 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 3953; doi:10.3390/app10113953

Article

Grammar guided genetic programming for network architecture search and road detection on aerial orthophotography

Víctor de la Fuente Castillo [1,†,*], Alberto Díaz-Álvarez [1,†], Miguel-Ángel Manso-Callejo [1,†] and Francisco Serradilla García [1,†]

[1] Universidad Politécnica de Madrid; victor.fuente@upm.es (V.F.C.), alberto.diaz@upm.es (A.D-A.), m.manso@upm.es (M-A.M-C.), francisco.serradilla@upm.es (F.S.G.)
* Correspondence: victor.fuente@upm.es (V.F.C.)
† Current address: Escuela Técnica Superior de Ingenieria de Sistemas Informáticos, Universidad Politécnica de Madrid, Calle Alan Turing s/n (Ctra. de Valencia, Km. 7) C.P. 28031, Madrid, Spain

Featured Application: This system can be applied to automatically design the architecture of deep artificial neural networks for any given labeled dataset.

Abstract: Photogrammetry involves aerial photography of the earth's surface and subsequently processing the images to provide a more accurate depiction of the area (Orthophotography). It's used by the Spanish Instituto Geográfico Nacional to update road cartography but requires a significant amount of manual labor due to the need to perform visual inspection of all tiled images. Deep Learning techniques (artificial neural networks with more than one hidden layer) can perform road detection but it is still unclear how to find the optimal network architecture. Our system applies grammar guided genetic programming to the search of deep neural network architectures. In this kind of evolutive algorithm all the population individuals (here candidate network architectures) are constrained to rules specified by a grammar that defines valid and useful structural patterns to guide the search process. Grammar used includes well-known complex structures (e.g. Inception-like modules) combined with a custom designed mutation operator (dynamically links the mutation probability to structural diversity). Pilot results show that the system is able to design models for road detection that obtain test accuracies similar to that reached by state of the art models when evaluated over a dataset from the Spanish National Aerial Orthophotography Plan.

Keywords: Grammar Evolution; Deep Learning; Network Architecture Search; Grammar Guided Genetic Programming

---

1. Introduction

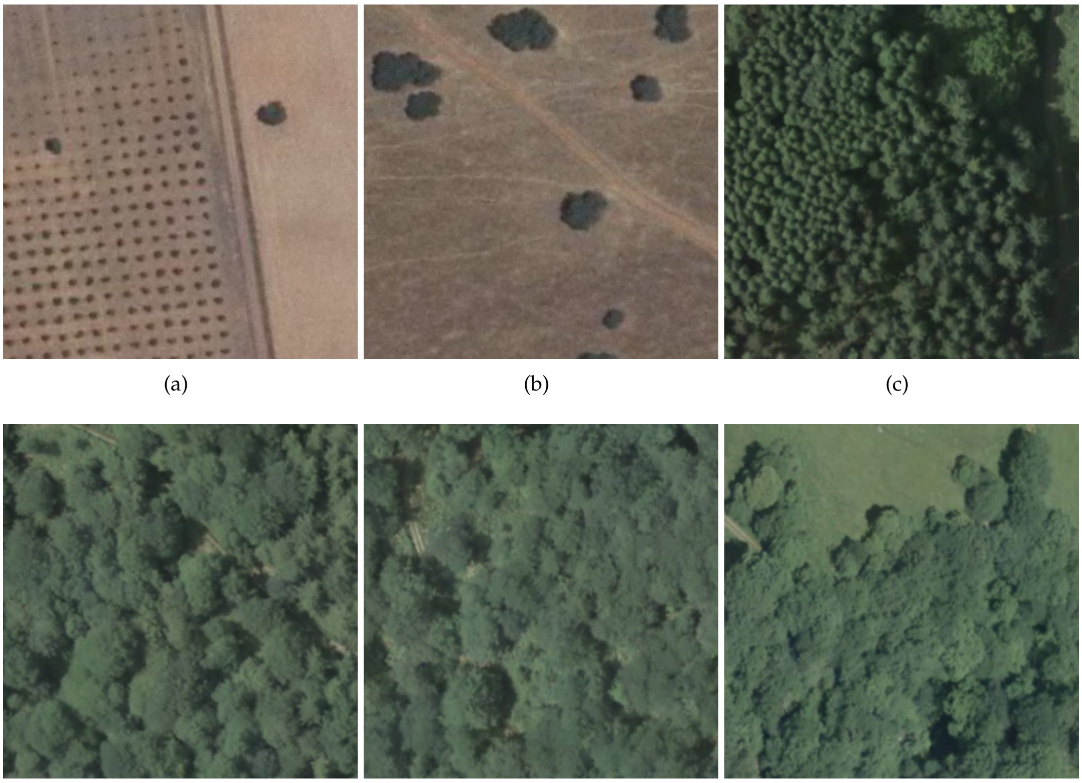
Two of the main problems with remote sensing information as noted by [1] are that the high volume of data exceeds by far the capabilities of human analysis (by manual revision) and at the same time is usually crucial to perform classification of said data. Under this category of data, large scale aerial image processed as orthophotography is a useful source of information for many domains. To give some examples of the broad array of applications we can find, there have been systems developed for the detection of: coastline changes [2], snow-avalanches [3], fires [4], bodies in disaster sites [5], trees [6], seedlings [7], roofs [8], transmission towers[9,10], vehicles [11,12], photo-voltaic arrays [13,14], vegetation and buildings [15].

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

We will focus on road detection on aerial images being it an important subject, among other things, due to the need to constantly update road maps. This importance is clear by the volume of research conducted and applied systems developed on this domain.

As an object of detection [16] divided the road characteristics into four main groups: spectral properties (those regarding surface characteristics), geometric (such as width and curvature), topological properties (those derived from their role as part of a transportation network) and contextual (e.g. different types of roads may have different restrictions to their shape, size or materials).

Put in the context of aerial image we can find many complications related to those groups or properties some even caused by the geographic context of the roads. One of this problems as shown on Figure 1 (examples a and b) is due to surrounding terrain having spectral properties similar to the secondary roads shown. Other common problem are occlusions due to their contextual properties (e.g. secondary mountain roads). Here we show an example of said occlusion at forest areas (Figure 1, examples c,d,e and f), on the first case due to the shadows and the rest the trees themselves covering the road. Other problems like some natural or artificial structures being similar to roads are shown on Figure 2.



(a) (b) (c)

(d) (e) (f)

Figure 1. Roads: contextual problems terrain and occlusion.

Some road detection systems rely entirely on non-neural models, some examples include [17] that combines texture information extraction with a knowledge-base road extraction and a post-processing step for noise elimination, [18] that applies Structural Support-Vector Machines (S-SVMs) [19] or [20] that uses GraphCut [21]. Though we can find many other systems that rely on road characteristics analysis [16–18,20,22–28] most recent research has favored the application of deep artificial neural networks (ANNs).

We find some systems include ANNs as part of their detection pipeline such as [29–33] or just preprocess the input images before they are feed to the network as in [34].

Is frequent to rely on a deep ANN to perform the whole road detection/segmentation process by either developing a novel network or using a preexisting architecture. Some researchers use custom

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

Figure 2. Sample images not containing roads that could be hard to classify.

network architectures [35,36], others customize well-known models [30,37–40], use those well-known models with transfer learning [41,42] or combine then to road-detection ensembles [43].

Systems that apply deep learning techniques to detection on such images can use quite different network dimensions and architectures. We can find ones that go from relatively small convolutional networks as the one seen on [44] to others using models like the Inception-V3 architecture [45] combined with transfer learning, like the one found on [14] for solar panel detection.

To try and solve the need to find the optimal ANN architecture automatically, or Network Architecture Search (NAS), has been extensively researched by using many different approaches but has proven to be a complex task given the dimensions of the search space. Some examples of such approaches include the use of reinforcement learning techniques, as seen on [46–48] or the application of evolutive algorithms like [49–60].

