Journal of Data Science 21 (2), 333–353
April 2023

DOI: 10.6339/23-JDS1098
Computing in Data Science

An Assessment of Crop-Specific Land Cover Predictions Using High-Order Markov Chains and Deep Neural Networks☆

Luca Sartore[1,2,*], Claire Boryan[2], Andrew Dau[2], and Patrick Willis[2]

[1]*National Institute of Statistical Science, 1750 K Street NW Suite 1100, 20006 Washington DC, USA*
[2]*United States Department of Agriculture, National Agricultural Statistics Service, 1400 Independence Avenue SW, 20250 Washington DC, USA*

Abstract

High-Order Markov Chains (HOMC) are conventional models, based on transition probabilities, that are used by the United States Department of Agriculture (USDA) National Agricultural Statistics Service (NASS) to study crop-rotation patterns over time. However, HOMCs routinely suffer from sparsity and identifiability issues because the categorical data are represented as indicator (or dummy) variables. In fact, the dimension of the parametric space increases exponentially with the order of HOMCs required for analysis. While parsimonious representations reduce the number of parameters, as has been shown in the literature, they often result in less accurate predictions. Most parsimonious models are trained on big data structures, which can be compressed and efficiently processed using alternative algorithms. Consequently, a thorough evaluation and comparison of the prediction results obtain using a new HOMC algorithm and different types of Deep Neural Networks (DNN) across a range of agricultural conditions is warranted to determine which model is most appropriate for operational crop specific land cover prediction of United States (US) agriculture. In this paper, six neural network models are applied to crop rotation data between 2011 and 2021 from six agriculturally intensive counties, which reflect the range of major crops grown and a variety of crop rotation patterns in the Midwest and southern US. The six counties include: Renville, North Dakota; Perkins, Nebraska; Hale, Texas; Livingston, Illinois; McLean, Illinois; and Shelby, Ohio. Results show the DNN models achieve higher overall prediction accuracy for all counties in 2021. The proposed DNN models allow for the ingestion of long time series data, and robustly achieve higher accuracy values than a new HOMC algorithm considered for predicting crop specific land cover in the US.

Keywords *categorical prediction; crop rotation patterns; deep neural networks; transition probability*

1 Introduction

Accurate prediction of crop-specific planting prior to the growing season is important for crop monitoring, yield assessment, and decision support, and to strengthen traditional survey programs conducted by the United States Department of Agriculture (USDA) National Agricultural Statistics Service (NASS). The NASS Cropland Data Layers (CDL; Boryan et al., 2011), which

☆The findings and conclusions in this presentation are those of the authors and should not be construed to represent any official USDA or US Government determination or policy. This research was supported in part by the intramural research program of the US Department of Agriculture, National Agriculture Statistics Service.

*Corresponding author. Email: lsartore@niss.org.

Received July 29, 2022; Accepted March 23, 2023

are 30m crop-specific land-cover datasets, are the basis for crop rotation modeling described in this paper. The CDLs are available for the 48 conterminous states from 2008 to 2022 and disseminated to the public via the CroplandCROS web application (https://croplandcros.scinet.usda.gov/). When blended with USDA Farm Service Agency (FSA) Common Land Unit (CLU) farmer-reported data (Heald, 2002; USDA, FSA, 2017), CDLs have the potential to improve the estimation and forecasting of planted acreage as well as the imputation and sample design of area-based surveys. However, due to the large volume of data required to run the models, standard processing algorithms are insufficient to produce results within the short time frame required by operational programs. Therefore, NASS has been investigating and developing more suitable approaches to produce crop-prediction results that can be run at the national scale within a relatively brief time frame (i.e., within a day or two when operating in a high-performing computational environment).

Any sequence of crops observed over time can be considered as a categorical time series, which can be analyzed using different modeling strategies described in the literature. The most prominent examples are High-Order Markov Chains (HOMC; Tong, 1975), discrete autoregressive moving average (Jacobs and Lewis, 1983), integer autoregressive processes (Latour, 1998), and generalized linear models (Fokianos and Kedem, 2003). Most of these strategies, however, commonly use or approximate Transition Probabilities (TP), which are a natural avenue to model the temporal dynamics of a discrete random process. Thus, a random variable X_t with a finite discrete support, $\mathcal{S} = \{0, 1, \ldots, m-1\}$, where m represents the total number of observable states at time t, is used to describe the temporal evolution of the process. In their simplicity, HOMC can model crop-rotation patterns using the previous p categorical values to determine the future rotations.

Hence, consider a Markov chain of order p, $X_{t-p}, \ldots, X_{t-1}, X_t$, which satisfies the following equation:

$$\Pr(X_t = j | X_{t-1} = s_1, X_{t-2} = s_2, \ldots) = \Pr(X_t = j | X_{t-1} = s_1, X_{t-2} = s_2, \ldots, X_{t-p} = s_p), \quad (1)$$

for any $j, s_1, \ldots, s_p \in \mathcal{S}$, where the TP in (1) is conditioned on a finite set of previous states, i.e. by making the assumption that any state observed before time $t-p$ does not provide any information on the status of the chain at time t. This assumption allows one to build a model that requires the estimation of $m^p(m-1)$ parameters, which are usually represented as a generic entry t_{ij} of a TP matrix $\mathbf{T} \in [0, 1]^{m^p \times m}$, where

$$t_{ij} = \Pr(X_t = j | X_{t-1} = s_1, \ldots, X_{t-p} = s_p),$$

so that all rows sum to 1, for any $i = s_1 + m s_2 + m^2 s_3 + \cdots + m^{p-1} s_p$. Even if TP matrices account for all possible combinations of events, they become computationally impractical, especially when m is very large and $p > 1$. Moreover, since the size of parametric space increases exponentially at the rate $O(m^p)$, most entries of the TP matrix are unidentifiable. This commonly occurs when the number of observed unique rotations is much smaller than the size of the parametric space. Pegram (1980), Logan (1981) and Raftery (1985) were among the first to propose parsimonious models to approximate the TP of a fully specified HOMC. However, these models are less flexible than Machine Learning (ML) models that are widely accepted by the remote sensing community.

To analyze a categorical spatio-temporal process, Li and Reynolds (1997) developed an event-driven approach based on cellular automata (Wolfram, 1984) and first order Markov chains. This hybrid methodology was used to simulate vegetation dynamics, where TPs govern temporal patterns and local rules determine the spatial configuration of the neighborhood.

While this approach is relatively accurate (Halmy et al., 2015), it does not take advantage of long-term dynamics in the observed patterns. Osman et al. (2015) overcame this issue by using Bayesian networks to model the temporal dynamics of short sequences of input data (up to six years with only five crop categories). The method proposed by Osman et al. (2015) resulted in a 60% crop prediction accuracy without using remotely sensed images. Parker et al. (2003) reviewed different methods to study crop rotation patterns, where neural networks were considered as evolutionary models. Fully connected Deep Neural Networks (DNN) were evaluated by Zhang et al. (2019) to predict major crops in the United States (US) Corn Belt, which is the Midwestern US region that dominates US corn production. Although more sophisticated extensions exist (Yaramasu et al., 2020), DNNs are still limiting when training and validating the model over large areas (even with dedicated hardware for parallel computations). The parametric space of these established models reduces to $O(mp)$ when using binary indicator variables to encode the m land-cover categories. The parametric space can be further reduced when using Recurrent Neural Networks (RNN) that take advantage of nonlinear dependencies across time (Hopfield, 1982; Hochreiter and Schmidhuber, 1997). In recent years, sampling algorithms, which can also reduce the number of input data and the resulting training time, have become more common. These algorithms allowed Johnson and Mueller (2021) to achieve an overall accuracy of about 71% even though their random-forest model was trained on only 0.25% of the available data. Because random forests are an ensemble of decision trees (Breiman, 2001), they randomly disrupt the rotation patterns in (1), and this can result in unstable approximations of TPs. This aspect is more accentuated when pruning a single Classification And Regression Tree (CART; Breiman et al., 1984). In fact, to reproduce the results of a HOMC of order p, a CART model that uses recursive binary partitioning still requires a number of branches of size $O(m^p)$.

