WebArrayDB: A Geospatial Array DBMS in Your Web Browser

https://wikience.github.io/webdb2022

Ramon Antonio Rodriges Zalipynis
HSE University
Moscow, Russia
rodriges@gis.land;arodriges@hse.ru

Nikita Terlych
HSE University
Moscow, Russia
naterlych@edu.hse.ru

ABSTRACT

Geospatial array DBMSs operate on georeferenced N-d arrays. They provide storage engines, query parsers, and processing capabilities as their core functionality. Traditionally, those have been too heavy for a Web browser to support. Hence, Web Applications, mostly Geographic Information Systems (GISs), run array management on their server back-ends that return small portions of the results via the network. We showcase WebArrayDB, the first geospatial array DBMS that can run completely inside a Web browser. We demonstrate that modern Web browsers, JavaScript, and respective software libraries are sufficiently mature to build and run such a feature-rich and powerful DBMS. A Web-based array DBMS should reduce server load, enable offline work, decrease network I/O, and improve user experience. We also present ArrayGIS, our new Web GIS based on WebArrayDB, and invite everyone to explore both via a freely accessible, informative, and interactive Web GUI.

PVLDB Reference Format:
Ramon Antonio Rodriges Zalipynis and Nikita Terlych. WebArrayDB: A Geospatial Array DBMS in Your Web Browser. PVLDB, 15(12): 3622 - 3625, 2022.
doi:10.14778/3554821.3554859

1 INTRODUCTION

Array DBMSs represent an important class of systems for efficient management, processing, and even visualization of N-d arrays [7]. Array DBMSs are experiencing an R&D surge: dozens of array systems have recently been proposed [7] with diverse focuses ranging from machine learning [12] to physical simulations [8]. Many array-centric problems, like top-k queries [2] and tunable queries [6], have recently been tackled for the first time.

An N-d array is a natural model for numerous important data types, e.g. satellite imagery, which is actively used for urban planning, emergency response, and food security, to name a few [1]. Hence, geospatial array management constitutes a large portion of array DBMS workloads. For example, Sentinels is a family of European satellite missions. About 203 TiB of Sentinel products are disseminated daily with total downloads of 80.5 PiB/year [9].

The rapid growth of big array data stimulates the development of client-server applications. Whereas servers run in a Cloud and perform the required array processing, web browsers are becoming increasingly popular platforms for client applications: Web GISs (Geographic Information Systems). However, they are still not mature as the power of modern web browsers is largely underutilized.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554859

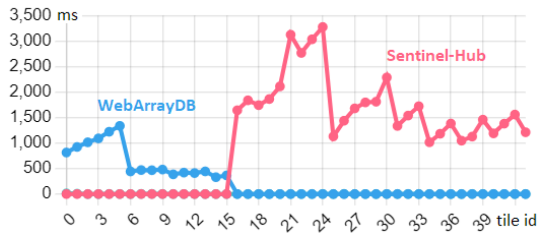


Figure 1: Latency: WebArrayDB vs. Sentinel-Hub

Modern Web GISs run all array processing on the server side, e.g., ArcGIS[1], Mango Map[2], NextGIS Web[3], GIS Cloud[4], Mapbox[5], and Carto[6]. They use proprietary and/or open geospatial protocols to send queries (array processing commands) to their servers and retrieve results in small portions (e.g., 2-d image tiles to display in a web browser on the client side). This increases response times, up to several seconds, significantly degrading the user experience.

Companies claim that only 0.1s of reduction in latency can influence the user journey and ultimately increases conversion rates [3]. We argue that Web GISs can submit loads of array queries to an array DBMS running entirely in a web browser to drastically reduce query response times. A Web GIS that relies on WebArrayDB can be over 2× faster compared to querying only Sentinel-Hub [10], a Cloud service for disseminating and processing Sentinel data, fig. 1.

WebArrayDB currently supports data ingestion over the `WMTS` protocol, provides a storage engine, SQL-like query syntax, and executes local map algebra operations directly in a web browser. For locally stored arrays, WebArrayDB enables ArrayGIS to adjust color palettes, get source array cell values under the mouse cursor, and run array computations completely in a web browser. Besides the latency reduction for improved user experience, other important benefits include client hardware utilization (e.g., GPU via WebGL) and reduced server load (potentially more clients with less cost). We are on the way to integrating vector data processing in ArrayGIS.