Some systems use grammars to define the rules to obtain valid structures that can encode either the neuron connectivity matrix and some other form of network topology [54,56,57] or other higher level expression of the operations being performed by the network layers and their connectivity [52,53,55,58–61]. Some of those systems like [52,53] work by generating architecture expressions with high level operations (such as convolutions, activation functions, dropout, etc) on [53] the architecture is sequential in term of layers while [52] allows more complex connectivities (parallel branches inside the network).

In order to design from scratch deep neural networks for road detection o aerial images, and guide the search with information about useful network structural patterns, the present work is conducted inside the Cartobot project for the Instituto Geográfico Nacional (IGN, Spain). This project is of real application on a dataset constructed (by IGN and Cartobot personnel) from Spanish National Aerial Orthophotography Plan (PNOA) aerial orthoimages. The two main research goals are:

- Design a grammar that can successfully encode the most typical high level architectural blocks of deep ANNs.
- Build a NAS system capable of designing deep RNA using that grammar for the purpose of successfully detecting roads in the aerial image.

We divert from the approach of [52] in that our grammar is going to define specific high level blocks, that will be latter discussed, instead of allowing for fully free connectivity for the generation of the blocks with parallel branches. We will also work on a dynamic mutation operator linked to the structural diversity of solutions (population) being contemplated at each instant.

We start by reviewing some of the well-known network models (AlexNet [62], VGG [63], GoogleLeNet [64], RestNet [65], Inception-v3 [45], Inception-v4 and Inception-ResNet [66], DPN [67] and ResNeXt [68]) used by top scoring results on the Imagenet Large Scale Visualization Recognition Challenge (ILSVRC) [69] (editions from 2012 to 2017) where many different architectural patterns can be found on them. As mentioned by defining a meta-grammar able to condense some of such structural rules we aim to guide an evolutive NAS system to find networks capable of solving the road detection problem on aerial orthoimages.

The problem of road detection (including secondary roads) on such orthoimages has, as we have explained, the added difficulty of natural or artificial structures that can be mistaken with roads or transportation infrastructures. Some samples of those formations can be seen on Figure 2 with various levels of complexity. On the other hand many different types of roads are being considered and are required to be detected by the system. We have a set of sample orthoimages with their corresponding numerical topographical cartography at a scale of 1:25,000 (BTN25) to allow for easier identification of the roads by the reader (see Figure 3).

2. Methods

Our system, VGNet, uses a Grammar Guided Genetic Programming (GGGP) [70] to solve the NAS problem. GGGP is a family of evolutionary algorithms that encode solutions to a given program as tree structures (constrained by a set of structural rules specified with a grammar) and follows the same basic process of a genetic algorithm with some variations.

The reasons for selecting this approach are:

- Evolutionary algorithms easily allow for a high level of parallelism. [55]
- The use of a grammar avoids exploring invalid solutions and allows for domain knowledge to be included to guide the search.

The general GGGP evolutive process follows this steps:

1. We define a coding system to represent the candidate solutions (individuals) for a given problem and a context-free grammar that defines restrictions to their structure.
2. A population (a set) of initial solutions is created following the grammar rules.
3. We check the fitness (a measure of how well they solve the problem at hand) of the individuals to see if we found an acceptable solution (given some stop criteria or goal to reach) and should stop the process or continue the search.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1
Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953


(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 3. Sample images containing roads and their corresponding road cartography.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

4. If the stop criteria is not met we create a new population by:
   (a) Select solutions (called parents), usually by pairs, with regard to their fitness value. Better solutions have higher chances of being selected.
   (b) With certain probability combine the parents (or leave them unchanged) to obtain new solutions (offspring) using a crossover operator. The goal here is to combine the information contained in each parent to try and find better solutions. In our scenario subtrees are exchanged between the parents to generate the offspring.
   (c) With certain probability the offspring individuals are checked to add some random variations (constrained once again by the grammar rules) to improve the exploration of new solutions (new areas of the search space).
5. We go back to step 3 with the new population we created.

For VGNet the individuals (candidate solutions) will represent an ANN constrained by a context-free grammar of highly used network architectural building blocks. The system has the internal workflow shown on Figure 4. Each individual generated is a derivation tree and represents a neural network architecture codified using a expression language designed to be easily read. We will refer as Deep Learning models, or deep neural networks, to ANNs with more than one hidden layer.

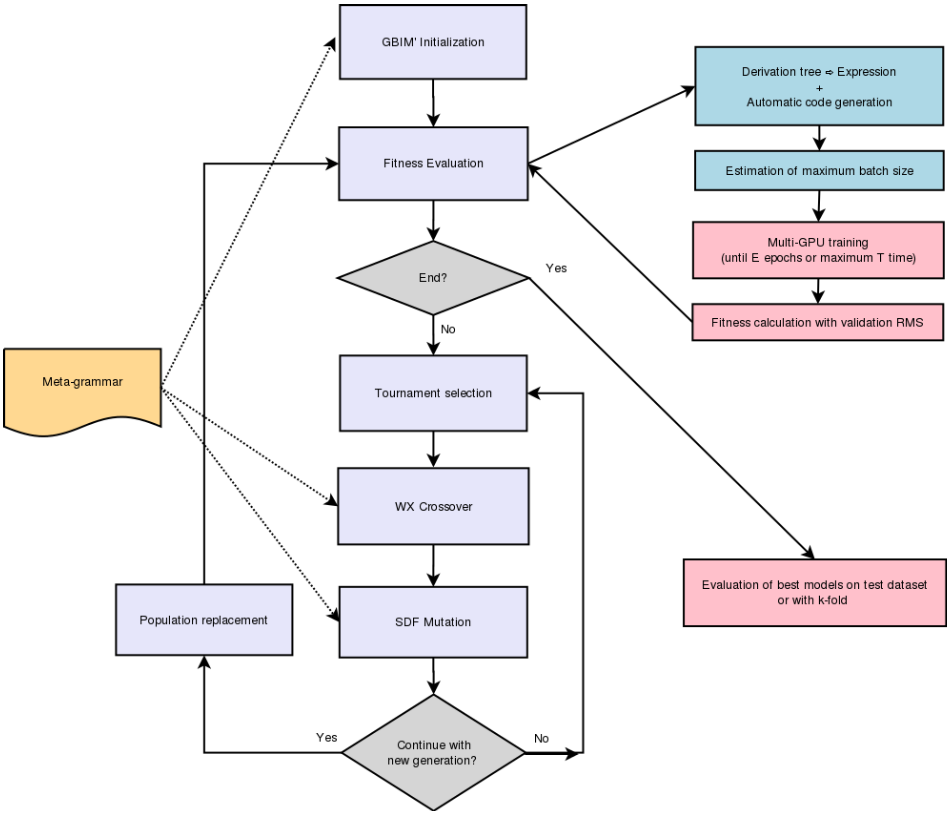


Figure 4. VGNet: network architecture design workflow.

The operators used during the evolution process are:

- Individual generation: based on Grammar-Based Initialization Method (GBIM) [71]. GROW [72] was used initially but was later discarded in favor of GBIM.
- Mutation: based on Grammar-Based Mutation (GBM) [73] and a variation of it of our own design, Schema Diversity Factor (SDF) mutation.
- Probabilistic Tournament Selection [74] using the formula from [75].
- Crossover: Whingham (WX) [70].
- Generational population replacement.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

Methods used by initialization and mutation are based around GBIM and GBM with the following modifications:

- Removing all non-terminals the restriction of having at least one terminal node only production (from hereon we will refer to those as default productions). This affects the guaranteed uniformity of depth level during random generation studied by the authors of the original method [71].
- Creating a derivation tree a random depth is chosen from the interval on the interval $[min\_depth, max\_depth]$ (original GBIM uses the interval $[1, max\_depth]$).