A feature engineering approach is proposed in this paper to decrease the dimension of the dataset (i.e., by reducing the number of variables). Like principal component analysis (PCA; Jolliffe, 1986), feature-reduction leads to parsimonious models that can achieve results comparable to more complex models (Buciluǎ et al., 2006; Ba and Caruana, 2014). However, these feature reduction algorithms result in information loss when compressing data. Thus, a "Dense" Binary Representation (DBR) of the categorical values is considered by taking inspiration from quantum information processing (Barnett, 2009), which uses a base-2 representation (Yuen, 1975; Miyashita et al., 2016) of the m integer numbers associated to each categorical value in $\mathcal{S}$. In fact, instead of using dummy variables (for the presence/absence of a particular categorical value), the base-2 representation can reduce the number of binary variables to $\lceil \log_2 m \rceil$ without losing information. Therefore, this allows one to operate with a reduced number of variables and to adopt more parsimonious models that may require less computational capacity.

After Deutsch (1985) introduced the first universal quantum Turing machine, its computational complexity was further investigated by Bernstein and Vazirani (1997), who compared it to a probabilistic Turing machine (Santos, 1969). Although these conceptual machines were shown to solve NP-hard problems in polynomial time (Berthiaume and Brassard, 1994; Shor, 1994; Grover, 1996), these ideas were not implemented in practice until the appearance of the first quantum computers. Nonetheless, Quantum-Inspired Neural Networks (QINN) were introduced by Menneer and Narayanan (1995), who developed a computationally efficient neural network where the weights assumed quantum properties. Following the introduction of the first quantum computational devices (Gibney, 2017), successive developments and examples of recent applications of quantum neural networks were presented in Jeswal and Chakraverty (2019). After the introduction of a python package named *tensorflow-quantum* (0.7.2; Broughton et al., 2020), the

development of hybrid neural network that operates with quantum layers became more accessible to researchers. However, the QINNs presented in the following sections can be processed using classical computers (even without the use of specialized hardware that might not be available to most practitioners).

The quantum neural models proposed in this paper are designed to study the predictive distribution of categorical outputs by mimicking the behavior of conceptual quantum circuits. Predictions and their associated uncertainties can be estimated using the random outputs generated by the circuits. One caveat of this approach is that while it is computationally more appealing than established bootstrap algorithms (Efron, 1979), it can be time-prohibitive when using stochastic algorithms for computing predictions on large datasets. Consequently, to reduce the computational burden, a deterministic solution is evaluated for its potential to expand the crop-rotation analysis to a national scale. Moreover, due to DBR, the QINN model can process crop-rotation patterns over longer time periods with less memory usage for data processing.

The remainder of this paper is structured as follows. Section 2 includes a description of the data preparation, the neural approximation of a quantum circuit, the prediction mechanism, and entropy-based statistics to evaluate the uncertainty associated with the crop predictions. Section 3 describes a real data application, and final conclusion and remarks are presented in Section 4.

2 Methodology

2.1 Data Preparation

Crop-rotation data obtained from the CDL are categorical time-series data that are produced through a classification process. This process is based on remotely sensed images, farmer-reported ground-reference data, and other ancillary data to ensure the production of high-quality, crop-specific, land cover products (Boryan et al., 2011). The CDLs are disseminated in GeoTIFF format (Ritter et al., 2000; Mahammad and Ramakrishnan, 2003) at the US national level. However, due to the large amount of data, specific areas of interest need to be subset before processing. This approach has been beneficial for processing longer time windows, and it allows the analysis to be focused on local rotation dynamics.

In their original format, the raw data are loaded in memory to construct a tabular dataset having the annual information stored by column. This implies that the raster files are not processed as a sequence of images, because the focus of this study is on the temporal evolution of crop rotation patterns. In fact, this perspective allows the proposed models to be used also for processing field-level data, where the crop information is stored in an attribute table that can be linked to a set of polygons delineating the fields (ESRI, 1998).

Categorical data can be used as integer numbers by models or algorithms leveraging HOMC theory (especially when both m and p are small). However, these data are typically converted into indicator variables to form a binary sequence of length $(p+1)\,m$, while the DBR can produce a shorter sequences of length $(p+1)\lceil \log_2 m \rceil$ without losing information. This is accomplished through the base-2 representation of integer numbers, which can also be used to operate with conceptual quantum circuits.

Rather than operating with bits that only represent either the state 0 or 1, quantum circuits operate with qubits (or quantum bit), which can simultaneously represent two eigenstates (or orthogonal states), i.e. $|0\rangle$ and $|1\rangle$, through a superposition $\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where the eigenstates $|0\rangle = (z_0, 0)^\top \in \mathbb{C}^2$, $|1\rangle = (0, z_1)^\top \in \mathbb{C}^2$, for any $z_0, z_1 \in \mathbb{C}$ that satisfy the condi-

tions $|z_0|^2 = |z_1|^2 = 1$, and $\alpha_0, \alpha_1 \in \mathbb{C}$, such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Therefore, the measurement of a physical qubit with quantum state $|\psi\rangle$ will return a zero with probability $|\alpha_0|^2$ or a one with probability $|\alpha_1|^2$.

When operating with q qubits, the superposition is a complex vector of dimension 2^q that describes the distribution of all possible combinations of zeros and ones as outputs from a quantum circuit. All these combinations can be found in the finite set $\{0, 1\}^q$, where the superscript indicates q Cartesian products of the set $\{0, 1\}$ with itself. Because the number of categorical values is less than the cardinality of the set of all possible outcomes from a quantum circuit, i.e. $m < |\{0, 1\}^q| = 2^q$, one can build a one-to-one correspondence between the m categorical values and the first m binary outcomes obtained through the measurement of the q qubits. For example, if $q = 2$, the superposition is formulated as

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \in \mathbb{C}^4, \tag{2}$$

where α_{00}, α_{01}, α_{10} and $\alpha_{11} \in \mathbb{C}$, such that $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$, and the eigenvectors are denoted by

$$\begin{aligned}
|00\rangle &= |0\rangle \otimes |0\rangle = (z_0 z_0, 0, 0, 0)^\top \in \mathbb{C}^4, \\
|01\rangle &= |1\rangle \otimes |0\rangle = (0, z_1 z_0, 0, 0)^\top \in \mathbb{C}^4, \\
|10\rangle &= |0\rangle \otimes |1\rangle = (0, 0, z_0 z_1, 0)^\top \in \mathbb{C}^4, \\
|11\rangle &= |1\rangle \otimes |1\rangle = (0, 0, 0, z_1 z_1)^\top \in \mathbb{C}^4,
\end{aligned}$$

where the symbol $\otimes$ denotes the Kroneker product. Therefore, the measurement of q physical qubits with the superposition in (2) will return the vector $(0, 0)^\top$ associated with the categorical value 0 with probability $|\alpha_{00}|^2$, the vector $(0, 1)^\top$ associated with the categorical value 1 with probability $|\alpha_{01}|^2$, the vector $(1, 0)^\top$ associated with the categorical value 2 with probability $|\alpha_{10}|^2$, and the vector $(1, 1)^\top$ associated with the categorical value 3 with probability $|\alpha_{11}|^2$.

Furthermore, quantum circuits process the information in input using unitary transformations (as one-qubit gate) on individual qubits, while multi-qubit transformations generate highly entangled states. Entanglement is achieved when two or more qubits provide information on the state of the others (i.e. when one qubit becomes dependent to a degree on the state of another). The state of a single qubit can be altered as $\mathbf{U}|\psi\rangle$, where $\mathbf{U} \in \mathbb{C}^{2\times 2}$ is a unitary matrix. However, q qubits are processed using adequate unitary matrices of size $q \times q$ to alter the degree of entanglement. Because unitary transformations preserve the inner product between two vectors, unitary matrices ensure the computational reversibility of a quantum algorithm.