Our major contributions include: (1) we describe WebArrayDB, the first geospatial array DBMS in pure JavaScript that runs entirely in a Web browser; (2) we describe ArrayGIS, a novel Web GIS based on WebArrayDB; (3) we demonstrate WebArrayDB and ArrayGIS by providing free access and by offering interactive lessons related to real-world tasks on multispectral satellite imagery.

[1]https://www.arcgis.com [2]https://mangomap.com [3]https://nextgis.ru/nextgis-web [4]https://www.giscloud.com [5]https://www.mapbox.com [6]https://carto.com

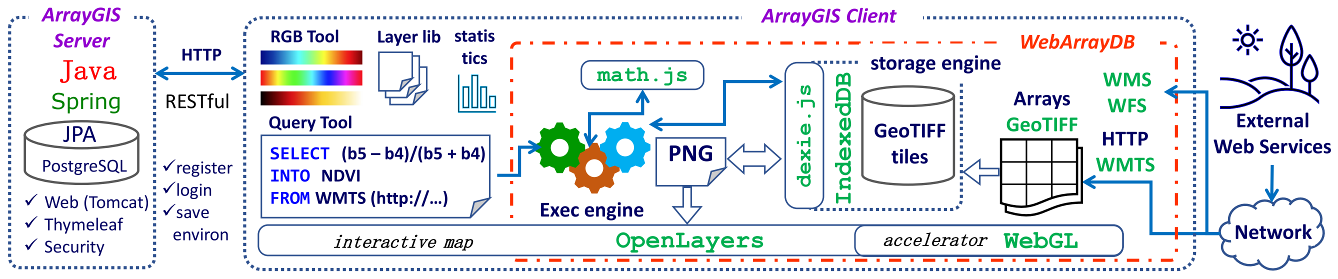


Figure 2: WebArrayDB and ArrayGIS Architectures

2 WEBARRAYDB OVERVIEW

Now, we briefly overview WebArrayDB, the first array DBMS running completely in a Web browser, fig. 2: http://webdb.gis.gg

Data Formats. For decades, Web GIS applications used widespread graphical formats for the Web like PNG or JPEG. These formats allowed a web browser only to visualize arrays rendered on the server side; web clients did not freely operate on raw array data. Hence, even basic queries like color adjustment required client-server communication and retrieval of updated images.

A key reason for such an architecture was the complexity of geospatial data formats that could not be addressed by web browsers. For example, **`GeoTIFF`**[7] supports diverse compression techniques, georeferencing, internal tiling, pixel/band interleaving, and color spaces, to name a few. However, recent advancements in web development made it possible to create **`geotiff.js`**[8]: a small library in pure JavaScript for reading **`GeoTIFF`** files.

WebArrayDB utilizes **`geotiff.js`** for working with **`GeoTIFF`** array tiles. However, we still also use the aforementioned image file formats for visualizing arrays in **`OpenLayers`**[9].

Data Ingestion. As we are running an array DBMS in a web browser, we should expect other web services (not local files) to be its primary data sources. **`WMTS`** (Web Map Tile Service) is a popular protocol for serving georeferenced map tiles over the HTTP [5].

A widespread **`WMTS`**[10] use-case is to retrieve map tiles in PNG or a similar file format. However, the **`WMTS`** standard foresees the dissemination of map tiles in more sophisticated file formats that can carry original array data, like **`GeoTIFF`**, and initially intended for desktop applications.

WebArrayDB uses **`OpenLayers`** to negotiate with Web services over the **`WMTS`** protocol, e.g., retrieve and parse `capabilities.xml`. However, instead of PNG files, WebArrayDB ingests **`GeoTIFF`** files with **`geotiff.js`**. WebArrayDB performs (1) on-the-fly tile-by-tile conversion of **`GeoTIFF`** files to the structure supported by **`OpenLayers`** (for visualization), and (2) saves **`GeoTIFF`** files into the WebArrayDB storage engine for future use (see below).

Array Storage Engine relies on **`IndexedDB`**[11] and **`dexie.js`**[12] (a minimalistic JavaScript wrapper for **`IndexedDB`**). **`IndexedDB`** is a JavaScript API for storing significant amounts of structured data in a web browser. It is attractive for a web-based array DBMS as it can also store BLOBs and uses indexes for fast searches.

