Island Transpeciation: A Co-Evolutionary Neural Architecture Search, applied to country-scale air-quality forecasting

Konstantinos Theodorakos, Oscar Mauricio Agudelo, Joachim Schreurs,
Johan A.K. Suykens, *Fellow Member, IEEE* and Bart De Moor, *Fellow Member, IEEE & SIAM*

***Abstract*—Air pollution causes around 400,000 premature deaths per year in Europe due to Particulate Matter, nitrogen oxides and ground-level ozone pollutants. Multiple-Input Multiple-Output Nonlinear Auto-Regressive eXogenous Deep Neural Networks are frequently used to predict a day before, air-quality pollution incidents, at a country-scale. With complexity and data sizes increasing, finding performant models becomes harder. We propose island transpeciation to optimize hyper-parameters and architectures. Unlike using a single optimizer, island transpeciation combines results from multiple optimizers, to consistently provide excellent performance. Moreover, we show that island transpeciation outperforms random model search and other previous modelling efforts. Island transpeciation is a Neural Architecture Search that uses co-evolution (genes), to combine (transpeciation) populations of incompatible optimizers (species) organized in island formations. In island transpeciation, architecture search is parallelized and utilizes a distributed pool of hardware resources. We have successfully used these techniques to predict next-day ozone concentrations across the Belgian territory.**

***Index Terms*—deep neural networks, neural architecture search, co-evolution, air quality forecasting**

I. Introduction

Preprint submitted April 12, 2021; revised November 10, 2021, February 25, 2022 and May 9, 2022. Acknowledgments: • KU Leuven: Research Fund (projects C16/15/059, C3/19/053, C24/18/022, C3/20/117, C3I-21-00316), Optimization frameworks for deep kernel machines C14/18/068, Industrial Research Fund (Fellowships 13-0260, IOFm/16/004) and several Leuven Research and Development bilateral industrial projects; • Flemish Government Agencies: ○ FWO: EOS Project no G0F6718N (SeLMA), SBO project S005319N, Infrastructure project I013218N, TBM Project T001919N; PhD Grants (SB/1SA1319N), projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant, ○ This research received funding from the Flemish Government (AI Research Program). All authors are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium, ○ EWI: the Flanders AI Research Program, ○ VLAIO: CSBO (HBC.2021.0076) Baekeland PhD (HBC.20192204). • European Commission: European Research Council under the European Union's Horizon 2020 research and innovation programme (ERC Adv. Grant grant agreement No 885682), ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. • Other funding: Foundation 'Kom op tegen Kanker', CM (Christelijke Mutualiteit), Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms). *(Corresponding author: Konstantinos Theodorakos.)*

The authors are with KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium (e-mails: {konstantinos.theodorakos, mauricio.agudelo, joachim.schreurs, johan.suykens, bart.demoor}@esat.kuleuven.be).

AIR pollutants are substances that harm the human health and the environment. They can be released from natural and anthropogenic sources: mining activities, agriculture, waste treatment, industrial processes, energy plants, burning fossil fuels, road transportation, residential activities and natural phenomena. Air pollution was the 5th global risk factor in 2017 [1], by total number of deaths from all causes, ages and both sexes and causes around 400,000 premature deaths per year in Europe [2]. There is also a negative economic impact by reducing productivity through working days lost, increasing health care costs and human life span shortening. The most dangerous pollutants in Europe are Particulate Matter (PM), Nitrogen Oxides (NO_x) and ground-level ozone (O_3). Air pollution forecasting is a prevention measure against the short-term (hours or days) and long-term (years) exposure to harmful substances for vegetation, animals and humans. With accurate forecasting, governments and policy-makers could raise real-time "low air-quality" alerts. With timely warnings, the public minimizes negative health effects by reducing outdoor activities during dangerous times. Our objective is to find the most accurate, country-scale, next-day ozone forecasting model possible.

In [3], the authors forecast daily PM in Belgium, using basic Artificial Neural Networks (ANN). A single model per monitoring station was proposed, with only a few exogenous variables. As a consequence, the numerical accuracy of the predicted concentrations was limited. The authors identified cloud cover, day of the week and wind direction as important exogenous features, which we also include in our proposed model. Ensemble Kalman Filters (EnKF) [4] and Optimal Interpolation (OI) [5], improved the O_3 and PM estimates of the air-quality model AURORA in Belgium. Although clustering cannot offer direct numerical predictions for air-quality stations, it is worth mentioning. Incremental Kernel Spectral Clustering (IKSC) [6] clustered drifting non-stationary time-series of multiple PM monitoring stations over Belgium and three surrounding countries. Their approach captured some spatial shifting patterns on the spread of a pollution episode. In [7], the authors used deep learning to forecast O_3. The exogenous variables were reduced using a decision classification tree. A human-in-the-loop approach was used to tune a few Deep Neural Networks (DNN) hyperparameters manually and to avoid overfitting. As exogenous parameters, other than meteorological variables the authors added chemical analytes. To our knowledge, chemical concentration forecasts are not

easily obtainable and may require the additional dependency of the outputs of (computationally demanding) deterministic air-quality models. The authors used only 2 years of training data, with 80% training, 20% testing split, without any form of cross-validation or automated architecture search. Finally, this approach seems to work only for single monitoring station predictions.

Multiple-Input Multiple-Output (MIMO), Nonlinear Auto Regressive eXogenous (NARX) DNN for air-quality forecasting is a modelling architecture that can predict next-day O_3 concentrations, at a country scale. In this work, we used real-world, publicly available data: daily ozone concentration time-series (1990 to 2018) from 46 O_3 Belgian monitoring stations retrieved from the European Environmental Agency (EEA) [8], which we combined with 51 weather variables [9] acquired from the surface-level ERA-interim European Centre for Medium-Range Weather Forecasts (ECMWF) database. The proposed DNNs successfully predicted one day before, an "inform-public" O_3 alert level in Belgium at critical times. With complexity and data sizes increasing, finding

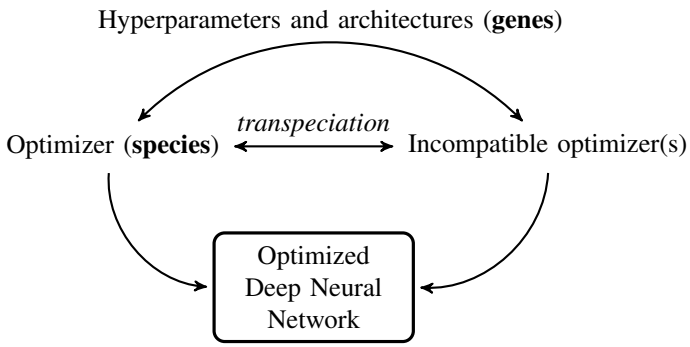


Figure 1. Island transpeciation is a Neural Architecture Search technique. Given hyperparameters and architectures (encoded as genes), it uses co-evolution (transpeciation) to combine populations of incompatible optimizers (species), in order to find highly performant Deep Neural Network (DNN) models.

performant models becomes harder. To improve the forecasting performance of DNNs, we developed *island transpeciation* [10] (Fig. 1). Island transpeciation can combine results from multiple optimizers, to consistently provide good performance. Island transpeciation is a Neural Architecture Search (NAS) technique that uses co-evolution (genes) to combine (transpeciation) populations of incompatible optimizers (species), in order to find highly performant DNN models. **Contributions:**

- Definition of a new operator for evolutionary algorithms: *transpeciation*.
- A new type of automated parallel and distributed NAS: Island transpeciation. Heterogeneous networked computing resources can cooperate with fault-tolerance. Computing hardware can be added/removed without interrupting the neural architecture search (also known as hot-plugging).
- MIMO NARX DNN: A prototype for country-scale air quality forecasting, using a single model.
- Deep learning model configuration suggestions in the context of ozone forecasting.

This document is divided into 5 sections. Section II provides a brief introduction to air-quality standards, DNNs and Neural Architecture Search. Section III discusses the island transpeciation neural architecture search. Section IV describes the MIMO-NARX DNN model prototype. Section V shows the performance of the air quality forecasting models and the effectiveness of island transpeciation. Section VI, contains insights and conclusions. The supplemental material contains implementation details, the settings of the global and local optimizers used in the experiments, the hyperparameters and optimizers of the machine learning models that we compared against, formulas and illustrations of the mathematical benchmarks that we used, the exogenous weather variables, and finally links to source code/data repositories.

II. Background

A. Air-quality standards

Ozone (O_3) is an inorganic molecule, a less stable allotrope of oxygen. It has industrial and consumer applications as an oxidant, but damages the mucous and respiratory tissues of humans and animals [11]. Exposure to O_3 with a concentration of 1 part per million (ppm) can affect the respiratory system. Exposure to 15 to 20 ppm, for 2 or more hours can be life-threatening. For the long term protection of human health and vegetation, specific O_3 concentration thresholds were set by the European Union [12]:

1) Background: 60 $\mu g/m^3$.
2) Healthy limit: 120 $\mu g/m^3$.
3) Public informing (>1 station): 180 $\mu g/m^3$.
4) Alert: 240 $\mu g/m^3$.

We used real-world, publicly available data: Time-series (1990 to 2018) from 46 O_3 Belgian monitoring stations, retrieved from the European Environmental Agency (EEA) [8]. To validate our approach, we also used data from two additional countries (from 2014 to 2019): Malta (3 stations) and Cyprus (3 stations). Data are sampled as a maximum daily 8-hour mean by the following monitoring station types:

1) Urban: Protection of human health. Range: a few km^2. Location: residential areas of cities.
2) Suburban: Protection of human health and vegetation. Range: tens of km^2. Location: city outskirts.
3) Rural: Protection of human health and vegetation. Range: sub-regional levels (a few km^2). Location: small settlements, crops, forests and natural ecosystems.
4) Background-rural: Protection of vegetation and human health. Range: regional, national or continental (1.000 up to 10.000 km^2). Location: crops, forests, natural ecosystems, very low population regions.