2.2 Model Development

Traditional neural networks were developed to mathematically imitate the flow of information as processed by biological systems (Bishop and Nasrabadi, 2006). McCulloch and Pitts (1943) were the first to attempt modeling neural synapses and introduced concepts that can be used to develop quantum neural computing. For example, the qubits initialized to a $0\rangle$ state are individually processed using the Hadamard gates, i.e. $\mathbf{H}|0\rangle$, where

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

This operation sets the qubits to a uniform superposition between the two eigenstates, and hence, a measurement of the state $\mathbf{H}|0\rangle$ will return a zero or a one with equal probability, i.e. $\frac{1}{2}$.

This operation that follows can be performed using rotation gates, that are defined as

$$\mathbf{R} = \begin{bmatrix} \cos(\frac{\beta}{2}) & -i\sin(\frac{\beta}{2}) \\ -i\sin(\frac{\beta}{2}) & \cos(\frac{\beta}{2}) \end{bmatrix} \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \begin{bmatrix} e^{-i\frac{\zeta}{2}} & 0 \\ 0 & e^{i\frac{\zeta}{2}} \end{bmatrix},$$

where $i = \sqrt{-1}$, and $\beta, \theta, \zeta \in [-\pi, \pi]$ are trainable parameters. This means that the input information in each qubit is processed as $\mathbf{RH}|0\rangle$ before being entangled through the use of controlled-not (CNOT) gates, i.e.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The CNOT gate entangles the quantum states of two qubits, and several CNOT gates can be used to combine the information in the qubits before it is passed through the next layer of a neural network (Panchi and Shiyong, 2008). Figure 1 illustrates a quantum circuit that processes four qubits using the Hadamard, rotation, and CNOT gates (Zhao and Wang, 2021). Although the neural quantum circuit in Figure 1 satisfies the requirements in Schuld et al. (2014), the implementation of a quantum neural network that uses this approach would have been challenging without using specialized software. However, the conceptual circuit shown in Figure 1 can be easily implemented using the *cirq* package (Cirq Developers, 2022) in python, and it can be used as a layer of a DNN model through the use of the *tensorflow-quantum* package.

However, when a neural network implementation based on the *tensorflow-quantum* package cannot be trained with traditional computational devices (due to the large number of observations), it is possible to develop a neural network model to mimic the behavior of such circuits. In practice, a QINN can process binary inputs through a sequence of hidden layers and can, finally, simulate a binary output from a multivariate Bernoulli distribution. This contrasts with more established architectures that compute the most likely category from a multinomial distribution.

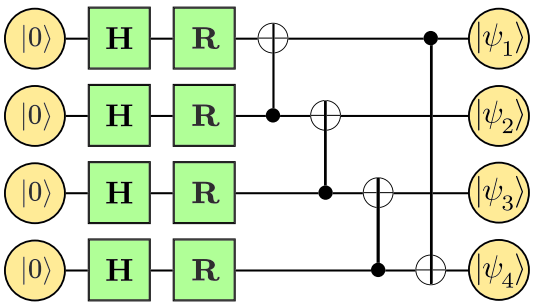

Figure 1: Quantum circuit that operates on four qubits (or quantum bits) with the capacity of processing $2^4 = 16$ eigenstates (or orthogonal states) in input. The green squares are quantum logic gates applicable to individual qubits. The quantum gates depicted with the matrix $\mathbf{H}$ are Hadamard gates, and those represented with the matrix $\mathbf{R}$ are rotation gates. Two-qubit operations, such as the CNOT (or controlled not), are represented by linking the symbols $\oplus$ and $\bullet$ with a straight vertical line. Single-qubit operations change the probability amplitudes of the qubits, while multi-qubit operations are used to entangle two or more qubits.

In fact, empirical predictive distributions can be constructed by looking at the relative frequencies of the random binary digits, and other inferential analyses are then conducted similarly to parametric bootstrap methods.

From a computational perspective, the feature "compression" performed by the DBR leads to parsimonious networks that can be trained much more quickly. Thus, the size of the parametric space decreases to $O(\lceil \log_2 m \rceil p)$ when using fully connected, feed-forward neural networks. However, the number of parameters shrinks to

$$O(\lceil \log_2 m \rceil) \tag{3}$$

when recurrent layers are introduced in the neural architecture. This occurs because the number of parameters in a recurrent layer does not depend on the number of time steps. E.g., the `SimpleRNN` implementation in the *tensorflow-keras* package (Chollet et al., 2015) requires a number of parameters that is equal to

$$(\lceil \log_2 m \rceil + 1)u + u^2,$$

where u represents the number of neurons (or units) within the recurrent layer, and $\lceil \log_2 m \rceil + 1$ denotes the number of coefficients associated to the input features and the bias parameter. Furthermore, Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU; Cho et al., 2014) also have a number of parameters that is expressed solely as a function of $\lceil \log_2 m \rceil$ and u, where u is here treated as a constant to obtain the same Landau notation shown in (3).

When the number of predictions to compute is large, the randomization required to mimic a quantum circuit becomes very expensive. Therefore, a deterministic collapsing mechanism of the output layer can be used instead of the actual measuring mechanism of the quantum-circuit states. This can be easily performed using a given threshold to compute the most probable binary values from the output layer of a QINN model. Denoting the vector-valued output of QINN by $(\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{q-1})^\top \in [0, 1]^q$, one can write the categorical prediction in its expanded form on base 2 as

$$\hat{X}_t = \sum_{j=0}^{q-1} 2^j \mathbb{1}_{[\tau_j,\, 1]}(\hat{y}_j), \tag{4}$$

where the indicator function $\mathbb{1}_{[\tau_j,\, 1]}(\hat{y}_j)$ is defined as

$$\mathbb{1}_{[\tau_j,\, 1]}(\hat{y}_j) = \begin{cases} 1, & \text{if } \hat{y}_j \in [\tau_j,\ 1], \\ 0, & \text{otherwise,} \end{cases}$$

where the thresholds $\tau_j \in (0, 1)$, for any $j = 1, \ldots, q$, can be either set to 0.5 or computed by minimizing prediction errors. In fact, optimized thresholds should be considered especially when operating with class-imbalanced samples.

Further computational benefits arise by using sigmoidal functions for modeling the output neurons. Because the derivatives of these functions can be formulated more easily, they require less computations resulting in enhanced computational performance.

2.3 Evaluation of Prediction Quality

Measures of prediction accuracy (Congalton and Green, 2019) are typically computed using historical test data to determine whether a trained model can produce good forecasts. This

often results in a set of statistics that provide a reasonable assessment of the overall prediction quality by losing small-scale information that could be relevant in other applications. E.g., field-level measures of prediction uncertainty can be used to identify fields for which the acreage of a specific crop can be accurately imputed as suggested by Sartore et al. (2022).

Although normalized entropy measures (Shannon, 1948) can be computed for DNN models (Zhang et al., 2019; Yaramasu et al., 2020), small-scale QINN-prediction uncertainty approximated using T-norms (Gupta and Qi, 1991) is preferable because it avoids the reconstruction of the empirical predictive distribution over 2^q possible outcomes. Moreover, if at least one binary output is more uncertain than others, the predicted category from the QINN-model is more likely to be incorrect. Therefore, the maximum normalized uncertainty value computed for each output neuron can provide an uncertainty index for the prediction in (4). This operation can be viewed as the union of q fuzzy sets (Zadeh, 1973), where normalized uncertainties are used as membership functions. Thus, the uncertainty of the categorical prediction can be computed as the maximum normalized entropy:

$$\mathsf{U}_{\mathsf{H}}\left[\hat{X}_t\right] = \max_{i=0,\dots,q-1}\left\{\hat{h}_i\right\},$$

where

$$\hat{h}_i = \begin{cases} -\left[(1-\hat{y}_i)\log(1-\hat{y}_i) + \hat{y}_i \log(\hat{y}_i)\right]\log(2)^{-1}, & \text{if } \hat{y}_i \in (0,1), \\ 0, & \text{if } \hat{y}_i \in \{0,1\}, \end{cases} \tag{5}$$

or the maximum normalized Gini index

$$\mathsf{U}_{\mathsf{G}}\left[\hat{X}_t\right] = \max_{i=0,\dots,q-1}\left\{4\hat{y}_i(1-\hat{y}_i)\right\},$$

which is more appropriate for large datasets because it avoids the computation of the logarithms in (5) and provides a reasonable approximation. In both cases, the uncertainty index takes values from the interval [0, 1] (as shown in Figure 2), where 0 and 1 denote high and low model confidence respectively.

