Technical Report

DomainTools

IPV4 Address Space in DNSDB

Joe St Sauver, Ph.D.
Distinguished Scientist

Aggregate Count Data for All Internet IPv4 Addresses Seen in DNSDB “A” Records

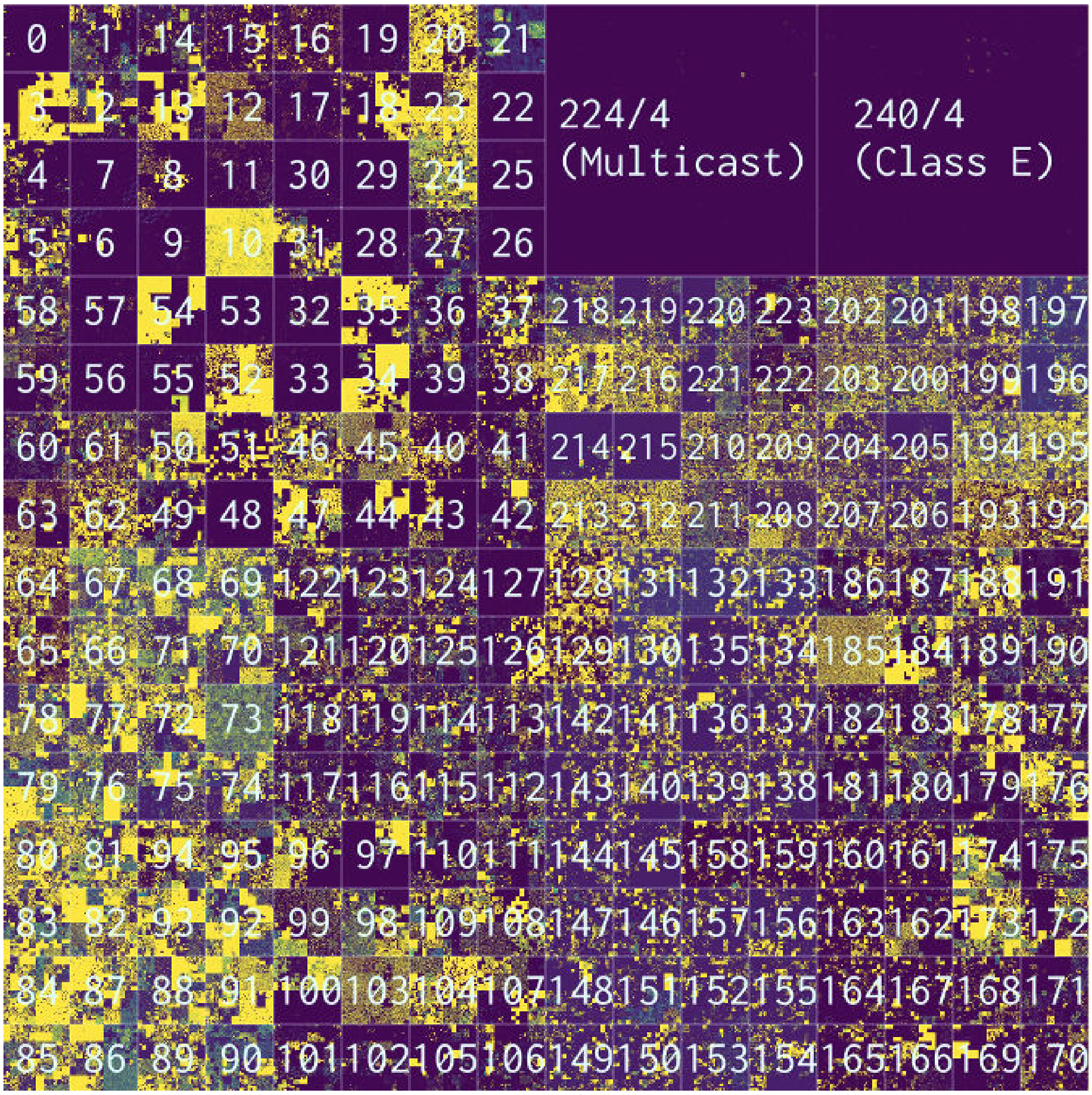


Table of Contents

Executive Summary

The DomainTools mission is “To map the Internet to detect and predict emerging threats.” The company previously reported on IPv4 addresses seen at least once in Farsight DNSDB in 2017. Here, we update and extend the earlier study to focus on the aggregated cache miss counts associated with each /24 netblock for the entire Internet IPv4 address space.