B. Hyperparameter optimization of neural networks

Recurrent Neural Networks (RNN) [13] is a type of ANN that is commonly used to model time series. RNNs exhibit temporal sequential behavior where memories are retained and recalled as stable entities collectively. An extension of RNN is the *Bidirectional* Recurrent Neural Networks (BRNN) [14], which use sequential data in both forward/backward

time directions without the requirement of the input data length to be fixed. *Long Short-Term Memory (LSTM)* [15] cells store information over extended time intervals, without losing shorter time predictive capabilities. *Gated Recurrent Units (GRU)* [16] are similar to the LSTM, with less trainable parameters. LSTMs with a forget gate, are expressed [15] as:

$$\begin{aligned}
\mathbf{f}_t &= \sigma_g(\mathbf{W_f}\mathbf{x}_t + \mathbf{U_f}\mathbf{h}_{t-1} + \mathbf{b_f}) \\
\mathbf{i}_t &= \sigma_g(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{h}_{t-1} + \mathbf{b_i}) \\
\mathbf{o}_t &= \sigma_g(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{h}_{t-1} + \mathbf{b_o}) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W_c}\mathbf{x}_t + \mathbf{U_c}\mathbf{h}_{t-1} + \mathbf{b_c}) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where d is the number of input features, h is the number of hidden units, $\circ$ the Hadamard product. $\mathbf{x}_t \in \mathbb{R}^d$ is the input vector to the LSTM unit at time t, $\mathbf{f}_t \in \mathbb{R}^h$ is the forget gate's activation vector, $\mathbf{i}_t \in \mathbb{R}^h$ is the input gate activation vector, $\mathbf{o}_t \in \mathbb{R}^h$ is the output gate activation vector, $\mathbf{c}_t \in \mathbb{R}^h$ is the cell state vector and $\mathbf{h}_t \in \mathbb{R}^h$ is the hidden state vector. The $\mathbf{W} \in \mathbb{R}^{h\times d}$, $\mathbf{U} \in \mathbb{R}^{h\times h}$ weight matrices and the bias vectors $\mathbf{b} \in \mathbb{R}^h$, are learned during training. σ_g is a sigmoid function and σ_c, σ_h are hyperbolic tangent functions. LSTMs fall under a more general class of *Nonlinear Auto-Regressive eXogenous (NARX)* models [17]. NARX is a predictive mathematical formulation with output variables depending on previous values as well as exogenous variables, along with a stochastic (random) term:

$$\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(t-1), ..., \mathbf{y}(t-l_y), \mathbf{u}(t-1), ..., \mathbf{u}(t-l_u)) + \mathbf{e}(t) \tag{2}$$

where $\mathbf{y}(t) \in \mathbb{R}^n$ is the output vector at time t (e.g., O_3 concentrations), $\mathbf{u}(t) \in \mathbb{R}^m$ is the vector of exogenous input variables (e.g., temperature, total cloud cover, etc.). l_y and l_u are the numbers of lags for $\mathbf{y}$ and $\mathbf{u}$ respectively. $\mathbf{e}(t) \in \mathbb{R}^n$ is the prediction error (white noise process assumed) and $\mathbf{F}(.)$ is a nonlinear vector function (ANN, polynomial function, etc). We focused on *Multiple-Input Multiple-Output (MIMO)* models, because they allow us to use multiple input time-series of air quality monitoring stations, to predict the outcomes of multiple stations. With *Multiple-Input Single-Output (MISO)*, we forecast the output of a single measuring station.

Due to the inherent difficulty of modern day prediction problems, such as the air quality forecasting that we focus on this paper, neural network models typically have an abundance of hyperparameters to tune. We used a combination of global and local search methods to explore and optimize these parameters. We here give an overview of the global optimizers we used: *Particle Swarm Optimization* (PSO) [18] solves problems iteratively, using populations of moving candidate solutions (particles), located within a parameter space. Particles (swarm) influence each-other based on their fitness, position, velocity etc. *Genetic Algorithms* (GA) [19] is a class of Evolutionary Algorithms (EA) [20] that generate candidate solutions via natural selection and biology-inspired operators: mutation, crossover and selection. *Differential Evolution* (DE) [21] is an EA that solves non-differentiable optimization problems, by combining multiple elite candidate solutions. *Bayesian Optimization* (BO) [22] defines prior and posterior distributions (Gaussian process) over objective functions, using exploration/exploitation trade-offs on bounded parameter spaces. *Random Search* (RS) [23] samples from random distributions to solve black-box optimization problems. Given the best hyperparameters found from the global optimizers, we extend the model architecture search with the following *Local Search (LS)* optimizers: *L-BFGS-B* [24] is a limited-memory, quasi-Newton algorithm that solves large nonlinear bounded optimisation problems. It approximates the Hessian of the objective function via gradient projection and a limited-memory matrix. *Sequential Least-Squares Quadratic Programming (SLSQP)* [25] is a gradient-based optimizer that applies first-order (affine) approximations on constraints (variable bounds in our case) and successive second-order (quadratic/least-squares) approximations of the objective function. The *TruNCated Newton (TNC)* [26] is a preconditioned conjugate-gradient method, where the direction of search at each Newton method iteration is defined as the solution of a quadratic sub-problem (truncation). Finally, the *trust-constraint* [27] reduces objective function calculation costs by building arbitrary trust regions around an initial guess solution. Instead of calculating the objective function at multiple (or all possible points), it guesses the shape of the objective function.

Neural Architecture Search (NAS) automatically architects Neural Network designs [28], [29]. Due to the inherent complexity of DNNs and the tasks at hand, empirically architecting DNN can be tedious, time-consuming and ineffective. NAS is overlapping with the fields of meta-learning [30] and hyper-parameter optimization and has three main components: (1) *Search space*: The bounded or unbounded range of architecture parameters. It can be prone to human bias. Careful treatment can significantly reduce the total NAS search space and execution time [31]. (2) *Search strategy*: The method to perform exploration and exploitation of the architecture search space. (3) *Performance estimation strategy*: The metric that can foresee the future performance of the trained ANN on unseen data and future tasks. In our case, DNN architecture search is performed in conjunction with hyperparameter tuning.

The primary motivation for developing island transpeciation is the fact that DNN models can be characterized by tenths up to hundreds of hyperparameters and architectures, with varying levels of sensitivity. Manually tweaking the available options does not guarantee optimal model performance [32]. With information theory applied to evolutionary algorithms [33], it was theoretically established that there are no free lunch theorems for optimization. In other words, a specific optimizer that performs well on specific datasets and use cases, may under-perform in others. There is also evidence that, combining multiple optimization algorithms leads to improvements in global model performance, compared to a homogeneous/single-optimizer case [34]. In this paper, we argue that for fast convergence it is helpful to combine different types of optimizers for NAS as well. The introduction of island transpeciation in NAS, allows the combination of DNN architectures generated from vastly different global optimizers. This ensures consistently good performance over different initializations, parameter spaces or difficulty of the problem.

III. Main result: Island Transpeciation

Speciation is the evolutionary process with which populations evolve to become distinct species [35]. Island transpeciation is a global search technique that uses evolution to combine incompatible optimizers, in order to find highly performant DNN architectures (Fig. 2). Artificial populations of optimizers can be combined into panmictic structures of complex system networks called *island or multi-population models* [36]. Every optimizer maintains a number of candidate solutions in their *internal representation* level, in an *island*. In Fig. 2 we have three islands of the following iterative optimizers: PSO, GA and BO. PSO and GA islands have a population of five (internal candidate solutions), whereas BO only one (distribution of parameters). An optimizer that can not directly use candidate solutions from other islands, is considered a unique *species*. E.g., a PSO algorithm (which contains individuals of the species "particle") can not directly accept and utilize candidate solutions from the "individual" species of a GA.

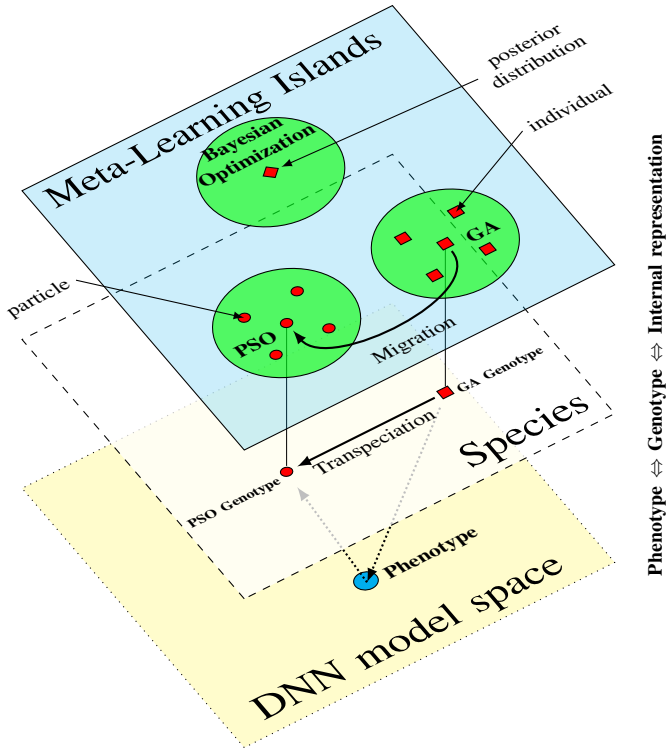


Figure 2. Island transpeciation is a co-evolutionary Neural Architecture Search (NAS) that optimizes DNN models (phenotype). Optimizers (top layer: meta-learning islands) like Bayesian Optimization, Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) can co-evolve DNN models (bottom layer: DNN model space). Cooperation and competition between incompatible optimizers is achieved via migration (transpeciation operator) of solutions between optimizers (mid layer: species) using: *genotype-to-internal representation* transformations.

Transpeciation is an evolutionary operator that allows *internal representation* to (globally compatible) *genotype* transformations. In other words, to enable transferring of candidate solutions between incompatible optimizers, we project solutions from the internal algorithm format to a commonly compatible form. *This way, we can achieve concurrent cooperation between vastly different optimization algorithms.* The *transpeciation operator* is optimizer/implementation-dependent. For example (Fig. 2), transpeciation can transform an individual (from the GA species) into a particle (to the PSO species) and allow GA and PSO to co-operate seamlessly. An individual GA solution can migrate to a PSO island, after it is transformed (transpeciated) into the particle species via a common *genotype*. The bottom layer represents the final form of an optimized and trained DNN model, expressed as *phenotype*.

A. Generalized Island Model formal extension

The Generalized Island Model (GIM) [34] is a paradigm that allows genetic algorithms to run in parallel over multiple processors. GIM improves the performance of genetic algorithm subpopulations via candidate solution migrations. From GIM, we have *Archipelago* $\mathbb{A}$ which is the tuple:

$$\mathbb{A} =< \mathbb{I}, \mathscr{T} > \tag{3}$$

where i is the island identification with $i = 1, 2, ...n$, $\mathbb{I} = \{I_1, I_2, ..., I_n\}$ is the set of islands and $\mathscr{T}$ is the migration topology (a multi-dimensional grid in our case). Every island I_i is a tuple:

$$I_i =< \mathscr{A}_i, P_i, \mathscr{P}_i, \mathscr{R}_i > \tag{4}$$

where $\mathscr{A}_i$ is the optimization algorithm (e.g. PSO, GA, ...), P_i is the candidate solution population, μ_i is the migration interval, $\mathscr{P}_i$ is the migration-selection policy and $\mathscr{R}_i$ is the migration-replacement policy of island i.