This modifications meta-grammar are more easy to read and, though the formal study is outside the scope of our present work, we believe that the use of the cited default productions on the meta-grammar can lead to the introduction of a bias in their favor during the derivation tree generation.

The following sections will detail the codification scheme (Sec. 2.1), with the detailed meta-grammar on (Sec. 2.1.2). The SDF mutation operator we apply is explained in (Sec. 2.2 and 2.3) and finally the fitness in (Sec. 2.4)

2.1. Codification Scheme

2.1.1. Expression Language

To encode the network structures we define an expression language capable of simply representing many of the classification architectures found in the state of the art with some exceptions (i.e. DenseNet [76] is not supported due to the syntax we use for parallel processing branches). At this point we only consider architectures with one input point and one classification output (architectures like Faster-RCNN [77] with multiple outputs at different levels of the network are not supported in the current version of the system).

The expressions have the following tokens as basic components:

- In, Out: They represent the network input and output respectively.
- "[", "]": Start and end of parallel branches of processing that share the same input (branches are separated by ",").
- direct: connection between two network points without any operator being applied (residual-like connection).
- Processing nodes with two main types:
  - Traditional operators: convolutions (CONV), max or average poolings (MP or AP), dropout [78], Fully-Connected layers (FC), softmax and flatten.
  - Aggregation operators: concatenation (CONCAT) or zero-padded sum (SUM). The last one adds the features of all branches with the same channel index and concatenates the rest. Both aggregation operators are applied immediately after the closing of parallel processing branches to unify the results.

Some of the traditional operators use a parametric syntax with format **operator[-parameter]{0,n}**. The specific number of required parameters varies between operators:

- CONV-i-j-k l: i (dimension), j (number of generated features), k (stride) and l (normalization used). Currently only Batch Normalization (BN) [79] or no normalization are supported by our system. We also performed some tests including both regular (CONV) and separable convolutions (SepCONV).
- MP-m-k and AP-m-k: m (dimension) and k (stride).
- DROPOUT-r: r (rate).
- FC-s: s (size).

Some important notes:

- Flatten: it is always present before the first FC layer.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

- Activation functions: all layers use leaky-ReLU [80], with the only exception of the output layer that uses Softmax.
- In, SOFTMAX, Out: The input and output dimensions are not specified on the operators themselves since they are problem dependent and configured at library level.

2.1.2. Meta-grammar

To guide the evolutionary search the overall network architecture we allow is as follows:

1. Input layer: dimensions are given by the images of the dataset being used.
2. Sequence of high level structural blocks. Each block is chosen from the following set:
    - Convolutional block. Sequence of convolution, optional batch normalization, dropout and optional pooling.
    - Inception based blocks. Parallel convolutional branches with their outputs being concatenated (all branches are constrained to apply the same spatial dimensional reduction or none).
    - ResNet based blocks. Consists of two parallel processing branches. A convolutional sequence and residual connection with their outputs being aggregated by zero padded sum (to allow for different number of features between the two branches).
    - Inception-ResNet based blocks. Here we have a Inception based block and a residual connection with aggregation of both their outputs via a zero padded sum.
3. Sequence of Fully Connected layers (leaky-Relu activation function) alternated with dropouts. Before the first FC layer a flatten operation is automatically performed.
4. Output layer: softmax layer with one output per class.

It worth mentioning that the dropout operator presence is forced allowing a value of $rate = 0$ to account for it being inactive. That way the operator block is always present and can be easily turned on/off via mutation or crossover.

Dimensions higher than three are not used for the convolutions and poolings in light of the results obtained by [45] using factorized convolutions to replace those with higher dimensions. Limit for number of FC layers exists mainly due to their computational cost compared to the convolutional ones.

At a lower level the meta-grammar has the following parameters:

- FC layers; maximum number of layers and set of dimensions allowed: **fM**, **fS**.
- Convolutions; sets of dimensions and number of features allowed: **cD**, **cF**.
- Pooling; sets of pooling types and dimensions: **pT** (e.g. MP and AP), **pD**.
- Convolution and pooling; set of valid strides: **cpS**.
- Normalizations; set of allowed normalizations: **norm**.
- Dropout allowed rates: **dR**.

The meta-grammar is expressed as a context-free grammar described here in Backus-Naur Form (BNF). To reduce the text extension we use the contraptions Block (B), Network (N, NS for plural), Parallel (PRL) and Reduce Dimension (RD):

$G_{fM,fS,cD,cF,cpS,pT,pD,norm,dR}$ = {S=ANN, $\Sigma_{N,cpS}$, $\Sigma_{T,fS,cD,cF,cpS,pT,pD,norm,dR}$}

$\Sigma_{N,cpS}$ =
{ ANN, HIGHLVL_N, BLOCKS, RD, FC, FC_B, RESN_B,
INCEPT_B, INCEPT_RESN_B, CONV_N, CONV_B,
CONV_N_S1, CONV_B_S1, PRL_CONV_NS, PRL_B_RD,
PRL_CONV_NS_RD, RD, CONV, POOL, DROPOUT }
∪ { CONV_Sk, POOL_Sk } $\forall k \in cpS$

$\Sigma_{T,fS,cD,cF,cpS,pT,pD,norm,dR}$ =
{ ‘In’, ‘Out’, ‘flatten’, ‘SOFTMAX’, ‘direct’, ‘CONCAT’, ‘SUM’, ‘DROPOUT’, ‘[’, ‘]’, ‘,’, ‘direct’ }

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

$\cup$ { 'CONV-i-j-k-l' } $\forall i \in cD, \forall j \in cF, \forall k \in cpS,$
$\forall l \in norm$
$\cup$ { 'p-m-k' } $\forall p \in pT, \forall m \in pD, \forall k \in cpS$
$\cup$ { 'DROPOUT-r' } $\forall r \in dR$
$\cup$ { 'FC-s' } $\forall s \in fS$

⟨ANN⟩ ::= 'In' ⟨HIGHLVL_N⟩ ⟨RD⟩ 'flatten' ⟨FC⟩ 'SOFTMAX' 'Out'
⟨HIGHLVL_N⟩ ::= ⟨BLOCKS⟩
| ⟨HIGHLVL_N⟩ ⟨HIGHLVL_N⟩
⟨BLOCKS⟩ ::= ⟨CONV_B⟩
| ⟨RESN_B⟩
| ⟨INCEPT_B⟩
| ⟨INCEPT_RESN_B⟩
⟨FC⟩ ::= ⟨DROPOUT⟩ ⟨FC_B⟩ ⟨DROPOUT⟩
| ...
| ⟨DROPOUT⟩ $\langle FC\_B \rangle_0$ $\langle DROPOUT \rangle_0$ ... $\langle FC\_B \rangle_{fM}$ $\langle DROPOUT \rangle_{fM}$
⟨CONV_N⟩ ::= ⟨CONV_B⟩ | ⟨CONV_B⟩ ⟨CONV_N⟩
⟨CONV_B⟩ ::= ⟨CONV⟩ ⟨DROPOUT⟩
| ⟨CONV⟩ ⟨DROPOUT⟩ ⟨POOL⟩
⟨CONV_B_S1⟩ ::= ⟨CONV_S1⟩ ⟨DROPOUT⟩
| ⟨CONV_S1⟩ ⟨DROPOUT⟩ ⟨POOL_S1⟩
⟨CONV_N_S1⟩ ::= ⟨CONV_B_S1⟩
| ⟨CONV_B_S1⟩ ⟨CONV_N_S1⟩
⟨RESN_B⟩ ::= '[' ⟨CONV_N_S1⟩ ',' 'direct' ']' 'SUM'
⟨INCEPT_B⟩ ::= '[' ⟨PRL_CONV_NS⟩ ']' 'CONCAT'
| '[' ⟨PRL_CONV_NS_RD⟩ ']' 'CONCAT'
⟨INCEPT_RESN_B⟩ ::= '[' '[' ⟨PRL_CONV_NS⟩ ']' 'CONCAT' ',' 'direct' ']' 'SUM'
⟨PRL_CONV_NS⟩ ::= ⟨CONV_N_S1⟩ ',' ⟨CONV_N_S1⟩
| ⟨PRL_CONV_NS⟩ ',' ⟨CONV_N_S1⟩
⟨PRL_CONV_NS_RD⟩ ::= ⟨CONV_N_S1⟩ ⟨RD⟩ ','
⟨CONV_N_S1⟩ ⟨RD⟩
| ⟨CONV_N_S1⟩ ⟨RD⟩ ',' ⟨RD⟩
| ⟨PRL_CONV_NS_RD⟩ ',' ⟨CONV_N_S1⟩ ⟨RD⟩
⟨RD⟩ ::= ⟨CONV_S2⟩
| ⟨POOL_S2⟩
⟨CONV⟩ ::= ⟨CONV_Sk⟩ | ... $\forall k \in cpS$
⟨CONV_Sk⟩ ::= 'CONV-i-j-k l' | ... $\forall i \in cD, \forall j \in cF, \forall l \in norm$
⟨POOL⟩ ::= 'p-m-k' | ... $\forall p \in pT, \forall m \in pD, \forall k \in cpS$
⟨POOL_Sk⟩ ::= 'p-m-k' | ... $\forall p \in pT, \forall m \in pD, \forall k \in cpS$
⟨DROPOUT⟩ ::= 'DROPOUT-r' | ... $\forall r \in dR$
⟨FC_B⟩ ::= 'FC-s' | ... $\forall s \in fS$

