03	0.76	**0.75**	0.78	0.77	0.77	0.77	0.76	900	12200
5	0.74	0.72	**0.70**	0.72	0.75	0.73	0.76	0.70	1360	14500
6	1.09	0.51	**0.48**	0.51	0.53	0.56	0.58	0.52	1140	14700
7	0.75	0.72	**0.69**	0.70	0.71	0.73	0.76	0.73	1150	15700
8	1.11	0.94	0.94	0.94	0.94	0.94	**0.89**	0.96	2267	14100
9	1.10	0.69	**0.63**	0.65	0.66	0.66	0.67	0.63	1140	12600
Avg.	1.17	0.76	**0.70**	0.72	0.73	0.73	0.78	0.75	1635	13489

Table 2.1: Crossvalidation averaged RMSE over test dataset (in terms of yield standard deviation from each field), yield standard deviation, and yield mean [Kg/ha].

stacked input with L different variables (each one is a channel in the stacked frame). Now let $w_{i,j}$ be the filter's weights, $y_{u,v}$ be the convolution output at position (u, v), and $x_{k,i,j}$ be the value at variable k aligned with position (i, j) in the filter. Then we have that y_{uv} is given by equation (2.1):

$$y_{uv} = \sum_{k=1}^{L} \sum_{i=1}^{n} \sum_{j=1}^{n} (w_{ij} x_{kij}) \tag{2.1}$$

Figure 2.10 shows an example where two stacked inputs are convolved with a single 2D filter with stride equal to one. It is evident in this example that the output of the convolutional layer only contains a linear combination of the values from each input, and that the information from each separate input is not preserved. As a consequence, any nonlinear interaction between the input variables will not be modeled at the fully connected layers (that work as function approximators).

One possible solution to overcome this limitation is to convolve the filter

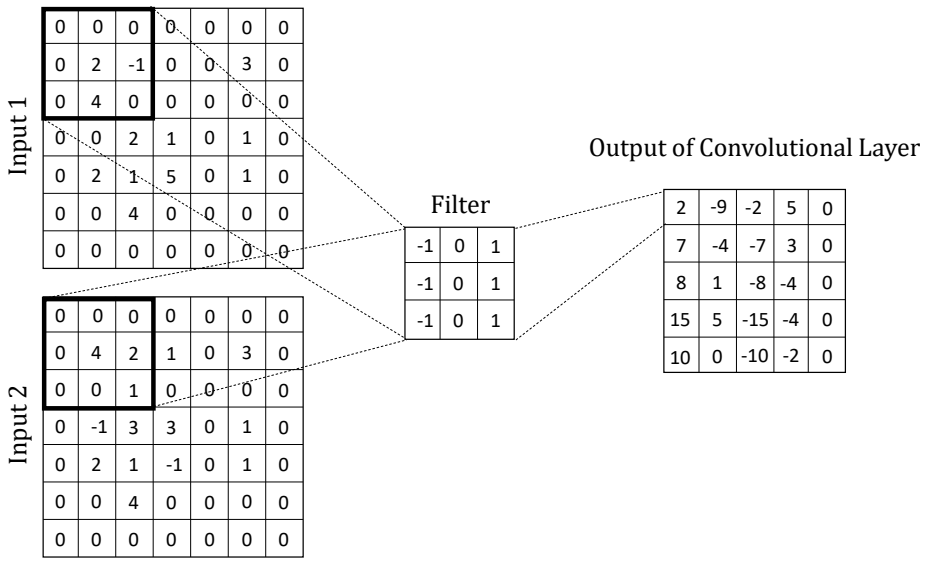


Figure 2.10: Two stacked inputs convolved with a 2D filter.

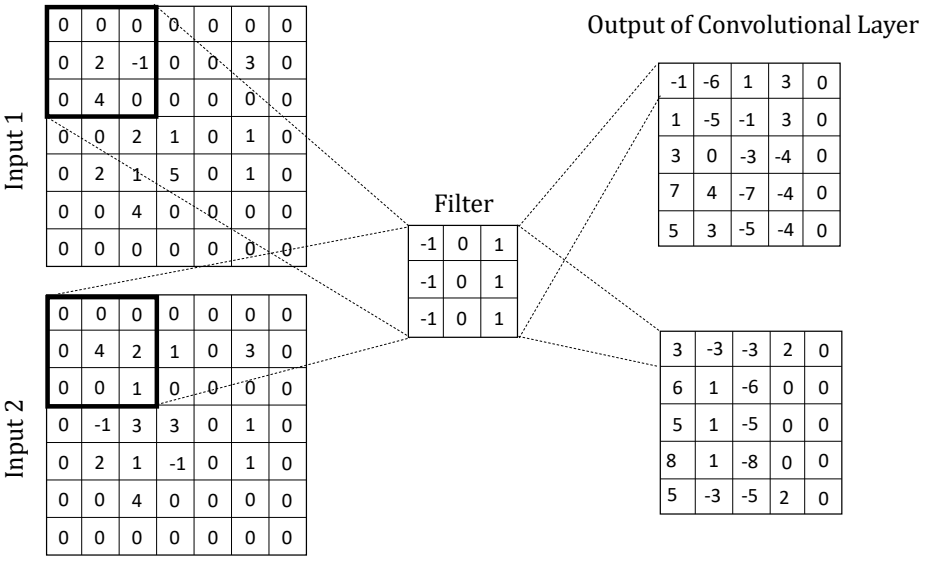


Figure 2.11: Two stacked inputs convolved with a 3D filter.

input-wise, so the information from each input is carried separately through the convolutional layer. The proposed 3D architecture considers the stacked inputs as a third dimension to convolve the filter, carrying the information from each input to be combined in the fully connected layers. However, this method leads to an unnecessary increase in the number of parameters in the network, once the same filter may not be relevant (as feature extractors) for all input variables, resulting in a model with a lower data efficiency.

The multi-stream architecture addresses both problems discussed above. It performs independent convolutions with the inputs while using a different set of filters for each one. When comparing the LF and EF models it is reasonable to say that LF focuses on better feature extraction from inputs, while EF can model more complex interaction between variables. Also, by reducing the dimension of each input before combining them, the LF model becomes easier to train, leading to higher data efficiency.

When looking to Table 2.1, one can notice that the CNN models have a modest advantage over their competitors in some fields, while in others, this advantage is much higher. To investigate the possible cause of this difference, the variability of the response variables is observed. For that, spherical models were fitted to the variogram. Let's define c_0 as the nugget variance and $(c_0 + c_1)$ as the sill variance. Then, c_0 represents the variance observed at a lag distance of 0m, and is a function of stochastic effects and measurement error [80], while c_1 accounts for the additional variance that comes from the spatial structure of the data for a given range. Since the CNN framework focuses on modeling unknown spatial structures of the data, we propose to estimate how much of the total variability observed in each field comes from such structure. This resembles the Cambardella metric [81] used to calculate spatial dependence, except here we use a ratio of c_1 to total

variance, given by equation (2.2):

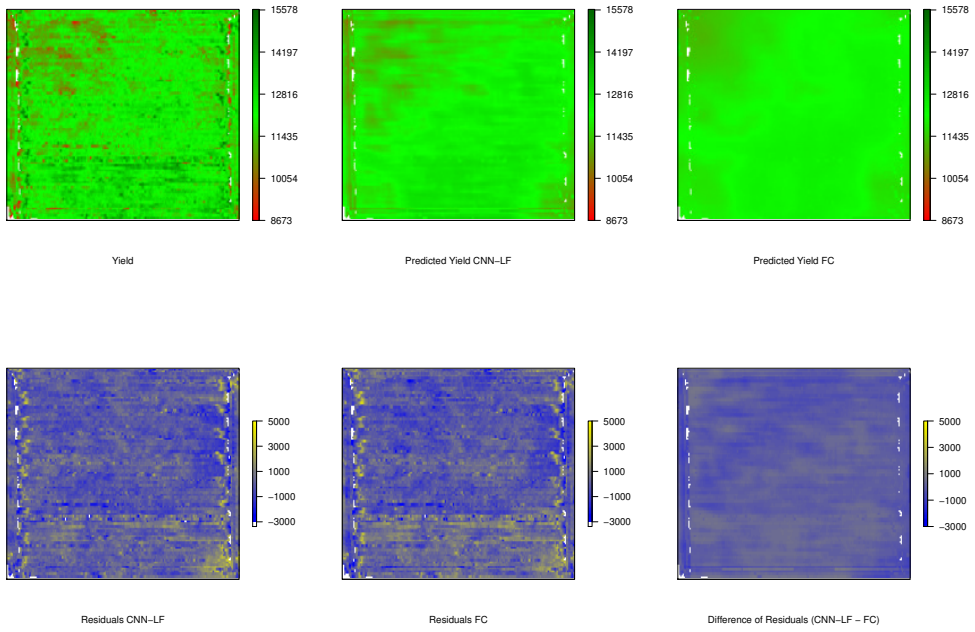


Figure 2.12: True yield, predicted yield using CNN-LF, predicted yield using FC, residuals for both models, and difference of residuals for field 4 [Kg/ha]

$$C_{\text{index}} = 1 - \frac{c_0}{c_0 + c_1} \tag{2.2}$$

Table 2.2 shows the C_{index} calculated for the yield data of all nine fields and the RMSE reduction of the LF architecture when compared to MLR, FC, SVM, and RF. It is clear and also expected that in fields with high C_{index} (i.e., yield variability is highly dependent on the spatial structure of the data), the proposed model presents a higher reduction on the RMSE value when compared to FC and MLR. On the other hand, in fields with low C_{index}, a simpler model as FC or even MLR also has good performance, getting closer to CNN model, which results in a lower improvement when

Field	Decrease in RMSE [%]				C_{index}
	MLR	FC	SVM	RF	
1	57	32	24	29	0.47
2	36	5	3	8	0.36
3	70	8	33	3	0.18
4	27	1	3	1	0.10
5	5	3	8	1	0.12
6	56	6	17	8	0.50
7	7	3	8	4	0.29
8	16	0	-5	2	0.04
9	43	8	6	0	0.33
Avg.	35	7	11	6	0.27

Table 2.2: Percent decrease in RMSE value using CNN-LF when compared to FC and MLR for each field, and their respective C index for the yield map

comparing them.

Then, data from yield or even from an explanatory variable can be used to assess the expected improvement on yield prediction before using the proposed framework, which is more computationally expensive than commonly used methods.

Figure 2.12 shows a qualitative comparison between predicted yield maps using the LF and FC models in field four. It is possible to observe that the spatial structure of the data is better captured by the LF model, while FC results in a "blurrier" map. The difference between the residual maps from both models shows a spatial structure that resembles the one observed on the true yield map. This indicates that the main difference between the two models lies in their ability to capture spatial features in the data. Moreover, one can observe that most of the high values in the residuals maps are located close to the field's borders. The data in such regions are expected to have more noise, due to yield monitoring system errors caused by the change in harvester speed and direction, and cannot be explained by the selected

explanatory variables.

2.4 Discussion

A CNN model for yield prediction based on pre-season treatments and environmental variables was proposed. Four different architectures based on convolutional neural networks were tested on nine corn fields and compared to a multiple linear regression model, a fully connected neural network, a support vector machine, and a random forest regression. Tests were conducted using data from the same field in which each model was trained, and after computing the RMSE values for the test sets, the LF architecture demonstrated the best performance among all. This model has the advantage of allowing nonlinear combinations among input variables, while keeping the model with a relatively low number of parameters.

Fields with yield variability highly associated with the spatial structure of the data get the most benefit from this framework, and this can be estimated beforehand using the proposed C_{index}. The index is a proxy for the level of variability explained by the spatial structure of the input data and uses information from a variogram of the response variable.

The demonstrated generalization power and low predictive error make this model suitable for nitrogen and seed rates optimization for future seasons to maximize expected yield or even profit. All variables used in this work are available at the beginning of the season when the farmers must decide on which fertilization rates to apply in the field.

The proposed CNN framework brings the power of neural networks to geospatial problems. It allows creating models with no need of handcrafting features or making spatial distribution assumptions, which are incorporated

into the learning process. This framework can be also easily extended to incorporate more input variables and also information on climate conditions when data from different years become available.

CHAPTER 3

OPTIMIZATION OF FERTILIZATION RATES

Choosing the optimal rates of nitrogen and seeds is one of the main reasons to create a good yield prediction model, and the complexity of optimizing these rates depends on the chosen model. In Chapter 2, we proposed a Multi-Stream Convolutional Neural Network with Late Fusion (MSCNN-LF) for predicting yield. This model has a much larger input search space when compared to other ML models since it maps patches of the input maps to a predicted yield value at a single point in the field. Also, such patches overlap between themselves, making the sequential optimization of each point in the field not possible. Then this chapter presents an optimization framework for the MSCNN-LF model to find the spatial allocation of nitrogen and seeds that result in the maximum expected net revenue to the farmer. We propose a gradient ascent with a momentum term algorithm under the assumption that this is a non-convex optimization problem. An evolutionary algorithm is also explored and detailed in this chapter.

3.1 Gradient of Net Revenue

The first step is to derive the gradient of the net revenue (i.e., the total yield value discounting nitrogen and seed costs) with respect to nitrogen and seed rate maps.

Let $N, S \in \mathbb{R}^{m \times n}$ be the matrices representing the applied nitrogen and

seed rates maps respectively. Now, let $\mathcal{C}, \mathcal{S} \in \mathbb{R}^{l \times l}$ be sets of matrices cropped from matrices N and S such that $l < n < m$. We define a function $\tilde{f}(\bar{N}, \bar{S}) : \mathbb{R}^{l \times l} \times \mathbb{R}^{l \times l} \longrightarrow \mathbb{R}$, with $N \in \mathcal{C}$ and $S \in \mathcal{S}$, that represents the MSCNN-LF model, mapping a patch from nitrogen and seed rate maps to a yield value, taking environmental attributes as constants. The total crop yield is then given by $f(N, S) = \sum^{i} \tilde{f}(\bar{N}_i, \bar{S}_i)$, where $i \in \mathbb{N}$ is the index of each element in the yield map. We want to derive $\nabla f(N, S)$ as:

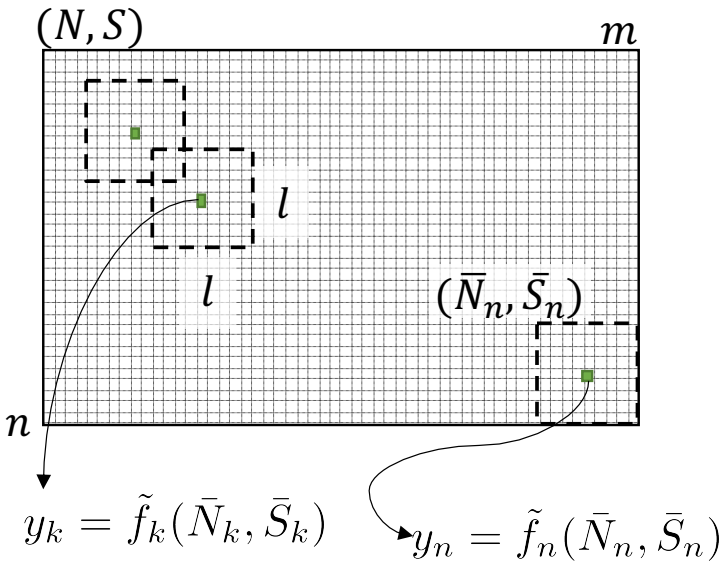


Figure 3.1: Yield as a function of a cropped patch from the nitrogen and seed maps.

$$\nabla f(N, S) = \left(\frac{\partial \sum^{i} \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial N}, \frac{\partial \sum^{i} \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial S} \right). \tag{3.1}$$

The result of each partial derivative also belongs to $\mathbb{R}^{m \times n}$. However, as $\tilde{f}(\bar{N}, \bar{S})$ is a function of just a cropped patch of the input map, the partial derivative at all elements outside the patch will be equal to zero. We also

define a zero padding function $z(C,i) : \mathbb{R}^{l\times l} \times \mathbb{N} \longrightarrow \mathbb{R}^{m\times n}$ that centers the $l \times l$ C matrix (i.e. a patch cropped from the input map) at element i in a $m \times n$ matrix, and completes the remaining elements with zero. Thus, we have:

$$\begin{aligned}\frac{\partial \sum^{i} \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial N} &= \sum^{i} z\left(\frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_i}, i\right) \\ &= \sum^{i} z\left(\begin{bmatrix} \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_{i_{11}}} & \cdots & \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_{i_{1l}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_{i_{l1}}} & \cdots & \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_{i_{ll}}} \end{bmatrix}, i\right).\end{aligned}$$

Similarly,

$$\frac{\partial \sum^{i} \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}} = \sum^{i} z\left(\begin{bmatrix} \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_{i_{11}}} & \cdots & \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_{i_{1l}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_{i_{l1}}} & \cdots & \frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_{i_{ll}}} \end{bmatrix}, i\right).$$

Fortunately, all elements of $\frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_i}$ and $\frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_i}$ are easily obtained from the MSCNN-LF model's gradients computed for the backpropagation algorithm. With (3.1) derived, our final goal is to maximize the crop yield return discounting the costs from nitrogen and seeds, which can be formulated as:

$$\max_{N,S} \quad \left(p_V f(N,S) - p_N \sum^{i} N_i - p_S \sum^{i} S_i\right) \tag{3.2}$$

$$\text{subject to: } N_{\min} \leq N_i \leq N_{\max}, \forall\ i$$

$$S_{\min} \leq S_i \leq S_{\max}, \forall\ i,$$

where p_V, p_N, and p_S are the prices per smallest unit area of corn, nitrogen, and seed, respectively, and the rates of nitrogen and seed are bounded by minimum and maximum values. The boundary values come from the rates applied during the on-farm experiment, since the MSCNN-LF model is not trained with values above or below them. Finally, let P_N and P_S be $m \times n$ matrices containing all their elements equal to p_N and p_S respectively. Then we write the gradient of (3.2) as:

$$\nabla Y(N,S) = \left(p_V \sum^{i} z \left(\frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{N}_i}, i \right) - P_N \ , \right.$$
$$\left. p_V \sum^{i} z \left(\frac{\partial \tilde{f}(\bar{N}_i, \bar{S}_i)}{\partial \bar{S}_i}, i \right) - P_S \right). \tag{3.3}$$

3.2 Gradient Ascent

With the model's gradient derived in (3.3), we propose a gradient ascent algorithm with momentum term to maximize the expression in (3.2). Maps N and S are initialized with the status quo rates usually applied by farmers and then updated according to Algorithm 1.