Algorithm 1: Island I_i with transpeciation operator $\mathbb{T}_i$

1 initialize P // Generate model architectures
2 **while** *NOT(stop criteria)* **do**
3 $P' \leftarrow \mathscr{A}_i(P, \mu_i)$ // Update architectures
4 $\mathbb{M} \leftarrow \mathbb{T}_i(\mathscr{P}_i(P'))$ // Transpeciation
5 Send $\mathbb{M}$ to islands adjacent to $I_i \in \mathscr{T}$
6 Let $\mathbb{M}'$ be the set of solutions received from adjacent islands
7 $P'' \leftarrow \mathscr{R}(P', \mathbb{T}_i'(\mathbb{M}'))$ // Invert transpeciation and replace worst models
8 $P \leftarrow P''$ // Update island population
9 **end**

First, an island generates its initial model architecture population P (Line 1 of Algorithm 1). The optimization algorithm $\mathscr{A}_i$ updates the architectures and after the migration interval μ_i generates the population P' (Line 3, $\mu_i = 5$ iterations for our case). The migration-selection policy $\mathscr{P}_i$ (Line 4) determines which deme $\mathbb{M}$ will be migrated to the adjacent islands. The *deme* $\mathbb{M}$ is a globally compatible sub-population, generated by the island i. Our extension on the original GIM [34] formalism is the transpeciation operator $\mathbb{T}$ (Lines 4, 7). $\mathbb{M}$ is created by the *internal representation* population of the island i, using the transpeciation operator $\mathbb{T}_i$:

$$\mathbb{M} = \mathbb{T}_i(internal\_representation_i) \tag{5}$$

where $\mathbb{T}_i$ is the *transpeciation operator* for island i, and $\mathbb{M}$ contains globally compatible candidate solutions that will be sent to adjacent islands (Line 5). The internal representation of an algorithm (or species) could be e.g.: a "particle" for PSO island, or a "distribution" for a BO island. Typically, we would only be able to combine candidates from the same species, with the same number and type of parameters (in other words, we would only accept migrating particles for PSO). However, transpeciation allows the back-and-forth cooperation of incompatible optimizers. After we have sent the deme $\mathbb{M}$ (Line 5) and received the deme $\mathbb{M}'$ (Line 6), the operator $\mathbb{T}'$ (Line 7) performs the reverse transpeciation operation. Island i converts the globally compatible deme $\mathbb{M}'$ that was received from the adjacent islands, into the $internal\_representation_i$ of our island:

$$internal\_representation_i = \mathbb{T}'_i(\mathbb{M}') \quad (6)$$

where $\mathbb{M}'$ is the *deme* received from islands adjacent to I_i and $\mathbb{T}'_i$ is the *Transpeciation' operator* of island i, for reverse transformations (global to internal representations). E.g., a PSO can convert candidate solutions coming from a GA or even a BO island into a particle. A lossy transpeciation case is when we convert a particle from the PSO species into an individual of the DE species: we retain and convert all the particle position parameters. However, we lose the internal parameters regarding particle velocity or acceleration. For an optimization algorithm like random search there is no inverse transpeciation operation, because this algorithm only generates demes, it does not accept demes internally. Finally, using the migration-replacement policy $\mathscr{R}$, the worst internal candidates of P' are replaced (Line 7) and P'' becomes the new island population of candidate solutions (Line 8). Practically, the transpeciation operator rescales hyperparameter bounds, from the internal representation format of each optimizer to the globally compatible genotype bounds (see Table I). The transpeciation operation with the Bayesian optimizer [37] is a special case: performs a "maximize" function call, to get the Maximum A-Posteriori (MAP) hyperparameter estimation. The inverse transpeciation operation performs a "probe" function call, to "guide/suggest" solutions (coming from the other islands) to the Bayesian optimizer. The supplemental material (Appendix A) contains information about the parallel and distributed implementation, the transpeciation operator in practice and a flowchart of island transpeciation.

Table I
ISLAND SPECIES (OPTIMIZERS) THAT COOPERATE IN ISLAND TRANSPECIATION NAS FOR GLOBAL AND LOCAL SEARCH.

Island Species		
Optimizer	**Internal representation** (of hyperparameters)	**Description**
Global Optimizers		
Random Search (RS)	$[0, 1]$	Uniform distribution sampling, with island population of one [23].
Particle Swarm Optimization (PSO)	position and velocity	Multi-dimensional representation of the parameters as real valued positions, along with rate of change of the positions [38].
Bayesian Optimization (BO)	parameter distribution	Candidates are sampled from a bounded posterior distribution of parameters [37].
Genetic Algorithms (GA)	energy $\in [0, 1]$	Variants used: simple, μ, $\mu+\lambda$, Covariance Matrix Adaptation (CMA) [39], [40].
Differential Evolution (DE)	energy $\in [0, 1]$	Similar to GA [41].
Local (bounded) optimizers		
L-BFGS-B	-	First derivative Quasi-Newton method [24].
SLSQP	-	Sequential Least Squares Quadratic Programming [25].
TNC	-	Truncated Newton algorithm [26].
Trust-constr	-	Trust-region for unconstrained optimization solving quadratic subproblems [27].

IV. METHODS USED FOR OZONE FORECASTING

A. Model

MIMO NARX DNN: Single-station forecasting of air quality has been tried in the past with LSTM [7]. For multi-station predictions, we have the MIMO NARX DNN (Fig. 3):

$$\hat{\mathbf{y}}(t) = \mathbf{F}(\mathbf{y}(t-1), ..., \mathbf{y}(t-l_y), \mathbf{u}(t-1)..., \mathbf{u}(t-l_u)) \quad (7)$$

where t is the time-step (in days), n is the number of O_3 monitoring stations (46 for Belgium), m is the number of exogenous variables (51 weather/environmental variables and 8 calendar cyclical features), $\hat{\mathbf{y}}(t) \in \mathbb{R}^n$ is the output vector forecast at time t (O_3 concentration at each monitoring station) with maximum lags l_y, $\mathbf{u}(t) \in \mathbb{R}^m$ the exogenous variables vector with maximum lags l_u and $\mathbf{F}(.)$ is the nonlinear vector function (3-layered DNN in our case).

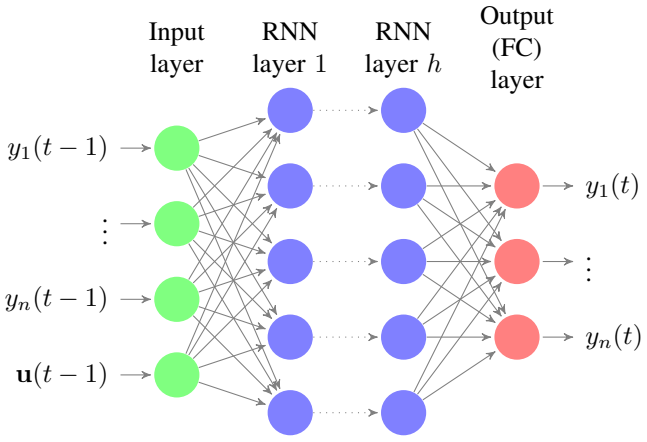


Figure 3. Diagram of Multiple-Input Multiple-Output (MIMO) Nonlinear AutoRegressive eXogenous (NARX), for next-day air-quality forecasting: $\hat{\mathbf{y}}(t) = \mathbf{F}(\mathbf{y}(t-1), \mathbf{u}(t-1))$. Predict next-day (time t) air-quality for n measuring station outputs $\mathbf{y}$, using exogenous variables $\mathbf{u}(t-1)$ (e.g., weather forecasts) and previous-day measurements $\mathbf{y}(t-1)$. Recurrent Neural Networks (RNN) act as h hidden layers and the final Fully Connected (FC) layer provides the output numerical predictions.

We use only one-day lags because we did not notice any accuracy improvements experimentally by increasing them further. In addition, we had an increment (almost double or more per extra delay) in dataset size and model training times. We used daily time-step instead of hourly for two

Table II
DISCRETE HYPERPARAMETER BOUNDS (ARCHITECTURE SEARCH GENES IN THE COMMON GENOTYPE).

Hyperparameter	Values
Layer	
Simple or Bidirectional	LSTM, RNN, GRU.
Auxiliary	Gaussian Noise, Batch Normalization.
Weight Initializer	Zeros, uniform (random, lecun, he), ones, normal (random, lecun, he, glorot, truncated).
L1/L2 weight decay	Activity regularizers, bias regularizers, kernel regularizers.
Deep Neural Network	
Optimizer	Adam [46], nadam [47], amsgrad [48], adagrad [49], adadelta [50].

Table III
NUMERICAL HYPERPARAMETER BOUNDS (FOR EACH DNN LAYER).

Hyperparameter	Min	Max	Description
batch size	7	31	Samples per model update.
epoch size	350	600	Training data pass count.
units	64	512	Neuron count per layer.
dropout	0.01	0.25	Unit fraction to leave-out.
recurrent dropout	0.01	0.25	Recurrent unit fraction to leave-out.
gaussian noise STD	0.1	0.5	Noise distribution standard deviation.
L1, L2 regularizers	0.0	0.01	L1 and L2 layer optimization penalty.

reasons: (1) our exogenous weather data had a minimum time-step of 3 hours and (2) because the environmental agencies like the Belgian Interregional Environment Agency (IRCEL - CELINE) usually inform the public with 24-hour means.

B. Search space

Representation: For the architecture and hyperparameter search space representation, upper and lower bound arrays were used as constraints. The arrays are of fixed length and contain integer (nominal) or float value ranges. The genotype of a model is a value set, sampled from this bounded space. The candidate model phenotypes are the trained sequential DNN models, using the architecture defined by the genotype. Tensorflow Keras [42] was the model implementation framework.

The candidate models are fixed to three base recurrent layers (see Table II). Architecture search genes are expressed as bounded integer values. They represent: (1) the type of the core recurrent layer (LSTM [15], simple RNN [13] or GRU [16]) and (2) the recurrent layer sequence causality (simple or bidirectional [14]). The auxiliary/utility layers are placed between the base layers. Auxiliary placements are determined by the evolutionary search. Auxiliary layer types can be: (1) Batch Normalization [43] and (2) Gaussian Noise [44]. Layer weight initializer genes (Table II), determine the sampling distribution of the initial neuron weights. *Optimizers* are the algorithms that guide the weight training of ANN [45]. We used all the available optimizers that work with recurrent DNN architectures [42] (see Appendix B of supplemental material for optimizer details). Note that in this paper an ample amount of hyperparameters are considered in the search space. The reasoning behind this is twofold: First, to demonstrate the proposed solution in a large search space setting. Second, restricting the hyperparameter search too much could result in a sub-optimal model. In practice one could reduce the search space by expert knowledge about the problem at hand.

The continuous floating-point value bounded parameters that we optimized, are shown in Table III. We chose the hyperparameter bounds both empirically and from suggested ranges on previous works [51], [52], [53]. Mini-batches increase DNN model generalization [53] but we have to accept that model training times will increase. We chose a minimum batch size of 7 (days), because a week time-span may exhibit cyclical calendar patterns. The epoch size parameter acts mostly as a max range in our setup. We empirically chose large ranges, since we also use early (model training) stopping [42] to avoid overfitting. Due to model memory constraints, we chose to have at maximum three RNN layers with at most 512 units each. In terms of dropout and recurrent dropout, suggested values range from 0.1 up to 0.5 [51], [52]. However, the upper ranges are mostly destined for very large models, so we empirically chose a conservative range of max 0.25. Gaussian noise [54] and regularizers [52] improve model generalization. Again, we chose noise bounds empirically.

C. Search strategy

In the ring topology, each island is a subpopulation that connects virtually and directly to another subpopulation using one dimension. Grid island or cellular topologies [36] of two dimensions can be folded into a torus. Due to high graph connectivity, a torus interconnect architecture [55] can have vast amounts of optimization islands, while requiring very few hops to propagate messages through the whole structure (Fig. 4). Solution diversity is maintained for multiple generations and the most performant models will not rapidly dominate the island structure. The neighborhoods can be expanded to four dimensions (hypercubes) or more.