2.2. Diversity control via structural schema clusters

Due to the high number of parameters used by the expression language to define the architectures, measuring diversity by checking for unique expressions of population individuals can be misleading.

By taking those expressions and ignoring their specific operator parameters we can group them into clusters of unique abstract structural schemas. Those clusters represent groups of solutions where the connection structure between generic operations is the same.

To give a simple example both this two sub-expressions:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

- AP-3-1 [CONV-3-128-1 DROPOUT-0.4 , direct] SUM CONV-3-256-2 DROPOUT-0.5
- AP-3-1 [CONV-3-56-1 DROPOUT-0.2 , direct] SUM CONV-3-128-1 DROPOUT-0.3

Belong to this generic structural schema:

- AP [CONV DROPOUT , direct] SUM CONV DROPOUT

The number of schema clusters gives us an idea of how many structurally different architectures are being explored by the current population.

2.3. Schema Diversity Factor Mutation

We propose a new grammar based mutation operator based on Grammar Based Mutation (GBM) [73]. Seeing how GBM only chooses one non-terminal node from the tree for mutation and uses a fixed mutation probability we propose some modifications to adapt it to our NAS problem. We will refer to our variant of GBM as Schema Diversity Factor (SDF) mutation and it differs from GBM in two main aspects:

- The derivation tree non-terminal nodes are checked for mutation following a width first search.
- The mutation probability (mP) is calculated for each particular individual derivation tree based around its non-terminal node number and the population diversity (d) following equations (1, 2 and 3).

$$mP(i) = \beta \cdot \frac{1}{count(i.nonTerminalNodes)} \tag{1}$$

Where mP is the mutation probability for the individual i and β is an scaling factor that can be set manually or calculated in relation to the population diversity. We use the population structural diversity (d) measured via unique structural schema clusters with equations (2 and 3).

$$\beta = (1 - d) \cdot (\beta_M - \beta_m) + \beta_m. \tag{2}$$

$$d = numberOfUniqueSchemasClusters/populationSize \tag{3}$$

Values of β will belong to the interval $[\beta_m, \beta_M]$. During our experiments the interval was set to $[0.5, 2]$.

This has the effects of:

- Increasing the mutation probability when the structural diversity is low in order to improve exploration and lowering it when said diversity is high to allow exploitation.
- Since we check for mutation each node individually:
  - Mutation of more than one subtree at a time is allowed (more than one part of the same network architecture can be changed).
  - We want a lower probability as the total node number grows to avoid an excessive number of mutations. That's why we adjust it for each individual depending on the the number of non-terminal nodes.

2.4. Fitness

To obtain an individual fitness, its architecture expression is automatically translated into code and the resulting model is trained from scratch on the training partition of the dataset. For loss we use either the Root Mean Square Error (RMSE) or the Categorical Cross-Entropy (CCE) over the validation partition to obtain the fitness using the following equation:

$$fitness(indv) = \frac{1}{1 + \frac{1}{n} \cdot \sum_{i=lastTrainEpoch-(n-1)}^{lastTrainEpoch} loss(indv, validation, i)}$$

We use $n = 3$ to smooth the oscillations of the loss value and obtain an average estimation.

3. Experiments and results

3.1. Experimental setup

Two different servers where used to conduct the evolution tests (Table 1).

Table 1. Test servers: hardware specifications.

Server	GPUs	CPU	RAM
IGN1	4× Tesla V100 16GB	$2\times Intel(R)Xeon(R)Gold6148@2.40GHz$	128GB
Cartobot1	2× RTX 2080Ti 11GB	i7-8700 3.2GHz	64GB

We restrict the search space of valid architectures for our experiments as follows:

- Maximum number of fully connected layers was set to one in order to reduce the memory and computational cost of training the network during the evolutive design stage. The maximum number of neurons on those layers was kept small for the same reason (8, 16, 32).
- Only convolutions of dimensions (1, 2, 3) where allowed, according to findings on [45] regarding factorized convolutions. The number of features is a set of typical used values in the range [2,256].
- We allowed the use of the two most frequently used types of pooling (max-pooling and average-pooling).
- Regarding the stride of convolutions and poolings we use 1 to keep spatial dimensions and 2 to allow their reduction at certain parts of the architecture (those places are specified inside the meta-grammar restrictions).
- For the dropout rates, valid values for the drop-rate parameter where set inside the interval [0, 0.4] with increments of 0.1. Here 0.0 is interpreted as an inactive dropout operator. The higher value 0.4 was set empirically to avoid TensorFlow-Keras warnings obtained for values higher than 0.5.
- We allow the use of no normalization operation or Batch Normalization (BN) to let the design process choose where to place the BN operations.
- Only for the models labeled with the preffix "v2" (version 2) we allowed the following settings (due to higher computational capabilities of the server used for the experiments):
  - Maximum number of FC layers is 3 instead of 1.
  - Bigger dimensions for FC layers (up to 256).
  - Separable 2D convolutions are used in addition to regular convolutions.
  - BN momentum was empirically set to 0.5 (due to average size of training batches caused by GPU number and available memory).
- On all cases during design the network was trained for its evaluation only for a maximum of 20 epochs (early stop if during 5 there was no improvement on the training metrics).

On all of the experiments the dataset was partitioned by experiment into train/validation/test subsets with a ratio of 70:20:10. Tournament selection used size $k = 4$ ($selectionPressure$ took values 0.7 or 1 depending on the experiment).

3.2. Results

For all the tests conducted here we use a dataset from the Spanish National Aerial Orthophotography Plan (PNOA) composed of 256x256x3 images (size-bands) with two mutually exclusive labels. Number of examples by class: 9173 (contains_road) and 7986 (doesnt_contain_road). This adds to a total of 17159 samples. We will refer to it as IGN-PNOA dataset from hereon.

Sample results of a designed network's output for three random sample groups outside of the dataset (orthophotos are from random geographical areas of Spain with no associated cartography

 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 3953; doi:10.3390/app10113953

available) to check their predictions over unknown data are shown on Figures (5, 6 and 7). Under each image of the group is the network predictive output for the class "contains road".