In reviewing the results, we identify some /24 netblocks with unusually large aggregate counts – aggregate cache miss counts in excess of **175 billion**. We dig in on some of those “heavy hitters,” reporting on the domains that appear to be most heavily contributing to those huge cache miss counts. At the other end of the cache miss “aggregate count spectrum,” we find that over 75% of all /24 netblocks have aggregate cache miss counts of 100 or less.

Violin plots and individual Hilbert curves are provided for each /8 for those who may wish to dig in more deeply.

This report also introduces and demonstrates the value of improvements to query meta-data reporting in the company’s command line client, `dnsdbq`, when `dnsdbq` is used for bulk queries.

Introduction

The DomainTools mission is, “To map the Internet to detect and predict emerging threats.” As part of that “mapping,” we’ve previously analyzed and reported on domain names as seen in Farsight DNSDB, and in 2017, we reported on IPv4 addresses appearing in DNSDB.

Much has changed since that time. With the five regional Internet registries running out of IPv4 addresses[1], we can now analyze IP address space as a whole. Here we have updated and extended the earlier 2017 IP address study. After replicating the earlier study’s review of IPv4 address incidence in DNSDB “A” records, we shifted focus to the aggregated cache miss counts associated with each /24 netblock for the entire Internet IPv4 address space.

The culmination of this effort, the cover image on page 2 of this report, presents aggregate count data for all Internet IPv4 addresses seen in DNSDB “A” records shown as a Hilbert curve heatmap. Areas with the highest observation counts are bright yellow, while areas with the lowest observation counts are purple. The image represents totals for over 16.7 million `dnsdbq /24` summary queries, timefenced to the preceding 90 days.

Before we describe exactly how we generated that image, we will first talk about the scope of this report, review the limitations of the 2017 study, and discuss the methodology and data collection methods used for this study. We will then not only show Hilbert curves for each /24 IP Address block but also show violin graph visualizations of IPv4 address activity per /8.

Focus for this Report

Curiosity about the Internet is “in our DNA.” This is not the first time we have reported on what’s in Farsight DNSDB. A few domain name-related examples include:

- “A Decade of Passive DNS: A Snapshot of Top-Level Domain Traffic”[2]
- “TLD Traffic Volumes in DNSDB. Year-by-Year Graphic For 2010 - 2019”[3]

Most closely related to this effort, in 2017, Mr. Ben April, then Director of Research for Farsight Security, wrote a blog post summarizing IPv4 addresses as seen in DNSDB:

- “Visualizing the Incidence of IPv4 addresses in Farsight’s DNSDB”[4]

That study looked at data on an IPv4 “/24” netblock-level of granularity – each pixel in the image from 2017 represented 256 IPv4 addresses. It extracted its data directly from three MTBL files: the daily MTBL file for 3-Feb-2017; the monthly MTBL file for Jan-2017, and the 2015 yearly

[1] https://www.theregister.com/2019/11/25/ipv4_addresses_gone/
[2] https://info.farsightsecurity.com/a-decade-of-passive-dns
[3] https://info.farsightsecurity.com/tld-traffic-volumes-in-dnsdb
[4] https://www.domaintools.com/resources/blog/visualizing-the-incidence-of-ipv4-addresses-in-farsights-dnsdb

MTBL file. The 2016 file was still in the process of being prepared at the time the study was performed, and wasn't available for analysis.
Much has happened since that time, including:

- The Internet has largely exhausted readily-available IPv4 address space and has begun moving to IPv6. Will that result in even more complete utilization of IPv4 address space or lower utilization as IPv6 and carrier grade NAT replace traditional public IPv4 IPs?
- Public cloud uptake and mobile device usage have rapidly increased which has caused "deperimeterization" and may result in changes to connection and access patterns. Likewise, the "Internet of Things" is now firmly a reality. Many more devices are now online such as smart speakers, cars, and refrigerators, in addition to traditional desktops, laptops, tablets, or smartphones.
- Consumer performance expectations have also evolved, driving adoption of content delivery networks (CDNs) to accelerate and scale traffic to ever-growing traffic volumes. CDNs shift what we see and from where. For example, content that might previously have lived in a company's own address space may now be getting served via a CDN.
- The world suffered through the COVID-19 pandemic which sickened or killed millions and caused many businesses, employees, and family members to shift their lives online (work-from-home, "Zoom school," etc.).
- Artificial intelligence has gone from rom ELIZA's primitive Rogerian dialogs[5] to large language models (LLMs) exemplified by ChatGPT.[6] While some may worry that "Skynet"[7] is now real and "Judgment Day" will soon be upon us, the most readily-foreseeable result of LLMs is AI-assisted development of new Internet applications with an associated increase in demand for IP addresses.
- The company itself has changed. Farsight Security is now part of DomainTools. Our sensor network and the volume of traffic we observe has greatly expanded. Finally, Farsight DNSDB and the clients used to access it have matured.

With all those changes and more, we think it's appropriate for us to re-examine the IPv4 address space we see in Farsight DNSDB. Therefore in this study we chose to do a deeper analysis than seen-or-not IPv4 "/24" netblock analysis from 2017. Here we look to gauge the activity seen via each /24 as measured by cache miss counts in our stored RRSets. This acts as a proxy for how heavily used these IP addresses are. To convey these findings in more detail, we will present visualizations of each /8 netblock in detail as Violin Plots as well as the more traditional Hilbert Curves. We hope this provides a more nuanced understanding of activity in the IPv4 address space as viewed from DNSDB.

To avoid potential confusion, we'll note that there are some questions that this report explicitly does not address:

- "IPv6 Addresses." It is our intention to cover IPv6 in a separate follow-up report. There is too much to show in just one document!

[5] https://en.wikipedia.org/wiki/ELIZA
[6] https://en.wikipedia.org/wiki/ChatGPT
[7] https://en.wikipedia.org/wiki/Skynet_(Terminator)

- "The 'most popular' or 'most heavily used' IPv4 address space." The cache miss traffic our sensors see is heavily influenced by the TTL (time to live) set for each name and impacts of DNS caching. A very popular name with a long TTL may have lower counts than a much less popular name set to use a short TTL.
- "Parts of the IPv4 address space that aren't in use." While some netblocks may not appear in the DNSDB "A" record data described in this report, that doesn't necessarily mean that that address space isn't "in use." For instance:
    - The space may be in use, and may even have DNS "A" records defined, but we may not have seen any successful attempts to resolve those names during our study period, or we may have filtered that traffic from DNSDB.
    - The space may be used but not have DNS "A" records defined. Devices using IPv4 addresses may be accessed solely by their numeric addresses, perhaps as part of a dynamic address pool or for infrastructural purposes. DNS is not mandatory for the Internet to function.
    - The space may use names from locally-created or "alternative" TLDs We only index traffic associated with ICANN-approved TLDs.
    - The space in question may in fact not be used for "traditional" purposes, but may be legitimately used for less-common things such as a "network telescope", a.k.a. darknet analysis[8]
- "Where DNSDB gets its data." As a matter of company policy, we do not discuss specifics of our sensor network, including who may be contributing data to DNSDB or the location of DNSDB sensors.
- "Queries DNSDB API customers have made." While queries get routinely logged for accounting and debugging purposes, we don't "mine" query logs for reports like this one.
- "DNS traffic is not guaranteed to translate into actual packet traffic flows." There may be IPs that show up in DNS query traffic that may not actually be "in use." After all, any DNS administrator can create records pointing at any arbitrary IPs. If those records get queried, that data may then end up in passive DNS, potentially creating unjustified "indications of use" which are not borne out in fact.