Algorithm 1 Pseudocode for the gradient ascent algorithm with momentum.

```
[N, S] ← status quo rate
V ← 0
for i = 1 : max iterations do
   compute ∇Y(N, S) for the current [N, S]
   V = αV + ∇Y(N, S)    (compute the momentum)
   [N, S] = [N, S] + λV    (update the input maps)
   λ = λd_r    (decrease the step size)
end for
return [N,S]
```

The momentum term α is used to prevent the algorithm from converging to local maxima or saddle points [82]. This term is adjusted for optimizing each field since they represent different problems with different search spaces. The term λ is the step size, and it is decreased over the iteration for a more refined search for the global maxima.

3.3 Sensitivity Index

In order to ensure our optimization algorithm is based on a model where the manageable variables (i.e., nitrogen fertilizer and seed rates) are really relevant and not vanished by more relevant environmental inputs, a sensitivity index based on partial derivatives [83] is obtained as follows:

$$\zeta_v = \frac{1}{L}\sqrt{\sum_{i=1}^{L}\left(\frac{\partial \tilde{f}}{\partial v_i}(\bar{N}_i, \bar{S}_i)\right)^2}, \tag{3.4}$$

where L is the number of available training samples, and v is the input label. In our experiment, v indexes the set $\{\text{NR}, \text{SR}, \text{Elev.}, \text{EC.}, \text{Soil}\}$. As the value of the partial derivative depends on the point of the input space it is being evaluated, our sensitivity index takes in account the partials for every sample in our training dataset. Notice that each element v_i in the input map v has its gradient dependent on all cropped inputs (Figure 3.2), with $\bar{N}_i$ and $\bar{S}_i$ being the only manageable inputs. Table 3.1 shows the index ζ for each input for the nine studied fields, revealing that the model is in fact sensitive to the selected manageable variables. Field 1 is an exception, showing an index of zero for nitrogen rate, which is expected since a constant rate was

used for this field during the OFPE.

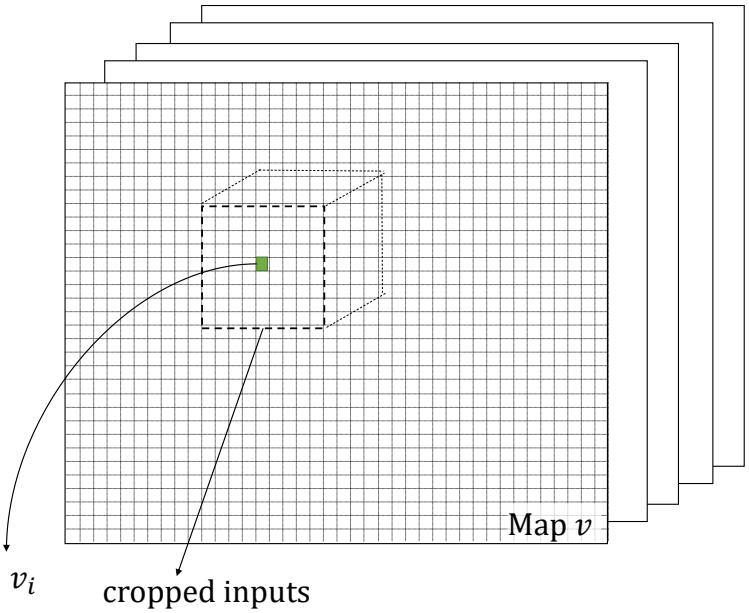


Figure 3.2: Region for gradient evaluation of v_i.

Table 3.1: Sensitivity index ζ for each input.

Field	NR	SR	Elev.	EC.	Soil
1	0.00	2.86	0.84	0.14	0.45
2	1.98	1.54	1.22	0.65	0.71
3	1.22	5.23	0.11	0.25	0.56
4	0.86	1.11	1.48	0.74	0.31
5	0.76	0.22	0.83	0.47	0.38
6	2.81	3.51	0.17	0.29	0.29
7	1.45	0.92	0.38	0.22	0.46
8	0.63	0.55	2.32	0.02	0.17
9	1.16	0.50	0.04	0.32	0.06

3.4 Genetic Algorithm

As shown in Chapter 2, the yield value at one point in the field depends on a limited area around it. Then, if we consider a larger scale, the problem of optimizing fertilization rates on part of a field becomes more decoupled from optimizing the remaining areas. In this sense, Genetic Algorithms (GA) are a promising solution to our problem. As new solutions in a generation are a result of combinations and mutations of previous solutions, this algorithm can explore the input space more efficiently by keeping parts of previous solutions that are close to optimal for the next generation. Genetic algorithms have been used before in spatial allocation problems [84, 85] and demonstrated good performance when compared to other optimization methods.

Using the rate of nitrogen and seed at each unit of analysis in the field (i.e., 5×5m) is the most obvious choice for encoding a solution. However, preliminary experiments revealed the algorithm did not converge to a solution due to the high number of genes in each individual. Hence, we propose using control points to encode a candidate solution in the algorithm. We define a grid of control points spaced 50 meters between them. This spacing is big enough to reduce the solution's size by a factor of a hundred. To construct the nitrogen and seed maps corresponding to a solution, we interpolate the control points using kriging.

The fitness function used to evaluate each individual in a generation (i.e., a candidate solution) is given by the same expression as in (3.2). Each population has 30 individuals from which the best three are kept for the next generation. The probabilities of crossover and mutation were set to 0.8 and 0.2 respectively. The constant rate from status quo management and the nitrogen and seed maps used during the OFPE are included in the

initial population to provide the algorithm with good candidates. Algorithm 2 shows the pseudo-code for the GA.

Algorithm 2 Pseudocode for the genetic algorithm.

```
Initialize a random population.
Replace 2 individuals with status quo and OFPE rates.
for n = 1 : max generations do
  for j = 1 : population size do
    [N, S] = kriging(j)
    Y_j = p_V f(N, S) − p_N Σ^i N_i − p_S Σ^i S_i
  end for
  Perform Crossover operation.
  Perform Mutation operation.
  Reproduce the best five individuals.
  Generate the next population.
end for
```

3.5 Experiments and Results

Experiments were conducted with the same nine fields used in Chapter 2 to demonstrate the monetary potential of the proposed optimization algorithms. The total net revenue (i.e., yield value discounting costs of nitrogen and seed) is estimated for the initial condition (status quo rates applied by the farmer) and for the resulting maps from the gradient ascent algorithm and GA. Table 3.2 shows the total percent change on expected net revenue, nitrogen, seed, and yield values after optimization.

Results show a 3.5% average increase in the expected net revenue when using the gradient ascent algorithm, with field 2 going up to 5.2%. It can be observed that the increase in the net revenue is obtained through different strategies, depending on the field. For some fields, the rate of nitrogen and seeds was increased to generate more yield, while others showed to be unresponsive to high rates of nutrients. In such cases, the algorithm was able

Table 3.2: Change in net revenue, nitrogen, seed and yield after optimization.

Field	Gradient Ascent				Genetic Algorithm			
	N.R.	N	Seed	Yield	N.R.	N	Seed	Yield
1	3.6%	0%	10.5%	4.6%	1.3%	0%	15.1%	2.4%
2	5.2%	-17.3%	-18.2%	-0.9%	4.1%	-9.5%	-15.6%	-0.2%
3	3.8%	18.0%	13.4%	5.6%	2.1%	18.7%	15.4%	5.1%
4	4.2%	-20.8%	-11.4%	-0.2%	3.9%	-17.9%	-10.1%	0.1%
5	2.7%	-12.8%	-11.8%	0.0%	2.3%	-10.2%	-11.1%	0.2%
6	1.8%	10.1%	4.6%	2.4%	0.4%	16.1%	7.9%	3.2%
7	2.2%	-30.4%	-17.6%	-0.7%	1.3%	-28.0%	-15.7%	-0.1%
8	3.7%	-42.3%	-13.0%	0.1%	3.2%	-40.1%	-11.6%	0.4%
9	4.1%	-12.7%	-11.7%	-0.2%	3.3%	-10.1%	-9.7%	0.2%

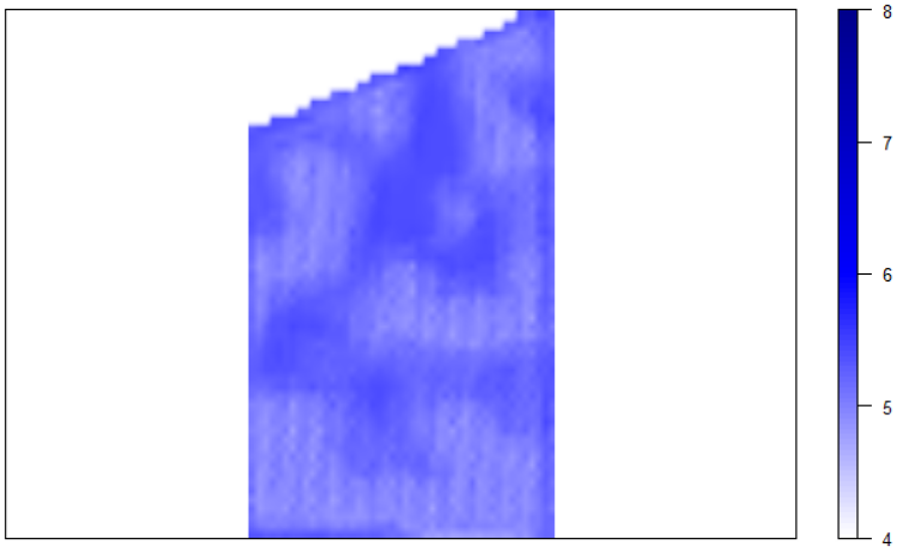


Figure 3.3: Optimized nitrogen map (UAN28/acre) showing low response areas on field 7.

to select areas where nutrients could be reduced without reducing the yield. Field 7 is a good example of how the algorithm reduced the nitrogen rate in unresponsive areas. Figure 3.3 shows the final nitrogen map for this field.

With tested values for the step size and momentum terms, the algorithm converged within an average of 30 iterations. Additional experiments were made initializing the input maps with random values. In such experiments, the optimization algorithm also converged within 30 iterations to very similar maps to the ones obtained with different initial conditions. These results indicate that the algorithm can find either the global maxima or a flat local maxima. The second case could be considered even better than a sharp global maxima since it is more robust to undesired variations during the nitrogen application process.

As we use control points with a lower resolution in the genetic algorithm, it is natural to observe a less detailed map when compared with the one from gradient ascent. To illustrate this, Figure 3.4 compares the nitrogen and seed maps obtained from the gradient ascent algorithm with the ones obtained through GA in field 6. Notice that the difference between the algorithms goes beyond the chosen resolution. As Table 3.2 reveals, the genetic algorithm resulted in a lower increase in the expected net revenue. The algorithm took, on average, 150 generations to converge. Figure 3.5 shows the convergence in field 6.

3.6 Discussion

This chapter presented an optimization framework to find the manageable inputs of the MSCNN-LF that maximize the expected crop's net revenue. When compared to traditional farming practices, this framework showed an

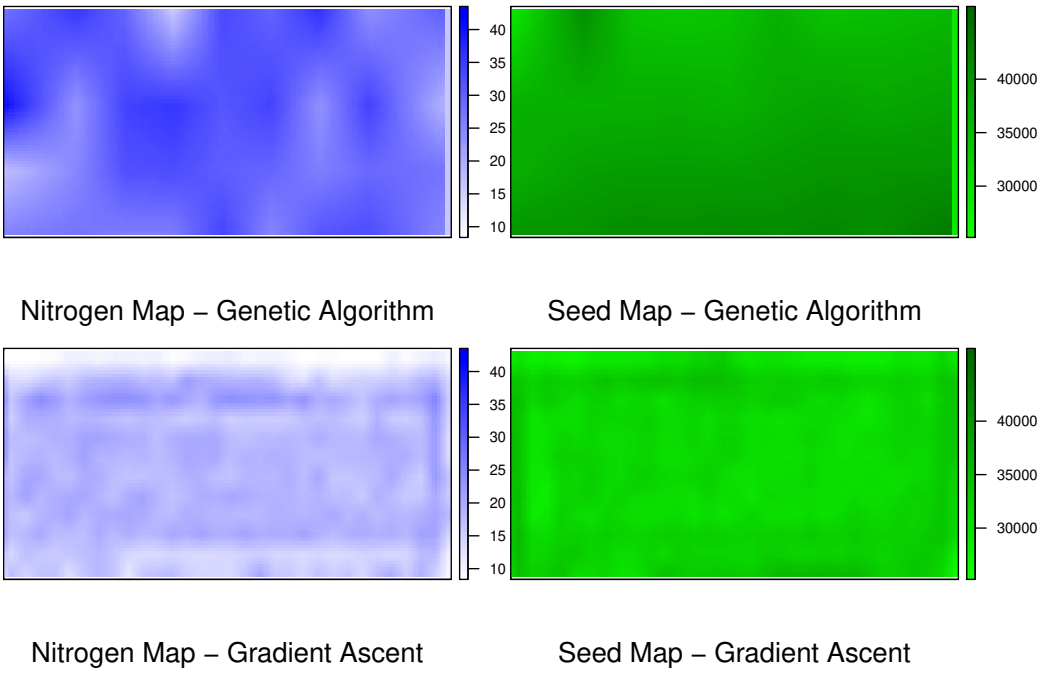


Figure 3.4: Comparison between maps obtained by gradient ascent and genetic algorithm on field 6.

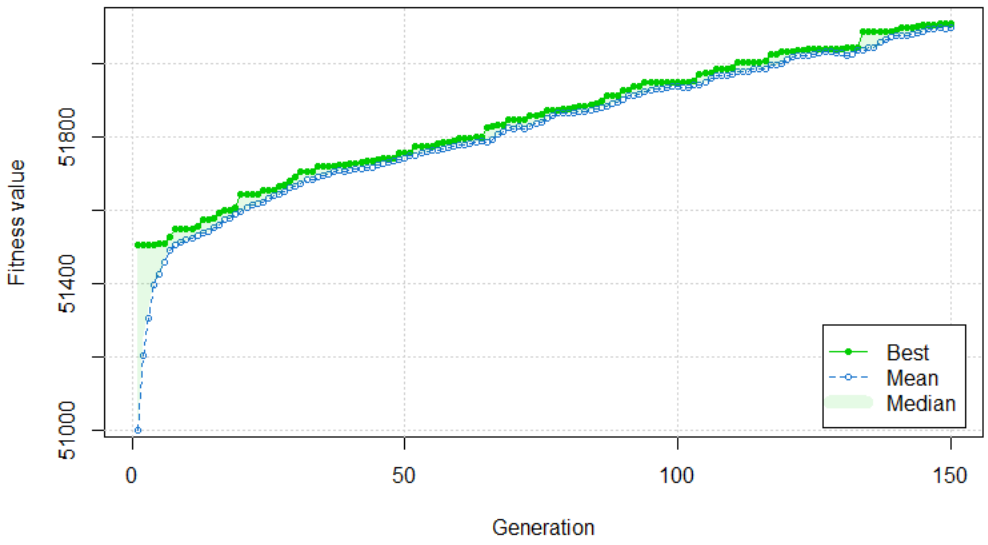


Figure 3.5: Net revenue in USD over the generations in field 6.

increase in crop yield return discounting the costs for all fields. The benefits of decreasing nitrogen while maintaining yield are of great importance to subdue the environmental impact caused by water pollution.

We also proposed a method to calculate a sensitivity index that can be used for ranking the importance of each input variable in the CNN trained model. Having this information allows farmers to understand the causes of productivity losses and variability in their fields.

Notice that the results depend on the initial conditions and are based on the model's predictions, rather than on real experiments. Nevertheless, this experiment aims to evaluate the optimization framework regarding its ability to drive the output of the yield prediction model to maximize an objective function. The performance of this algorithm in real experiments will depend on the model's accuracy, which was evaluated in Chapter 2 and showed to be better than other machine learning methods.

CHAPTER 4

UNCERTAINTY QUANTIFICATION AND RISK AVERSE-OPTIMIZATION.

This chapter proposes two complementary solutions to leverage the applicability of the previously proposed CNN models into DSS. Firstly, the model with the best performance proposed in Chapter 2 is modified under the deep ensemble framework to provide uncertainty estimation of predictions. Then, a gradient-based optimization algorithm is proposed to find the nitrogen and seed maps that maximizes the farmer's net revenue with safety bounds. We formulate a tractable objective function in which the risk constraints are easily adjusted according to each farmer's risk aversion. Data from six cornfields in the US were used to train the models and compare the results of the proposed optimization algorithm against the status quo management.