2.4 An Alternative Markov-Based Algorithm

For comparison purposes, a new predictive algorithm based on HOMC theory was specifically developed to select the most likely category without estimating the whole TP matrix. Because the number of matches between the most recent rotation patterns and those observed in the past (i.e. having a temporal lag of one year) diminishes as p increases, it is also required to produce predictions using shorter patterns (i.e. having length $p^* < p$). This becomes particularly helpful to produce predictions for the entire area without missing certain locations due to the possible absence of a certain pattern on record. For implementation purposes, the algorithm iteratively replaces the predictions made at the previous step with the most likely category using increasingly longer patterns of length p^*. The algorithm stops when the last updates have been made using crop rotation patterns of length $p^* = p$.

At first, the algorithm assigns to the whole county the most frequent category observed at time $t-1$. Successively, p^* is set to one, and the categories observed at time $t-p^*-1$ are matched to those at time $t-p^*$. The most frequent categories for each pattern that have been matched in the past are then computed using simple counts that satisfy the matching conditions. For example, if corn is most frequently planted after soybeans, then corn is predicted for time t in a field, which was planted to soybeans at time $t-1$. These predictions are then used to update

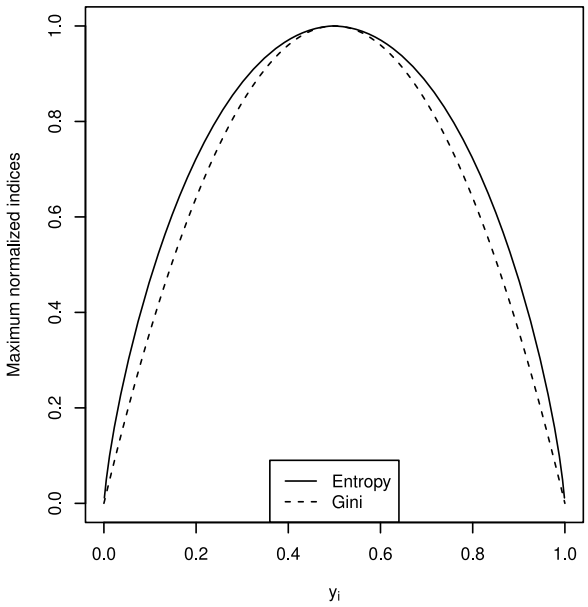

Figure 2: Maximum normalized Gini index (dashed line) and maximum normalized entropy index (solid line).

the categorical values for the locations with a matching pattern. Afterward, the value of p^* is increased to add the information from one extra year in the study. Therefore, more complex patterns, which include more than one crop in a sequence, are matched and counted to identify the most frequent crops. This iterative process continues until the final predictions are made with $p^* = p$.

This algorithm can be computationally more appealing because it is based on counting crop transitions rather than identifying optimal parameter values through gradient-based optimization algorithms. Furthermore, it does not require a lot of memory to run, because it operates with the input data in their native format without using the base-2 representation that requires at least $32\lceil \log_2 m \rceil$ bits to store binary values according to the IEEE 754 standard (Kahan, 1996).

3 Real Data Application

To ensure the reproducibility and replicability of the methodology and algorithms described in Section 2, crop rotation patterns were studied using exclusively publicly available CDL data (Boryan et al., 2011). Six agriculturally intensive counties, which reflect the range of major crops grown in the Midwest and southern regions of the US, were selected to assess the performance of the considered models. The county locations are highlighted in Figure 3 with the letters A–F. All CDL categories were used for the predictive analysis. The six counties include: Renville, North Dakota (A); Perkins, Nebraska (B); Hale, Texas (C); Livingston, Illinois (D); McLean, Illinois (E); and Shelby, Ohio (F). Eight major crops including barley, cover crops, oats, rye, durum wheat, spring wheat, winter wheat and soybeans are grown in Renville, North Dakota (A). Corn and winter wheat are grown in Perkins, Nebraska (B). Cotton, corn, winter wheat, and sorghum are the major commodities grown in Hale, Texas (C). Corn and soybeans are the major

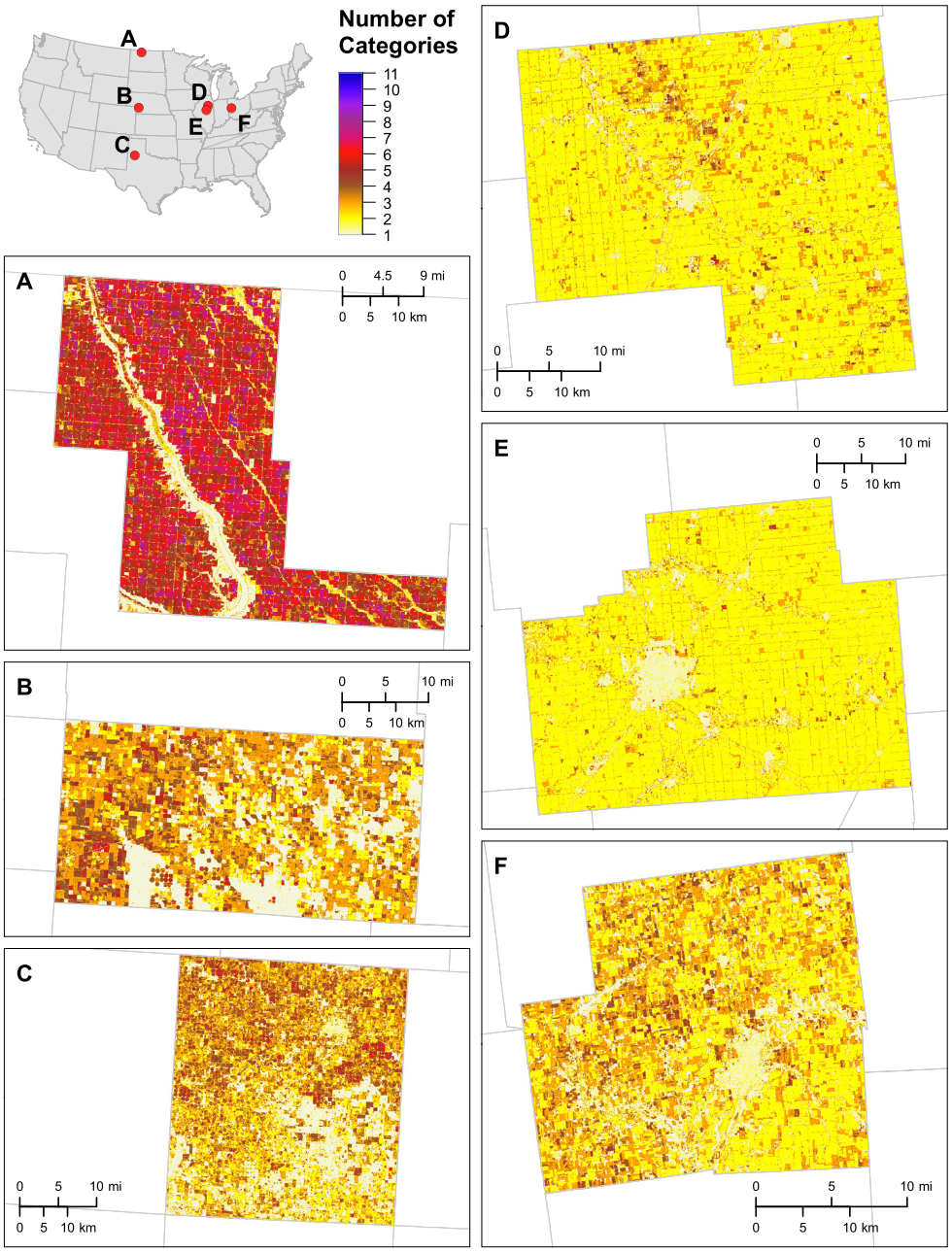

Figure 3: Six agriculturally intensive US counties were selected to assess model performance. The county locations are identified with the letters A – F. The six counties include: Renville, North Dakota (A); Perkins, Nebraska (B); Hale, Texas (C); Livingston, Illinois (D); Mclean, Illinois (E); and Shelby, Ohio (F). The number of CDL categories that rotate from 2011 to 2021 are shown. Pale yellow indicates no change. Yellow indicates two crops rotating back and forth. Brown and red indicate three to five crops rotating and purple and blue indicate as many as 10 to 11 crops or CDL categories in rotation.

commodities grown in Livingston (D) and McLean (E) Counties in Illinois, as well as Shelby, County in Ohio (F).