[7]https://trac.osgeo.org/geotiff [8]https://geotiffjs.github.io [9]https://openlayers.org
[10]https://www.ogc.org/standards/wmts [11]https://w3c.github.io/IndexedDB
[12]https://dexie.org [13]https://mathjs.org [14]https://www.khronos.org/webgl

In its storage engine, WebArrayDB keeps raw array tiles (N-d arrays) together with some metadata, e.g., URL key and extent. A parameter controls the array data volume that WebArrayDB can keep on the client side. The least frequently used array tile is removed when the space runs out.

Query Parsing. WebArrayDB uses an SQL-like query syntax, similar to AQL or RasQL [7]. A query example is given below. It computes NDVI, a popular vegetation index [13]. As input, the query takes 2-d arrays: Sentinel bands 8 and 4. The `FROM` clause instructs WebArrayDB to retrieve the input arrays from a remote service via the **`WMTS`** protocol. The result, a new 2-d array called `NDVI`, will be saved by the storage engine in the web browser (**`IndexedDB`**).

```
SELECT (band8 - band4)/(band8 + band4)
INTO NDVI
FROM WMTS (https://services.sentinel-hub.com/ogc/wmts/
     <personal_api_key>?REQUEST=GetCapabilities)
```

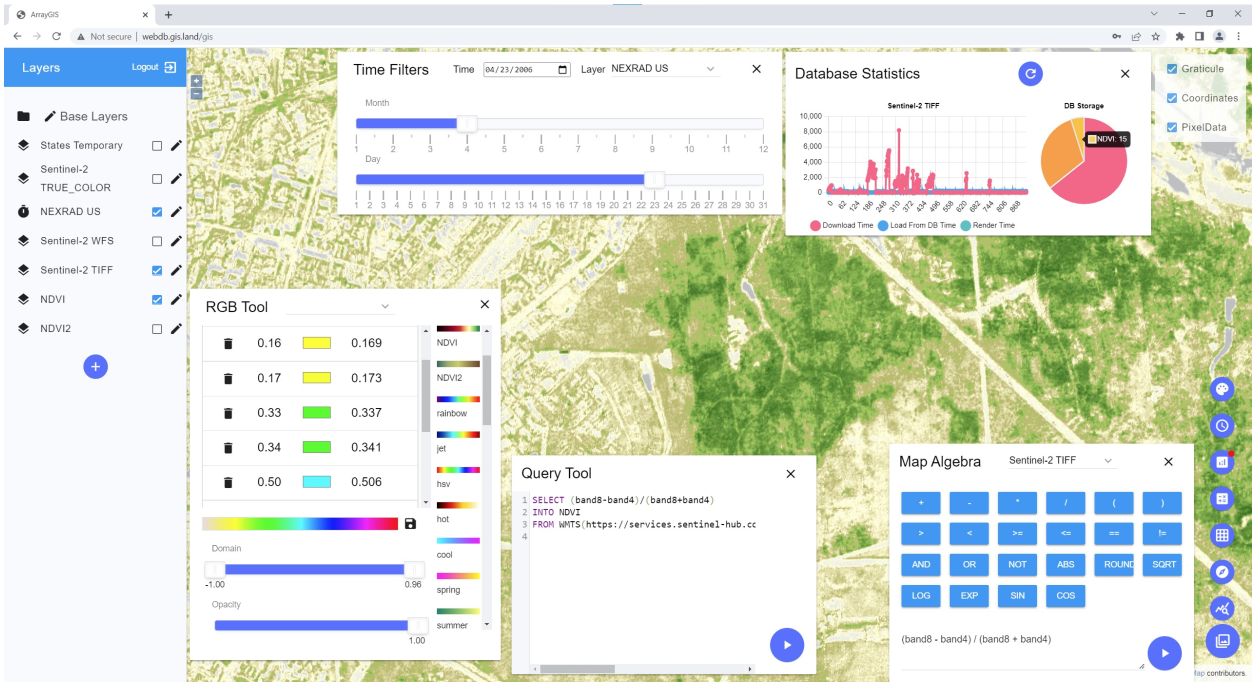
To parse queries, we use **`math.js`**[13]. It is a purely JavaScript library that supports symbolic computations, a large set of built-in functions, and different data types like numbers, units, and matrices.