4.1 Uncertainty Quantification of the MSCNN Model

We consider two sources of uncertainty: epistemic and aleatoric [86]. The first corresponds to the "lack of knowledge" in our data, meaning that there are unconsidered hidden variables in the process we are modeling. It results in an ambiguity in the response variable for a given input over the training dataset. The aleatoric uncertainty is the inherent variability in the modeled phenomena combined with the randomness in the training process of the MSCNN model. The last causes high variability in the predictions from out-of-distribution inputs. In a general sense, one can reduce the epistemic

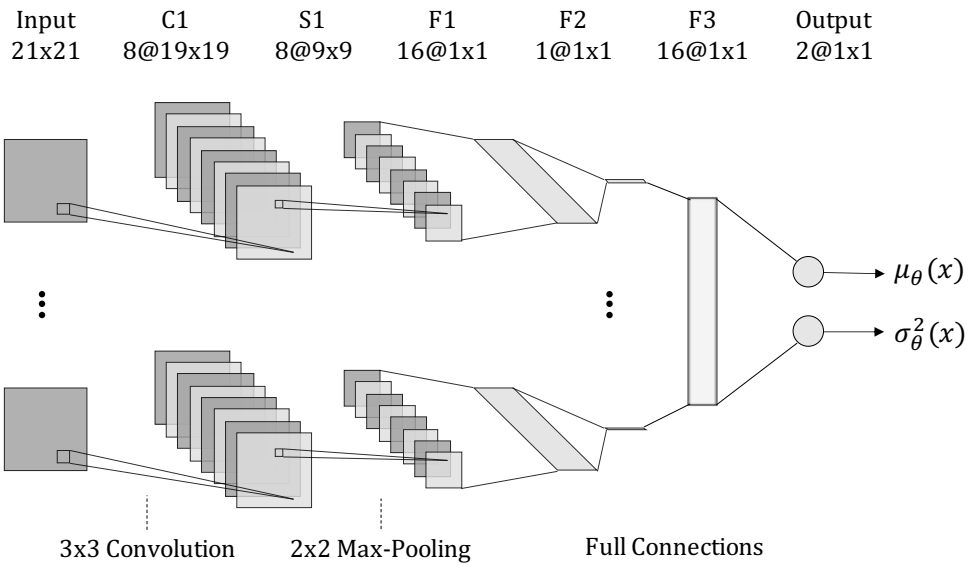


Figure 4.1: RA-CNN-LF with prediction's mean and variance.

uncertainty using additional information, but not the aleatoric [87].

In Chapter 2, we proposed a Multi-Stream Convolutional Neural Network with Late Fusion of inputs (MSCNN-LF) to predict yield. It was trained in a supervised fashion via backpropagation to minimize the Mean Squared Errors (MSE) of predictions as in a regression problem. The model outputs a single value that best represents the data from a set of similar or identical inputs. It is evident, in this case, that predictive uncertainty quantification is not considered.

To incorporate uncertainty quantification into the MSCNN-LF model, we use the concept of Deep Ensembles proposed in [27]. We create a Risk-Aware Convolutional Neural Network with Late Fusion of inputs (RA-CNN-LF) by assuming our uncertainty follows a Gaussian distribution. We first change the output layer of the MSCNN-LF architecture to have two neurons representing the mean ($\mu_\theta(x)$) and variance ($\sigma^2_\theta(x)$) of the predicted distribution. The full architecture is depicted in Figure 4.1.

We want – for a given input – the network to produce an output distri-

bution that is consistent with the epistemic uncertainty in the training set. For this, instead of optimizing the network's weights by minimizing the MSE value, we now minimize the Negative Log-Likelihood (NLL) between the distribution defined by the predicted mean and variance, and the samples from the training set, considering them as sampled from a Gaussian distribution. Then, the loss function used in the training process is given by:

$$-\log p_\theta(y_n|x_n) = \frac{\log \sigma_\theta^2(x)}{2} + \frac{(y - \mu_\theta(x))^2}{2\sigma_\theta^2(x)} \tag{4.1}$$

By training the new architecture with the above loss function, the model makes predictions with uncertainty quantified by the value of $\sigma_\theta^2(x)$. So, for an input x, the model approximates the output distribution observed in the training set. This is good but insufficient to safely explore the input space looking for the optimal input management x. The epistemic uncertainty quantification works well when inputs are close to the training set's domain. However, we want our model to also output high predictive uncertainty for inputs not in the training set. Hence, we make use of the aleatoric nature of the training process to estimate the uncertainty from the domain shift, as explained next.

Instead of training and using a single network for our predictions, we train an ensemble containing networks with identical architectures combined as a mixture model. The difference between members of the ensemble comes from the training process. Each member is initialized with a different set of random weights and trained on different shuffled versions of the training set. By following this procedure, the randomness injected in the process increases the difference between predictions from each member given inputs far from

the domain of the training set.

After training the members of the ensemble, we need to combine each member to obtain the final prediction. The mean of the mixture model is given by equation (4.2), and the variance by (4.3):

$$\mu_\star(X) = \frac{\sum^m \mu_{\theta_m}}{M} \tag{4.2}$$

$$\sigma^2_\star(X) = \frac{\sum^m (\sigma^2_{\theta_m}(X) + \mu^2_{\theta_m}(X)) - \mu^2_\star(X)}{M} \tag{4.3}$$

4.2 Risk-Averse Gradient Ascent

In this section, we describe how to use the RA-CNN-LF model to optimize manageable input variables with risk estimation and constraints. Similar to the optimization algorithm proposed in Chapter 3, the challenge posed by the proposed model lies in the increased input space resulting from using "patches" of the input maps instead of single points as inputs. In addition, the patches that are used for predicting yield at two points close to each other in the field have overlap between them, making the sequential and independent optimization over the field impossible. As a result of this increased input space, and for the previous experiments in Chapter 3, we opt for not using numerical optimization algorithms. Instead, we propose a gradient ascent method for maximizing an objective function yet to be defined.

4.2.1 Problem Formulation

Remember that the predictive model has five input variables: nitrogen and seed rate prescription maps, elevation map, soil's shallow electroconductivity, and a pre-season satellite image from the field. We train the RA-CNN-LF model to map patches cropped from the input maps (i.e., a 21×21 raster for each input variable representing a 100×100m area) to a predicted normal distribution (defined by μ and σ^2) of the yield at the patch's center element. A predicted yield map is then constructed by making predictions for each element in the yield map. Notice that, again, only the nitrogen and seed maps are manageable variables, so for our optimization algorithm, we consider the other three inputs as constants.

More formally, we define a function $\tilde{f}(\bar{N}, \bar{S}) : \mathbb{R}^{21\times21} \times \mathbb{R}^{21\times21} \longrightarrow \mathcal{N}(\mu, \sigma^2)$ representing our RA-CNN-LF model, where $\bar{N}$ and $\bar{S}$ respectively represent patches cropped from the nitrogen and seed maps. We also define the total productivity of a field as the sum of the yield predicted over the field: $f(N, S) = \sum^i \tilde{f}(\bar{N}_i, \bar{S}_i)$, where i is the index of the position of each element in the yield map, and N and S represent the entire nitrogen and seed maps respectively. The expected farmer's net revenue is defined in (4.4), where p_v, p_N, and p_S are the prices of the yield, nitrogen and seeds, respectively. In addition, we define the standard deviation of net revenue in (4.5) to be used as a measurement of risk:

$$\mu_r(N, S) = p_V \ \mathbb{E}(f(N, S)) - p_N \sum^i N_i - p_S \sum^i S_i \tag{4.4}$$

$$\sigma_r(N, S) = p_V \ \sqrt{\text{var}(f(N, S))} \tag{4.5}$$

Then, our final objective function is a composition of μ_r and σ_r^2, so we can

maximize the expected net revenue while keeping a certain level of confidence in our predictions. We propose to weigh the uncertainty by a constant β in our objective function that can be defined according to each farmer's risk aversion. We also create a constraint on the Value at Risk (VaR) ρ [88], which is, in this case, the minimum net revenue the farmer expects with a confidence of 97% (z-score = 2). Finally, our optimization problem is formulated as:

$$\begin{aligned} &\max_{N,S} \ \mu_r(N,S) - \beta\sigma_r(N,S) \\ &\text{subject to: } \mu_r(N,S) - 2\sigma_r(N,S) \geq \rho \\ &N_{min} \leq N_i \leq N_{max}, \forall \ i \\ &S_{min} \leq S_i \leq S_{max}, \forall \ i. \end{aligned} \tag{4.6}$$

The two additional constraints ensure that the search for rates of nitrogen and seed stay bounded by the minimum and maximum value utilized during the experiment that generated the data used to train the model. Notice that such rates are chosen based on what the farmer considers safe and is willing to test.

4.2.2 Gradient Ascent with Log Barrier Function

We propose a gradient-based solution for the posed optimization problem. The log barrier method [89] is used to account for the first constraint, so we need to re-write our objective function as in (4.7):

$$Y = \mu_r(N,S) - \beta\sigma_r(N,S) - \frac{1}{t}\log(c) \tag{4.7}$$

$$c = \mu_r(N,S) - 2\sigma_r(N,S) - \rho \tag{4.8}$$

The gradient of (4.7) is defined as $\nabla Y = \left(\frac{\partial Y}{\partial N}\ ,\ \frac{\partial Y}{\partial S}\right)$ and is given by equations (4.9) and (4.10), where P_N and P_S are matrices of same size of the yield raster, containing all elements equal to the price of nitrogen and seeds respectively (i.e., resulting on an element-wise subtraction). The detailed derivation of these equations can be found in the Appendix, as well as the methodology to obtain the gradients of the expectation and variance of the productivity w.r.t. the nitrogen and seed maps:

$$\frac{\partial Y}{\partial N} = \left(p_Y \frac{\partial \mathbb{E}(f(N,S))}{\partial N} - P_N\right)\left(1 + \frac{1}{tc}\right) - \left(\frac{p_Y}{2\sqrt{\text{var}(f(N,S))}}\frac{\partial\ \text{var}(f(N,S))}{\partial N}\right)\left(\beta + \frac{1}{tc}\right) \tag{4.9}$$

$$\frac{\partial Y}{\partial S} = \left(p_Y \frac{\partial \mathbb{E}(f(N,S))}{\partial S} - P_S\right)\left(1 + \frac{1}{tc}\right) - \left(\frac{p_Y}{2\sqrt{\text{var}(f(N,S))}}\frac{\partial\ \text{var}(f(N,S))}{\partial S}\right)\left(\beta + \frac{1}{tc}\right) \tag{4.10}$$

The incorporation of the VaR constraint into the objective function through the log barrier function is conditioned by starting the optimization algorithm from some point in the set of feasible solutions (i.e., $[N_0, S_0]$ must satisfy the constraint). Hence, we first need to find an initial solution that satisfies the VaR constraint by having our algorithm look for it as a first step. Thus, the first step of our algorithm maximizes the VaR itself, by maximizing the expression in (4.8), the gradient of which is given by simply choosing $t = \infty$ and $\beta = 2$ in the expressions (4.9) and (4.10).

The multidimensional nature of the problem's input space suggests that the problem is unlikely to be convex, so a momentum term [82] is incorporated to prevent the search from stopping in a local maxima or a saddle point. The momentum term V is used by integrating the gradients' values at every iteration with a discount factor, so it works as a moving average of the gradient. It is updated every time step using the expression $V_{i+1} = \alpha V_i + \nabla Y(N, S)$, and the update of the input maps uses this term instead of the gradient values. The overall optimization technique is summarized in Algorithm 4.

Algorithm 3 Pseudocode for the constrained gradient ascent with momentum

1: $[N, S] \leftarrow$ constant rate ; $V \leftarrow 0$
2: **while** $(\mu_r(N, S) - 2\sigma_r(N, S) \leq \rho)$ and $(i \leq$ max iterations$)$ **do**
3: $\quad t \leftarrow \infty, \beta \leftarrow 2$
4: $\quad V = \alpha V + \nabla Y(N, S)$
5: $\quad [N, S] = [N, S] + \lambda V$
6: $\quad i = i + 1$
7: **end while**
8: **for** $i = 1 :$ max iterations **do**
9: $\quad t \leftarrow 1, \beta \leftarrow$ desired β
10: $\quad V = \alpha V + \nabla Y(N, S)$
11: $\quad [N, S] = [N, S] + \lambda V$
12: $\quad t = t + \epsilon$, $\lambda = \lambda d_r$
13: **end for**

By the end of the while loop (line 7 in Algorithm 4), the pair of nitrogen and seed maps will be a solution for a VaR strictly greater than ρ, if such solution exists. Once in the feasible set, the algorithm starts to maximize the objective function (4.7). The term t is used to weigh the log barrier function and must be increased at every iteration. By increasing t, the additional term representing the constraint approximates an indicator function, and the objective function becomes closer to its original form (eq. (4.6)). The step size λ is reduced by a decay rate d_r at every iteration to refine the search for the global maxima.

4.3 Experiments and Results

The same fields and procedures from Chapter 2 were used to evaluate the predictive performance of the RA-CNN-LF model. Nine fields averaging 40 ha were selected from the Data Intense Farm Management (DIFM) project [68] dataset. The data from each field was recorded during On-Farm Precision Experiments (OFPE) in the 2017 season with approximately 200 experimental units each, represented by 85 m long and 18 m wide polygons. Six fields are rain-fed fields in Illinois (Fields 2, 4, 7, 8, and 9) and Ohio (Field 5), and three are irrigated fields in Nebraska (Fields 3 and 6) and Kansas (Field 1). The nitrogen and seed rates in each experimental unit were randomized, ranging from 20% below to 20% above the status quo rate usually applied by the farmer.

As proposed in [78], partitions of the data were created by spatially dividing each field in stripes perpendicularly to the longest field's dimension to maximize the physical distance between the data collected from each partition. Each field was divided into five partitions, four being used for training the model, and one for testing. Experiments were conducted using cross-validation over the five folds in each field. The members of the deep ensemble were trained independently over the training sets, starting from random initial weights, and using the Adam optimizer [79]. The metric used to evaluate the predictive performance of the ensemble is the averaged RMSE values between predicted mean and true values over the five test sets from the cross-validation setup.

4.3.1 Model's Performance

The first experiment tests the influence of the number of members in the ensemble and the number of epochs during the training process. Even the NLL optimizes the weights of the network to approximate both μ and σ^2, the effect of overfitting affects them differently. The predicted mean is unlikely to get worse generalization power due to the averaged prediction from the members of the ensemble. On the other hand, the predicted variance is affected by any disagreement between the members of the ensemble (see eq. 4.5). Table 4.1 shows the averaged RMSE between predicted means and true values in the test sets for different numbers of members and training epochs. Results suggest that using more than three members in the ensemble does not increase the predictive performance of the mean values. Also, training the model for more epochs does not decrease the predictive performance of μ.

To assess the predicted variance over the training epochs, for each sample in the test set we computed the distance between the predicted mean value and the true value in terms of the predicted standard deviation ($e = \frac{\mu(x)-\bar{y}}{\sigma(x)}$). This method provides a standardized residual, and we expect the distribution of the residuals to follow a normal distribution with zero mean and unitary variance. Figure 4.2 shows an example of the distribution of the standardized residuals in three cases: (a) an underfitted model with "conservative" predictions; (b) a good model, and (c) an overfitted model with "optimistic" predictions.

The NLL between the ensemble's predicted distributions and the validation data was computed online during training as a metric for early stopping. Figure 4.3 shows the averaged NLL over fields' test sets for different num-

Table 4.1: RMSE between predicted means and true values for different numbers of members (M) and training epochs.

Field	M=2				M=3				M=5			
	1	5	10	15	1	5	10	15	1	5	10	15
1	0.75	0.67	0.66	0.66	0.70	0.64	0.63	0.63	0.67	0.65	0.63	0.63
2	0.84	0.81	0.80	0.80	0.83	0.80	0.79	0.79	0.83	0.79	0.78	0.78
3	0.53	0.60	0.60	0.59	0.55	0.58	0.59	0.58	0.53	0.58	0.58	0.59
4	0.75	0.76	0.75	0.74	0.75	0.74	0.73	0.73	0.74	0.74	0.73	0.72
5	0.73	0.72	0.71	0.72	0.72	0.71	0.71	0.70	0.72	0.71	0.71	0.72
6	0.50	0.47	0.47	0.47	0.49	0.47	0.47	0.46	0.50	0.48	0.47	0.46
7	0.71	0.68	0.68	0.68	0.70	0.69	0.68	0.68	0.72	0.69	0.68	0.68
8	0.92	0.92	0.92	0.92	0.94	0.93	0.92	0.93	0.93	0.92	0.92	0.93
9	0.66	0.63	0.63	0.61	0.65	0.62	0.62	0.62	0.65	0.63	0.63	0.62
Avg.	0.71	0.70	0.69	0.69	0.70	0.69	0.68	0.68	0.70	0.69	0.68	0.68

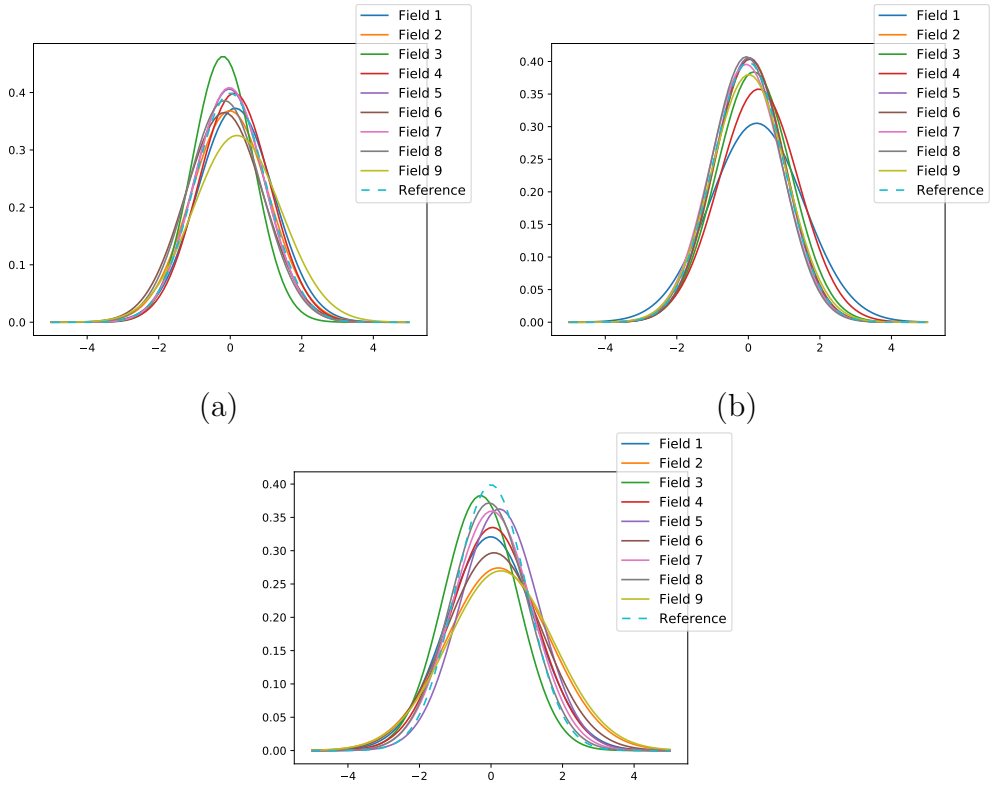


Figure 4.2: Distribution of the standardized residuals for (a) an under fitted model, (b) a good model, and (c) an over fitted model.

bers of members in the ensemble. Although an ensemble with five members demonstrated a similar predictive mean performance as one with three, the first resulted in lower NLL on the test set. No significant reduction in the NLL is observed in ensembles with more than five members, so we have adopted this number in further experiments.