Figure 3 identifies the number of CDL categories (crops and other land cover types) that change or rotate from 2011 to 2021 at the pixel level. For example, in Renville, North Dakota (A), there are many crops with varied rotation patterns. The CDL categories that do not generally change or rotate, such as water, forest, grass/pasture, or urban areas, are illustrated in pale yellow. Much of the land cover in Renville, North Dakota rotates between different crops from four to eight times (as shown in the brown-red areas) over the eleven years considered. Some pixels illustrate a rotation pattern between different crop every year as shown in blue. In Perkins, Nebraska (B) there are fewer crops grown with more consistent rotations. These are shown in predominantly yellow and brown. Four major crops are grown in Hale, Texas (C) and rotate consistently with two to five identified crops. The corn and soybeans grown in Livingston (D) and McLean (E) Counties in Illinois follow a regular rotation pattern (corn followed by soybeans or soybeans followed by corn), which is illustrated with the yellow color. Corn and soybeans are also grown in Shelby, County in Ohio (F) but the rotation pattern is not as perfectly consistent back and forth from one crop to the next as shown in the brown to red colors indicating two to five crops in the rotations.

To compare the performance of the methodology and algorithms proposed in Section 2, the crop rotation patterns at the county-level were studied using:

- The HOMC-based algorithm described in Section 2.4.
- A feed-forward DNN (as discussed by Zhang et al., 2019) that uses indicator/dummy variables (i.e., one for the presence of a specific categorical value, and zero for its absence).
- A RNN that uses indicator/dummy variables in input and output (named RNN1).
- A RNN that uses the DBR for the variables in input and indicator/dummy variables in output (named RNN2).
- A RNN that uses indicator/dummy variables in input and the DBR for the variables in output (named RNN3).
- A RNN that uses the DBR for both input and output variables (named QINN).

A DNN using a quantum layer was implemented in *tensorflow-quantum* (named TFQN, as in TensorFlow-Quantum Network), but it could not produce results due to technical limitations of the computational environment used for this study. In fact, the TQFN model requires a lot of memory because it needs to transform a large number of data points into quantum data.

The training set was formed by randomly selecting 80% of the rotations observed between 2011 and 2020, while the remaining 20% were used for validation. By moving one step forward in time, all data between 2012 and 2021 were used for testing purposes. The learning rate of the ADAM optimizer (Kingma and Ba, 2014; Chollet et al., 2015) was set to 0.001 while the other settings were kept fixed to their default values. For all networks, the batch size was set to 2,048 units and number of epochs to 4. Furthermore, to homogenize the variety of training examples, the batches were randomly formed at each epoch using a shuffling algorithm. These choices were made for assessing the convergence of the optimization algorithm using a limited number of iterations. However, a higher number of epochs can help provide a better solution for the models where the optimization algorithm requires more iterations to fully converge.

The several architectures developed for this application are shown in the tables in Appendix A. Because the original data are natively stored as 8-bit integers taking values from the set $\{0, 1, \ldots, 255\}$, the output layers have 256 neurons when indicator variables are considered and 8 neurons when the "dense" binary representation is used. The activation function in output is the softmax for all neural networks with exception for the logistic sigmoid

Table 1: Average overall accuracy (%) at the county level obtained with six different models/algorithms. Highest county-level accuracy is bolded.

County	HOMC	DNN	RNN1	RNN2	RNN3	QINN
Livingston, IL	91.3	**93.8**	91.4	92.9	93.0	89.9
McLean, IL	91.5	**95.2**	94.3	94.9	94.9	92.8
Perkins, NE	79.1	**91.9**	89.2	89.1	89.1	87.0
Renville, ND	74.1	**85.3**	83.4	84.1	80.8	70.4
Shelby, OH	80.7	**90.7**	85.9	86.9	87.2	87.2
Hale, TX	69.4	**77.0**	76.3	75.7	68.6	66.7

used for RNN3 and QINN. The total number of trainable parameters dropped precipitously from 592,388 (when using standard indicator variables) to 133,636 (when adopting recurrent layers) and around 70,000 or less (when considering "dense" binary variables and recurrent layers). Rectified Linear Units (ReLU; Nair and Hinton, 2010) were considered as an activation function of most hidden layers to address possible gradient vanishing problems. Dropout layers were inserted in the architectures as a form of regularization (Hinton et al., 2012). Moreover, simple (unrolled) RNN layers were considered to take advantage of temporal relationships in the data.

Through the analysis of the resulting confusion matrix, standard accuracy measures (Congalton and Green, 2019) are computed to compare the performances of the two models (see Figure 4). The overall accuracy indicates general agreement between the 2021 CDL and the predictions. However, since the initialization of the parameters is randomized, the distribution of the accuracy results is reported for each network rather than focusing on the best results achieved by each architecture. This is achieved by computing the overall accuracy values for 50 different replicates of the predicting procedures. The resulting accuracy of the fitted models are shown in the boxplots in Figure 4. Furthermore, the average overall accuracy (shown in Table 1) is computed by county for each neural model, and can be compared to the overall accuracy produced using the HOMC algorithm described in Section 2.4 with the same input information provided to the neural models, i.e. by considering crop rotation patters of length $p = 9$. In Livingston and McLean County in Illinois where the primary crops are corn and soybeans in a regular rotation, the DNN model achieves overall prediction accuracy values that are over 93%. In Shelby, Ohio (predominantly planted to corn and soybeans), and in Perkins, Nebraska (where corn, soybeans and wheat are grown), the DNN prediction accuracy values are between 90% and 92%. In Renville, North Dakota, where there is a much larger variety of crops grown with a varied crop rotation pattern, the accuracy of the DNN model is 85%. In Hale, Texas, the DNN overall prediction accuracy is 77%, the lowest when compared to the other counties. On average, the neural models perform quite similarly, although the HOMC outperformed RNN3 in Hale, Texas, and the QINN in Hale, Texas; Renville, North Dakota; and Livingston, Illinois.