(a) p_road=0.00 (b) p_road=1.00 (c) p_road=1.00 (d) p_road=1.00

(e) p_road=1.00 (f) p_road=1.00 (g) p_road=1.00 (h) p_road=0.01

(i) p_road=0.49 (j) p_road=1.00 (k) p_road=1.00 (l) p_road=1.00

Figure 5. Sample predictions for randomly selected section of map outside the dataset. p_road shows the network prediction confidence for the class "contains road" for each image.

We selected the nine highest scoring networks (measured on the validation partition) designed by VGNet, all models where trained from scratch using the train and validation examples. To evaluate their test accuracy k-fold cross-validation (CV) is used with typical values of k=5,10 [81,82]. Regarding k=10 [81,83] shows that 10-fold CV obtain similar results to leave-one-out cross-validation (LOOCV) and has lower computational cost. Test metrics are included on Figures 8, 9 and detailed on Table 2. Their respective accuracy 95% t confidence intervals (using the calculation method detailed by [84]) are on (Table 3) to show the expected accuracy for each model. In all cases the same parameters used for designing the networks ($epochs = 200$, $alpha = 1e^{-4}$, $optimizer = Adam$ and batch size calculated for each model based on GPU memory available) where used to train them on this final stage.

(a) p_road=1.00 (b) p_road=1.00 (c) p_road=0.00 (d) p_road=0.00

(e) p_road=1.00 (f) p_road=0.95 (g) p_road=1.00 (h) p_road=1.00

(i) p_road=0.03 (j) p_road=0.96 (k) p_road=1.00 (l) p_road=0.00

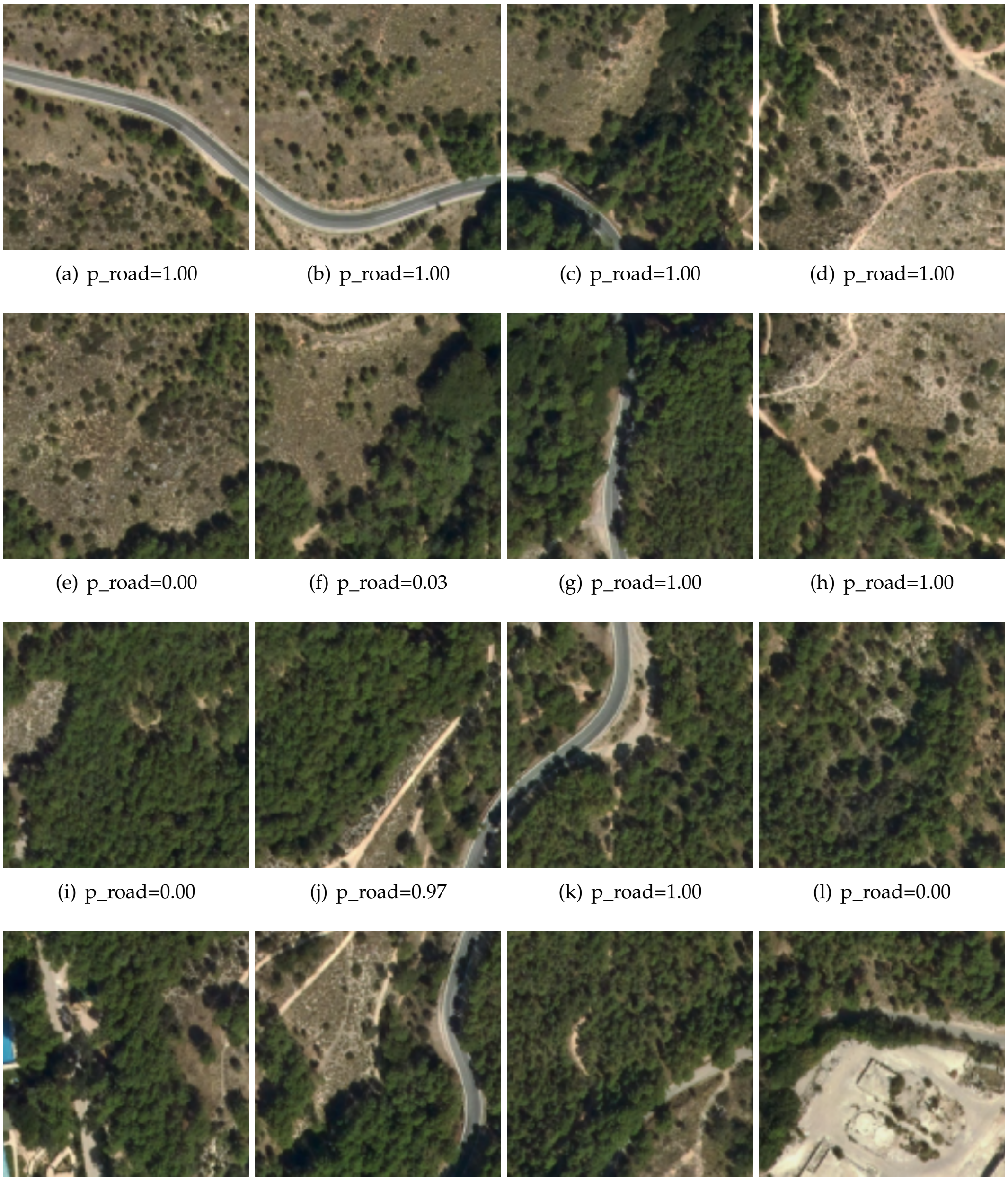
Figure 6. Sample predictions for randomly selected section of map outside the dataset. p_road shows the network prediction confidence for the class "contains road" for each image.

Table 2. K-fold test accuracy results for best designed networks (selected by validation accuracy) for the IGN-PNOA dataset.

Model	k	Accuracy min	Accuracy max	Accuracy avg	Accuracy stdev
v1m1	5	0.784	0.857	0.829	0.0284
	10	0.551	0.918	0.833	0.103
v1m2	5	0.788	0.890	0.860	0.042
	10	0.714	0.892	0.825	0.058
v1m3	5	0.777	0.879	0.843	0.040
	10	0.699	0.909	0.812	0.077
v1m4	5	0.712	0.871	0.810	0.073
	10	0.808	0.877	0.852	0.021
v1m5	5	0.716	0.874	0.829	0.0657
	10	0.851	0.892	0.874	0.012
v1m6	5	0.788	0.877	0.845	0.036
	10	0.777	0.913	0.867	0.0362
v2m1	**5**	**0.895**	**0.922**	**0.915**	**0.011**
	10	**0.843**	**0.931**	**0.907**	**0.033**
v2m2	**5**	**0.906**	**0.931**	**0.922**	**0.010**
	10	**0.910**	**0.938**	**0.927**	**0.007**
v2m3	**5**	**0.889**	**0.935**	**0.916**	**0.017**
	10	**0.900**	**0.937**	**0.925**	**0.012**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2020 doi:10.20944/preprints202005.0002.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 3953; doi:10.3390/app10113953



(a) p_road=1.00 (b) p_road=1.00 (c) p_road=1.00 (d) p_road=1.00

(e) p_road=0.00 (f) p_road=0.03 (g) p_road=1.00 (h) p_road=1.00

(i) p_road=0.00 (j) p_road=0.97 (k) p_road=1.00 (l) p_road=0.00

(m) p_road=0.98 (n) p_road=1.00 (o) p_road=1.00 (p) p_road=1.00

Figure 7. Sample predictions for randomly selected section of map outside the dataset. p_road shows the network prediction confidence for the class "contains road" for each image.

Table 3. Test accuracy 95% t confidence intervals for all models calculated from k-fold results for the IGN-PNOA dataset. For each model and k value we show the calculated accuracy expected interval mean, standard deviation and the interval itself.