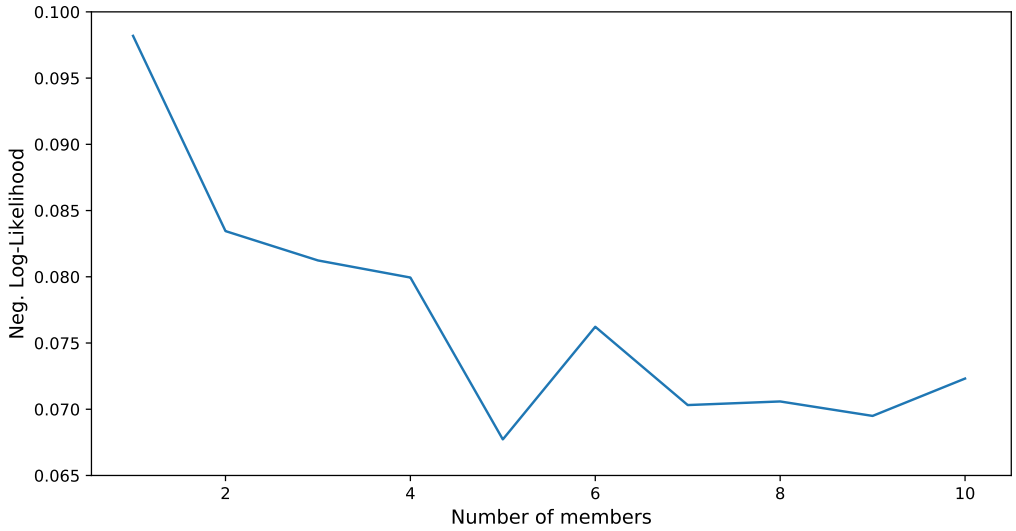


Figure 4.3: Averaged negative log-likelihood over fields for different numbers of members in the ensemble..

After following this procedure for training a model for each field, we compared their RMSE with the ones from the MSCNN-LF. Table 4.2 shows the averaged RMSE over the folds' test sets for the CNN-LF and a five-members RA-CNN-LF. The table also includes the observed mean and standard deviation of the true yield data. As the data was previously standardized for training, all values must be multiplied by the standard deviation of the field to get the absolute errors. Nevertheless, the standardized results help to compare the model's performance over the fields.

Results show that the RA-CNN-LF outperformed its former architecture in seven of the nine tested fields, and achieved equal performance in the other two. The better result can be explained by the ensemble nature of the RA-CNN-LF, which, as shown in [90], significantly increases the prediction's performance. The better predictive performance, although desirable, is not the main result of this chapter, which focuses on incorporating predictive uncertainty for optimizing the crop inputs.

We constructed maps for predicted yield mean and standard deviation by

Table 4.2: Crossvalidation averaged RMSE over test dataset, yield standard deviation, and yield mean.

Field	RMSE		Yield [Kg/ha]	
	RA-CNN-LF	CNN-LF	Stdv.	Mean
1	0.64	0.66	3240	12500
2	0.80	0.83	2290	10700
3	0.56	0.58	1230	14400
4	0.74	0.75	900	12200
5	0.70	0.70	1360	14500
6	0.47	0.48	1140	14700
7	0.68	0.69	1150	15700
8	0.92	0.94	2267	14100
9	0.63	0.63	1140	12600
Avg	0.68	0.70	1635	13489

making predictions for each element in the yield map, using the RA-CNN-LF model. Figure 4.4 compares the predicted and true yield maps and also shows the "uncertainty map" based on the predicted standard deviation on fields 4, 5, and 7. The uncertainty map may bring insightful information to farmers regarding which parts of the fields present more variation resulting from unobserved variables. Notice that higher uncertainty is predicted on the borders of each field, where the data generated by the yield monitor is not reliable due to changes in the harvester's speed and direction.

4.3.2 Optimization Results

To evaluate the proposed optimization, we followed the steps described in Algorithm 4 to find the optimal nitrogen and seed maps on fields 2, 3, 4, 5, 6, and 7. In all of the following experiments, the nitrogen and seed maps were initialized with the status quo constant rate used by each farmer, so we can demonstrate the profitability potential of our optimization algorithm. We predicted the expected net revenue (i.e., revenue discounting nitrogen

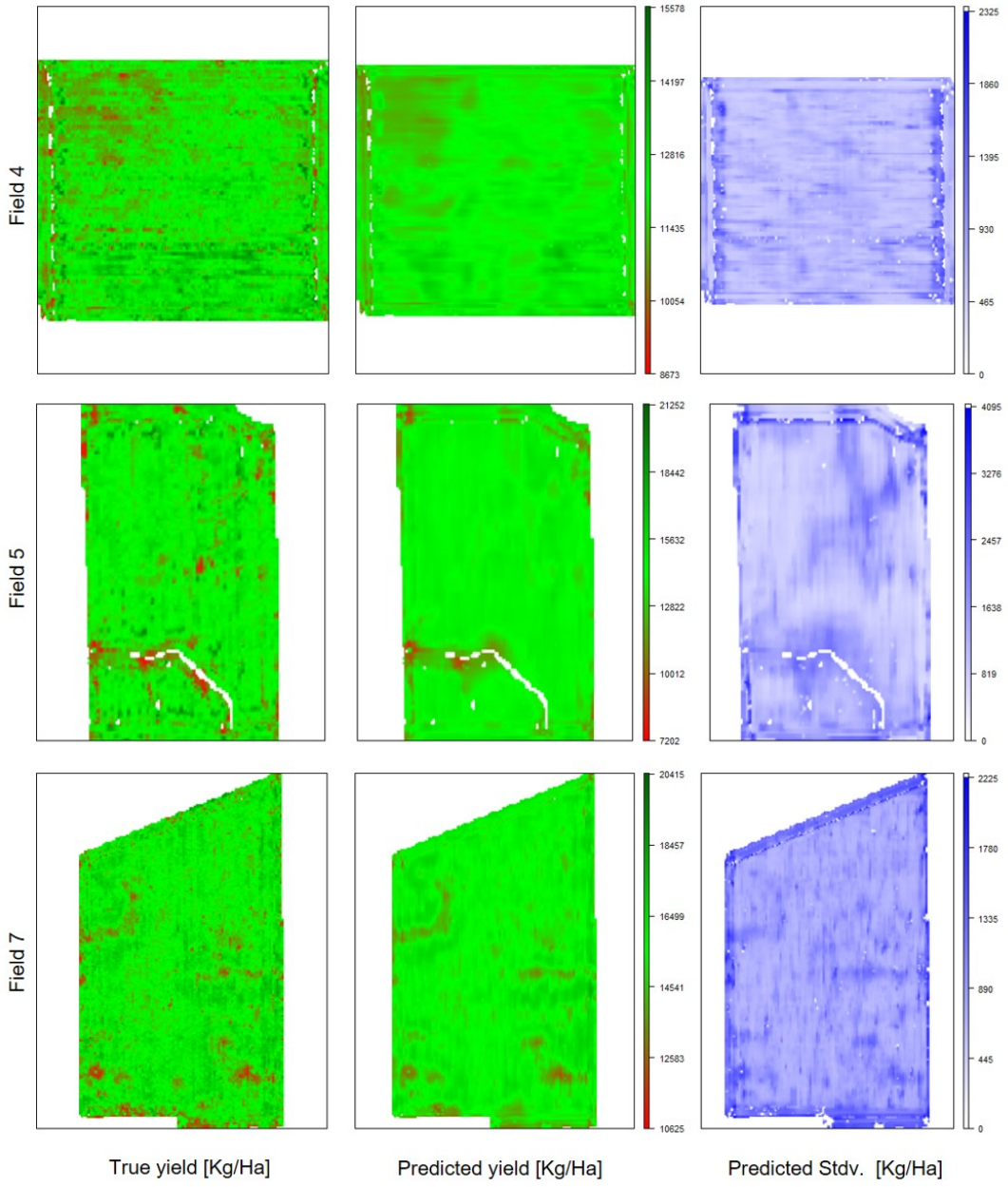


Figure 4.4: True yield, predicted yield, and predicted standard deviation [Kg/Ha] in fields 4, 5, and 7.

Table 4.3: Percent change in net revenue, predicted standard deviation, total nitrogen and seed costs, and total yield.

Fields		2	3	4	5	6	7	Avg.
Net Revenue	$\beta = 0$	6.4	0.4	1.8	1.9	5.8	2.7	3.2
	$\beta = 1$	5.8	-0.5	-0.9	1.2	5.4	2.9	2.3
	$\beta = 2$	4.7	-0.7	-2.2	0.8	5	3.2	1.8
	$\beta = 5$	3.3	-1	-2.3	0.3	4.6	2.7	1.3
Stdv	$\beta = 0$	-0.2	-4.6	-1	-0.3	-20.3	-1.3	-4.6
	$\beta = 1$	-7.4	-10.9	-28.4	-5.7	-25.9	-13.7	-15.3
	$\beta = 2$	-8.5	-11.2	-32.6	-6.2	-26.8	-16.5	-17.0
	$\beta = 5$	-8.8	-10.5	-30.9	-6	-26.7	-17.6	-16.8
Nitrogen	$\beta = 0$	-15.4	-4.3	-4.8	-8.2	-4.9	-16.9	-9.1
	$\beta = 1$	-7.3	12.2	-10.7	-7.6	8.9	-13	-2.9
	$\beta = 2$	-4.5	12.9	-8.3	1.9	12.7	14.1	4.8
	$\beta = 5$	-1.7	16	-11.3	3.2	17.4	25.1	8.1
Seed	$\beta = 0$	-10.2	12	-10.7	-7.6	8.9	-13	-3.4
	$\beta = 1$	-16.9	6.5	-11.2	-11.6	7.7	-17.3	-7.1
	$\beta = 2$	-6.8	14	-7.8	-5.2	9.6	-8.5	-0.8
	$\beta = 5$	-3.3	15.9	-3.3	-1.7	11.1	-0.2	3.1
Yield	$\beta = 0$	0.9	0.6	-0.2	0.1	5.5	-0.2	1.1
	$\beta = 1$	2.3	1.2	-3.1	0.3	5.8	1.5	1.3
	$\beta = 2$	2.2	1.2	-3.3	0.3	5.7	2.2	1.4
	$\beta = 5$	1.9	1.3	-3.1	0.3	5.6	2.7	1.5

and seed costs) for the initial condition and the maps obtained from the optimization process for comparison. The prices used in the algorithm are $p_S = \$0.25$/kseed, $p_N = \$0.44$/lb, and $p_Y = \$3.85$/bushel.

We start testing the optimization algorithm with no VaR constraint and different values of β to assess how the expected profitability changes when more weight is given to minimize the uncertainty. Table 4.3 show the percent change in expected net revenue, yield, nitrogen and seed costs, and also in the total variance in such predictions. As expected, increasing β reduces the predicted variance of the final solution, and, in most cases, results in lower expected net revenue. Results also suggest that more confident solutions are associated with higher rates of nitrogen and seeds.

Figures 4.5 and 4.6 show the nitrogen and seed maps resulting from the optimization algorithm with $\beta = 1$. As showed in Table 4.3, the nitrogen is reduced in most parts of the fields, indicating that the status quo constant rate is higher than necessary.

As Table 4.3 suggests, increasing β may reduce the expected net revenue. We then provide an alternative way to match the farmer's risk aversion by including the VaR constraint. It sets a minimum net revenue, after which the farmer is willing to take more risks to maximize the expected net revenue. After finding a solution that satisfies the constraint, the expected net revenue is maximized regardless of the increase in uncertainty (i.e., $\beta = 0$), unless it gets close to violating the constraint. Figure 4.7 shows the evolution of the net revenue during the optimization process for the constrained (with VaR = \$53.500) and the unconstrained (with $\beta = 0$) algorithms on field 6.

4.4 Discussion

A novel framework for risk-averse optimization of crop inputs was proposed. It encompasses a powerful yield prediction model with uncertainty quantification and a flexible optimization algorithm that can be tuned to match each farmer's risk aversion.

The MSCNN-LF architecture proposed in Chapter 2 was re-designed into the Deep Ensemble framework and trained using the negative log-likelihood loss function. The new model outputs a Gaussian probability distribution defined by predicted mean and variance. Results show that the new architecture not only incorporates uncertainty quantification measured by the predicted variance but also outperforms its original version in seven out of nine tested fields. The predicted variance in each element in the yield map

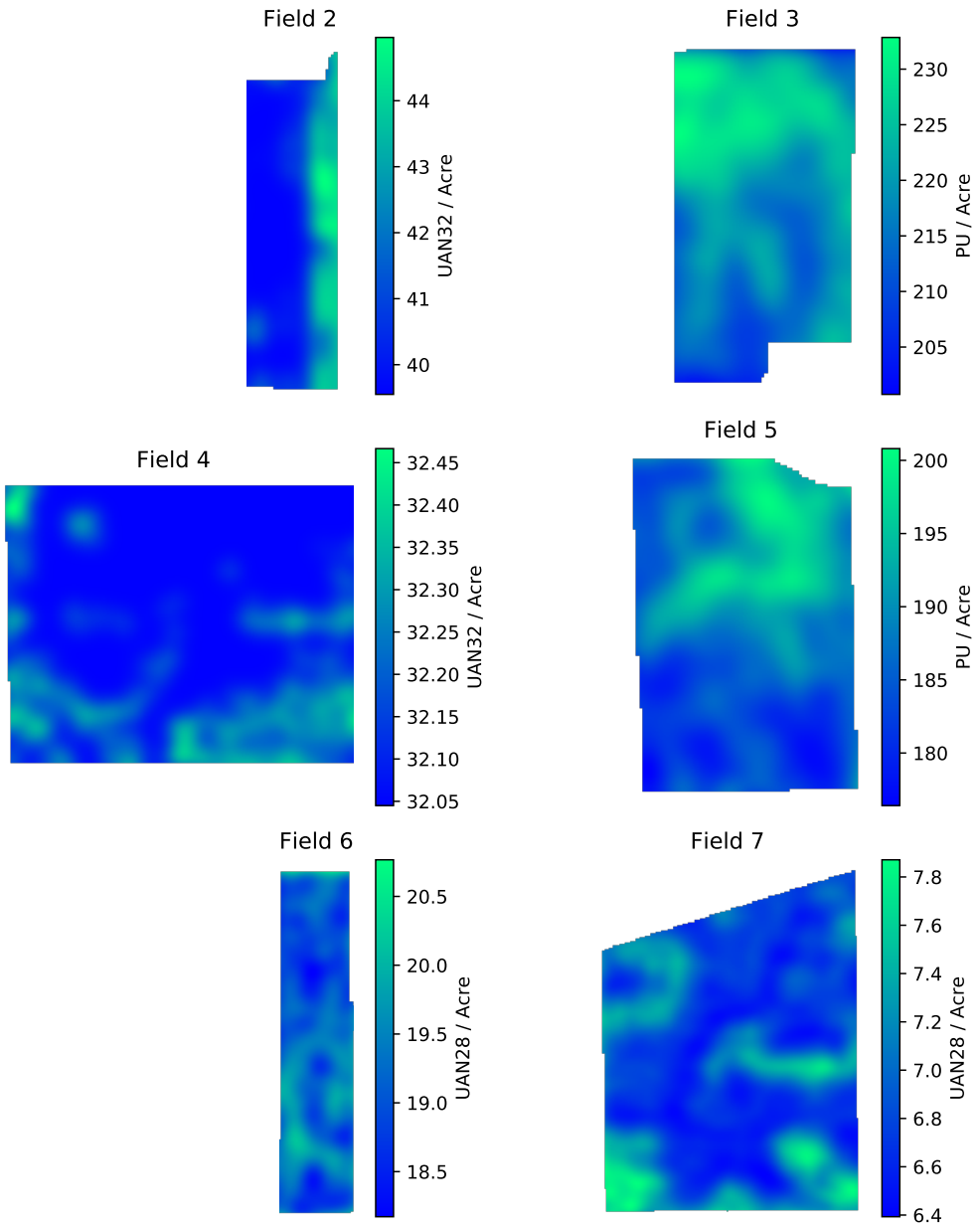


Figure 4.5: Optimized nitrogen maps for fields 3, 4, 5, 6, and 7.