The results obtained through the approach based on the HOMCs are computed for the six counties shown in Table 2. The highest accuracy values were produced for the Illinois counties due to more stable rotations dominated by corn and soybeans. However, the algorithm produced better predictions using the information from the previous year for the counties in Illinois and Nebraska. The highest HOMC accuracy was produced using three-year rotations in North Dakota, four-year rotations in Ohio, and five-year rotations in Texas. It appears that the accuracy is dependent on both the variety of crops planted in a county and on the lengths of the

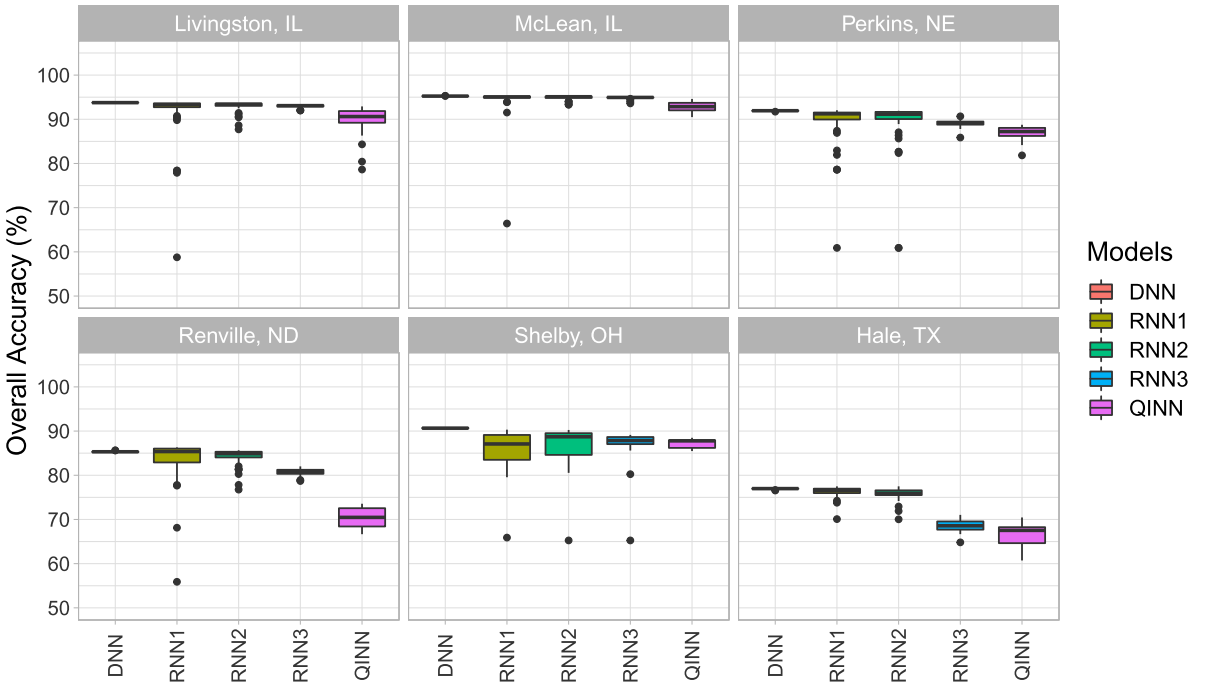

Figure 4: Accuracy indices (%) for the two neural architectures considered. Five statistics (minimum, first quartile, median, third quartile, and maximum) are reported to summarize the distribution of the results obtained after 50 experimental replicates.

Table 2: Overall accuracy (%) obtained with HOMCs at different time steps p. Best performance are bolded.

County	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	$p=9$
Livingston County, IL	**91.8**	91.4	91.7	91.5	91.5	91.5	91.5	91.5	91.3
McLean County, IL	**92.3**	92.1	91.8	91.9	91.9	91.9	91.8	91.6	91.5
Perkins County, NE	**81.4**	78.1	79.0	79.0	80.3	79.9	79.3	79.3	79.1
Renville County, ND	73.7	76.5	**77.0**	76.1	75.0	74.8	74.6	74.3	74.1
Shelby County, OH	80.5	80.3	81.5	**81.6**	81.1	80.9	80.9	80.7	80.7
Hale County, TX	71.4	70.6	70.1	71.4	**71.4**	70.6	70.4	69.7	69.4

sequences of crops provided in input to the models. However, none of the accuracies in Table 2 are higher than those produced using the DNN and showed in Table 1.

From a computational perspective, the results obtained to evaluate the computational performances were produced on a virtual machine with 20.04 Ubuntu Linux operative system running on a cloud environment operated by Google with an Intel Xeon CPU having 6 dual-threading cores at 2.20GHz, 504GB of RAM. The programs were written in python (3.8.10) to operate with *tensorflow* (2.7.0; Abadi et al., 2015). Furthermore, two C functions were implemented to convert native raster data into the binary format specific to the network under consideration. To accelerate such conversion, the *OpenMP* library (Dagum and Menon, 1998) and Single Instruction-stream Multiple Data-stream (SIMD; Cypher and Sanz, 2012) technology were used to allow the code to run in parallel on several CPU cores. Specialized hardware, such

as GPUs, TPUs or optical/photonic devices, have the potential to improve the computational efficiency required by the neural models considered in this paper. However, the HOMC-based algorithm was implemented in R, and it does not require a supercomputer to produce results when using data at the county level. Although the HOMC algorithm has been designed to operate with the least amount of memory, the differences in accuracy with respect to the best neural network model do not justify its potential use for operational purposes.

The time required for data preparation from a GeoTIFF format into feather datasets ranged between 1.34 seconds (for the smallest county, i.e. Shelby, OH) to about 7 seconds (for the largest county, i.e. Renville, ND). The conversion of the input variables into indicator variables ranged from 0.12 to 0.33 seconds, while the conversion into "dense" binary variable ranged from 0.03 to 0.14 seconds. The difference in time observed between the two type of conversion is mostly attributed to memory allocation rather than computation time. As shown in Table 8, the average total running time by model and county was mostly dependent on the number of data points to process. For the smallest county, the running time ranged between 9.3 seconds (for QINN) and 55.9 seconds (for RNN3). For the largest county, the running time ranged between 39.8 seconds (for QINN) and 178 seconds (for RNN3). In general, the training time required by RNN1, RNN2 and RNN3 models tends to be longer than the training time required by other network models. The time elapsed for the HOMC is not comparable with those reported for the neural networks. However, the results reported in Table 2 were produced on a laptop with Windows 10 mounting an Intel Core i5-8350 CPU at 1.7 GHz and with 8 GB of RAM.

When using DNN the total runtime is dependent on the number of samples in each batch to compute the stochastic gradient descent and the total number of epochs. On the one hand, if the number of batches is relatively small, more time is required but faster convergence might be achieved within an epoch. On the other hand, a larger number of epochs can produce models that are more stable. In this study, it was observed that the networks using the DBR could require more epochs, but this does not guarantee that the overall accuracy is going to be as high as that produced by the DNN model. In an operational environment, where a tradeoff between accuracy and training time must be considered, the use of recurrent layers should be avoided. However, the adoption of cloud technologies is required to allow several DNN models to be trained in parallel for processing much larger spatial datasets.

4 Conclusion

This paper provides a thorough evaluation and comparison of the crop specific prediction results obtained using the HOMC algorithm and five types of DNN models for six major crop-production counties, which reflect the range of major crops grown and a variety of crop rotation patterns in the Midwest and southern US. In this paper, five models are applied to crop rotation data between 2011 and 2021 from Renville, North Dakota; Perkins, Nebraska; Hale, Texas; Livingston, Illinois; McLean, Illinois; and Shelby, Ohio (as shown in Figure 3). Although the DNN is the less parsimonious model among those considered, the DNN model achieved the highest overall accuracies for the six counties when compared to those obtained with the other models (as shown in Table 1). All model results were compared to the official CDL in 2021.

Specifically, in Livingston and McLean Counties, Illinois; where there are two major crops and a consistent corn and soybean rotation pattern; the DNN achieved the highest accuracies with Livingston at 93.8% and Mclean at 95.2%. This is compared with the corresponding neural model results which range from 89.9% to 94.9%. In Shelby, Ohio, where corn and soybeans

are the major crops, but grown in a less consistent crop rotation pattern than Illinois. The DNN achieved a 90.7% prediction accuracy while the other approaches achieved accuracy values ranging from 80.7% to 87.2%. In Perkins, Nebraska; where a wider range of crops including corn, soybeans and winter wheat are grown; the DNN accuracy is 91.9% and the accuracy values for the other models range from 79.1% to 89.2%. In Hale County, Texas; the DNN accuracy value is 77% while the accuracy values for the other models range from 69.4% to 76.3%. In Renville, ND, where a wide variety of crops are grown with range of rotation patterns; the DNN accuracy is 85.3% and the accuracy values for the other models range from 74.1% to 84.1%. Based on these results, the proposed DNN model, which allows for the incorporation of long time series data, provides the most robust algorithm for crop prediction modeling across six agriculturally intensive counties and potentially the 48 conterminous states.