Model	Test accuracy 95% t Confidence Interval (k-fold k=5)	Test accuracy t 95% Confidence Interval (k-fold k=10)
v1m1	0.823 ± 0.035 = [0.793, 0.864]	0.833 ± 0.074 = [0.759, 0.907]
v1m2	0.860 ± 0.052 = [0.808, 0.912]	0.826 ± 0.042 = [0.784, 0.867]
v1m3	0.843 ± 0.050 = [0.793, 0.893]	0.812 ± 0.055 = [0.757, 0.867]
v1m4	0.810 ± 0.091 = [0.719, 0.901]	0.852 ± 0.015 = [0.836, 0.867]
v1m5	0.829 ± 0.082 = [0.747, 0.911]	0.874 ± 0.009 = [0.865, 0.882]
v1m6	0.845 ± 0.045 = [0.800, 0.890]	0.867 ± 0.026 = [0.842, 0.893]
v2m1	**0.915 ± 0.013 = [0.901, 0.928]**	**0.907 ± 0.024 = [0.884, 0.931]**
v2m2	**0.922 ± 0.012 = [0.910, 0.934]**	**0.927 ± 0.005 = [0.922, 0.932]**
v2m3	**0.916 ± 0.021 = [0.894, 0.937]**	**0.925 ± 0.008 = [0.917, 0.934]**

Peer-reviewed version available at Appl. Sci. 2020, 10, 3953; doi:10.3390/app10113953

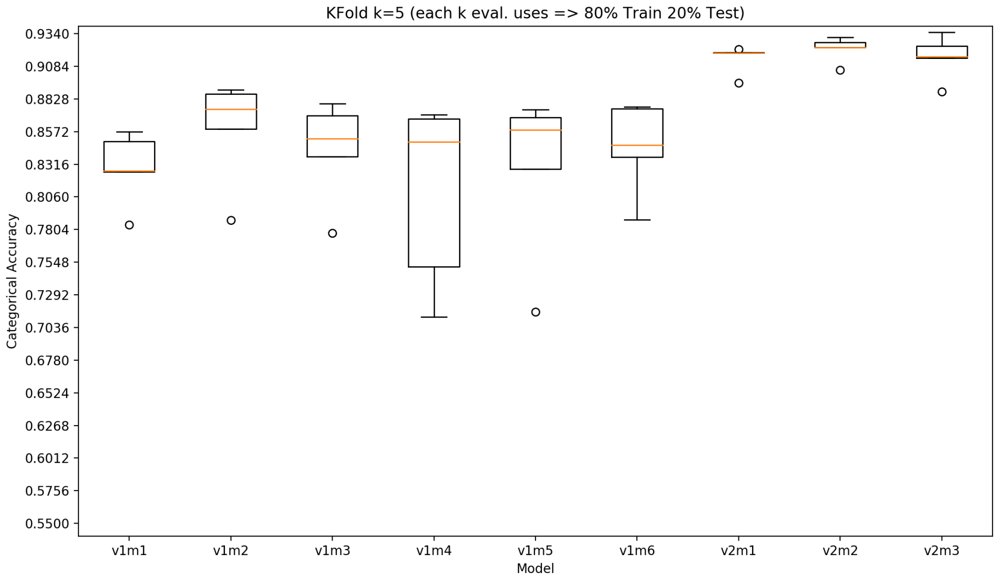


Figure 8. K-fold (k=5) test accuracy for selected models on the IGN-PNOA dataset.

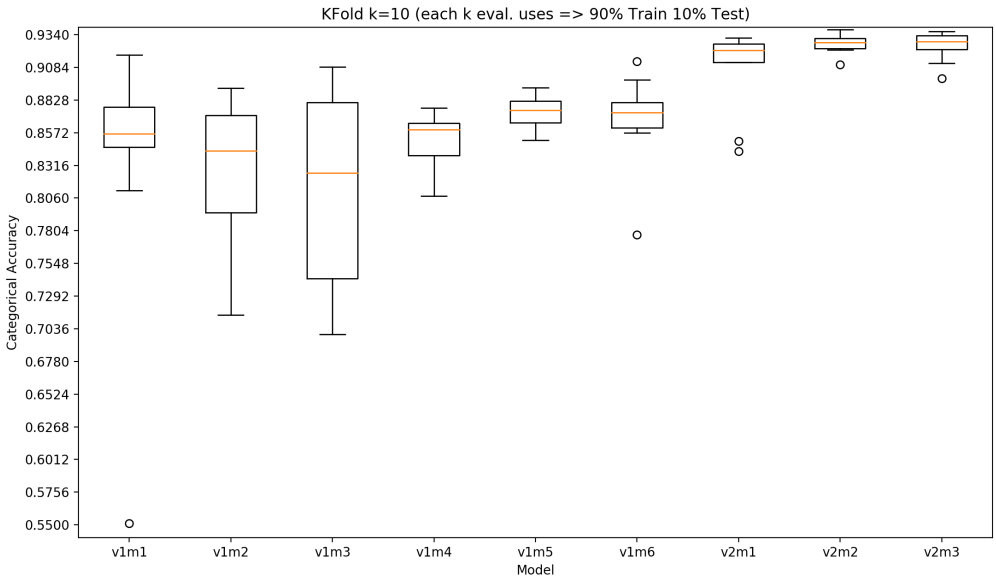


Figure 9. K-fold (k=10) test accuracy for selected models on the IGN-PNOA dataset.

Finally, in order to give a point of common reference for other authors to be able to compare results, we run preliminary tests on a typical NAS benchmark, the CIFAR-10 dataset (we set the maximum number of FC layers to 3 on this experiments). Here the best model (selected by validation loss) designed by VGNet obtained test categorical accuracies that oscillate on the range shown on Table 4.

Table 4. Test metrics for best VGNet's designed model for the CIFAR10 dataset.

Model	accuracy mean	accuracy stdev	accuracy mean, with data augmentation	accuracy stdev, with data augmentation
c10g49m2	0.911	0.003	0.918	0.002

4. Discussion and Future Work

Results found by [41] applying transfer learning techniques over a similar version of this dataset (in this case using partitioning train/validation/test with ratio 50:25:25) report test accuracies for some well known architectures like Inception-v3 (87.88%-91.19%) or Inception-ResNet (91.28%-92.53%). On [43] mean accuracies are reported, among others for ResNet-50 (91.7%), VGG (93.9%) and for two different ensembles (94.6% and 95.6% respectively). Here ensemble refers to the combinations of multiple networks into one meta-classifier, namely their results are combined in order to improve predictions.

Our system was able to successfully design full networks that, trained from scratch (without any manual fine-tuning nor data augmentation or application of ensembles), yielded for the best scoring model a test accuracy of 90.6%-93.1% with kfold (k=5) and 91%-93.8% (with k=10). Therefore the GGGP NAS approach shows results similar to well-know architectures. The main withdrawal we find is the time it takes to run the evolutive algorithm. We should note that, as mentioned above, some of the meta-grammatic parameters were set in our tests according to hardware constraints (e.g. GPU memory) and therefore should not be interpreted as optimal settings in any case.

On the other hand, meta-grammars allow new applications of the system as a hypothesis checker using different variations of meta-grammars and comparing the results obtained. For example: is it better to apply desertion only in a certain stage of the architecture? or are desertion rates more adequate

We are currently working on methods to speed up the design process. From reducing the number of epochs at early design stages to using caching methods for the convolutional block weights or migrating to training on cloud servers. Research is also being conducted on ways to gain knowledge reusable among different evolution experiments. Finally regarding improving the method exploration/exploitation during the NAS future research will further explore enhancements to the presented SDF mutation operator.