Query Execution plans consist of the following phases: (1) load, (2) join, (3) compute, and (4) render. ArrayGIS asks WebArrayDB to emit only those resulting array portions which will be immediately visible to the user. When they pan/zoom the map, ArrayGIS and WebArrayDB generate new resulting tiles on-the-fly.

WebArrayDB loads array tiles from its storage engine or requests those via **`WMTS`** if they are absent locally. Then, WebArrayDB performs array join (see below) if several arrays participate in the query. Next, the resulting array values are computed running the code written in pure JavaScript. Finally, a map tile image (PNG) corresponding to the raw array tile (numeric values) is rendered via **`WebGL`**[14] (a GPU-based accelerator) and displayed in **`OpenLayers`**.

Array Joins are vital for queries involving multiple arrays [7]. WebArrayDB supports extracting input arrays (1) from a single **`WMTS`** response or (2) different layers. In the latter case, an output array is tiled using the smallest input tile. Instead of full retiling [4], we emit a tile by taking all source tiles overlapping the output tile.

Performance Evaluation. Given a query, resulting map tiles are emitted on-the-fly. Latency is the time to emit a tile (steps 1–4), up to the moment the user sees it on the screen, fig. 1. For the NDVI query, tiles № 0–15 happened to be in the WebArrayDB storage engine (locally, emitted in ≈0.5–1.5s), while tiles № 16–43 had to be retrieved from Sentinel-Hub (remotely, emitted in ≈1–3.5s).



🗑	0.16	0.169
🗑	0.17	0.173
🗑	0.33	0.337
🗑	0.34	0.341
🗑	0.50	0.506

```
SELECT (band8-band4)/(band8+band4)
INTO NDVI
FROM WMTS(https://services.sentinel-hub.cc
```

(band8 - band4) / (band8 + band4)

Figure 3: ArrayGIS GUI, available at https://wikience.github.io/webdb2022

3 ARRAYGIS OVERVIEW

ArrayGIS is a front-end for WebArrayDB that features interactive Web GUI and is freely accessible at http://webdb.gis.gg

Layer Library (Layers panel on the left) allows users to add (round button with +), turn on/off 2-d (latitude, longitude) and 3-d (time, latitude, longitude) layers. ArrayGIS supports WMS, WFS, and WMTS protocols that provide layers with or without temporal axes. WMTS can deliver tiles in GeoTIFF or as already rendered images. As an example, we added several layers that represent all supported layer types; the Nexrad US layer has a temporal axis.

A pencil button next to each layer opens a dialog where users can explore layer properties (is temporal, is visible, opacity percent).

Toolbox Gallery (a series of vertically aligned buttons) opens by pressing the large round button in the bottom right corner. Each smaller round button views/hides the respective toolbox.

Time Travel Toolbox controls the display of layers with temporal axes. A user can adjust year, month, and day with the respective slider or enter the date as a string in the calendar box.

Interactive Map displays layer data, provides panning/zooming, and respects cartographic projections. With the checkboxes in the top right, users can turn on/off the graticule and the display of current map coordinates (in the top left corner) of the mouse cursor.

Value under cursor is an extremely important capability for improved user experience. As ArrayGIS operates on raw source data, it does not require client-server communication and is lightning-fast in displaying array cell values. The user should tick the "Pixel data" checkbox in the top right corner to display source cell values (in the bottom left corner) on mouse click events.

At a glance, this feature is simple. However, it respects current cartographic projection and zoom level and takes data from the WebArrayDB storage. For WMTS, we extended `OpenLayers` to get global coordinates, convert them to the tile index, and then within the tile itself. It took about 100 lines of code.