Comparing to the Old Study

Given the existence of the prior 2017 study, one option would be to just rerun the prior study on current data, imposing as few changes as possible. However, given all of the changes we've previously mentioned, we felt that for this report a deeper analysis was called for.

Even with that, we did want to perform some data comparisons. The 2017 study reported seeing 12,904,017 unique IPs for Feb 3, 2017, and 37,266,823 unique IPs for January 2017. Checking the May 15th, 2023 daily MTBL file, after manually removing a handful of spurious entries that weren't actually IPv4 addresses, we found our sample daily file had 41.9 million unique IPs. An increase of 325%:

[8] We distinguish between "darknet", address space that's announced and monitored but which has no users or servers and which should see no legitimate traffic, and "darkweb," which is often accessed via ToR and uses the onion routing protocol.

```
$ dnstable_dump -d dns.20230515.D.mtbl | fgrep " IN A " | \
awk '{print $4}' | sort -u > ips.txt
$ wc -l ips.txt
41942511 ips.txt
```

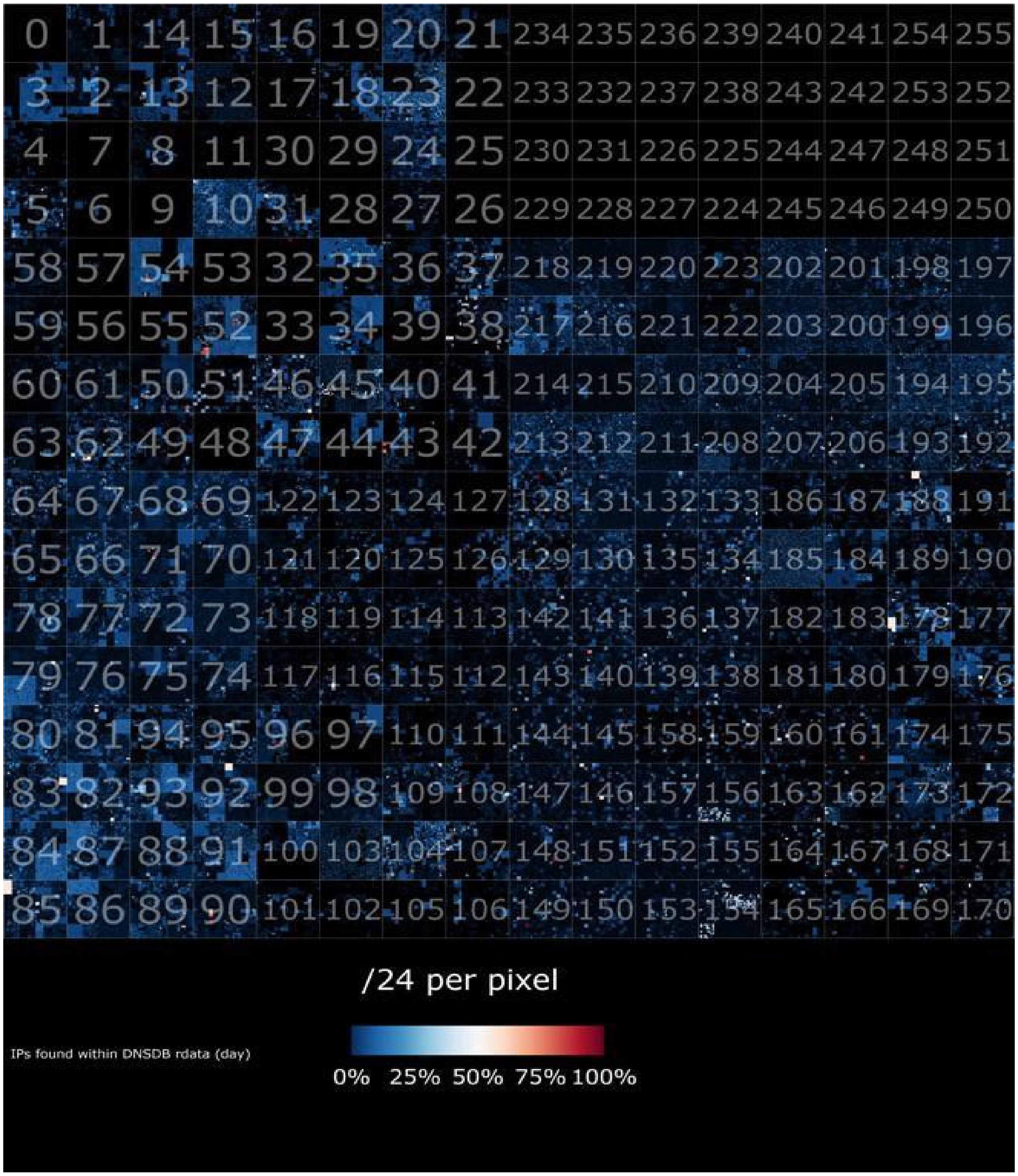
We plotted those using the same `ipv4-heatmap`[9] visualization tool the 2017 study used:

```
$ cat ips.txt | ipv4-heatmap -u "/24 per pixel" -h -d \
-i -P rdbu -f LiberationSans-Regular.ttf \
-t "IPs found within DNSDB rdata (day)" \
-a block-labels-4.txt -o map_day.png
```