Author Contributions: Conceptualization, V.F.C, A.D-A. and F.S.G.; methodology, V.F.C., A.D-A. and F.S.G.; software, V.F.C.; validation, V.F.C., A.D-A. and F.S.G.; formal analysis, V.F.C.; investigation, V.F.C.; resources, V.F.C., A.D-A., M-A.M-C. and F.S.G.; data curation, V.F.C.; writing–original draft preparation, V.F.C.; writing–review and editing, V.F.C, A.D-A., M-A.M-C. and F.S.G.; visualization, V.F.C; supervision, F.S.G.; project administration, M-A.M-C.; funding acquisition, F.S.G. and M-A.M-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Instituto Geográfico Nacional (IGN, Spain).

Acknowledgments: To the staff at the Instituto Geográfico Nacional (IGN, Spain) and all team members of the Cartobot project. To Carolina Temprado for her support and graphical design tips. To José Eugenio Naranjo for additional draft reviewing and suggestions.

Conflicts of Interest: The founders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Risojević, V.; Momić, S.; Babić, Z. Gabor descriptors for aerial image classification. International Conference on Adaptive and Natural Computing Algorithms. Springer, 2011, pp. 51–60.
2. Topouzelis, K.; Papakonstantinou, A.; Doukari, M. Coastline change detection using Unmanned Aerial Vehicles and image processing technique. *Fresen. Environ. Bull* **2017**. Query date: 2020-04-10 12:57:44.

3. Korzeniowska, K.; Bühler, Y.; Marty, M.; .... Regional snow-avalanche detection using object-based image analysis of near-infrared aerial imagery. *Natural Hazards & Earth ...* **2017**. Query date: 2020-04-10 12:57:44.
4. Yuan, C.; Ghamry, K.; Liu, Z.; .... Unmanned aerial vehicle based forest fire monitoring and detection using image processing technique. *2016 IEEE Chinese ...* **2016**. Query date: 2020-04-10 12:57:44.
5. Zacharie, M.; Fuji, S.; Minori, S. Rapid Human Body Detection in Disaster Sites Using Image Processing from Unmanned Aerial Vehicle (UAV) Cameras. *2018 International Conference on ...* **2018**. Query date: 2020-04-10 12:57:44.
6. Selim, S.; Sonmez, N.; Coslu, M.; Onur, I. Semi-automatic tree detection from images of unmanned aerial vehicle using object-based image analysis method. *Journal of the Indian Society of ...* **2019**. Query date: 2020-04-10 12:57:44.
7. Buters, T.; Belton, D.; Cross, A. Seed and seedling detection using unmanned aerial vehicles and automated image classification in the monitoring of ecological recovery. *Drones* **2019**. Query date: 2020-04-10 12:57:44.
8. Alidoost, F.; Arefi, H. A CNN-Based Approach for Automatic Building Detection and Recognition of Roof Types Using a Single Aerial Image. *PFG–Journal of Photogrammetry, Remote Sensing ...* **2018**. Query date: 2020-04-10 12:57:44.
9. Dutta, T.; Sharma, H.; Vellaiappan, A.; .... Image analysis-based automatic detection of transmission towers using aerial imagery. *... recognition and image ...* **2015**. Query date: 2020-04-10 12:57:44.
10. Tragulnuch, P.; Chanvimaluang, T.; .... High Voltage Transmission Tower Detection and Tracking in Aerial Video Sequence using Object-Based Image Classification. *... on Information and ...* **2018**. Query date: 2020-04-10 12:57:44.
11. Lu, J.; Ma, C.; Li, L.; Xing, X.; Zhang, Y.; Wang, Z.; .... A Vehicle Detection Method for Aerial Image Based on YOLO. *Journal of Computer and ...* **2018**. Query date: 2020-04-10 12:57:44.
12. Cao, Y.; Wang, G.; Yan, D.; Zhao, Z. Two algorithms for the detection and tracking of moving vehicle targets in aerial infrared image sequences. *Remote Sensing* **2016**. Query date: 2020-04-10 12:57:44.
13. Malof, J.; Bradbury, K.; Collins, L.; .... Image features for pixel-wise detection of solar photovoltaic arrays in aerial imagery using a random forest classifier. *2016 IEEE ...* **2016**. Query date: 2020-04-10 12:57:44.
14. Yu, J.; Wang, Z.; Majumdar, A.; Rajagopal, R. DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States. *Joule* **2018**, *2*, 2605–2617.
15. Shorter, N.; Kasparis, T. Automatic vegetation identification and building detection from a single nadir aerial image. *Remote Sensing* **2009**, *1*, 731–757.
16. Fortier, A.; Ziou, D.; Armenakis, C.; Wang, S. Survey of work on road extraction in aerial and satellite images. *Center for Topographic Information Geomatics, Ontario, Canada. Technical Report* **1999**, *241*.
17. Wang, J.; Qin, Q.; Gao, Z.; Zhao, J.; Ye, X. A new approach to urban road extraction using high-resolution aerial image. *ISPRS International Journal of Geo-Information* **2016**, *5*, 114.
18. Mattyus, G.; Wang, S.; Fidler, S.; Urtasun, R. Enhancing road maps by parsing aerial images around the world. Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1689–1697.
19. Tsochantaridis, I.; Joachims, T.; Hofmann, T.; Altun, Y. Large margin methods for structured and interdependent output variables. *Journal of machine learning research* **2005**, *6*, 1453–1484.
20. Zhou, H.; Kong, H.; Wei, L.; Creighton, D.; Nahavandi, S. Efficient road detection and tracking for unmanned aerial vehicle. *IEEE transactions on intelligent transportation systems* **2014**, *16*, 297–309.
21. Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence* **2001**, *23*, 1222–1239.
22. Lin, Y.; Saripalli, S. Road detection from aerial imagery. 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 3588–3593.
23. Unsalan, C.; Sirmacek, B. Road network detection using probabilistic and graph theoretical methods. *IEEE Transactions on Geoscience and Remote Sensing* **2012**, *50*, 4441–4453.
24. Hu, J.; Razdan, A.; Femiani, J.C.; Cui, M.; Wonka, P. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing* **2007**, *45*, 4144–4157.
25. Hinz, S.; Baumgartner, A. Automatic extraction of urban road networks from multi-view aerial imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* **2003**, *58*, 83–98.
26. Trinder, J.C.; Wang, Y. Automatic road extraction from aerial images. *Digital Signal Processing* **1998**, *8*, 215–224.