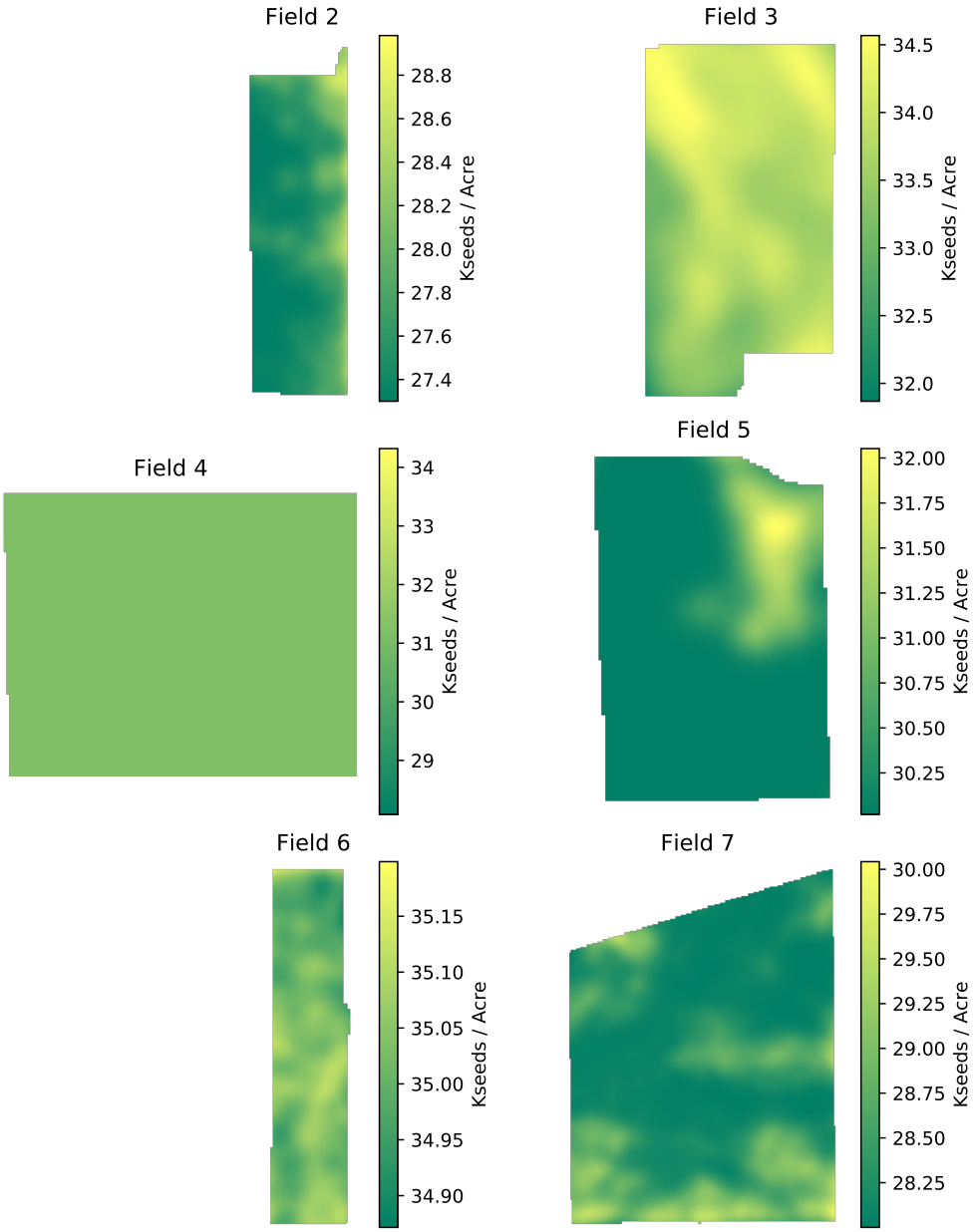


Figure 4.6: Optimized seed maps for fields 3, 4, 5, 6, and 7.

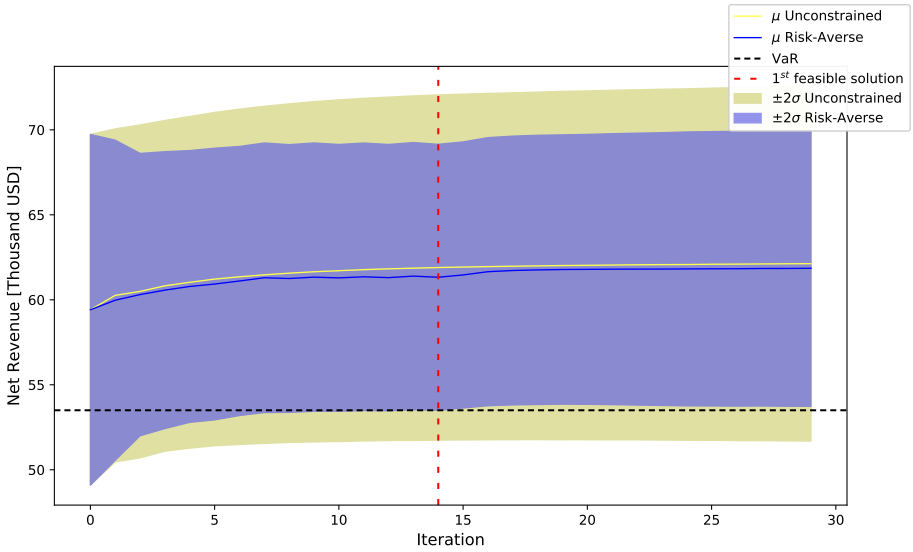


Figure 4.7: Net revenue over the iterations of the optimization algorithm for the constrained and unconstrained cases on field 6.

was used to construct an uncertainty map that can give valuable insights to farmers regarding the field's areas with high variability associated with unobserved variables.

The proposed optimization algorithm is based on the gradient ascent method and maximizes the expected net revenue by changing the manageable crop inputs while minimizing the predicted variance. A trade-off is observed between the increase in the expected net revenue and the level of confidence in the obtained solution. To account for this trade-off, we created a constraint on the value at risk. It prevents the algorithm from finding solutions with low confidence that can result in a net revenue below a feasible chosen threshold. We compared the expected net revenue from the optimization algorithm's solution with the one obtained from the status quo constant rate management. Experiments show an average increase of 3.3% in the expected net revenue and up to 6.8% in one of the tested fields. A significant reduction in the nitrogen and seed rates was observed, while the expected yield remained

almost at the same level. The obtained results demonstrate environmental and economic benefits over the status quo management, which is characterized by the over-application of fertilizer and seeds. Results also suggest that higher rates of nitrogen fertilizer and seeds result in less variability in the yield response, which may explain the farmer's choice for such rates.

CHAPTER 5

COVERAGE PATH PLANNING BY LEARNING FROM OFFLINE ALGORITHMS.

This chapter proposes a three-step framework to learn online coverage control strategies using demonstrations from offline algorithms. We start by constructing a grid representation for different planar and non-planar maps and creating a coverage path using offline methods. Then, an agent based on a CNN is trained in a supervised fashion over pairs state-action using a condensed representation of the map centered at the agent's position. The final step consists of using a policy gradient reinforcement learning algorithm to refine the learned strategy. The performance of this approach is tested on different maps and compared with their offline planning.

5.1 Construction of a training set

A reasonable number of maps and their respective optimized paths must be generated to create examples of state and action pairs and train the CNN agent. The agent's performance is expected to depend on the number of training examples and on how representative the training maps are when compared to the testing ones. In this dissertation, instead of using real crop fields, we use a methodology to generate simulated maps for training and testing. This choice comes from the fact that there is a limited number of fields in our dataset, and they don't have information regarding potential obstacles. The simulated maps' construction procedure is explained next.

Each map is represented by a 20×20 square grid containing at least one obstacle (i.e., nodes the agent cannot navigate through). We assume this representation fully covers the map and that visiting every node in the grid is sufficient to complete the task. The maps are constructed by randomly choosing the number of obstacles, varying them from one to three. The obstacles' size is also defined by random sampling from a set containing integers from one to ten. The obstacles are then constructed by choosing a starting node in the map and setting the adjacent nodes as obstacles according to a pre-defined probability distribution. Such distribution assigns a higher probability for nodes that result in a straight line obstacle.

Two types of maps are generated: planar and non-planar. In planar maps, the cost of moving from one node to its neighbor is homogeneous all over the map, depending only on whether the agent makes a turn or overlaps a previously covered node. In non-planar maps, there is an extra cost associated with the change in the elevation between two nodes. The elevation profile of each generated map is defined automatically by a parametric linear function with randomly selected parameters. Figure 1 shows an example of planar and non-planar maps. For simplification purposes, the agent can perform only four actions while navigating the map (UP, DOWN, LEFT, and RIGHT).

To calculate the cost of each action in a path, a cost function must first be defined. In a general form, the cost function of moving from node i to node j is defined as:

$$C_{(i,j)} = k_z \cdot |z_j - z_i| + k_t \cdot \mathbb{1}_{\text{turn}} + k_o \cdot \mathbb{1}_{\text{overlap}}, \tag{5.1}$$

where $\mathbb{1}_{\text{turn}}$ and $\mathbb{1}_{\text{overlap}}$ are indicator functions on whether the step from i to j was a turn and/or overlapped a previously covered node. The constants

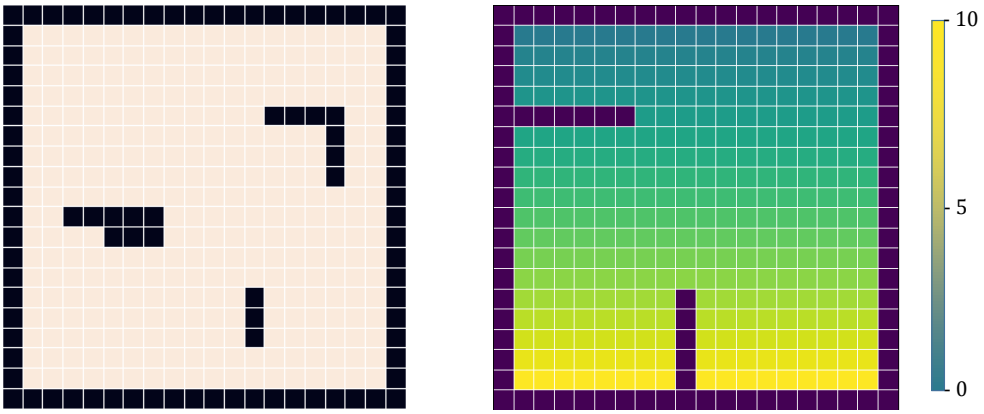


Figure 5.1: Examples of planar (left) and non-planar (right) maps.

k_z, k_t, and k_o are used to weigh the elevation, turning and overlapping costs respectively.

The offline solution for each planar map is obtained through cell decomposition and Boustrophedon motion. The order in which each cell is visited is determined using dynamic programming to minimize the cost function for the final path. The same solution is not applicable for non-planar maps since it doesn't consider the costs associated with the change in elevation. Next, we propose a heuristic Dijkstra's algorithm to solve the non-planar maps.

5.1.1 Heuristic Dijkstra's algorithm

A naive approach for this problem would be to create a tree in which each child node contains all the sequence of visited nodes leading to it. This approach would branch exponentially fast and is not scalable. Then, we propose a branching reduction technique based on a heuristic function to approximate this problem to the shortest path problem.

In the shortest path problem, the path from a node u to a node v in the optimal global path is also optimal. In the coverage problem, an action at

step t influences the cost of covering future nodes by modifying the map and changing the agent's position. In this case, a partition of the optimal global path is not necessarily the optimal path to cover the corresponding area (i.e., Bellman's principle of optimality does not hold). However, if we could estimate how much of the future cost is accounted for each action in the past, we could make the principle of optimality hold.

Let $J_k(u)$ be the cost-to-go of covering the k remaining nodes in the map starting from u, and $c_{(u,v|\tau)}$ be the edge cost of moving from node u to v given the previous sequence of nodes τ such that the number of covered nodes is increased by one. Then a Bellman equation is defined as:

$$J_k(u) = c_{(u,v|\tau)} + J_{k-1}(v) \tag{5.2}$$

As mentioned before, $J_k(u)$ depends on the previous actions that defined the path τ leading to u. Choosing the next node v that minimizes $J_k(u)$ requires to minimize it over all possible paths. So, we propose to decompose the edge cost as follows:

$$c_{(u,v|\tau)} = \bar{c}_{(u,v)} + L(\tau), \tag{5.3}$$

where the function $L(\tau)$ is the total extra cost created by all previous actions in the sequence τ, and $\bar{c}_{(u,v)}$ is the edge cost ignoring the overlap cost. Minimizing $L(\tau)$ at each step is the same as minimizing the expected extra cost created by every previous action. Then we can re-write the Bellman equation as

$$J_k(u) = \bar{c}_{(u,v)} + f(v) + J_{k-1}(v) \tag{5.4}$$

The function $f(v)$ represents the expected extra cost in future steps as a result of choosing the next node v. It differs from the heuristic function in the A* algorithm [91], since it does not predict the total cost to the goal, and predicts only the additional future cost caused by how the next choice changes the map instead. We propose a heuristic approximation of f, based on the intuition that we want to keep as many clear paths to cover the remaining nodes as possible. We start by identifying a set of nodes that are at the border of the remaining set of nodes to cover. Next, consider a border containing N nodes. For each node in the border, the normal distance $d(i)$ to the next node in the border is computed for the actual map configuration, and the one resulting from choosing the next node. The function is then defined as:

$$f(v) = k_f \cdot \sum_{i=1}^{N} 1 - \frac{d(i)_{t+1}}{d(i)_t} \tag{5.5}$$

This approximation penalizes actions that create narrow spaces, keeping the remaining map with a higher number of possible paths to cover the remaining nodes. The constant k_f was obtained by testing different values in a set of maps and choosing the one that resulted in the minimal averaged total cost across them. Similar to the A* algorithm, a good approximation for the function f considerably reduces the search space for the Dijkstra's algorithm and yields a near-optimal solution for the non-planar maps used

as a demonstration for training the agent.

5.1.2 State observation

Perceiving the environment and making inferences about its spatial structure is a fundamental step in building an online coverage algorithm. In a learning framework as the one proposed, having a high-dimensional input state will result in the need for a higher number of examples to train a model that maps such states into actions. Moreover, the model (here a CNN) will increase in size and will be more prone to overfit the examples and lose generalization power. Hence, as in many learning problems, it is desired to create a minimal and sufficient representation of the input state. This is particularly important for the policy improvement step since the reinforcement learning algorithm may not converge due to having a high-dimensional input [92].

The observed state also depends on assumptions on the agent's ability to perceive the environment and is not easily defined. We start from the assumption that the agent needs information regarding the map's coverage and obstacles, and the elevation of surrounding nodes. In this work, we assume the agent (e.g, a harvester or a variable-rate applicator) can accurately scan its surrounding area with a limited range sensor (e.g., a laser scanner or ultrasonic sensors) and estimate the remaining area to cover with accuracy inversely proportional to distance (e.g., a 360 degrees camera). Then, by following such assumptions and trying to reduce the state's dimension, we propose the following state representation. The short-range scanner yields two patches cropped from the original map around the agent, one containing the information from coverage and obstacles, while the other containing the elevation of each node in the patch. The first experiments have demonstrated that 7×7 patches are sufficient for good convergence of the algorithm. The

estimation of the remaining nodes to cover in the map is done through a novel dimension reduction technique called ring-pooling, which is explained in the next paragraph.

The ring-pooling is a condensed representation of the remaining area to cover in the grid, and its resolution is inversely proportional to the distance from the agent. This is a way to reduce the state dimension where the information is less important. The ring-pooling is defined by the initial seed size and the order of compaction. The seed size s determines the size of the seed patch containing $s \times s$ nodes centered at the agent's position. Subsequential rings of nodes are constructed around the seed patch, such that each ring's width equals the previous ring's width multiplied by the order c. So, the width l of the i^{th} ring is given by (5.6):

$$l_i = sc^{i-1}, \quad i \in \{1, 2, 3, ...\} \tag{5.6}$$

With the rings defined, the percent coverage of each $l_i \times l_i$ patch in the i^{th} ring is computed and stored into a square matrix with an odd number of rows and columns. The center element in the matrix stores the percent area coverage for the seed patch. The elements around it store the percent coverage of patches in the subsequential ring, and so on. Notice that the i^{th} ring greater than one is stored in $8(i-1)$ elements in the matrix. So the stride d_i between the $l_i \times l_i$ patches in the i^{th} ring is defined by (5.7):

$$d_i = \frac{l_i + l_{i-1}}{2(i-1)}, \quad i \in \{2, 3, 4, ...\} \tag{5.7}$$

Figure 5.2 shows an example of state observation using the ring-pooling

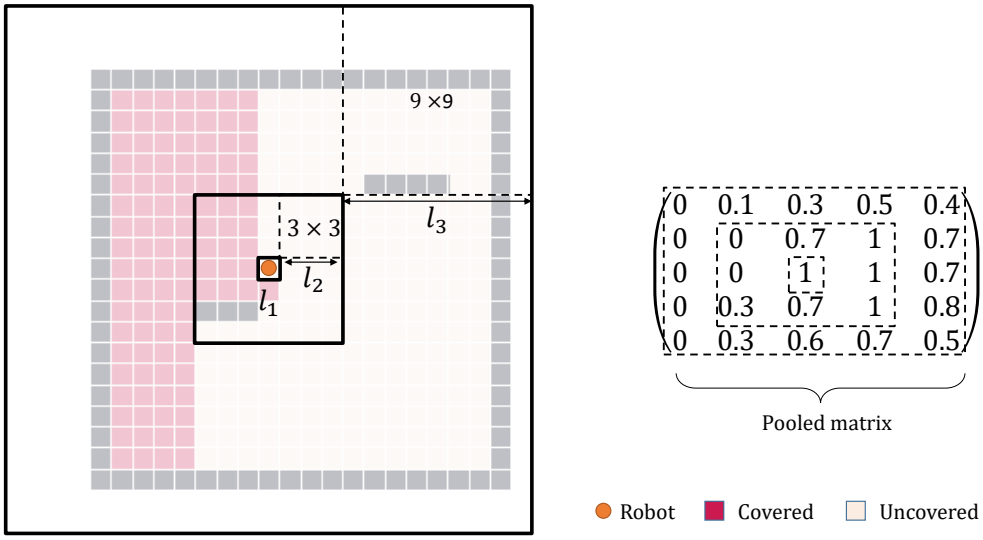


Figure 5.2: Example of the ring pooling with seed size $s = 1$ and order $c = 3$.

technique with seed $s = 1$ and order $c = 3$. The size of the resulting pooled matrix depends on the number of rings to be used, which is ideally chosen to cover the entire map after the seed and order are defined. Once chosen, the size of the pooled matrix must not change since it has to match the input of the CNN.