The resulting “daily” graph is shown on the next page. We can’t compare it to the 2015 daily graph because that graph wasn’t included in the 2017 blog article as the earlier article only reported the summary count for the daily file.

[9] https://github.com/hrbrmstr/ipv4-heatmap

IPv4 Coverage for the May 15th, 2023 Daily



While the earlier article did not include a graph for the daily file, it did include a graph for the monthly file, so let's run a current monthly file for comparison. We will use the file for April 2023. That file, dns.202304.M.mtbl, is 612GB in size. Given that file size, we processed it in a step-by-step fashion, rather than just "pipelining" the results the way we did for the smaller daily MTBL file.

```
$ dnstable_dump -d dns.202304.M.mtbl | \
fgrep " IN A " | awk '{print $4}' > ips-from-202304.txt
$ sort -T . ips-from-202304.txt > ips-from-202304-sorted.txt
$ uniq ips-from-202304-sorted.txt > ips-from-202304-sorted-uniqued.txt
[handful of spurious entries removed manually]
$ wc -l ips-from-202304-sorted-uniqued.txt
274,253,827 ips-from-202304-sorted-uniqued.txt    [commas added]
```

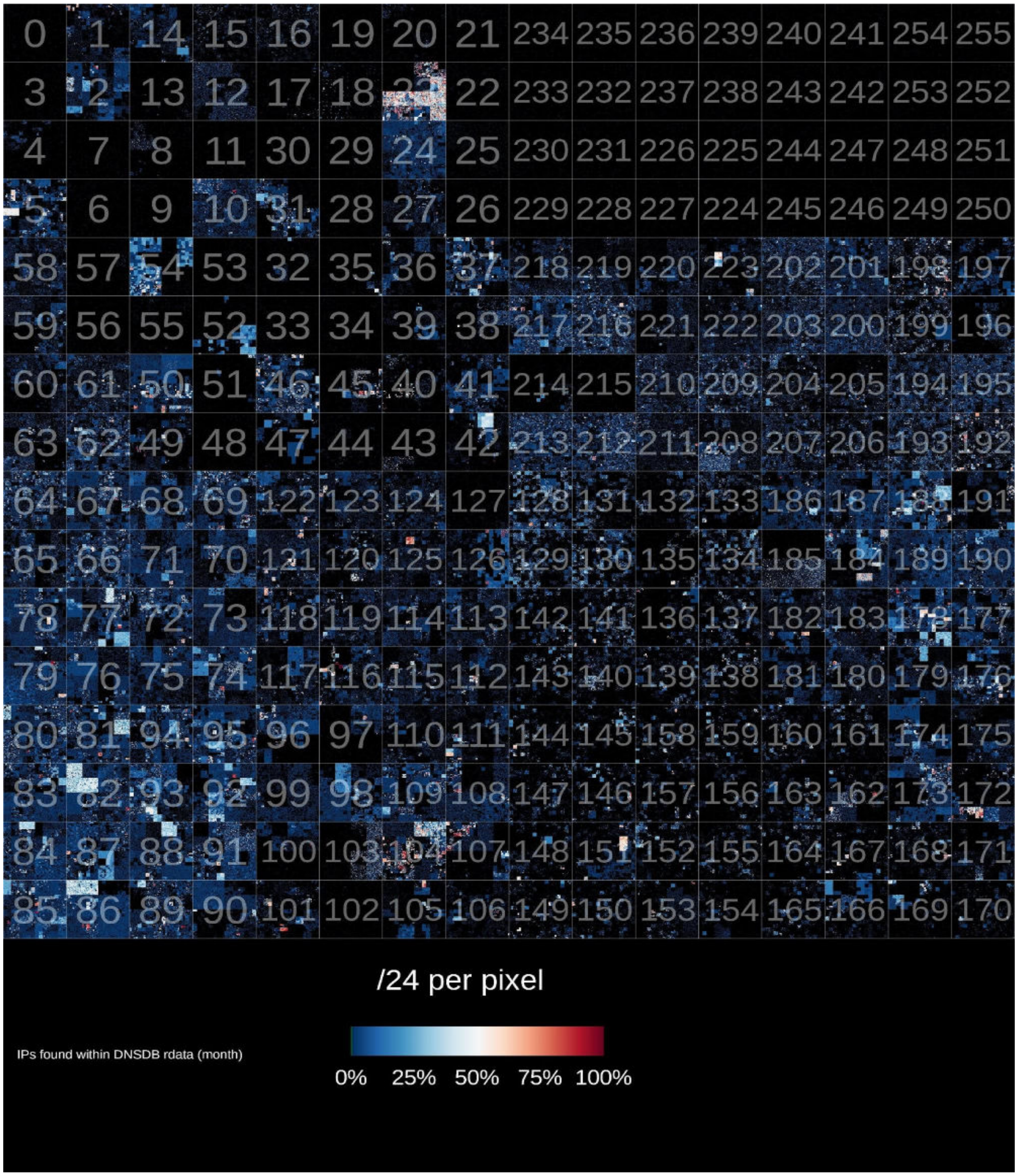
For comparison, the 2017 study found just 37,266,823 entries for the January 2017 monthly, meaning IPv4 inclusion has improved by a factor of `274253827/37266823 = 7.359` since 2017.

Let's now plot the "old style" monthly data:

```
$ cat ips-from-202304-sorted-uniqued.txt | \
ipv4-heatmap -u "/24 per pixel" -h -d -i -P rdbu \
-f LiberationSans-Regular.ttf \
-t "IPs found within DNSDB rdata (month)" -a block-labels-4.txt \
-o map_month.png
```