The islands iterate asynchronously. Training and evaluation times depend on the count of the trainable parameters of the model. In addition, every optimization algorithm-island does not iterate at the same pace. Removing any implicit or explicit synchronization barriers after model training and evaluation, allows the evolved models to complete not only in accuracy but also training speed. The removal of synchronization barriers allows for the usage of the competing consumers pattern, which is discussed in the supplemental material (Appendix A). Migrations of candidate solutions can still occur but after a predefined count of island iterations (migration interval μ). In our case, each island receives migrating candidate solutions (demes) from its neighbors every $\mu = 5$ fitness evaluations. An n-dimensional grid yields ν neighbors per island (for our case we have a 3-dimensional grid, that yields $\nu = 3$ neighbors). But which migrating candidate to accept? We used *Linear Ranking (LR) selection* [56]:

1) Rank ν neighbors by fitness (worst rank: i = 0).

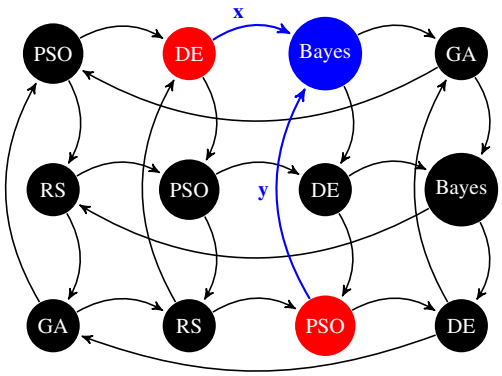


Figure 4. Illustration of an adjacent island structure in two dimensional island transpeciation. In the example above, models are sent using a single-direction Von Neumann neighborhood (radius 1) of a two-dimensional torus interconnect architecture. The *Bayes* optimization island receives adjacent demes (candidate models) from the *DE* (x dimension) and the *PSO* (y dimension) islands. By constraining the migration direction and limiting the amount of candidate models that we send/receive, we retain high model diversity in the whole island structure.

2) Adjust selection pressure $s \in (1, 2]$:
   - Small $s \Rightarrow \approx$uniform random.
   - Large $s \Rightarrow$ best ranks have higher selection probability.
3) Random choice given probabilities:

$$P_{LinearRank}(i) = \frac{(2-s)}{\nu} + \frac{2i(s-1)}{\nu(\nu-1)} \quad (8)$$

For GA, DE and PSO islands, we applied the *replace worst* strategy [57]. This fitness-based replacement strategy is a survivor selection mechanism, that ranks the DNN architecture candidates of an island by fitness. The worst "internal representation" candidate is replaced by the migrating candidate from the adjacent islands (e.g., for PSO, the particle with the worst fitness is replaced). We opted for this deterministic approach, because each island retains a relatively small number of candidates (e.g., five particles for PSO islands). The negative effect of the replace worst strategy, is that there may be premature convergence and genotype duplicates inside a population. The positive effect is that the mean fitness of an island's population is rapidly increased.

Phenotypic plasticity is the adaptation of an individual to a specific environment [58]. These adaptations are temporary, act only during a lifetime and do not propagate to offspring. To achieve a similar effect on DNN NAS, we applied random *phenotypic mutations* [59]. Naturally, seemingly similar DNNs can have vastly different performance due to neuron weight training randomness. With phenotypic mutations we can introduce additional DNN model variations, not guided by evolution or the common genotype. The bounded architecture search space is expanded from 25 variables to 53: we included layer-based regularization hyperparameters. These regularizers apply additional penalties to the loss function. L1 and L2 norm weight decay penalty is randomly applied (sampled from a uniform distribution) during the neuron weight training (Table II), with: activity, bias and kernel regularizers. Linear Ranking selection in combination with the replace worst (candidate model) strategy and random phenotypic (neuron weight) mutations, allow for iterative increases in the average performance of solutions, while avoiding the negative side-effects of the evolutionary elitism (having only one architecture dominate the archipelago of candidates).

D. Performance estimation strategy

To compare forecasting model performance in NAS, we used two metrics. First, the symmetric Mean Absolute Percentage Error (sMAPE) [60]:

$$sMAPE = \frac{2}{T}\sum_{t}^{T}\frac{|y(t) - \hat{y}(t)|}{|y(t)| + |\hat{y}(t)|} * 100(\%) \quad (9)$$

where t is timestep of maximum T, $\hat{y}(t)$ the forecast and $y(t)$ the observation at time t. Second, the Mean Absolute Scaled Error (MASE) [60]:

$$MASE = \frac{1}{J}\sum_{j}^{J}|\frac{y(j) - \hat{y}(j)}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y(t) - y(t-m)|}| \quad (10)$$

where j is the forecast period out of total forecasts J and m is the seasonal period (with $m = 1$ for non-seasonal as the Naive-1 forecast error).

V. Results

A. Description of the data and the experiments

Real-world, publicly available data are used in all experiments. Ozone (O_3 max daily 8-hour rolling mean concentrations) data were gathered from the European Environmental Agency (EEA) [8]. Data was augmented with 51 weather variables (see Appendix D of Supplemental material), acquired from the surface-level ERA-interim ECMWF public database [9], [3]. The first set of air quality data from EEA, start from 1990 and end up in 2011. The dataset contains daily, hourly and yearly values. For the years 2012 to 2018 the data provided are only hourly, which require pre-processing in order to be converted to daily. In addition, all the time-series were standardized on the training phase and unstandardized for the predictions. For the initial MISO models, we used as input 51 exogenous weather variables and a single monitoring station time-series with 1-lag (1 day before). The output is the forecast for the current day. The MIMO models use the same inputs, but the output is the current day forecast of all the time-series. All the data available for Belgium are used. For the MIMO and MISO models predicting 2018, we added 8 calendar cyclical features ($sine$ and $cosine$ of: month, day of week/year, week of year). Fig. 5 shows the geo-locations of all the 46 O_3 stations used in the experiments: urban, suburban, rural and rural background. The datasets have two significant limitations. First, the exogenous variables are only sampled by 1 location in the middle of Belgium. The reason is data scaling issues. Adding up to 46 stations x 51 variables would significantly increase the model training times of the experiments. In future work we will use local weather information, along with feature reduction from hyperparameter tuning and expert knowledge. Second, only 1-day lag was used in all cases. We tested adding one extra lag-day, but we did not

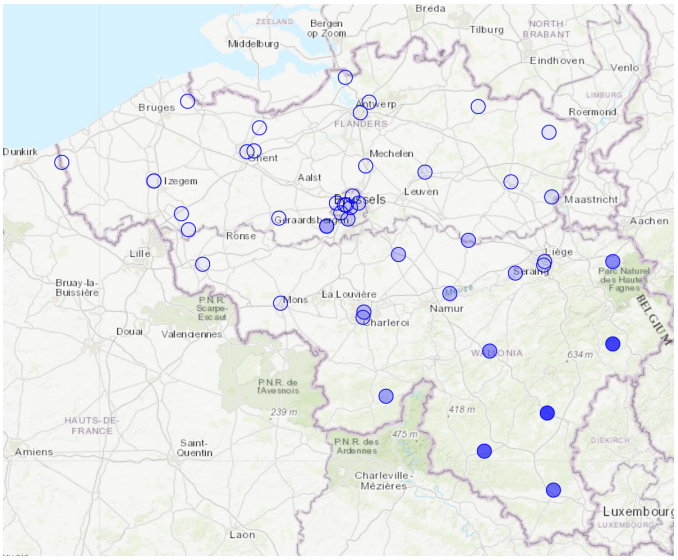


Figure 5. All the available 46 O_3 monitoring station locations that we used for country-scale (Belgium) air-quality forecasting. The full time-series dataset (1990 to 2018) is publicly accessible from the European Environmental Agency (EEA) [8].

notice any increment in accuracy, while we almost doubled the amount of training data. The Partial AutoCorrelation Function (PACF) for O_3 showed high partial autocorrelation for 1-day lag (0.82) but much lower (0.09) for 2-day lags or more.

For pre-processing, ECMWF performs 4D-variational data assimilation on the weather data. EEA flags air-quality measurements as "verified" if they pass full quality assurance and quality control and "valid" if values are above detection limit and not due to station maintenance or calibration. We removed any invalid/unverified data and filled them with values from the most correlated stations time-series of the same timestep. For any remaining missing values, we applied linear interpolation (forward/backward fill). Hourly values, were aggregated as daily max of 8-hour rolling mean. Finally, features are standardized. To avoid model overfitting and allow for better generalization for time-series predictions, we applied Time-series Cross-Validation [61]. This technique does not shuffle the data sequence and avoids information leakage from future steps. The mean MSE of a 5-fold cross-validation determines whether a model is performant and generalizes well. The "test" data are held-out from the start, never used in model training. Next to that, we employ *Early Stopping* and *Learning-Rate reduction on plateau* [42]. In case there are no model fitness improvements after a predetermined amount of iterations, the global island search stops. In our experiments, we disabled the automatic stopping criteria, in order to examine the co-evolution for around 350 to 500 iterations.

B. Results for MISO models

1) Island transpeciation versus other optimizers: With the GIM model [34], it was shown that combinations of different global hyperparameter optimizers in island structures, can outperform single optimizers. We tested island transpeciation against random search in two use cases: (1) on mathematical benchmarks and (2) on a subset of the real-world data. The first experiment also serves the role of an ablation study, to examine the contribution of each component of island transpeciation on optimization. We used four mathematical functions [39] for optimization performance comparisons: (1) the Rosenbrock, (2) the Ackley, (3) the Rastrigin function and (4) the Schaffer function (see Appendix E of the supplemental material for illustrations and formulas). Rosenbrock (the "valley" function) is mostly used for gradient-based optimizers. It has a smooth surface and its global minimum lies in a banana-shaped narrow valley. Ackley is a non-convex function with central global minima, while having several local minima. It can easily trap hill-climbing algorithms in sub-optimal minima. Rastrigin has a large number of local minima and it is a non-convex, multimodal non-linear function. Schaffer is a smooth function, with central global minima and has a wave-like pattern. For the experiments we used problems of 50 dimensions and compared the following methods: (1) LS only (5 islands), (2) RS only and (3) island transpeciation with 8, 16 and 24 islands, with either non-communicating (no model co-evolution) or fully communicating islands (island transpeciation). The supplemental material (Appendix B) contains detailed parameters for all the global and local islands that we used. In almost

Table IV
OPTIMIZERS VERSUS MINIMIZATION MATHEMATICAL BENCHMARKS (50 DIMENSIONS, NUMBER INDICATES FOUND MINIMA, MEAN ± STANDARD DEVIATION, 500 ITERATIONS PER CASE, 30 EXPERIMENT RUNS). THE WELCH'S t-TEST OF UNEQUAL VARIANCES [62] IS POSITIVE, IF THE ISLAND CONFIGURATION HAS LESS ERROR THAN RANDOM SEARCH (FIRST NUMBER IN PARENTHESIS) OR THE BENCHMARK BEST (SECOND NUMBER). STATISTICAL SIGNIFICANCE: *** FOR P-VALUE < .001, ** FOR P-VALUE < .01 AND * FOR P-VALUE < .05.