The state observation is a set containing a 7×7 patch cropped from the grid map centered at the agent containing the elevation of each node, a patch with the same size containing detailed information from the coverage and obstacles around the agent, and a square pooled matrix with $2n-1$ rows and columns, where n is the number of rings. Finally, for each of the maps the training set is constructed to contain pairs state-action for each step in the path defined by the offline algorithm.

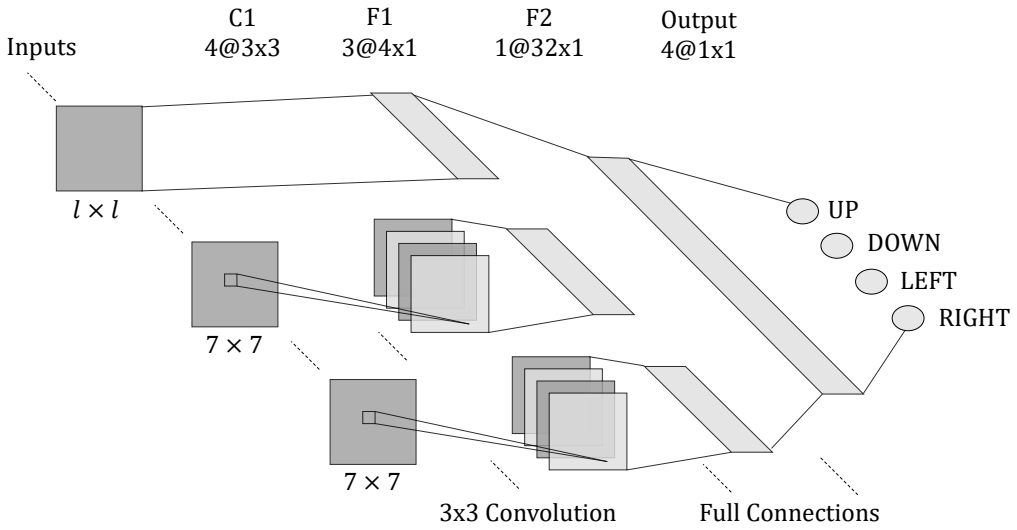


Figure 5.3: CNN architecture.

5.2 Imitation learning

Imitation learning is the first learning step in our framework. More precisely, we use behavioral cloning [93] to learn a policy in a supervised fashion over state-action pairs from demonstrations obtained offline. As mentioned before, the coverage control problem is highly dependent on the spatial structure of the environment. So, we propose a Convolutional Neural Network (CNN) architecture to be able to learn the most relevant features in the observed state to map it to a probability distribution over the possible actions. The policy is then defined by taking the action with the highest probability of being the optimal one.

5.2.1 CNN Architecture

As detailed in the previous section, our observation of the state contains three different types of information. The pooled matrix contains low-resolution information of the remaining nodes in the map; a cropped patch contains

high-resolution information of coverage and obstacles around the agent, and the remaining patch contains information regarding the elevation of each surrounding node. Next, we define a multi-stream architecture (Figure 5.3), in which each independent input is processed by at least one hidden layer before being combined. A grid search was performed to define hyperparameters such as the number of filters in the convolutional layers, the number of layers and neurons in the fully connected layers, and the dropout regularization probability.

The pooled matrix contains a decreasing resolution representation of the remaining nodes, so we assume that the only relevant spatial structure in this information is the position of each element in the matrix. Then, no convolutional layer is applied to this input, and it is fully connected to a layer with four neurons. Each input patch, on the other hand, is connected to a convolutional layer containing four 3×3 filters with stride equal to three. The output of the convolutional layer is then fed to a fully connected layer with four neurons. The intuition behind adding a layer with four neurons in each stream is having a sufficient number of output neurons coming from each stream to represent all desired actions. The layers are then concatenated and fully connected to a new layer with 32 neurons. All neurons up to this point in the network have ReLu as the activation function.

The output layer is designed to have four neurons, being one for each possible action of the agent. Each neuron gives a probability of its respective action being the optimal choice given the input state. A softmax function [94] is used to create a probability distribution over the output neurons as:

$$\sigma_i(z) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}, \ 1 < i < n \tag{5.8}$$

It takes the outputs z_i of the linear activation from each output neuron and assigns a probability to each action such that their sum equals one. In our case, n equals 4.

5.2.2 Training

The state-action pairs from offline paths created for each map were mixed into a single training dataset. We divided the training set into 80% of the examples for training and 20% for validation. The network was trained using Adam optimizer [79], and the validation set was used online as an early stop criterion (allowing at most 15 consecutive iterations of increase in its RMSE value) to avoid overfitting the examples provided. The weights values were restored to the ones with the best performance on the validation data. In this way, the network can generalize its predictions for different states than the ones in the provided examples.

5.3 Policy improvement

The policy learned by behavioral cloning is at most as good as the offline method used to generate the demonstration paths. Its performance depends on the number of examples provided and on how well they represent the search space. It is almost inevitable that many state-action pairs are very similar to each other (e.g., the Boustrophedon motion in spaces far from obstacles), while crucial decisions in the path appear fewer times in the examples. This unbalanced set of examples may lead to lower probabilities of the right action for the least represented states. So we propose to improve our previously learned policy using Reinforcement Learning (RL). Notice that starting the RL algorithm with a pre-trained agent considerably reduces its

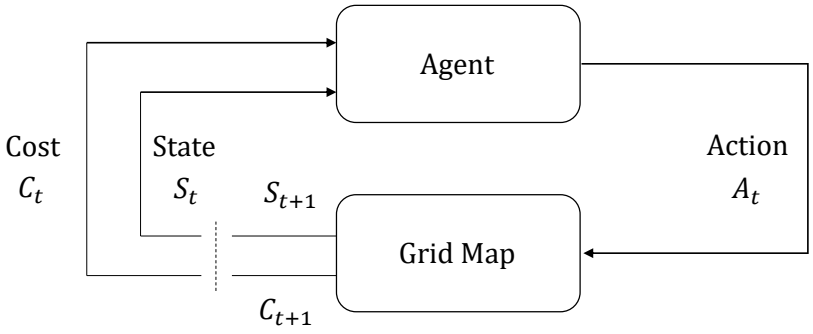


Figure 5.4: Interaction between agent and environment during policy improvement.

exploration space, which increases the chance and speed of convergence. Earlier experiments conducted by the authors didn't achieve convergence of the RL algorithm when an end-to-end approach was used starting from a random policy.

In this approach, the previously trained agent interacts with the same grid maps, generating new paths by itself with no need to use the offline algorithm. For each action performed by the agent, a cost and new state are returned from the grid map. This iterative process is depicted in Figure 5.4.

Our objective is to minimize the expected total cost of a path τ when following a policy π. In our case, the policy π is defined by the weights θ in the CNN, so we define the expected total cost as:

$$J(\theta) = \mathbb{E}_{\pi}[c(\tau)]. \tag{5.9}$$

More formally, we want to find the optimal set of weights θ^* such that $J(\theta)$ is minimal. Following the same approach used when first training the CNN, we use a gradient descent algorithm to iteratively update the weights, starting from θ_0, which is the set of parameters resulting from the behavioral

cloning phase. The iterative process evolves as follows:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t). \tag{5.10}$$

The problem now becomes to find the gradient of the expected total cost of a path τ that covers the entire map. By (5.9) and the policy gradient theorem [95] we have that:

$$\begin{aligned} \nabla J(\theta_t) &= \mathbb{E}_\pi[c(\tau) \nabla \log \pi_\theta(\tau)] \\ &= \mathbb{E}_\pi \left[c(\tau) \left(\sum_{t=1}^{T} \nabla \log \pi_\theta(a_t|s_t) \right) \right] \end{aligned} \tag{5.11}$$

To approximate the expectation in (5.11), we use a Monte-Carlo approach, in which a large number of simulations (each one is an episode) is conducted to collect samples of the term inside the expectation. For each episode, we sample five available grid maps, run the agent with the current policy, and store the costs along the paths. During the simulations, an action is chosen based on the probability distribution given by the current policy (i.e., actions assigned with higher probability more likely to be chosen).

The term $c(\tau)$ is somehow problematic if we think of it as a source of variance in our expectation. By definition, this term represents the total cost accumulated along the path. However, not every decision contributed equally to the final cost, especially if we consider the nature of our problem where a wrong action may result in an immediate or delayed cost higher than average. So, to minimize this effect, we replace the expectation of the total cost along the path by the sum of discounted costs, as proposed in the classic REINFORCE algorithm [96]. Then, we get::

$$\nabla J(\theta_t) = \mathbb{E}_\pi \left[\sum_{t=1}^{T} G_t \nabla \log \pi_\theta(a_t|s_t) \right], \tag{5.12}$$

where

$$G_t = \sum_{k=t+1}^{T} \gamma^{(k-t-1)} C_k,$$

and γ is the discount factor, a parameter to weigh the effect of past action on the actual cost. The discounted costs are then normalized to zero mean and unitary variance to remove the bias in the gradient. Thus, the final pseudo-code of our policy update becomes:

Algorithm 4 Policy gradient update

1: $\pi_\theta \leftarrow \pi_{\theta_0}$
2: $S \leftarrow$ grid maps
3: **for** episode $= 1,\ M$ **do**
4: map $\leftarrow$ random_sample(S)
5: **while** coverage $<$ desired coverage, and $t < T_{\text{limit}}$ **do**
6: choose an action based on $\pi_\theta(a_t|s_t)$
7: save (a_t, s_t) and the returned cost(C_t)
8: **end while**
9: calculate discounted costs $G_t(C(\tau))$
10: calculate the gradient of log probabilities
11: $\theta = \theta - \alpha \nabla J(\theta)$
12: **end for**

In practice, calculating the gradient of the log probabilities (line 10 of our pseudo-code) is done by the backpropagation algorithm used to train the CNN. Notice that the cross-entropy loss function used in multi-class classification problems is given by:

$$L = -\frac{1}{N}\left(\sum_{y=1}^{N} y_i \cdot \nabla \log(\hat{y}_i)\right), \tag{5.13}$$

where y_i is the desired distribution, and $\hat{y}_i$ is the one predicted by the network. So, lines 9 to 11 in the pseudo-code are implemented by training the neural network for one epoch using a batch of state-action pairs sampled from the trajectories using a cross-entropy loss function. To account for G_t in (5.12), we weigh the gradient of each sample by its respective discounted cost calculated on line 9.

It is important to observe that, at each episode, different maps are randomly sampled from the dataset. This process increases the generalization power of the refined policy on unseen maps. However, if we always sample the same map, we expect the resulting policy to converge to the optimal solution for this map. The advantages of this approach will be explored in future work.

5.4 Simulations and results

Experiments were conducted in a simulated environment. For the first step in our framework, we generated 40 planar and 40 non-planar maps according to the methodology described in Section 5.1. A path that minimizes the cost function in equation (5.1) was defined for each map by using the offline techniques also described in Section 5.1. This setup resulted in a dataset with approximately 32K examples containing state-action pairs. The constants chosen for the cost function are $k_z = 1$, $k_t = 1$, and $k_o = 5$. They were chosen such that overlapping a previously covered node has five times the

cost of turning, simulating, for instance, the task of spraying fertilizer in a field.

The CNN agent was trained using the resulting dataset as described in Section 5.2.2, and its policy improved using the method in Section 5.3. The policy improvement algorithm converged in around 450 episodes, as shown in Figure 5.6, which corresponds to one hour when running on a Core i7 CPU with 16Gb of RAM. This time can be significantly reduced when running this algorithm using a Graphical Processing Unit (GPU). The convergence of the policy improvement algorithm is considered achieved, when no significant reduction in averaged cost is obtained in the last 100 episodes. The sudden drop in the cost happens when the policy learns to identify the preferable movement direction according to the obstacles in a way that the number of overlapped nodes is reduced. An example of such improvement can be observed in Figure 5.5, where a path sampled during the episode number 50 shows a higher cost due to overlaps when compared with the path for the same map at episode 200.

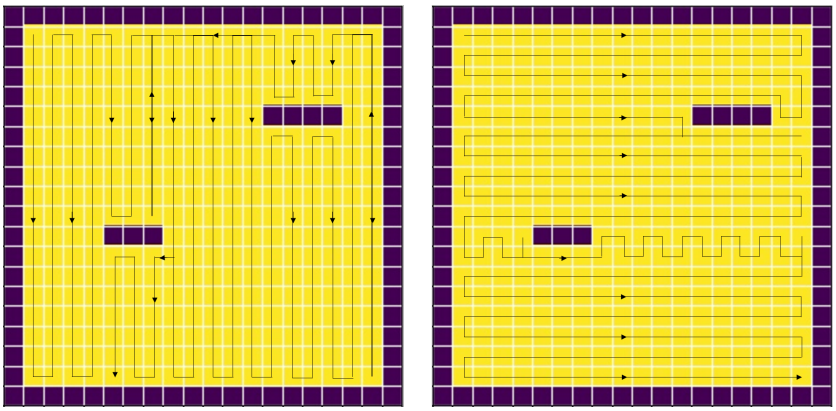


Figure 5.5: Sample of a path from episode 50 (on the left) and episode 200 (on the right).

The offline algorithm, the behavioral cloning agent, and the agent with the improved policy were tested on 12 additional maps, not in the training set:

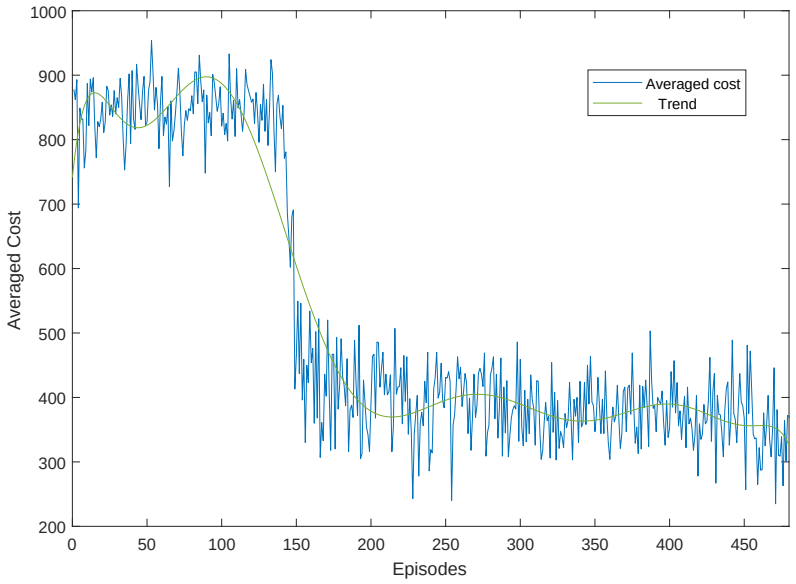
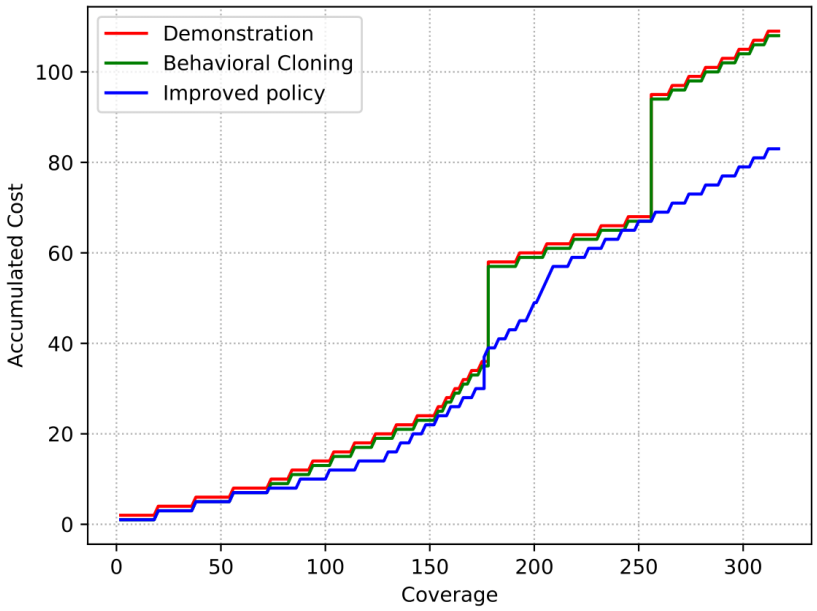
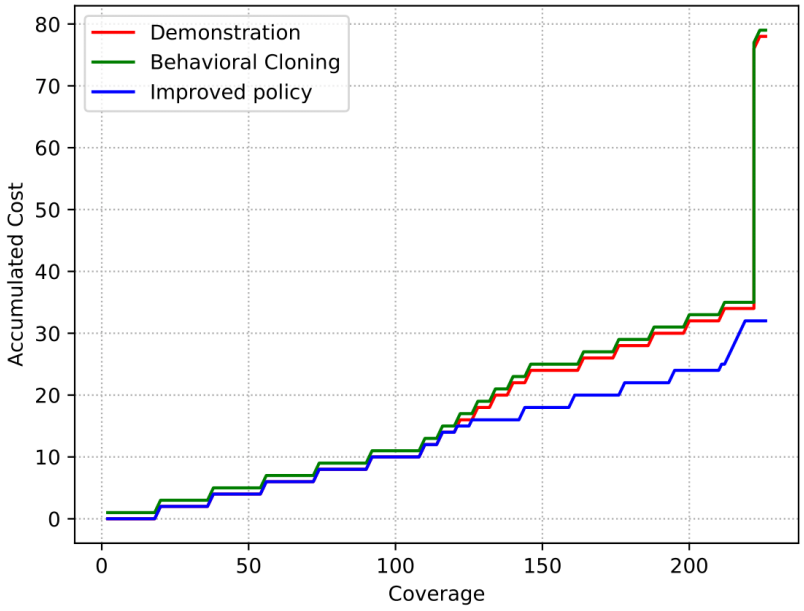


Figure 5.6: Averaged cost from paths simulated in each episode during policy improvement.

six planar and six non-planar. The coverage path of each agent was obtained by a greedy strategy that selects the action with the highest probability of being optimal, given a state. The agents' performance on two planar maps (A and B) are shown in Figure 5.7, and on two non-planar maps (C and D) are shown in Figure 5.8. The plots contain the accumulated cost as a function of the number of covered nodes for the obtained path. We evaluate the performance of our algorithm based on the cost of the path and the final coverage, as shown in Tables 5.1 and 5.2.