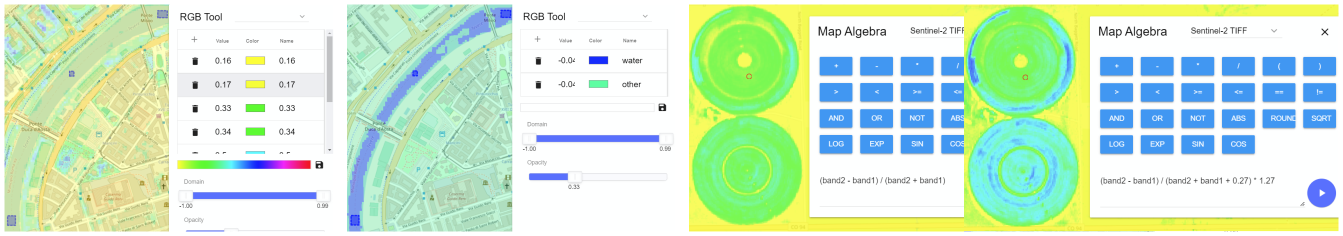
RGB Tool is another vivid example of the benefits of having source array data on the client side. Unlike other Web GISs, ArrayGIS can set/tune color palettes without client-server communication: a layer in new colors is re-rendered in a split second.

The user should select a layer, for which a color palette must be set/tuned, from the drop-down box. ArrayGIS offers a collection of predefined palettes. The user can modify an existing palette or create a new one for future use. New and modified palettes can be saved on the server side. The user can add/remove/re-arrange colors and change opacity. The layer is redrawn on the map on-the-fly.

Map Algebra is an analysis language loosely based on the concepts presented in [11]. Map Algebra is one of the most frequent classes of Geospatial Array DBMS queries [4]. Source array data are on the client side, so ArrayGIS and WebArrayDB run computations directly in a web browser. It takes less than 10 ms to compute a vegetation index [13] for a map tile, much less than a client-server communication. This toolbox lets users quickly apply map algebra on-the-fly to the selected layer without creating a new layer.

Query Tool accepts SQL-like queries. Syntax highlight is supported. The user can create a new layer with a query. ArrayGIS and WebArrayDB execute queries on raw array data, see section 2. It is typically 2× faster than the thin client approach, fig. 1.

Info Boxes display charts with diverse statistics: time to execute a query, i.e., load array tiles from the WebArrayDB storage engine, from a `WMTS` service, time to join, compute, and render, fig. 1. The charts also show the volume occupied by raw array data in WebArrayDB. Please, explore them live at http://webdb.gis.gg



(band2 - band1) / (band2 + band1)

(band2 - band1) / (band2 + band1 + 0.27) * 1.27

Figure 4: WebArrayDB and ArrayGIS Lessons: Initial States and Solutions

4 WEBARRAYDB & ARRAYGIS LESSONS

Let us present the lessons, following which the readers will be able to efficiently explore WebArrayDB and ArrayGIS. We expect that the audience will appreciate the speed of these systems for query answering. The readers will also get insights on the WebArrayDB and ArrayGIS functionality and extend their knowledge of Array DBMSs, Geospatial Cloud Services and Protocols.

Introductory Lesson **Goal:** getting familiar with ArrayGIS interface and WebArrayDB features. **Scenario:** the user starts by creating an account/logging in. At the start screen, the user will see four main GUI components: Interactive Map, Layer Library, Toolbox Gallery, and Main Checkboxes. We added several layers beforehand. The user should try to turn on/off the layers, and see how they look on the map. The user should also click the pencil icon next to a layer to check its properties in a separate dialog box.

If a `WMTS` layer is enabled, e.g. Sentinel-2 TIFF, the user can open the RGB tool from the Toolbox Gallery, select this layer from the drop-down list and try switching/modifying color palettes. When Sentinel-2 TIFF is enabled, the user can go to the Map Algebra tool and select this layer in the drop-down list: the NDVI formula will appear (the user can change it). Try computing NDVI: it should happen instantly[15] because the source array tiles, the user sees on the map, have already been stored locally in WebArrayDB.

Water Lesson *Mapping Water Bodies.* **Goal:** demonstrate near instantaneous computations on arrays and updates of array color palettes due to keeping raw array data in the web browser. **Motivation:** unlike other Web GISs, ArrayGIS relies on WebArrayDB and accesses source array data without client-server communication. **Area:** Rome, Italy. **Background:** clean water absorbs near-infrared light, so NDVI values close to zero or negative represent zones with the presence of water [1]. Hence, water is often quickly mapped using NDVI without referring to complex machine learning techniques. **Input data:** (1) Sentinel-2 TIFF layer, (2) ground truth (polygons) in the GeoJSON format: water.json (can be added from the Layer Library). **Scenario:** we will execute the query **SELECT NDVI** $< \beta$, where β is a tunable parameter, e.g. 0.1. We will use the ArrayGIS interface for convenience. Please, navigate to the area of interest by clicking the "Fly to" icon next to the layer name. Create the NDVI layer as described in the Introductory Lesson. Open the RGB Tool, pick any palette, e.g. `hsv`, and remove all colors except one, change this color to blue. Tune the value in front of this color (call it β). **The task** is to tune β such that the NDVI layer parts that overlap with the polygons get colored in blue (water).