Optimizer	**Rosenbrock** (smooth, valley-shaped)	**Ackley** (non-smooth, central global minima)	**Rastrigin** (non-smooth, many local minima)	**Schaffer** (smooth, wave-like, central global minima)
Local Search (5 islands)	$\mathbf{8748 \pm 904}$	19.75 ± 0.15	686 ± 60	533 ± 23
Random Search	43525 ± 4593	19.64 ± 0.22	705 ± 25	520 ± 14
8 islands	29410 ± 9868 (7.1^{***}, -11.42^{***})	19.09 ± 0.38 (6.92^{***}, -0.92)	649 ± 71 (4.11^{***}, -1.25)	$\mathbf{462 \pm 57}$ ($\mathbf{5.47}^{***}$)
8 islands transpeciation	17605 ± 10655 (12.24^{***}, -4.54^{***})	$\mathbf{19 \pm 0.43}$ ($\mathbf{7.27}^{***}$)	$\mathbf{626 \pm 73}$ ($\mathbf{5.66}^{***}$)	474 ± 46 (5.24^{***}, -0.9)
16 islands	28399 ± 9488 (7.86^{***}, -11.29^{***})	19.46 ± 0.26 (2.92^{**}, -5.01^{**})	665 ± 58 (3.49^{**}, -2.31^{*})	487 ± 36 (4.65^{***}, -2.1^{*})
16 islands transpeciation	27462 ± 8231 (9.33^{***}, -12.38^{***})	19.1 ± 0.39 (6.64^{***}, -0.94)	672 ± 46 (3.45^{**}, -2.94^{**})	497 ± 35 (3.39^{**}, -2.86^{**})
24 islands	33565 ± 6694 (6.72^{***}, -20.12^{***})	19.36 ± 0.24 (4.78^{***}, -3.96^{***})	691 ± 35 (1.78, -4.44^{***})	506 ± 27 (2.48^{*}, -3.87^{***})
24 islands transpeciation	30577 ± 7338 (8.19^{***}, -16.17^{***})	19.39 ± 0.3 (3.72^{***}, -4.11^{***})	687 ± 40 (2.19^{*}, -4.01^{***})	517 ± 19 (0.8, -5.04^{***})

Table V
ISLAND CONFIGURATIONS VERSUS 50 DIMENSION MINIMIZATION MATHEMATICAL BENCHMARKS (NUMBER INDICATES FOUND MINIMA, MEAN ± STANDARD DEVIATION, 20 ISLANDS, 500 ITERATIONS PER CASE, 30 EXPERIMENT RUNS). THE WELCH'S *t*-TEST[62] IS POSITIVE, IF THE ISLAND CONFIGURATION HAS LESS ERROR THAN RANDOM SEARCH (FIRST NUMBER IN PARENTHESIS) OR THE BENCHMARK BEST (SECOND NUMBER).

Island types	Rosenbrock	Ackley	Rastrigin	Schaffer
all optimizers	**12713 ± 6266** (**18.05*****)	18.75 ± 0.51 (6.67^{***}, -0.98)	621 ± 59 (6.21^{***}, -0.81)	460 ± 37 (7.45^{***}, -3.26^{**})
no GA	14556 ± 5065 (19.38^{***}, -1.02)	18.81 ± 0.51 (6.31^{***}, -1.27)	630 ± 64 (5.14^{***}, -1.27)	449 ± 29 (11.14^{***}, -2.51^{*})
no BO	38471 ± 5996 (3.07^{**}, -13.28^{***})	19.54 ± 0.23 (0.53, -6.59^{***})	682 ± 38 (2.74^{*}, -4.78^{***})	512 ± 22 (2.28^{*}, -11.14^{***})
no DE	15670 ± 6147 (16.53^{***}, -1.51)	18.95 ± 0.57 (4.63^{***}, -2.01)	622 ± 51 (7.02^{***}, -0.94)	445 ± 24 (13.72^{***}, -2.28^{*})
no RS	14185 ± 3855 (22.53^{***}, -0.89)	18.75 ± 0.56 (6.16^{***}, -0.91)	614 ± 36 (10.11^{***}, -0.55)	**427 ± 26** (**15.57*****)
no PSO	14015 ± 5148 (19.56^{***}, -0.72)	**18.58 ± 0.61** (**6.93*****)	**605 ± 60** (**7.22*****)	442 ± 28 (12.25^{***}, -1.77)

all cases, random search performed the worst (see Table IV). On Rosenbrock (smooth, valley-shaped), LS was best (statistically significant), as expected: the problem surface is smooth so local search (gradient) methods usually are the most performant. Transpeciation of 8 islands was second best on Rosenbrock. On Ackley (non-smooth function with central global minima), transpeciation of 8 islands was the best, the 8 non-communicating islands configuration was second and LS was last. On Rastrigin (non-smooth function with many local minima), island transpeciation of 8 islands was the best again, and RS was last. Finally on Schaffer, the 8 island configuration was again the most performant, but it was also the only case where communication between optimizers was detrimental. In our opinion, having only high optimizer diversity is not enough (non-communicating islands case) as a search space like Schaffer's for NAS is quite unusual. The Welch's *t*-test of unequal variances statistic [62] against random search passed in all cases and with high confidence (in most cases p-value < 0.05). From the experiments, we can see that the co-evolution (enabling island-to-island communication with transpeciation) does improve the optimization performance. The "24 islands" case was (statistically) worse than the "8 islands" case: having too many optimizers is not ideal, as there will be too few iterations allocated (from the 500 total) to each individual search technique.

To examine the effect of the island types in transpeciation, we have fixed the number of islands to twenty (four of each type), while we shuffle the optimizer parameters on each island. Table V shows the minimization results for 6 total scenarios of two main configurations: (1) "all optimizers" as islands and (2) all types except a specific optimizer. In most cases, the "all optimizers" configuration offered performance in the top three (with high statistical significance against the random search optimizer). For all functions, removing the BO islands introduces the largest statistically significant detriment in performance. In contrast, except for Schaffer, the absence of the PSO islands improved the average performance (however, not statistically significant). For Schaffer (the wave-like function), random search had the largest negative impact. Worth noting is that the archipelago iterates asynchronously, so the islands compete in speed against each other. This means that the BO optimizer does not experience as many iterations as e.g. PSO or Random Search (RS), because after some iterations BO becomes slower than the other optimizers in finding new candidates (in other words, BO is computationally expensive). However, this only happens in the mathematical benchmark case, where the objective function takes only a few milliseconds to calculate. To reduce that effect, we added a delay of 0.1 seconds in the objective function. In the NAS model search case, the objective function calculation takes several minutes (training and evaluating a DNN), so the BO islands do have time to catch up in calculating the next iterations versus the other faster optimizers.

For the context of air-quality, we used a subset of the real-world O_3 measurements, to compare island transpeciation in NAS against random search. Due to computational resource constraints, we had to halve the max neuron count per layer and use a smaller dataset (a single measuring station and only six weather variables). Fig. 6 shows the evolution of the best model accuracy during NAS. The horizontal axis denotes the count of iterative model improvements and the vertical axis the test accuracy (100% - sMAPE). Each scenario was repeated ten times and the total experiment runtime was

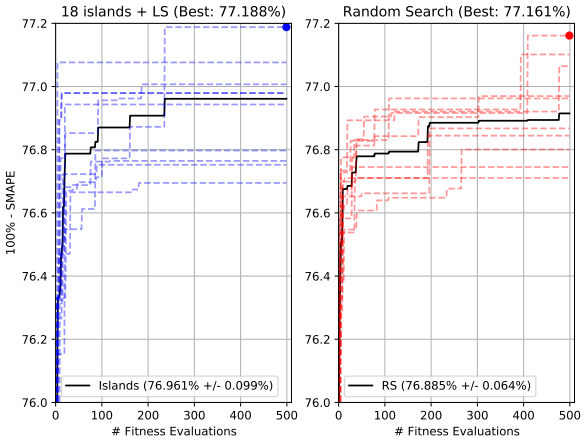


Figure 6. Island transpeciation (left) versus random search (right) NAS, on a real-world O_3 data subset: Island transpeciation generated more accurate models (best: 77.188%, median: 76.961% ± 0.099% mean absolute deviation, model samples: 5000) than Random Search (best: 77.161%, median: 76.885% ± 0.064%, model samples: 5000). According to Welch's *t*-test of unequal variances [62], the null hypothesis that the samples of each optimizer come from the same population can be rejected (*t*-statistic=3.109, p-value=0.00187). Each NAS search was repeated 10 times, for 500 iterations each. Target model type is MISO NARX, for the background-rural station BETN073 (code-name for the Belgian city Andenne). Train: 2000 to 2009, test: 2010, exogenous data: 6x weather variables, three cross-validation folds.

around 12 days (without the transpeciation parallelization, it could add up to 48 days). Out of 5000 models trained per case, island transpeciation found a better model (77.188%) than RS (77.161%). Island transpeciation, finds models with 0.076% higher accuracy and also surpasses 76.8% accuracy ≈100 iterations earlier than RS. To examine if the results from island transpeciation and RS are drawn from the same population distribution (the null hypothesis), we run the Welch's unequal variances *t*-test [62]. The *t*-test showed that the two samples have different means (statistic = 3.109) and that there is enough evidence to reject the null hypothesis (low p-value of 0.00187). The empirically chosen island transpeciation setup was: 18x optimization islands (4x GA, 4x BO, 4x DE, 3x RS, 3x PSO) on a 3x3x3 Cellular Automata grid, 3 directional neighborhood, 500 iterations (350 global search + 150 for LS polishing). The supplemental material (Appendix B) contains more details about the island setup.

2) Island DNN versus other models: Fig. 7 compares one-day-ahead O_3 forecasting MISO machine learning models. "Naive-1" at the bottom is the baseline and simplest model: the forecast is the value of the previous day. So, the baseline MASE is 1. We tried additional models but we show only the ones that surpassed the baseline. The Island DNN at the top was optimized by island transpeciation, both in architecture and hyperparameters. It was the best model and achieved the lowest MASE (0.655). Most of the other model types were trained with Matlab Regression learner, 10-fold cross-validation, regularization and 350 iteration hyperparameter Bayesian optimization. The Support Vector Machines (SVM) had a medium gaussian kernel and performed the worst with a MASE of 0.802. The best Gaussian Process Regression (GPR) had an rational quadratic kernel and performed better (MASE: 0.737) than the plain SVM. Least-Squares Support Vector Machines (LS-SVM) [63] are reformulations to the standard SVMs and exploit primal-dual interpretations. LS-SVM was the third best model (MASE: 0.705) and the second best was the Tree Ensemble model (MASE: 0.678). One argument against DNN is that they require long training times (≈ 5 days per search, ≈ 15 minutes per model). However, accounting for the hyperparameter optimization, the other model types did require multi-day training also (≈ 3-5 days). In addition, MISO NARX DNN can be easily expanded to MIMO, in order to forecast multiple measuring stations concurrently. To our knowledge, this is not readily possible for most of the other model types. Adding one or several extra outputs would also considerably increase the convergence training times for the other model types, but not for the DNN case. Fig. 8 shows the day-ahead prediction output of the best model, the optimized MISO NARX DNN model. In ozone forecasting, peaks are very important. We can see that the model is following the trend of the data and the sudden peaks in ozone concentration. The supplemental material contains detailed information on the optimization islands (Appendix B) and the machine learning methods used (Appendix C).