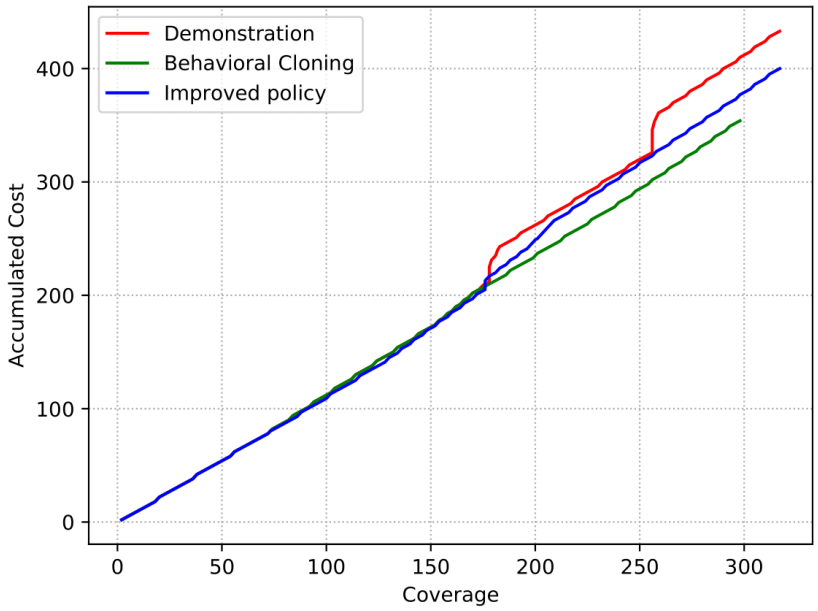


(A)

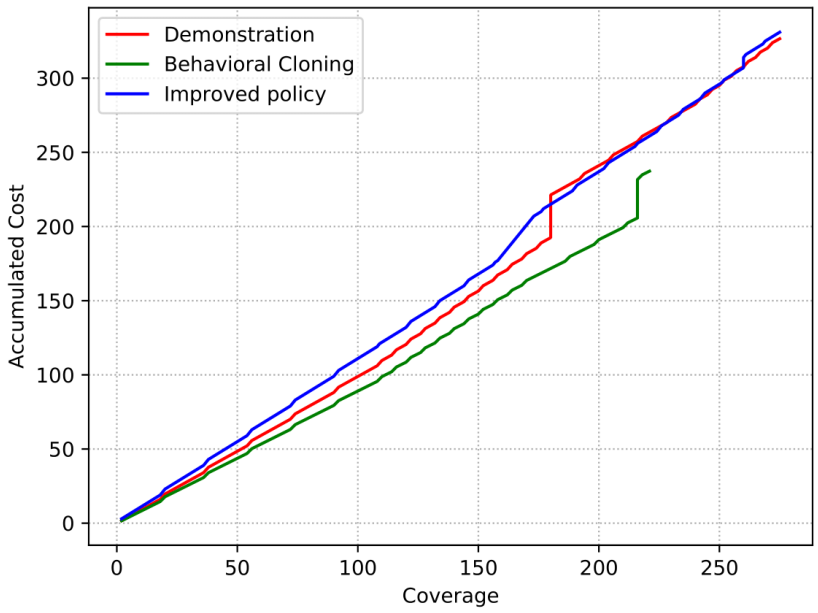


(B)

Figure 5.7: Performance of the offline algorithm, the behavioral cloning agent, and the agent with improved policy on planar maps.



(A)



(B)

Figure 5.8: Performance of the offline algorithm, the behavioral cloning agent, and the agent with improved policy on non-planar maps.

Map	Behavioral Cloning		Improved Policy	
	Cost	Cvg.	Cost	Cvg.
1	0.99	1	0.78	1
2	1.01	1	0.41	1
3	1.00	1	0.53	1
4	1.01	1	0.76	1
5	1.00	1	0.70	1
6	1.02	1	0.75	1

Table 5.1: Percent cost and coverage of behavioral cloning and improved policy when compared with the offline algorithm on planar maps.

Map	Behavioral Cloning		Improved Policy	
	Cost	Cvg.	Cost	Cvg.
1	0.77	0.86	0.87	1
2	0.73	0.79	1.04	1
3	0.88	0.92	0.78	1
4	0.69	0.72	1.20	1
5	1.14	1	0.94	1
6	0.71	0.86	1.10	1

Table 5.2: Percent cost and coverage of behavioral cloning and improved policy when compared with the offline algorithm on non-planar maps.

In all tested planar maps, the behavioral cloning strategy matched the offline algorithm's performance and achieved full coverage of the map. One can observe that the small steps in the cost are caused by the turns generated by the boustrophedon motion, while the big steps in the cost are caused by overlapping nodes to reach the next decomposed cell. The agent with

improved policy through reinforcement learning reduced, on average, by 35% the cost to cover the maps, reducing by almost 60% the cost of map 2. This result is explained in part by the fact the cellular-decomposition algorithm is not a good heuristic for high overlapping costs. Nevertheless, the experiment demonstrates that the policy improvement can generate a significantly better policy by reducing the number of overlaps.

Results from the non-planar maps show that the behavioral cloning strategy failed to achieve full coverage in five maps. In these cases, the agent started a loop of actions without covering any other node on the map. However, after refining the agent's policy, it was able to complete the map with a lower cost than the heuristic Dijkstra's algorithm in three fields, and with a slightly higher cost in the remaining ones. The Dijkstra's algorithm generates solutions closer to the optimal, which are not based on an evident behavior as the cellular decomposition algorithm. Therefore, it is harder to mimic this solution with the same number of demonstrations. Similar to the planar maps, there are big steps in the accumulated cost for the offline solution and for the behavioral cloning agent that result from overlapping nodes to reach non-covered areas. This result shows that the improved policy can avoid creating areas only reachable by overlapping previously covered nodes and hence reducing the accumulated cost.

5.5 Discussion

A novel framework for area coverage control was proposed. This framework is based on designing an agent to learn an online policy using demonstrations from offline algorithms and further refinement using a reinforcement learning algorithm. After training the agent in a supervised fashion with demonstra-

tions from 80 maps, the performance obtained was comparable to offline algorithms when testing in six new planar maps. The policy improvement strategy resulted in an agent with performance higher than the one obtained from behavioral cloning and eventually outperforming the offline algorithm even on non-planar maps. The main advantage of this framework lies in its flexibility in learning different and possibly multiple heuristic strategies from offline algorithms, improve their performances, and use them online. Also, the cost function can be arbitrarily defined, not depending on the heuristic that is usually focused on reducing specific costs.

The local and reduced representation of the observed state makes the algorithm scalable. The size of the map is expected to influence only the time required to obtain the offline solution and the duration of each episode in reinforcement learning.

CHAPTER 6

CONCLUSIONS

This dissertation presented a class of learning-based algorithms and methods for improving crop management. It started describing a novel spatial dependency model for yield prediction based on pre-season treatments and environmental variables. This model leverages a multi-stream architecture of CNN in order to model nonlinear dependencies among input variables, while accounting for variable-wise spatial feature extraction. We provided experimental evidence to show the superior performance of the LF realization of the MSCNN, achieving a reduction of up to 17% on the RMSE value when compared to a conventional 2D CNN with stacked input channels, and up to 29% when compared to an RF, which was shown to be the model with the best performance not using a CNN. An accurate yield prediction model is the key element for making decisions on fertilization rates.

Besides better performance, the MSCNN-LF model introduces a more challenging optimization problem. We have detailed a gradient ascent algorithm that maximizes the farmer's net revenue by optimizing the input fertilization maps. The proposed objective function focuses on farmer's net revenue to increase the chances of adoption of this method. However, the gradient ascent framework is flexible to be used for different objectives. For instance, since we have derived the gradient of the yield map with respect to the input variables, the objective function can be easily adapted to maximize yield. Experiments show that fertilization rates can be drastically reduced with no

or small decrease in productivity, which subdue environmental impacts and financial losses at almost no social costs.

To leverage the applicability of the proposed methods, the models and algorithms were re-designed to incorporate uncertainty quantification. The RA-CNN-LF model is trained following the deep ensemble framework and outputs a probability distribution rather than a single value. The variance of such distribution was used to construct an uncertainty map to provide farmers with the information of areas of their fields that have high variability associated to variables not in the dataset. The new architecture not only provides uncertainty estimation but also outperforms its former version in the experiments.

The proposed risk-averse optimization algorithm uses the RA-CNN-LF model to maximize the expected net revenue while reducing the risk associated with the uncertainty of explored solutions. We proposed two methods for avoiding risk. In the first, the output variance of the RA-CNN-LF is factored in the objective function, and the optimization algorithm minimizes it over the iterations. In the second approach, we created a constraint on the value at risk, so the output variance is not factored in the objective function until it gets close to the constraint. In this way, the net revenue is maximized regardless of the output variance as long as it complies with the level of risk-aversion of each farmer. As the gradient ascent method may find solutions that are too different from the training data domain, the risk-aversion framework is important to avoid them, since they are unlike the work in practice.

Finally, this dissertation proposed an online coverage control algorithm for applying the resulting fertilization maps in the field. The algorithm uses an MSCNN agent to learn a policy from offline algorithms, and solve the

coverage problem online. The resulting policy is further improved through RL, and experiments demonstrated its potential to outperform the offline algorithms. Although we only presented simulated results, the grid representation is easily transferable from simulation environments to real-world problems. For instance, for creating CPP for agricultural fields, we can use the elevation map of different fields and randomly place obstacles of the size of tractors, harvesters, water ponds, and other obstacles typically found in a field.

6.1 Future Work

The research and developments presented in this dissertation hold promise for improving crop management in an accessible and scalable way. Nevertheless, there is room for extending this work to many topics that require further research.

6.1.1 Generalization of the Yield Prediction Model

The framework proposed for yield modeling uses site-specific data generated through OFPE. However, not all farmers have access to this kind of data, especially the ones that own small fields. Future work will explore the use of transfer learning [97] to create models for fields where data is not available or is limited. It is expected that filters trained with data from a field can learn features also relevant to different ones. In this sense, additional research should explore training the MSCNN model with data from different fields and re-train only the fully connected layers with small-plot experiments [98] data from the target fields.

6.1.2 Use of Time Series Data for Yield Prediction

The residual nitrogen fertilizer in the soil from previous years is known to affect the yield response to crop management [99]. Also, there is a great interest in reducing the residual nitrogen in the soil to avoid water contamination [100]. Having data from multiple years allows us to use the time-series data from nitrogen application to better estimate yield. The time series may be incorporated into the model by stacking nitrogen maps from past years as a single multi-channel input in the MSCNN architecture. Moreover, the optimization algorithm proposed in this dissertation will factor this information, and it is expected that the level of residual nitrogen in the soil decreases over the seasons.

6.1.3 Extension of the CPP Framework

Future work encompasses testing how the framework scales to bigger maps, and how to obtain optimized offline solutions by always sampling the same map during reinforcement learning. The algorithm still lacks some practical features such as heuristics to avoid unexpected or wrong behaviors. Then, the transition from the simulated environment to real crop fields should consider potential failure scenarios where the online policy should be ignored and improved. Besides, methods for transitioning from the grid environment to continuous paths must be investigated.

6.1.4 Risk-Aware Online CPP

Although the proposed CPP converged to good control policy, the MSCNN agent only outputs the expected probability of each action being optimal at each state. Experiments demonstrate that during the imitation learning

phase, the agent selected actions that resulted in an infinity loop not fully covering the map. As the environment is only partially observable, the same observed state may have different optimal actions, which may lead to these unstable behaviors. Future work will incorporate uncertainty estimation into the agent's architecture using a deep ensemble. New policies can be explored such that this ambiguity is modeled, and the risk is avoided at a certain level [101, 102].

APPENDIX A

DERIVATIONS

A.1 Gradient of the Objective Function in Chapter 4

Let the mean and variance of the predicted net revenue be defined by:

$$\mu_r(N,S) = p_V\ \mathbb{E}(f(N,S)) - p_N \sum^{i} N_i - p_S \sum^{i} S_i$$
$$\sigma_r(N,S) = p_V\ \sqrt{\text{var}(f(N,S))}$$

We want to find the gradient of the expression given by Y w.r.t the nitrogen and seed maps:

$$Y = \mu_r(N,S) - \beta\sigma_r(N,S) - \frac{1}{t}\log(c)$$
$$c = \mu_r(N,S) - 2\sigma_r(N,S) - \rho$$

So, we have that the partials of $\nabla Y = \left(\frac{\partial Y}{\partial N}\ ,\ \frac{\partial Y}{\partial S}\right)$ are given by:

$$\frac{\partial Y}{\partial N} = \frac{\partial \mu_r}{\partial N} - \beta\frac{\partial \sigma_r}{\partial N} - \frac{1}{tc}\left(\frac{\partial \mu_r}{\partial N} - 2\frac{\partial \sigma_r}{\partial N}\right) = \frac{\partial \mu_r}{\partial N}\left(1 + \frac{1}{tc}\right) - \frac{\partial \sigma_r}{\partial N}\left(\beta + \frac{1}{tc}\right)$$
$$\frac{\partial Y}{\partial S} = \frac{\partial \mu_r}{\partial S} - \beta\frac{\partial \sigma_r}{\partial S} - \frac{1}{tc}\left(\frac{\partial \mu_r}{\partial S} - 2\frac{\partial \sigma_r}{\partial S}\right) = \frac{\partial \mu_r}{\partial S}\left(1 + \frac{1}{tc}\right) - \frac{\partial \sigma_r}{\partial S}\left(\beta + \frac{1}{tc}\right)$$

where

$$\frac{\partial \mu_r}{\partial N} = p_Y \frac{\partial \mathbb{E}(f(N,S))}{\partial N} - P_N$$
$$\frac{\partial \sigma_r}{\partial N} = \frac{p_Y}{2\sqrt{\text{var}(f(N,S))}} \frac{\partial \text{ var}(f(N,S))}{\partial N}$$

and

$$\frac{\partial \mu_r}{\partial S} = p_Y \frac{\partial \mathbb{E}(f(N,S))}{\partial S} - P_S$$
$$\frac{\partial \sigma_r}{\partial S} = \frac{p_Y}{2\sqrt{\text{var}(f(N,S))}} \frac{\partial \text{ var}(f(N,S))}{\partial S}$$

The partials of the yield map's mean and variance w.r.t the nitrogen and seed maps are obtained from the mixture model of the ensemble. Remember that the mean ($\mu_\star$) and var ($\sigma^2_\star$) of the mixture model are :

$$\mu_\star(X) = \frac{\sum^m \mu_{\theta_m}}{M}$$
$$\sigma^2_\star(X) = \frac{\sum^m (\sigma^2_{\theta_m}(X) + \mu^2_{\theta_m}(X)) - \mu^2_\star(X)}{M}$$

So, the partials of the mixture w.r.t to a cropped patch are:

$$\frac{\partial \mu_*}{\partial \bar{N}} = \frac{1}{M} \sum_{m=0}^{M} \frac{\partial \mu_{\theta_m}}{\partial \bar{N}} \quad ; \quad \frac{\partial \mu_*}{\partial \bar{S}} = \frac{1}{M} \sum_{m=0}^{M} \frac{\partial \mu_{\theta_m}}{\partial \bar{S}}$$
$$\frac{\sigma^2_*}{\partial \bar{N}} = \frac{1}{M} \sum_{m=0}^{M} \left(\frac{\partial \sigma^2_{\theta_m}}{\partial \bar{N}} + 2\mu_{\theta_m} \frac{\partial \mu_{\theta_m}}{\partial \bar{N}} - 2\mu_* \frac{\partial \mu_*}{\partial \bar{N}} \right)$$
$$\frac{\sigma^2_*}{\partial \bar{S}} = \frac{1}{M} \sum_{m=0}^{M} \left(\frac{\partial \sigma^2_{\theta_m}}{\partial \bar{S}} + 2\mu_{\theta_m} \frac{\partial \mu_{\theta_m}}{\partial \bar{S}} - 2\mu_* \frac{\partial \mu_*}{\partial \bar{S}} \right)$$

The partials $\frac{\partial \mu_{\theta_m}}{\partial \bar{N}}$, $\frac{\partial \mu_{\theta_m}}{\partial \bar{S}}$, $\frac{\partial \sigma^2_{\theta_m}}{\partial \bar{N}}$, and $\frac{\partial \sigma^2_{\theta_m}}{\partial \bar{S}}$ are obtained using the backpropagation algorithm (function tf.gradients in Tensorflow).

Finally, to combine the partials of the cropped patches into the full map we need to define a zero padding function $z(c, i) : \mathbb{R}^{l \times l} \times \mathbb{N} \longrightarrow \mathbb{R}^{m \times n}$ that centers the 21×21 patch at element i in a $m \times n$ map, and completes the remaining elements with zero. Then, we have that:

$$\frac{\partial \mathbb{E}(f(N,S))}{\partial N} = \sum^{i} z\left(\frac{\partial \mu_*}{\partial \bar{N}}, i\right) \quad ; \quad \frac{\partial \mathbb{E}(f(N,S))}{\partial S} = \sum^{i} z\left(\frac{\partial \mu_*}{\partial \bar{S}}, i\right)$$
$$\frac{\partial \text{ var}(f(N,S))}{\partial N} = \sum^{i} z\left(\frac{\sigma^2_*}{\partial \bar{N}}, i\right) \quad ; \quad \frac{\partial \text{ var}(f(N,S))}{\partial S} = \sum^{i} z\left(\frac{\sigma^2_*}{\partial \bar{S}}, i\right)$$

REFERENCES

[1] H. C. J. Godfray, J. R. Beddington, I. R. Crute, L. Haddad, D. Lawrence, J. F. Muir, J. Pretty, S. Robinson, S. M. Thomas, and C. Toulmin, "Food security: The challenge of feeding 9 billion people," Science, vol. 327, no. 5967, pp. 812–818, 2010.