27. Airault, S.; Ruskone, R.; Jamet, O. Road detection from aerial images: a cooperation between local and global methods. Image and Signal Processing for Remote Sensing. International Society for Optics and Photonics, 1994, Vol. 2315, pp. 508–518.
28. Zlotnick, A.; Carnine, P. Finding road seeds in aerial images. *CVGIP: image understanding* **1993**, *57*, 243–260.
29. Bastani, F.; He, S.; Abbar, S.; Alizadeh, M.; Balakrishnan, H.; Chawla, S.; Madden, S.; DeWitt, D. Roadtracer: Automatic extraction of road networks from aerial images. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4720–4728.
30. Máttyus, G.; Luo, W.; Urtasun, R. Deeproadmapper: Extracting road topology from aerial images. Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3438–3446.
31. Wang, J.; Song, J.; Chen, M.; Yang, Z. Road network extraction: A neural-dynamic framework based on deep learning and a finite state machine. *International Journal of Remote Sensing* **2015**, *36*, 3144–3169.
32. Mnih, V.; Hinton, G.E. Learning to detect roads in high-resolution aerial images. European Conference on Computer Vision. Springer, 2010, pp. 210–223.
33. Hu, X.y.; Zhang, Z.x.; Zhang, J. An approach of semiautomated road extraction from aerial image based on template matching and neural network. *International Archives of Photogrammetry and Remote Sensing* **2000**, *33*, 994–999.
34. Ichim, L.; Popescu, D. Road detection and segmentation from aerial images using a CNN based system. 2018 41st International Conference on Telecommunications and Signal Processing (TSP). IEEE, 2018, pp. 1–5.
35. Cheng, G.; Wang, Y.; Xu, S.; Wang, H.; Xiang, S.; Pan, C. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing* **2017**, *55*, 3322–3337.
36. Saito, S.; Aoki, Y. Building and road detection from large aerial imagery. Image Processing: Machine Vision Applications VIII. International Society for Optics and Photonics, 2015, Vol. 9405, p. 94050K.
37. Zhang, Z.; Liu, Q.; Wang, Y. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters* **2018**, *15*, 749–753.
38. Rao, Y.; Liu, W.; Pu, J.; Deng, J.; .... Roads detection of aerial image with FCN-CRF model. *... and Image Processing ...* **2018**. Query date: 2020-04-10 12:57:44.
39. Wei, Y.; Wang, Z.; Xu, M. Road structure refined CNN for road extraction in aerial image. *IEEE Geoscience and Remote Sensing Letters* **2017**, *14*, 709–713.
40. Zhong, Z.; Li, J.; Cui, W.; Jiang, H. Fully convolutional networks for building and road extraction: Preliminary results. 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, 2016, pp. 1591–1594.
41. Cira, C.I.; Alcarria, R.; Manso-Callejo, M.Á.; Serradilla, F. Evaluation of Transfer Learning Techniques with Convolutional Neural Networks (CNNs) to Detect the Existence of Roads in High-Resolution Aerial Imagery. Applied Informatics; Florez, H.; Leon, M.; Diaz-Nafria, J.M.; Belli, S., Eds.; Springer International Publishing: Cham, 2019; pp. 185–198.
42. Cira, C.I.; Alcarria, R.; Manso-Callejo, M.Á.; Serradilla, F. A Deep Convolutional Neural Network to Detect the Existence of Geospatial Elements in High-Resolution Aerial Imagery. *Proceedings* **2019**, *19*, 17. doi:10.3390/proceedings2019019017.
43. Cira, C.I.; Alcarria, R.; Manso-Callejo, M.Á.; Serradilla, F. A Framework Based on Nesting of Convolutional Neural Networks to Classify Secondary Roads in High Resolution Aerial Orthoimages. *Remote Sensing* **2020**, *12*, 765.
44. Volpi, M.; Tuia, D. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing* **2017**, *55*, 881–893.
45. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
46. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. *CoRR* **2016**, *abs/1611.01578*, [1611.01578].
47. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. *CoRR* **2017**, *abs/1707.07012*, [1707.07012].

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 3953; doi:10.3390/app10113953

48. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via Parameter Sharing. *CoRR* **2018**, *abs/1802.03268*, [1802.03268].
49. Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; Kavukcuoglu, K. Hierarchical Representations for Efficient Architecture Search. *CoRR* **2017**, *abs/1711.00436*, [1711.00436].
50. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized Evolution for Image Classifier Architecture Search. *CoRR* **2018**, *abs/1802.01548*, [1802.01548].
51. Liang, J.; Meyerson, E.; Hodjat, B.; Fink, D.; Mutch, K.; Miikkulainen, R. Evolutionary Neural AutoML for Deep Learning. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2019), 2019.
52. Assunçao, F.; Lourenço, N.; Machado, P.; Ribeiro, B. DENSER: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines* **2019**, *20*, 5–35.
53. Lima, R.H.R.; Pozo, A.T.R. Evolving Convolutional Neural Networks through Grammatical Evolution. Proceedings of the Genetic and Evolutionary Computation Conference Companion; Association for Computing Machinery: New York, NY, USA, 2019; GECCO '19, pp. 179–180. doi:10.1145/3319619.3322058.
54. Couchet, J.; Manrique, D.; Porras, L. Grammar-guided neural architecture evolution. International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer, 2007, pp. 437–446.
55. Tsoulos, I.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269 – 277. Machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007), doi:https://doi.org/10.1016/j.neucom.2008.01.017.
56. Jacob, C.; Rehder, J. Evolution of neural net architectures by a hierarchical grammar-based genetic system. Artificial Neural Nets and Genetic Algorithms. Springer, 1993, pp. 72–79.
57. Mahmoudi, M.T.; Taghiyareh, F.; Forouzideh, N.; Lucas, C. Evolving artificial neural network structure using grammar encoding and colonial competitive algorithm. *Neural Computing and Applications* **2013**, *22*, 1–16.
58. Assunção, F.; Lourenço, N.; Machado, P.; Ribeiro, B. Automatic generation of neural networks with structured Grammatical Evolution. 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1557–1564.
59. Soltanian, K.; Tab, F.A.; Zar, F.A.; Tsoulos, I. Artificial neural networks generation using grammatical evolution. 2013 21st Iranian Conference on Electrical Engineering (ICEE), 2013, pp. 1–5.
60. Ahmadizar, F.; Soltanian, K.; AkhlaghianTab, F.; Tsoulos, I. Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence* **2015**, *39*, 1 – 13. doi:https://doi.org/10.1016/j.engappai.2014.11.003.
61. Ahmad, Q.; Rafiq, A.; Raja, M.A.; Javed, N. Evolving MIMO Multi-Layered Artificial Neural Networks Using Grammatical Evolution. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing; Association for Computing Machinery: New York, NY, USA, 2019; SAC '19, pp. 1278–1285. doi:10.1145/3297280.3297408.
62. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, pp. 1097–1105.
63. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.
64. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
65. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
66. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-resnet and the impact of residual connections on learning. AAAI, 2017, Vol. 4, p. 12.
67. Chen, Y.; Li, J.; Xiao, H.; Jin, X.; Yan, S.; Feng, J. Dual path networks. Advances in Neural Information Processing Systems, 2017, pp. 4467–4475.
68. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. IEEE, 2017, pp. 5987–5995.

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 3953; doi:10.3390/app10113953

69. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.C.; Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **2015**, *115*, 211–252. doi:10.1007/s11263-015-0816-y.
70. Whigham, P.A.; others. Grammatically-based genetic programming. Proceedings of the workshop on genetic programming: from theory to real-world applications, 1995, Vol. 16, pp. 33–41.
71. García-Arnau, M.; Manrique, D.; Rios, J.; Rodríguez-Patón, A. Initialization method for grammar-guided genetic programming. International Conference on Innovative Techniques and Applications of Artificial Intelligence. Springer, 2006, pp. 32–44.
72. Koza, J.R.; Koza, J.R. *Genetic programming: on the programming of computers by means of natural selection*; Vol. 1, MIT press, 1992.
73. Couchet, J.; Manrique, D.; Ríos, J.; Rodríguez-Patón, A. Crossover and mutation operators for grammar-guided genetic programming. *Soft Computing* **2007**, *11*, 943–955.
74. Goldberg, D.E.; Deb, K. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. FOGA, 1990.
75. Hingee, K.; Hutter, M. Equivalence of probabilistic tournament and polynomial ranking selection. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008, pp. 564–571. doi:10.1109/CEC.2008.4630852.
76. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
77. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*; Cortes, C.; Lawrence, N.D.; Lee, D.D.; Sugiyama, M.; Garnett, R., Eds.; Curran Associates, Inc., 2015; pp. 91–99.
78. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* **2012**, *abs/1207.0580*, [1207.0580].
79. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* **2015**, *abs/1502.03167*, [1502.03167].
80. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. Proc. icml, 2013, Vol. 30, p. 3.
81. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An introduction to statistical learning*; Vol. 112, Springer, 2013.
82. Kuhn, M.; Johnson, K. *Applied predictive modeling*; Vol. 26, Springer, 2013.
83. Molinaro, A.M.; Simon, R.; Pfeiffer, R.M. Prediction error estimation: a comparison of resampling methods. *Bioinformatics* **2005**, *21*, 3301–3307.
84. Caffo, B. Statistical inference for data science. *Victoria, BC: Leanpub* **2015**.