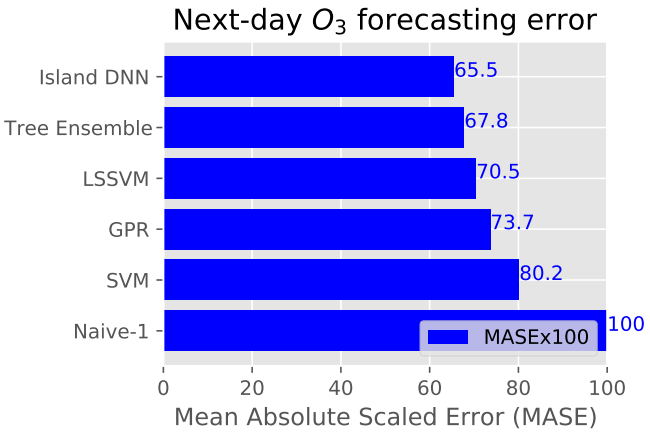


Figure 7. Next-day O_3 background-rural station forecasting error (BETN073 at Andenne, Belgium): Baseline (previous day in-sample value: Naive-1) against the best DNN, GPR, SVM and LS-SVM MISO NARX models. Most models were trained with Matlab Regression learner, 10-fold cross-validation, regularization and 350 iteration hyperparameter Bayesian optimization. Island transpeciation NAS trained the Island DNN, which had the lowest error. Island transpeciation setup: 18x optimization islands (4x GA, 4x BO, 4x DE, 3x RS, 3x PSO) on a 3x3x3 Cellular Automata grid, 3 directional neighbors, 500 iterations (350 global search + 150 for local search polishing). Training: 2000 to 2009, test: 2010, with 51 weather exogenous variables and linear interpolation on missing values.

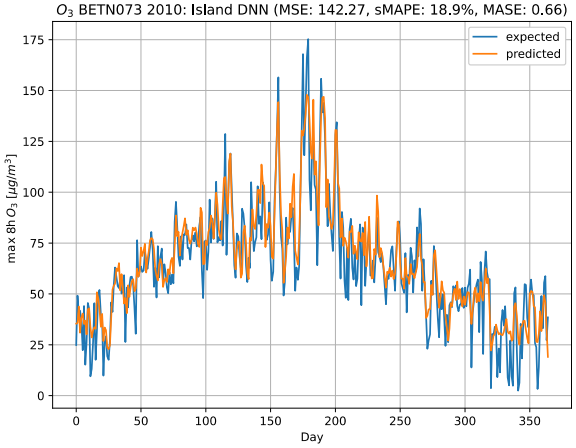


Figure 8. One-day ahead predictions of O_3 (single-station): Multiple-Input Single-Output (MISO) NARX DNN, optimized by island transpeciation NAS. The x-axis denotes time, 365 days of the year 2010, with a day as a time-step. The y-axis denotes the sensor value for an O_3 station, specifically the background-rural station BETN073 (code-name for the Belgian city Andenne). The blue line is the ground truth, the sensor values at the monitoring station. The orange line shows the model prediction. This model outperforms the other ML approaches. Train: 2000 to 2009, test: 2010, data: 51x weather variables [9].

C. Results for MIMO models

Fig. 9 shows country-scale observations and predictions of 25 O_3 monitoring stations for the year 2012, projected one on top of the other. It seems counter-intuitive to show a collective view; the reason is to show the spike around day 210. During the summer of 2012, there was an "inform public" alert, which happens when more than one stations have a measurement greater than 180 $\mu g/m^3$. The MIMO island DNN predicted the alert level successfully.

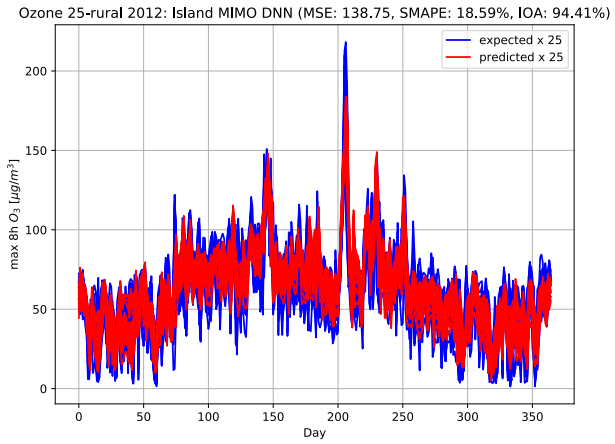


Figure 9. Country-scale one-day ahead predictions of O_3 (25 Belgian stations) with a MIMO NARX DNN optimized by island transpeciation: At day 207 our model successfully predicted a real-world incident, the **"inform public" level** one-day earlier. Train: 1990 to 2011, test: 2012. Index Of Agreement (IOA) indicates the match between the data and the forecasts (with 100% as the perfect match).

Fig. 10 compares the MISO versus MIMO models for single-station prediction performance. We want to predict one station, BETN073 (background-rural O_3 station). We compare three models: (1) "1-station model", (2) "all-stations model" (inputs from all the 46 O_3 stations in Belgium) and (3) "background-rural-stations model" (inputs from only the 18 background-rural O_3 stations). The "background-rural-stations

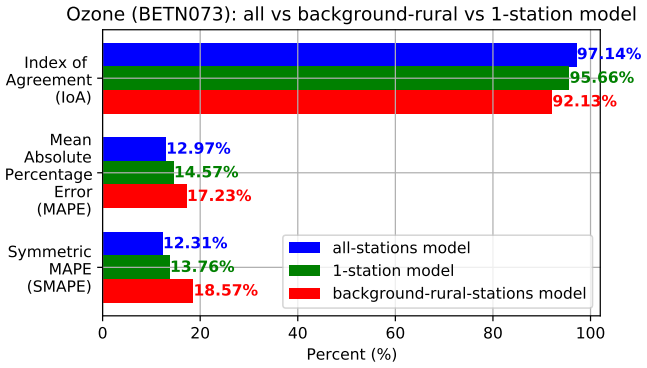


Figure 10. Single O_3 station (BETN073) prediction comparison: all-station vs 1-station model vs background-rural. All the models are optimized by island transpeciation NAS. "1-station model": MISO NARX trained only with the BETN073 station time-series and weather/calendar data. "background-rural-stations model": MIMO NARX trained only with the O_3 background-rural stations (18 total) of Belgium. "all-stations model": MIMO NARX trained with all the stations O_3 of Belgium (46 total). The all-stations model was the most accurate. Train: 2010 to 2017, test: 2018.

model" had the worst performance in all metrics. The "1-station model" was mediocre initially, but improved vastly after 500 island transpeciation iterations. The additional 8 cyclical calendar features (sine and cosine of day of the week/month/year) helped in accuracy even more. The "all-stations model" was the most accurate: reduced the sMAPE error by 1.45% against the single-station and 6.26% against the background-rural stations model.

D. Island transpeciation as NAS

We will see a case of global optimization with island transpeciation for O_3. In Fig. 11 we illustrate the NAS model accuracy progression when trying to find a suitable architecture for a MIMO NARX DNN model. The vertical axis denotes the model accuracy on the test set and the horizontal the total number of fitness evaluations. In the left figure, we can see a scatter plot where each dot type represents DNN accuracy from different island/hyperparameter optimizers. The dashed line shows the accuracy trend of all the models. We used 18 island variants of the following global optimizers: 4x GA, 4x BO, 4x DE, 3x RS and 3x PSO. The global optimizers were organized into a 1-dimensional ring of 18 slots and every island had one neighbor. NAS overall had a positive trend, during the 504 total fitness evaluations (total experiment runtime ≈5 days). The first 381 iterations were devoted to global search and the last 123 for local search model polishing. The local search was done by three non-communicating islands, running different local optimizers. Overall, the median model had 90.74% ± 2.6% accuracy. There was a vast range in model performance (worst 74.01% and best 92.60%), which further shows the need to optimize DNNs. Even though LS could not further improve the best found architecture, we can see that island transpeciation NAS indeed generated a consistently performant architecture, because the final 123 fitness evaluations with local search polishing had similarly high accuracy (from 90% to 92.5%). On the right figure, the black line represents the convergence to the best found model from the whole archipelago (92.6%). If we would have used only BO optimizers, DNN model improvements would get stuck to 92.34% from 145 to 244 fitness evaluations. However, DE and GA kept improving and eventually migrated their optimal solutions to the BO islands. With this co-evolution, the best solution was found rather early, on iteration 245 by the BO island. A general practical recommendation for a convergence strategy is *early stopping*: we should have converged to a good enough DNN architecture, if there are no further fitness improvements after a specific amount of consecutive iterations.

To validate our approach on different datasets and also assess the effect of island count against the MIMO DNN accuracy, we carried out experiments on two additional countries (Table VI). We examined Cyprus and Malta (both with three O_3 measuring stations), using five years of training/validation years (from 2014 to 2018) and 2019 as the out-of-sample test year. Cyprus and Malta are small Mediterranean countries located at the southern parts of Europe, with a climate much warmer than Belgium's. Even though both country data are aggregated and curated by the European Environmental Agency (EEA) [8], the original measuring station instrumentation and logging is fulfilled by different organizations: the Belgian Interregional Environment Agency (IRCEL - CELINE) (see https://www.irceline.be/), the Ministry

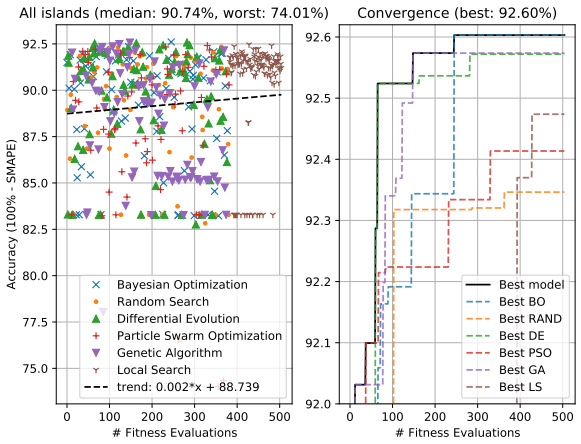


Figure 11. Accuracy progression with island transpeciation NAS on O_3 2018 (all 46 stations): left plot shows the accuracy of all the island models, right plot illustrates the convergence to the best solution (92.6% at iteration 245 by the BO island). For every iteration the DNN was trained from scratch, using the evolved architecture. The first 381 iterations were devoted to global search and the last 123 for local search polishing. The models had a vast accuracy range: worst 74.01% and best 92.60%. Local Search iterations even though they did not further polish the final architecture on the illustrated experiment, they had similarly good accuracy (from 90% to 92.5%, close to the best), showing that island transpeciation indeed found a performant DNN architecture. Setup: 18 islands (1D grid, one neighbor per island), 5 individuals per island, calendar cyclical features.