[2] S. R. Carpenter, N. F. Caraco, D. L. Correll, R. W. Howarth, A. N. Sharpley, and V. H. Smith, "Nonpoint pollution of surface waters with phosphorus and nitrogen," Ecological Applications, vol. 8, no. 3, pp. 559–568, 1998.

[3] W. Whalley, E. Dumitru, and A. Dexter, "Biological effects of soil compaction," Soil and Tillage research, vol. 35, no. 1-2, pp. 53–68, 1995.

[4] H. Auernhammer, "Precision farming—the environmental challenge," Computers and electronics in agriculture, vol. 30, no. 1-3, pp. 31–43, 2001.

[5] R. Gebbers and V. I. Adamchuk, "Precision agriculture and food security," Science, vol. 327, no. 5967, pp. 828–831, 2010.

[6] J. M. Antle, J. W. Jones, and C. E. Rosenzweig, "Next generation agricultural system data, models and knowledge products: Introduction," Agricultural Systems, vol. 155, pp. 186 – 190, 2017.

[7] L. Abbott and D. Murphy, What is Soil Biological Fertility?, 2007, pp. 1–15.

[8] D. M. Lambert, K. P. Paudel, and J. A. Larson, "Bundled adoption of precision agriculture technologies by cotton producers," Journal of agricultural and resource economics, pp. 325–345, 2015.

[9] R. Bramley, "Lessons from nearly 20 years of precision agriculture research, development, and adoption as a guide to its appropriate application," Crop and Pasture Science, vol. 60, no. 3, pp. 197–217, 2009.

[10] H. S. Pathak, P. Brown, and T. Best, "A systematic literature review of the factors affecting the precision agriculture adoption process," Precision Agriculture, vol. 20, no. 6, pp. 1292–1316, 2019.

[11] J. W. Jones, J. M. Antle, B. Basso, K. J. Boote, R. T. Conant, I. Foster, H. C. J. Godfray, M. Herrero, R. E. Howitt, S. Janssen, B. A. Keating, R. Munoz-Carpena, C. H. Porter, C. Rosenzweig, and T. R. Wheeler, "Brief history of agricultural systems modeling," Agricultural Systems, vol. 155, pp. 240 – 254, 2017.

[12] S. Osborne, J. Schepers, D. Francis, and M. R. Schlemmer, "Use of spectral radiance to estimate in-season biomass and grain yield in nitrogen- and water-stressed corn," Crop science, vol. 42, pp. 165–171, 2002.

[13] M. Kaul, R. L. Hill, and C. Walthall, "Artificial neural network for corn and soybean yield prediction," Agricultural Systems, vol. 85, pp. 1–18, 2005.

[14] P. Aggarwal, "Uncertainties in crop, soil and weather inputs used in growth models: Implications for simulated outputs and their applications," Agricultural Systems, vol. 48, pp. 361–384, 1995.

[15] A. Meyer-Aurich, N. Muhammad, and R. Herbst, "On-farm experimentation for identification of site specific fertilization potentials," International Conference on Precision Agriculture, 2008.

[16] D. G. Rodriguez, D. S. Bullock, and M. Boerngen, "The origins, implications, and consequences of yield-based nitrogen fertilizer management," Agronomy Journal, vol. 111, 01 2019.

[17] Y. Yang, J. Zhu, X. Tong, and D. Wang, "The spatial pattern characteristics of soil nutrients at the field scale," in Computer and Computing Technologies in Agriculture II, Volume 1, D. Li and C. Zhao, Eds. Boston, MA: Springer US, 2009, pp. 125–134.

[18] M. Hamza and W. Anderson, "Soil compaction in cropping systems: A review of the nature, causes and possible solutions," Soil and tillage research, vol. 82, no. 2, pp. 121–145, 2005.

[19] H. Choset, "Coverage for robotics–a survey of recent results," Annals of mathematics and artificial intelligence, vol. 31, no. 1-4, pp. 113–126, 2001.

[20] Y.-H. Choi, T.-K. Lee, S.-H. Baek, and S.-Y. Oh, "Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform," in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009, pp. 5788–5793.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Neural Information Processing Systems, vol. 25, 2012.

[22] A. Chlingaryan, S. Sukkarieh, and B. Whelan, "Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review," Computers and electronics in agriculture, vol. 151, pp. 61–69, 2018.

[23] M. Bachmaier and M. Gandorfer, "A conceptual framework for judging the precision agriculture hypothesis with regard to site-specific nitrogen application," Precision agriculture, vol. 10, no. 2, p. 95, 2009.

[24] A. K. Prasad, L. Chai, R. P. Singh, and M. Kafatos, "Crop yield estimation model for iowa using remote sensing and surface parameters," International Journal of Applied Earth Observation and Geoinformation, vol. 8, no. 1, pp. 26–33, 2006.

[25] J. Peters and S. Schaal, "Policy gradient methods for robotics," in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006, pp. 2219–2225.

[26] J. M. Bernardo and A. Smith, "Bayesian theory, vol. 405," Hoboken: Wiley, 2009.

[27] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in Advances in neural information processing systems, 2017, pp. 6402–6413.

[28] B. H. Baltagi, A companion to theoretical econometrics. Blackwell, 2003.

[29] R. Plant, Spatial Data Analysis in Ecology and Agriculture Using R. Boca Raton: CRC Press, 2012.

[30] L. Anselin, R. Bongiovanni, and J. Lowenberg-DeBoer, "A spatial econometric approach to the economics of site-specific nitrogen management in corn production," American Journal of Agricultural Economics, vol. 86, no. 3, pp. 675–687, 2004.

[31] Y.-x. Su, H. Xu, and L.-j. Yan, "Support vector machine-based open crop model (sbocm): Case of rice production in china," Saudi journal of biological sciences, vol. 24, no. 3, pp. 537–547, 2017.

[32] S. Brdar, D. Culibrk, B. Marinkovic, J. Crnobarac, and V. Crnojevic, "Support vector machines with features contribution analysis for agricultural yield prediction," in Second International Workshop on Sensing Technologies in Agriculture, Forestry and Environment (EcoSense 2011), Belgrade, Serbia, 2011, pp. 43–47.

[33] J. H. Jeong, J. P. Resop, N. D. Mueller, D. H. Fleisher, K. Yun, E. E. Butler, D. J. Timlin, K.-M. Shim, J. S. Gerber, V. R. Reddy et al., "Random forests for global and regional crop yield predictions," PLoS One, vol. 11, no. 6, 2016.

[34] J. You, X. Li, M. Low, D. Lobell, and S. Ermon, "Deep gaussian process for crop yield prediction based on remote sensing data," in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[35] D. I. Patricio and R. Rieder, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," Computers and Electronics in Agriculture, vol. 153, pp. 69 – 81, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169918305829

[36] K. Golhani, S. K. Balasundram, G. Vadamalai, and B. Pradhan, "A review of neural networks in plant disease detection using hyperspectral data," Information Processing in Agriculture, vol. 5, no. 3, pp. 354 – 371, 2018.

[37] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, "Identification of rice diseases using deep convolutional neural networks," Neurocomputing, vol. 267, pp. 378–384, 2017.

[38] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," 2015 IEEE International Conference on Image Processing (ICIP), pp. 452–456, 2015.

[39] L. A. da Silva, P. O. Bressan, D. N. Gonçalves, D. M. Freitas, B. B. Machado, and W. N. Gonçalves, "Estimating soybean leaf defoliation using convolutional neural networks and synthetic images," Computers and Electronics in Agriculture, vol. 156, pp. 360 – 368, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169918307907

[40] P. Nevavuori, N. Narra, and T. Lipping, "Crop yield prediction with deep convolutional neural networks," Computers and Electronics in Agriculture, vol. 163, p. 104859, 2019.

[41] Y. Sun, L. Zhu, G. Wang, and F. Zhao, "Multi-input convolutional neural network for flower grading," Journal of Electrical and Computer Engineering, vol. 2017, pp. 1–8, 2017.

[42] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," CoRR, vol. abs/1406.2199, 2014.

[43] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, June 2014, pp. 1725–1732.

[44] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in 2015 IEEE International Conference on Computer Vision, 2015, pp. 4489–4497.

[45] B. Basso, J. T. Ritchie, D. Cammarano, and L. Sartori, "A strategic and tactical management approach to select optimal n fertilizer rates for wheat in a spatially variable field," European Journal of Agronomy, vol. 35, no. 4, pp. 215–222, 2011.

[46] B. Basso, D. Cammarano, P. R. Grace, G. Cafiero, L. Sartori, M. Pisante, G. Landi, S. De Franchi, and F. Basso, "Criteria for selecting optimal nitrogen fertilizer rates for precision agriculture," Italian Journal of Agronomy, pp. 147–158, 2009.

[47] D. Pokrajac and Z. Obradovic, "Neural network-based software for fertilizer optimization in precision farming," in IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), vol. 3. IEEE, 2001, pp. 2110–2115.

[48] Y. Saikai, V. Patel, and P. D. Mitchell, "Machine learning for optimizing complex site-specific management," Computers and Electronics in Agriculture, vol. 174, p. 105381, 2020.

[49] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," Robotics and Autonomous systems, vol. 61, no. 12, pp. 1258–1276, 2013.

[50] W. Ji, J. L. Li, D. A. Zhao, and Y. Jun, "Obstacle avoidance path planning for harvesting robot manipulator based on maklink graph and improved ant colony algorithm," in Advances in Measurements and Information Technologies, ser. Applied Mechanics and Materials, vol. 530. Trans Tech Publications Ltd, 5 2014, pp. 1063–1067.

[51] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," Autonomous Robots, vol. 40, no. 6, pp. 1059–1078, 2016.

[52] H. Chen, T. Fuhlbrigge, and X. Li, "Automated industrial robot path planning for spray painting process: a review," in 2008 IEEE International Conference on Automation Science and Engineering. IEEE, 2008, pp. 522–527.

[53] H. Liu, J. Ma, and W. Huang, "Sensor-based complete coverage path planning in dynamic environment for cleaning robot," CAAI Transactions on Intelligence Technology, vol. 3, no. 1, pp. 65–72, 2018.

[54] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges." J. Inf. Sci. Eng., vol. 33, no. 1, pp. 101–121, 2017.

[55] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," Computational Geometry, vol. 17, no. 1-2, pp. 25–50, 2000.

[56] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in 2010 IEEE International conference on robotics and automation. IEEE, 2010, pp. 5525–5530.

[57] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," Annals of mathematics and artificial intelligence, vol. 31, no. 1-4, pp. 77–98, 2001.

[58] M. Kapanoglu, M. Alikalfa, M. Ozkan, O. Parlaktuna et al., "A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time," Journal of intelligent manufacturing, vol. 23, no. 4, pp. 1035–1045, 2012.

[59] A. Zelinsky, R. A. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in Proceedings of international conference on advanced robotics, vol. 13, 1993, pp. 533–538.

[60] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," in Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), vol. 2. IEEE, 2003, pp. 1685–1690.

[61] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-driven planning via imitation learning," The International Journal of Robotics Research, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[62] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," The International Journal of Robotics Research, vol. 36, no. 10, pp. 1073–1087, 2017.

[63] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," Automation in Construction, vol. 112, p. 103078, 2020.

[64] A. Barbosa, R. Trevisan, N. Hovakimyan, and N. F. Martin, "Modeling yield response to crop management using convolutional neural networks," Computers and Electronics in Agriculture, vol. 170, p. 105197, 2020.

[65] A. Barbosa, T. Marinho, N. Hovakimyan, and N. F. Martin, "Multi-stream cnn for spatial resource allocation: a crop management application," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2020.

[66] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in Field and service robotics. Springer, 1998, pp. 203–209.

[67] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," Robotics and autonomous systems, vol. 57, no. 5, pp. 469–483, 2009.

[68] D. S. Bullock, M. Boerngen, H. Tao, B. Maxwell, J. D. Luck, L. Shiratsuchi, L. Puntel, and N. F. Martin, "The data-intensive farm management project: Changing agronomic research through on-farm precision experimentation," Agronomy Journal, p. in press, 2019.

[69] K. A. Sudduth, D. B. Myers, N. R. Kitchen, and S. T. Drummond, "Modeling soil electrical conductivity–depth relationships with data from proximal and penetrating eca sensors," Geoderma, vol. 199, pp. 12–21, 2013.

[70] N. F. Martín, G. Bollero, and D. G. Bullock, "Associations between field characteristics and soybean plant performance using canonical correlation analysis," Plant and Soil, vol. 273, no. 1-2, pp. 39–55, 2005.

[71] C. Planet Team, San Francisco, "Planet application program interface: In space for life on earth." https://api.planet.com, 2017.

[72] S. Khanal, J. Fulton, A. Klopfenstein, N. Douridas, and S. Shearer, "Integration of high resolution remotely sensed data and machine learning techniques for spatial prediction of soil properties and corn yield," Computers and electronics in agriculture, vol. 153, pp. 213–225, 2018.

[73] S. Dahikar and V. Rode, "Agricultural crop yield prediction using artificial neural network approach," International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 2, pp. 683–686, 2014.

[74] S. Vani, R. Sukumaran, and S. Savithri, "Prediction of sugar yields during hydrolysis of lignocellulosic biomass using artificial neural network modeling," Bioresource technology, vol. 188, 2015.

[75] E. Betiku and A. E. Taiwo, "Modeling and optimization of bioethanol production from breadfruit starch hydrolyzate vis-à-vis response surface methodology and artificial neural network." Renewable Energy, vol. 74, pp. 87–94, 2015.

[76] O. Marko, S. Brdar, M. Panić, I. Šašić, D. Despotović, M. Knežević, and V. Crnojević, "Portfolio optimization for seed selection in diverse weather scenarios," PloS one, vol. 12, no. 9, p. e0184198, 2017.

[77] S. Brdar, D. Culibrk, B. Marinkovic, J. Crnobarac, and V. Crnojevic, "Support vector machines with features contribution analysis for agricultural yield prediction," in Second International Workshop on Sensing Technologies in Agriculture, Forestry and Environment (EcoSense 2011), Belgrade, Serbia, 2011, pp. 43–47.

[78] S. Vucetic, T. Fiez, and Z. Obradovic, "A data partitioning scheme for spatial regression," in IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339), vol. 4. IEEE, 1999, pp. 2474–2479.

[79] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations, 2014.

[80] J. A. Taylor, J.-P. Praat, and A. F. Bollen, "Spatial variability of kiwifruit quality in orchards and its implications for sampling and mapping," HortScience horts, vol. 42, no. 2, pp. 246 – 250, 2007.

[81] C. Cambardella, T. Moorman, J. Novak, T. Parkin, D. Karlen, R. Turco, and A. Konopka, "Field-scale variability of soil properties in central iowa soils," Soil Sci Soc Am J, vol. 58, 01 1994.

[82] N. Qian, "On the momentum term in gradient descent learning algorithms," Neural networks, vol. 12, no. 1, pp. 145–151, 1999.

[83] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," Digital Signal Processing, vol. 73, pp. 1 – 15, 2018.

[84] W. Liu, W.-c. Gao, Y. Sun, and M.-j. Xu, "Optimal sensor placement for spatial lattice structure based on genetic algorithms," Journal of Sound and Vibration, vol. 317, no. 1-2, pp. 175–189, 2008.

[85] D. Demetriou, L. See, and J. Stillwell, "A spatial genetic algorithm for automating land partitioning," International Journal of Geographical Information Science, vol. 27, no. 12, pp. 2391–2409, 2013.

[86] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, and M. Woltereck, "An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties," Reliability Engineering & System Safety, vol. 77, no. 3, pp. 229–238, 2002.

[87] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" Structural safety, vol. 31, no. 2, pp. 105–112, 2009.

[88] R. T. Rockafellar, S. Uryasev et al., "Optimization of conditional value-at-risk," Journal of risk, vol. 2, pp. 21–42, 2000.

[89] S. J. Wright, "On the convergence of the newton/log-barrier method," Mathematical programming, vol. 90, no. 1, pp. 71–100, 2001.

[90] T. G. Dietterich, "Ensemble methods in machine learning," in International workshop on multiple classifier systems. Springer, 2000, pp. 1–15.

[91] S. Dogru and L. Marques, "A*-based solution to the coverage path planning problem," in Iberian Robotics conference. Springer, 2017, pp. 240–248.

[92] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in The 2012 international joint conference on neural networks (IJCNN). IEEE, 2012, pp. 1–8.

[93] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," Neural Computation, vol. 3, no. 1, pp. 88–97, 1991.

[94] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," arXiv preprint arXiv:1704.00805, 2017.

[95] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[96] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine learning, vol. 8, no. 3-4, pp. 229–256, 1992.

[97] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in International conference on artificial neural networks. Springer, 2018, pp. 270–279.

[98] P. Jackson and T. McRae, "Selection of sugarcane clones in small plots," Crop science, vol. 41, no. 2, pp. 315–322, 2001.

[99] A. D. Halvorson, F. C. Schweissing, M. E. Bartolo, and C. A. Reule, "Corn response to nitrogen fertilization in a soil with high residual nitrogen," Agronomy journal, vol. 97, no. 4, pp. 1222–1229, 2005.

[100] N. Hong, P. C. Scharf, J. G. Davis, N. R. Kitchen, and K. A. Sudduth, "Economically optimal nitrogen rate reduces soil residual nitrate," Journal of Environmental Quality, vol. 36, no. 2, pp. 354–362, 2007.

[101] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," Journal of Machine Learning Research, vol. 16, no. 1, pp. 1437–1480, 2015.

[102] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," Journal of Artificial Intelligence Research, vol. 24, pp. 81–108, 2005.