Even if the proposed algorithm based on HOMC counts crop transitions rather than optimizing a set of parameters through gradient-based algorithms, it is not always the most computationally efficient approach studied in this paper. Even if the DNN model is not the most parsimonious model it requires very few epochs to produce stable results. Consequently, the DNN model can be considered for large scale applications when powerful computational devices are available. Future research will involve generalizing the proposed methodology to the 48 conterminous states, which will require sampling the data from an even greater variety of agricultural practices. Strong prediction accuracies ranging from 77% to 95.2% were achieved with DNN over the six counties. For operational crop classification, like the methodology used to produce the NASS CDL, the highest accuracies are generally obtained at the end of the summer growing season (October 1) when satellite imagery acquired throughout the entire growing season (between April 1 to October 1) are available. In the future, improvements to the networks considered in this paper may be achieved by accounting for information on the early (and late) stages of the crop phenological cycle.

Supplementary Material

Supplementary materials of the paper entitled "An Assessment of Crop-Specific Land Cover Predictions Using High-Order Markov Chains and Deep Neural Networks"

A Neural Network Architectures

Table 3: Structure of the DNN model implemented to predict the next crop rotation using standard indicator/dummy variables as inputs and outputs.

Layer type	**Output shape**	**Num. param.**
Flatten	(None, 2304)	0
Dense ReLU	(None, 256)	590,080
Dropout at 50%	(None, 256)	0
Dense ReLU	(None, 4)	1,028
Dense softmax	(None, 256)	1,280
Total parameters: 592,388		
Trainable parameters: 592,388		
Non-trainable parameters: 0		

Table 4: Structure of the RNN1 model implemented to predict the next crop rotation using standard indicator/dummy variables as inputs and outputs.

Layer type	**Output shape**	**Num. param.**
SimpleRNN	(None, 256)	131,328
Dropout at 50%	(None, 256)	0
Dense ReLU	(None, 4)	1,028
Dense softmax	(None, 256)	1,280
Total parameters: 133,636		
Trainable parameters: 133,636		
Non-trainable parameters: 0		

Table 5: Structure of the RNN2 model implemented to predict the next crop rotation using "dense" binary variables as inputs and indicator/dummy variables as outputs.

Layer type	**Output shape**	**Num. param.**
SimpleRNN	(None, 256)	67,840
Dropout at 50%	(None, 256)	0
Dense ReLU	(None, 4)	1,028
Dense softmax	(None, 256)	1,280
Total parameters: 70,148		
Trainable parameters: 70,148		
Non-trainable parameters: 0		

Table 6: Structure of the RNN3 model implemented to predict the next crop rotation using indicator/dummy variables as inputs and "dense" binary variables as outputs.

Layer type	**Output shape**	**Num. param.**
SimpleRNN	(None, 256)	131,328
Dropout at 50%	(None, 256)	0
Dense ReLU	(None, 4)	1,028
Dense sigmoid	(None, 8)	40
Total parameters: 132,396		
Trainable parameters: 132,396		
Non-trainable parameters: 0		

B Computational Time

Table 7: Structure of the QINN model implemented to predict the next crop rotation using "dense" binary variables as inputs and output.

Layer type	**Output shape**	**Num. param.**
SimpleRNN	(None, 9)	162
Dropout at 50%	(None, 9)	0
Dense ReLU	(None, 64)	640
Dense sigmoid	(None, 8)	520
Total parameters: 1,322		
Trainable parameters: 1,322		
Non-trainable parameters: 0		

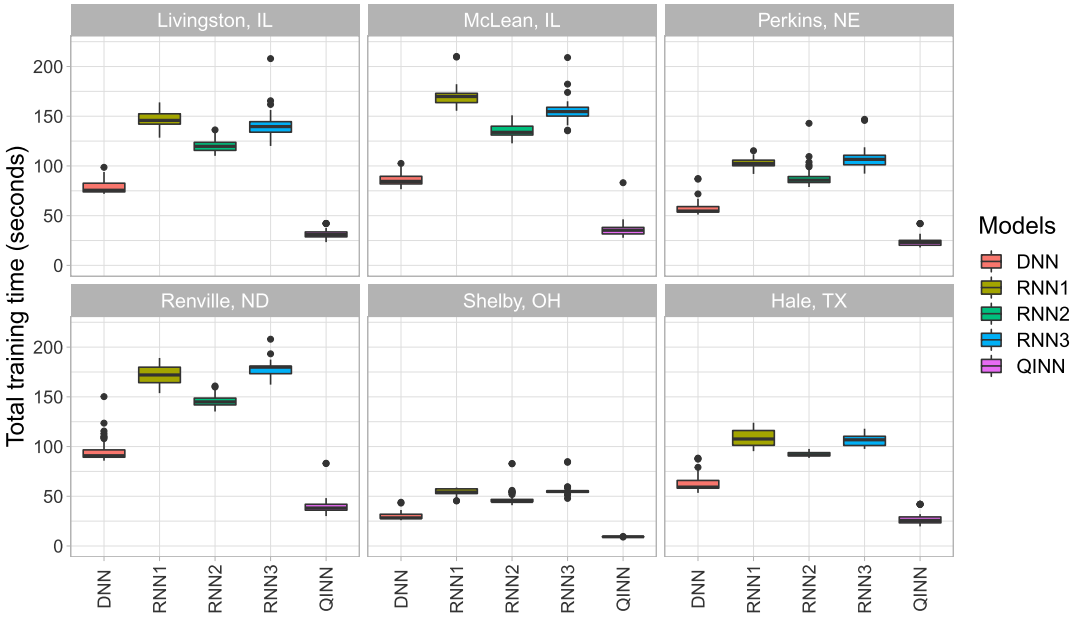

Figure 5: Distributions of total training time (expressed in seconds) of five neural models shown for each county.

Table 8: Average total training time (expressed in seconds) for the neural models and HOMC algorithm shown for each county. The fastest approach of each county is bolded.

County	**HOMC**	**DNN**	**RNN1**	**RNN2**	**RNN3**	**QINN**
Livingston, IL	**14.5**	79.0	147.2	120.2	140.9	31.4
McLean, IL	**19.0**	86.3	170.4	135.3	155.7	36.1
Perkins, NE	24.0	57.5	103.0	88.3	107.2	**23.9**
Renville, ND	88.7	95.5	171.5	145.6	178.0	**39.8**
Shelby, OH	15.0	30.0	54.1	47.4	55.9	**9.3**
Hale, TX	43.7	63.1	108.4	92.2	106.3	**27.0**

Acknowledgement

The authors would like to thank Avery Nagle for preparing the graphics in Figure 3, Valbona Bejleri, Eileen O'Brien, Linda J. Young, Michael Grosskopf, and two anonymous reviewers for providing useful comments on early drafts of this paper.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ba J, Caruana R (2014). Do deep nets really need to be deep? In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ (Eds.), *Advances in Neural Information Processing Systems* volume 27.

Barnett S (2009). *Quantum Information*, volume 16. Oxford University Press.

Bernstein E, Vazirani U (1997). Quantum complexity theory. *SIAM Journal on Computing*, 26(5): 1411–1473. https://doi.org/10.1137/S0097539796300921

Berthiaume A, Brassard G (1994). Oracle quantum computing. *Journal of Modern Optics*, 41(12): 2521–2535. https://doi.org/10.1080/09500349414552351

Bishop CM, Nasrabadi NM (2006). *Pattern Recognition and Machine Learning*. Springer.

Boryan C, Yang Z, Mueller R, Craig M (2011). Monitoring us agriculture: The US Department of Agriculture, national agricultural statistics service, cropland data layer program. *Geocarto International*, 26(5): 341–358. https://doi.org/10.1080/10106049.2011.562309

Breiman L (2001). Random forests. *Machine Learning*, 45(1): 5–32. https://doi.org/10.1023/A:1010933404324

Breiman L, Friedman J, Olshen R, Stone C (1984). *Cart. Classification and Regression Trees*. Routledge.

Broughton M, Verdon G, McCourt T, Martinez AJ, Yoo JH, Isakov SV, et al. (2020). Tensorflow quantum: A software framework for quantum machine learning. arXiv preprint: https://arxiv.org/abs/2003.02989.