of Labour, Welfare and Social Insurance of Cyprus (see: https://www.airquality.dli.mlsi.gov.cy/) and the Environment and Resources Authority of Malta (see: https://era.org.mt/topic/real-time-air-quality-network/). We trained MIMO DNN models with 4, 8, 16, 24, 32, or 40 islands for 500 island transpeciation iterations (no local search phase). To reduce the NAS computational time down to 3.5 hours for the 6 new model searches, we had to reduce our model hyperparameter space (max neuron units per layer: from 512 down to 128, max training iterations: from 600 down to 150, complete removal of the recurrent dropout ratio hyperparameter). Removing the recurrent dropout ratio hyperparameter allows Keras Tensorflow [42] to utilize the highly tuned NVIDIA CUDA Deep Neural Network library (cuDNN), that provides faster DNN back-propagation training than the default implementation. Using the knowledge that we gained from the previous experiments, we also reduced the variables from our datasets (from 50 weather variables down to 20: the 10 most positively correlated weather variables and the 10 most negatively correlated ones) using the Spearman's ρ rank-correlation coefficient [64]). We can see that in all island counts, the final models have good predictive accuracy (MASE metric is below 1.0). The Diebold-Mariano (DM) null-hypothesis test of predictive accuracies for forecasting models [65] shows that for all stations of both countries, the statistic is positive (test passes: forecasts are better than the Naive-1 persistence model) and with high confidence (except for some cases with station 3 in Malta, p-value < 0.05 for 4-40 islands). Depending on the metric, choosing 4-24 islands for 500 transpeciation iterations seems

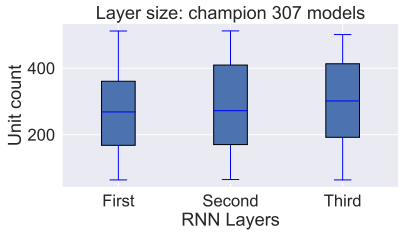
Table VI
ISLAND COUNT VERSUS METRICS OF BEST-FOUND MIMO DNN (500 NAS ITERATIONS). THE DIEBOLD-MARIANO (DM) NULL-HYPOTHESIS TEST OF PREDICTIVE ACCURACIES [65] COMPARES THE FORECASTING PERFORMANCE OF TWO MODELS AGAINST THE EXPECTED VALUES (FOR OUR CASE, METRIC: MEAN SQUARED ERROR (MSE), FORECASTING HORIZON: ONE-STEP AHEAD). DM IS POSITIVE, IF THE ISLAND DNN MODEL HAS LESS ERROR THAN THE NAIVE-1 MODEL (FIRST ROW) OR THE BENCHMARK BEST (SECOND ROW: VERSUS 24 ISLANDS FOR CYPRUS, VERSUS 16 ISLANDS FOR MALTA) ON THE TEST DATA.

	Islands					
Metric	**4**	**8**	**16**	**24**	**32**	**40**
Cyprus (3 ozone stations)						
MSE	**266.97**	272.75	283.42	273.44	270.39	280.74
RMSE	**16.34**	16.52	16.84	16.54	16.44	16.76
MAPE	15.69	15.9	16.03	**15.62**	15.72	15.72
sMAPE	**14.28**	14.56	14.82	14.3	14.31	14.74
IOA	82.99	82.32	82.18	**83.75**	82.97	82.9
MASE	0.8719	0.8894	0.9043	**0.8716**	0.872	0.899
Station 1 DM	5.19^{***}	5.14^{***}	4.73^{***}	5.17^{***}	5.51^{***}	$\mathbf{5.77}^{***}$
	-3.02^{**}	-2.31^{*}	-0.38		-1.08	-0.03
Station 2 DM	3.71^{***}	3.11^{**}	2.25^{*}	3.75^{***}	$\mathbf{3.78}^{***}$	2.53^{*}
	-1.91	0.54	2.95^{**}		-1.76	2.94^{**}
Station 3 DM	$\mathbf{3.8}^{***}$	3.72^{***}	3.2^{**}	3.52^{***}	3.77^{***}	3.78^{***}
	-2.25^{*}	-0.78	0.44		-1.52	-0.39
Malta (3 ozone stations)						
MSE	86.92	87.75	**84.18**	89.32	86.12	89.13
RMSE	9.32	9.37	**9.17**	9.45	9.28	9.44
MAPE	9.32	9.41	**9.11**	9.5	9.29	9.5
sMAPE	8.99	9.05	**8.83**	9.12	9.01	9.16
IOA	88.7	88.62	**89.62**	88.24	89.1	88.36
MASE	0.9509	0.9561	**0.9314**	0.9651	0.9527	0.9682
Station 1 DM	2.74^{**}	2.66^{**}	3.18^{**}	2.58^{*}	3.13^{**}	$\mathbf{3.39}^{***}$
	-0.01	-0.15		0.16	-1.4	-0.84
Station 2 DM	2.72^{**}	2.43^{*}	$\mathbf{3.89}^{***}$	2.27^{*}	2.19^{*}	1.64
	1.87	2.03^{*}		1.68	3.0^{**}	2.88^{**}
Station 3 DM	1.86	1.71	$\mathbf{3.22}^{**}$	1.09	2.41^{*}	1.2
	1.05	1.0		1.41	1.79	2.7^{**}

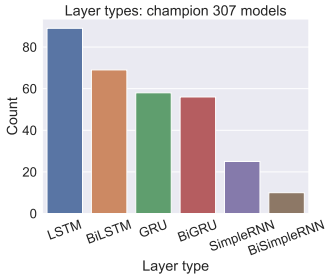
the most sensible amount. For Malta, 16 islands gave the best results and 40 islands were statistically the worst (for all stations, according to the DM test on 16 versus 40 islands). For Cyprus, the DM tests did not show clear differences on the island counts. Generally, for 32 or 40 islands, metrics show lower accuracy, possibly because each individual optimizer is not allowed enough iterations (500 iterations for 32-40 islands allowed $\approx$ 12-15 iterations per optimizer only). Worth noting is that outside of this experiment, using the full version of the hyperparameters (see Table II and Table III) and the data (all the 50 weather variables that make the NAS task more challenging) is still important, as they allow for much better final model accuracies (as seen in Fig. 7, models can reach a lower MASE of 0.65-0.8).

E. Champion NAS models

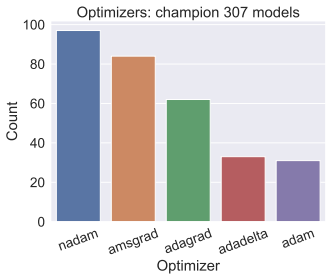
To examine the MIMO NARX DNN models in terms of architecture and hyperparameters, we aggregated all the 307 champion genotypes that were evolved from multiple experiment NAS runs (total 6659 models). Fig. 12 shows results on three important hyperparameters: layer size, layer type and DNN optimizer. GA optimizers may exhibit the "survival of the fattest" [66] side-effect: the average model size continuously grows, which makes the final optimized models rather large. With island transpeciation that is not the case; models compete both on accuracy and model training speed:



(a) Units hyperparameter (layer size in neurons).



(b) Layer type hyperparameter.



(c) DNN optimizer hyperparameter.

Figure 12. Hyperparameters from the top 307 (out of 6659) island transpeciation champion models, aggregated from different experiment runs. Layer size (a): champion models had a unit count close to the middle of the bounded space (64 to 512 units). With the implicit evolutionary pressure of the "model size vs training speed" trade-off, we can find performant and accurate DNN architectures, instead of architectures with the max trainable parameters possible ("survival of the fattest" [66]). Layer types (b): LSTM was selected about 4 times more than the bidirectional simple RNN layers. GRU layers stand in the middle of the selection preference. The plain layers were selected more often than the Bidirectional (Bi prefix). DNN (weight training) optimizers (c): most champion models used nadam and amsgrad.

the champion models (Fig. 12a) had a layer unit count close to the middle of the bounded space (≈300 units $\in [64, 512]$ units) and not the max. Models with the maximum trainable parameters have higher learning capacity, but can be slower to train. On the other hand, smaller models may perform worse than the larger ones initially, but they can be trained faster and evolve more. Experiencing more iterative improvements should yield better architectures. Fig. 12b examines the recurrent layer types selected by the optimizers. LSTM was preferred almost 4 times more than the bidirectional simple RNN layers. The bidirectional layers were chosen the least because their capabilities were not fully utilized: no future time-series steps were provided. Fig. 12c shows that most champion models had nadam and amsgrad as DNN weight optimizers. Adam and adadelta were chosen the least.

VI. Conclusion

MIMO NARX DNN can predict next-day O_3 pollution episodes, at a country-scale. This approach surpassed single-model performance on real-world, publicly available data: time-series (from 1990 to 2018) from 46 O_3 Belgian monitoring stations, combined with 51 meteorological variables. MIMO NARX DNN successfully predicted one day before, an "inform-public" O_3 alert level in Belgium for 2012. Predictions improved with: data standardization, time-series cross-validation, adding meteorological variables and cyclical calendar features. The main negative is long training times, especially with high input lag count.

Island transpeciation, optimized MIMO NARX DNN hyperparameters and architectures that outperformed random model search and previous modelling efforts. Island transpeciation combines results from multiple optimizers, to provide consistent performance. It is a co-evolutionary meta-learning method that combines NAS and iterative global/local optimizers. The asynchronous co-evolution balances model size versus training time trade-offs. Heterogeneous hardware resources can be parallelized with fault tolerance and distributed control. However, additional configuration is required compared to single hyperparameter optimizers. The encoding/decoding of internal representations to genotypes can be lossy, requires altering the optimizer source code internals and it is not always bidirectional (e.g., random search can not algorithmically accept model migrations). Finally, islands should iterate enough in order for the underlying optimizers to perform sufficiently.

For future work, we will examine: (1) increasing the island count to hundreds, (2) new optimizer types and (3) multi-objective optimization extensions. Finally, we will apply island transpeciation in particulate matter forecasting, and in vastly different domains like wind/solar energy forecasting.

References

[1] "Health Effects Institute. State of Global Air 2019," 2019. [Online]. Available: www.stateofglobalair.org

[2] C. Guerreiro, F. de Leeuw, A. G. Ortiz, M. Viana, and A. Colette, "Air quality in Europe — 2018 report," European Environment Agency, Tech. Rep., 2018.

[3] J. Hooyberghs, C. Mensink, G. Dumont, F. Fierens, and O. Brasseur, "A neural network forecast for daily average PM10 concentrations in Belgium," *Atmospheric Environment*, vol. 39, no. 18, pp. 3279–3289, 2005.

[4] O. M. Agudelo, O. B. Mendoza, P. Viaene, and B. De Moor, "Assimilation of ozone measurements in the air quality model AURORA by using the Ensemble Kalman Filter," *Proceedings of the IEEE Conference on Decision and Control*, no. ii, pp. 4430–4435, 2011.

[5] O. M. Agudelo, P. Viaene, and B. De Moor, "Improving the PM10 estimates of the air quality model AURORA by using Optimal Interpolation," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1154–1159, 2015.