[15]Users can inspect the WebArrayDB & ArrayGIS performance by experiencing GUI responsiveness and interactive charts in the Database Statistics infobox, figs. 1 and 3

Food Security Lesson *Assessing states of agricultural crops.* **Goal:** demonstrate ultra-rapid array query execution (map algebra) directly in a web browser. **Motivation:** unlike other Web GISs, ArrayGIS submits array queries to WebArrayDB which runs computations on the client side. **Area:** central pivot irrigation fields in Colorado, U.S.A. **Background:** Colorado features arid and semi-arid climates, so SAVI=(NIR−RED)/(NIR+RED+L)×(1+L) is used for arid regions with sparse vegetation and exposed soil surfaces since NDVI is very sensitive to soil brightness [13]. However, SAVI has a tunable parameter L$\in [0, 1]$, which is not known in advance and depends on the soil. Users tune L experimentally. **Input data:** (1) Sentinel-2 TIFF layer, (2) ground truth (points) in the GeoJSON format: field.json. **Scenario:** we will execute the query **SELECT (NIR−RED)/(NIR+RED+L)×(1+L)**. Click the "Fly to" icon for navigation (see Water Lesson). Open the Map Algebra Tool and convert the NDVI query that appears by default into the SAVI query. The NDVI color palette can also be used for the SAVI layer. Tune **L** and watch how fast the layer is updated on the screen. **The task** is to find the **L** value such that the points of the SAVI layer, that coincide with the GeoJSON points, get colored in green.

Takeaway insights WebArrayDB and ArrayGIS leverage the capabilities of modern web engines. They are very fast at executing geospatial array DBMS queries entirely on the client side. Web Array DBMSs and Web GISs will actively evolve in the near future.

Contributions. Zalipynis: ideas, approaches, software architecture & design, libraries' choice, this paper; Terlych: implementation.

REFERENCES

[1] ArcGIS book. 2022. https://learn.arcgis.com/en/arcgis-imagery-book/

[2] Dalsu Choi, Chang-Sup Park, and Yon Dohn Chung. 2019. Progressive top-k subarray query processing in array databases. *PVLDB* 12, 9 (2019), 989–1001.

[3] Milliseconds make Millions 2020. https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Consulting/Milliseconds_Make_Millions_report.pdf.

[4] Ramon Antonio Rodriges Zalipynis. 2018. ChronosDB: Distributed, File Based, Geospatial Array DBMS. *PVLDB* 11, 10 (2018), 1247–1261.

[5] Ramon Antonio Rodriges Zalipynis. 2019. ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud. In *SIGMOD*. 1985–1988.

[6] Ramon Antonio Rodriges Zalipynis. 2020. BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. *PVLDB* 13, 12 (2020), 2909–2912.

[7] Ramon Antonio Rodriges Zalipynis. 2021. Array DBMS: Past, Present, and (Near) Future. *PVLDB* 14, 12 (2021), 3186–3189.

[8] Ramon Antonio Rodriges Zalipynis. 2021. Convergence of Array DBMS and Cellular Automata: A Road Traffic Simulation Case. In *SIGMOD*. 2399–2403.

[9] Sentinel Data Access Annual Report 2021. https://sentinels.copernicus.eu/web/sentinel/-/copernicus-sentinel-data-access-annual-report-2021.

[10] Sentinel Hub 2022. https://www.sentinel-hub.com/.

[11] D. C. Tomlin. 1990. *Geographic Information Systems and Cartographic Modeling*.

[12] Sebastian Villarroya and Peter Baumann. 2020. On the Integration of Machine Learning and Array Databases. In *ICDE*. IEEE, 1786–1789.

[13] Jinru Xue et al. 2017. Significant remote sensing vegetation indices: A review of developments and applications. *J. of Sensors* (2017).