Buciluǎ C, Caruana R, Niculescu-Mizil A (2006). Model compression. In: Ungar L, Craven M, Eliassi-Rad T (Eds.), *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 535–541.

Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint: https://arxiv.org/abs/1409.1259.

Chollet F, et al. (2015). Keras. https://keras.io

Cirq Developers (2022). Cirq. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

Congalton RG, Green K (2019). *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*. CRC Press.

Cypher R, Sanz JL (2012). *The SIMD Model of Parallel Computation*. Springer Science & Business Media.

Dagum L, Menon R (1998). OpenMP: An industry standard API for shared-memory programming. *IEEE Computational Science & Engineering*, 5(1): 46–55. https://doi.org/10.1109/99.660313

Deutsch D (1985). Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical*

Sciences, 400: 97–117. 1818.
Efron B (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7: 1–26.
ESRI (1998). ESRI shapefile technical description. *Technical report*, Environmental Systems Research Institute, Inc.
Fokianos K, Kedem B (2003). Regression theory for categorical time series. *Statistical Science*, 18: 357–376. https://doi.org/10.1214/ss/1076102425
Gibney E (2017). D-wave upgrade: How scientists are using the world's most controversial quantum computer. *Nature*, 541(7638): 447–448. https://doi.org/10.1038/541447b
Grover LK (1996). A fast quantum mechanical algorithm for database search. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–219.
Gupta MM, Qi J (1991). Theory of t-norms and fuzzy inference methods. *Fuzzy Sets and Systems*, 40(3): 431–450. https://doi.org/10.1016/0165-0114(91)90171-L
Halmy MWA, Gessler PE, Hicke JA, Salem BB (2015). Land use/land cover change detection and prediction in the north-western coastal desert of Egypt using Markov-CA. *Applied Geography*, 63: 101–112. https://doi.org/10.1016/j.apgeog.2015.06.015
Heald J (2002). Usda establishes a common land unit. *ArcUser Online*. https://www.esri.com/news/arcuser/0402/usda.html
Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint: https://arxiv.org/abs/1207.0580.
Hochreiter S, Schmidhuber J (1997). Long short-term memory. *Neural Computation*, 9(8): 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
Hopfield JJ (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8): 2554–2558. https://doi.org/10.1073/pnas.79.8.2554
Jacobs PA, Lewis PA (1983). Stationary discrete autoregressive-moving average time series generated by mixtures. *Journal of Time Series Analysis*, 4(1): 19–36. https://doi.org/10.1111/j.1467-9892.1983.tb00354.x
Jeswal S, Chakraverty S (2019). Recent developments and applications in quantum neural network: A review. *Archives of Computational Methods in Engineering*, 26(4): 793–807. https://doi.org/10.1007/s11831-018-9269-0
Johnson DM, Mueller R (2021). Pre-and within-season crop type classification trained with archival land cover information. *Remote Sensing of Environment*, 264: 112576. https://doi.org/10.1016/j.rse.2021.112576
Jolliffe I (1986). *Principal Component Analysis*. Springer.
Kahan W (1996). IEEE standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776): 11.
Kingma DP, Ba J (2014). Adam: A method for stochastic optimization. arXiv preprint: https://arxiv.org/abs/1412.6980.
Latour A (1998). Existence and stochastic structure of a non-negative integer-valued autoregressive process. *Journal of Time Series Analysis*, 19(4): 439–455. https://doi.org/10.1111/1467-9892.00102
Li H, Reynolds JF (1997). Modeling effects of spatial pattern, drought, and grazing on rates of rangeland degradation: A combined Markov and cellular automation approach – Chapter 10. In: Goodchild MF, Quattrochi DA (Eds.), *Scale in Remote Sensing and GIS*, 211–230.

Logan JA (1981). A structural model of the higher-order Markov process incorporating reversion effects. *The Journal of Mathematical Sociology*, 8(1): 75–89. https://doi.org/10.1080/0022250X.1981.9989916

Mahammad SS, Ramakrishnan R (2003). Geotiff – A standard image file format for GIS applications. *Map India*, 28–31.

McCulloch WS, Pitts W (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4): 115–133. https://doi.org/10.1007/BF02478259

Menneer T, Narayanan A (1995). Quantum-inspired neural networks, *Tech. Rep. R329*.

Miyashita D, Lee EH, Murmann B (2016). Convolutional neural networks using logarithmic data representation. arXiv preprint: https://arxiv.org/abs/1603.01025.

Nair V, Hinton GE (2010). Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning*.

Osman J, Inglada J, Dejoux JF (2015). Assessment of a Markov logic model of crop rotations for early crop mapping. *Computers and Electronics in Agriculture*, 113: 234–243. https://doi.org/10.1016/j.compag.2015.02.015

Panchi L, Shiyong L (2008). Learning algorithm and application of quantum BP neural networks based on universal quantum gates. *Journal of Systems Engineering and Electronics*, 19(1): 167–174. https://doi.org/10.1016/S1004-4132(08)60063-8

Parker DC, Manson SM, Janssen MA, Hoffmann MJ, Deadman P (2003). Multi-agent systems for the simulation of land-use and land-cover change: A review. *Annals of the Association of American Geographers*, 93(2): 314–337. https://doi.org/10.1111/1467-8306.9302004

Pegram G (1980). An autoregressive model for multilag Markov chains. *Journal of Applied Probability*, 17(2): 350–362. https://doi.org/10.2307/3213025

Raftery AE (1985). A model for high-order Markov chains. *Journal of the Royal Statistical Society, Series B, Methodological*, 47(3): 528–539.

Ritter N, Ruth M, Grissom BB, Galang G, Haller J, Stephenson G, et al. (2000). Geotiff format specification geotiff revision 1.0. *SPOT Image Corp*, 1: 154–172.

Santos ES (1969). Probabilistic Turing machines and computability. *Proceedings of the American Mathematical Society*, 22(3): 704–710. https://doi.org/10.1090/S0002-9939-1969-0249221-4

Sartore L, Boryan CG, Willis P (2022). Developing entropies of predictive cropland data layers for crop survey imputation. In: *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, 1404–1407. IEEE.

Schuld M, Sinayskiy I, Petruccione F (2014). The quest for a quantum neural network. *Quantum Information Processing*, 13(11): 2567–2586. https://doi.org/10.1007/s11128-014-0809-8

Shannon CE (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shor PW (1994). Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134. IEEE.

Tong H (1975). Determination of the order of a Markov chain by Akaike's information criterion. *Journal of Applied Probability*, 12(3): 488–497. https://doi.org/10.2307/3212863

USDA, FSA (2017). Farm Service Agency (FSA) Common Land Unit (CLU) information worksheet. https://www.fsa.usda.gov/Assets/USDA-FSA-Public/usdafiles/APFO/support-documents/pdfs/clu_infosheet_2017_Final.pdf

Wolfram S (1984). Universality and complexity in cellular automata. *Physica D*, 10(1–2): 1–35. https://doi.org/10.1016/0167-2789(84)90245-8

Yaramasu R, Bandaru V, Pnvr K (2020). Pre-season crop type mapping using deep neural

networks. *Computers and Electronics in Agriculture*, 176: 105664. https://doi.org/10.1016/j.compag.2020.105664

Yuen C (1975). A note on base-2 arithmetic logic. *IEEE Transactions on Computers*, 100(3): 325–329. https://doi.org/10.1109/T-C.1975.224216

Zadeh LA (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3: 28–44. https://doi.org/10.1109/TSMC.1973.5408575

Zhang C, Di L, Lin L, Guo L (2019). Machine-learned prediction of annual crop planting in the US Corn Belt based on historical crop planting maps. *Computers and Electronics in Agriculture*, 166: 104989. https://doi.org/10.1016/j.compag.2019.104989

Zhao R, Wang S (2021). A review of quantum neural networks: Methods, models, dilemma. arXiv preprint: https://arxiv.org/abs/2109.01840.