[6] R. Langone, O. M. Agudelo, B. De Moor, and J. A. K. Suykens, "Incremental kernel spectral clustering for online learning of non-stationary data," *Neurocomputing*, vol. 139, pp. 246–260, 2014.

[7] B. S. Freeman, G. Taylor, B. Gharabaghi, and J. Thé, "Forecasting air quality time series using deep learning," *Journal of the Air and Waste Management Association*, vol. 68, no. 8, pp. 866–886, 2018.

[8] "AirBase - The European air quality database." [Online]. Available: https://www.eea.europa.eu/data-and-maps/data/airbase-the-european-air-quality-database-8

[9] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A. C. Beljaars, L. van de Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A. J. Geer, L. Haimberger, S. B. Healy, H. Hersbach, E. V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A. P. Mcnally, B. M. Monge-Sanz, J. J. Morcrette, B. K. Park, C. Peubey, P. de Rosnay, C. Tavolato, J. N. Thépaut, and F. Vitart, "The ERA-Interim reanalysis: Configuration and performance of the data assimilation system," *Quarterly Journal of the Royal Meteorological Society*, 2011.

[10] K. Theodorakos, *Air-quality forecasting in Belgium using Deep Neural Networks, Neuroevolution and distributed Island Transpeciation*. M.Sc. thesis. Katholieke Universiteit Leuven. Faculty of Engineering Science. Department of Electrical Engineering. ESAT-STADIUS, 9 2019.

[11] L. Hill and M. Flack, "The Physiological Influence of Ozone," *Proceedings of the Royal Society B: Biological Sciences*, vol. 84, no. 573, pp. 404–415, 12 1911.

[12] European Commission, "Directive 2002/3/EC of the European Parliament and of the council of 12 February 2002 relating to ozone in ambient air," *Official Journal of the European Union*, 2002.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1724–1734.

[17] H. Xie, H. Tang, and Y. H. Liao, "Time series prediction based on narx neural networks: An advanced approach," in *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, 2009, pp. 1275–1279.

[18] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "A Particle Swarm Optimization-Based Flexible Convolutional Autoencoder for Image Classification," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 8, pp. 2295–2309, 8 2019.

[19] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multiobjective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf*, 2019, pp. 419–427.

[20] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. IEEE, 12 2016, pp. 261–265.

[21] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 2018.

[22] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing, "Neural Architecture Search with Bayesian Optimisation and Optimal Transport," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 2020–2029.

[23] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[24] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.

[25] D. Kraft, "A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28," Braunschweig, p. 33, 1988.

[26] S. G. Nash, "Newton-Type Minimization Via the Lanczos Method," *SIAM Journal on Numerical Analysis*, vol. 21, no. 4, pp. 770–788, 10 1984.

[27] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Society for Industrial and Applied Mathematics, 10 2000.

[28] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search," in *Automated Machine Learning: Methods, Systems, Challenges*. Springer International Publishing, 2019, pp. 63–77.

[29] X. Zhou, A. K. Qin, M. Gong, and K. C. Tan, "A Survey on Evolutionary Construction of Deep Neural Networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 894–912, 2021.

[30] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning Methods, Systems, Challenges*. Springer International Publishing,, 2019.

[31] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[32] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2011.

[33] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[34] D. Izzo, M. Ruciński, and F. Biscani, "The generalized Island model," *Studies in Computational Intelligence*, vol. 415, no. January 2012, pp. 151–169, 2012.

[35] O. F. Cook, "FACTORS OF SPECIES-FORMATION." *Science*, vol. 23, no. 587, pp. 506–507, 1906.

[36] M. Tomassini, *Spatially Structured Evolutionary Algorithms*. Springer, 2005.

[37] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 4, 2012, pp. 2951–2959.

[38] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[39] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagńe, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012.

[40] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[41] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[42] F. Chollet and others, "Keras," 2015. [Online]. Available: https://keras.io

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, 2 2015, pp. 448–456.

[44] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 6 2017, pp. 972–981.

[45] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[46] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, San Diego, 2015.

[47] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning, ICML 2013*, no. PART 3, Atlanta, Georgia, USA, 2013, pp. 1139–1147.

[48] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[49] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *COLT 2010 - The 23rd Conference on Learning Theory*, vol. 12, 2010, pp. 257–269.

[50] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *CoRR*, 12 2012.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[52] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in Neural Information Processing Systems*, pp. 1027–1035, 2016.

[53] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–16, 2017.

[54] Ian Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[55] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas, "Blue Gene/L torus interconnection network," *IBM Journal of Research and Development*, vol. 49, no. 2-3, pp. 265–276, 10 2005.

[56] T. Bäck and F. Hoffmeister, "Extended Selection Mechanisms in Genetic Algorithms," in *Proceedings of the 4th International Conference on Genetic Algorithms*, 1991, pp. 92–99.

[57] D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proceedings of the Rocky Mountain conference on artificial intelligence, 1988*, 1988, pp. 88–101.

[58] T. D. Price, A. Qvarnström, and D. E. Irwin, "The role of phenotypic plasticity in driving genetic evolution," *Proceedings of the Royal Society B: Biological Sciences*, vol. 270, no. 1523, pp. 1433–1440, 2003.

[59] D. J. Whitehead, C. O. Wilke, D. Vernazobres, and E. Bornberg-Bauer, "The look-ahead effect of phenotypic mutations," *Biology Direct*, vol. 3, p. 18, 2008.

[60] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 10 2006.

[61] C. Bergmeir, R. J. Hyndman, and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction," *Computational Statistics and Data Analysis*, vol. 120, pp. 70–83, 10 2018.

[62] B. L. WELCH, "THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARIANCES ARE INVOLVED," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[63] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: WORLD SCIENTIFIC, 10 2002.
[64] D. I. F. Gregory W. Corder, "Comparing Variables of Ordinal or Dichotomous Scales: Spearman Rank-Order, Point-Biserial, and Biserial Correlations," in *Nonparametric Statistics for Non-Statisticians*. John Wiley & Sons, Ltd, 2009, ch. 7, pp. 122–154.
[65] F. X. Diebold and R. S. Mariano, "Comparing Predictive Accuracy," *Journal of Business & Economic Statistics*, vol. 20, no. 1, pp. 134–144, 2002.
[66] S. C. Cunnane, *Survival of the fattest: The key to human brain evolution*. World Scientific Publishing Co., 10 2005.

Konstantinos Theodorakos received the B.Sc. in informatics engineering (2013) from the Technological Educational Institute (TEI) of Crete, Greece and the Master of Computer Science in software engineering (2017) from the Universiteit Antwerpen in Belgium. He worked as an Electronic Hardware Technician (2008-2011) and as a Numerics Software Developer for Spatial and Spatio-Temporal Machine Learning on Geographic Information Systems (2013-2018). He is currently a Ph.D. student in Engineering Science, at the Department of Electrical Engineering (ESAT) and the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of KU Leuven, Belgium. He is working on time series analysis and modelling in data science applications. Mr. Theodorakos received the "Microsoft Azure credits for Research" award (2017) and the third place on the "Amazon Web Services IoT APP Challenge" contest (2017).

Oscar Mauricio Agudelo received the B.S. degree in electronics engineering from the Universidad Autónoma de Occidente, Cali, Colombia (1997), the M.S. degree in industrial control engineering from the Universidad de Ibagué (in cooperation with KU Leuven and Universiteit Gent), Ibagué, Colombia (2004), and the Ph.D. degree in electrical engineering from KU Leuven, Leuven, Belgium (2009). He worked at the Universidad Autónoma de Occidente, from 1997 to 2004, as a full time professor of control and automation. After his Ph.D., he held a postdoctoral position and later on a research manager position in the research group STADIUS at the Department of Electrical Engineering of KU Leuven. He is currently a project coordinator on systems and control in the same research group. His research interests are in model reduction techniques, systems and control theory, machine learning, model predictive control, data assimilation, deep learning, polynomial optimization, system identification, and analysis and design of intelligent control systems.

Joachim Schreurs was born in Hasselt Belgium, October 13, 1994. He received a Bachelor's degree in engineering science, electrical engineering and computer science (2015) and a Master's degree in engineering science, Mathematical Engineering (2017), both at the KU Leuven. He is currently a Ph.D. student in machine learning, at the STADIUS research division of the Department of Electrical Engineering (ESAT) at KU Leuven, under the supervision of prof. Johan A. K. Suykens. Joachim's scientific interests include machine learning, kernel methods, generative models, robust statistics and sampling algorithms for kernel methods.

Johan A.K. Suykens is full Professor with KU Leuven. He authored the books "Artificial Neural Networks for Modelling and Control of Non-linear Systems" (Kluwer Academic Publishers), "Least Squares Support Vector Machines" (World Scientific), co-authored "Cellular Neural Networks, Multi-Scroll Chaos and Synchronization" (World Scientific) and edited "Nonlinear Modeling: Advanced Black-Box Techniques" (Kluwer Academic Publishers), "Advances in Learning Theory: Methods, Models and Applications" (IOS Press) and "Regularization, Optimization, Kernels, and Support Vector Machines" (Chapman & Hall/CRC). He was associate editor for IEEE Transactions on Circuits and Systems (1997-1999 and 2004-2007), IEEE Transactions on Neural Networks (1998-2009), IEEE Transactions on Neural Networks and Learning Systems (from 2017) and IEEE Transactions on Artificial Intelligence (from 2020). He received the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as program co-chair for the International Joint Conference on Neural Networks 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007, co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning, and chair of ROKS 2013. He has been awarded an ERC Advanced Grant 2011 and 2017, has been elevated IEEE Fellow 2015 for developing least squares support vector machines, and is an ELLIS Fellow. He is currently the Master AI program director at KU Leuven.

Bart De Moor (Fellow, IEEE and SIAM) received a Ph.D. in applied sciences (1988) from KU Leuven. He was a research associate at Stanford University, is now full Professor at the Department of Electrical Engineering of KU Leuven and guest professor at the Universita di Siena, Italy. He served as head of cabinet of ministers of Science and Socio-Economic Policy in Belgium/Flanders and acted as vice-rector of International Policy of KU Leuven. His research fields are numerical linear algebra, system theory and control, machine learning and data science, in which he has been guiding more than 80 Ph.D. students and co-authored more than 400 scientific papers and 11 books. He has been serving in international scientific and science policy assessment committees, is member/head of the board of several (inter-)national scientific institutes (including the Flanders Biotech Institute) and funding agencies. He co-founded 8 spin-off companies (4 in Health 2.0, 4 in Industry 4.0). Since 2019, he co-architects and coordinates the Flanders AI program. He is the chairperson of the Capricorn Digital Growth Fund (venture capital), of Health-House (a high-tech science outreach centre) and the Alamire Foundation (digital humanities, polyphonic music) and co-founded Technopolis (children's science center). A member of the Royal Academy, he was awarded with numerous scientific prizes and recognitions (a.o. an ERC Advanced Grant) and honours (Fellowships of IEEE, SIAM). He received the Science Excellence Award from King Albert II of Belgium (2010) and was nominated a Commander in the Order of Leopold, by King Filip I (2020).