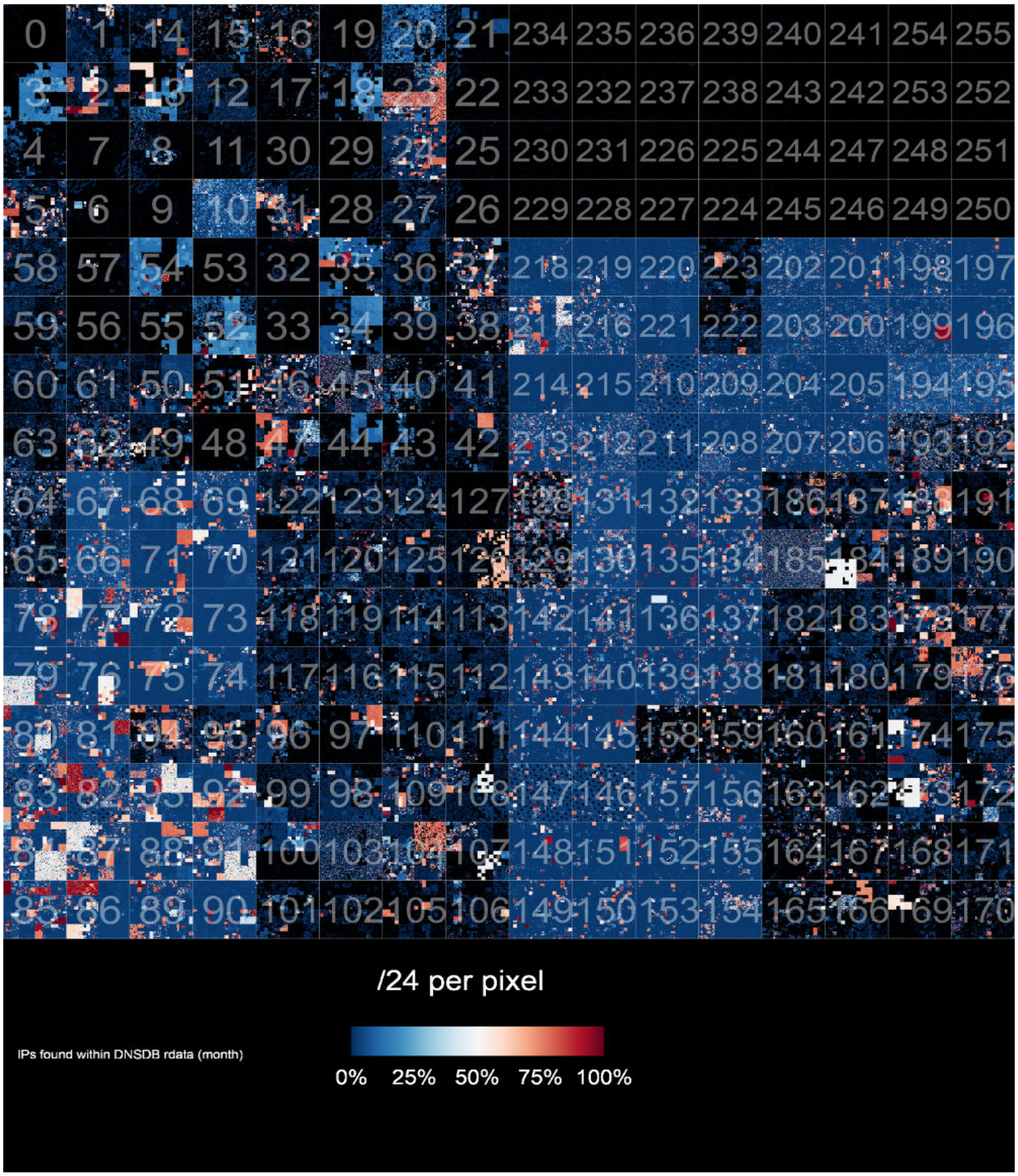
The graph from the 2017 article can be seen on the next page, with our updated 2023 graph on the page after that.

Graph From the Original 2017 Study



For comparison, the new April 2023 monthly graph can be seen on the next page.

IPv4 Coverage for the April 2023 Monthly



Clearly the current monthly shows significantly higher utilization than the 2017 study, with distinctly more pink and red blocks – as we'd suspected, a lot has changed since 2017.

This 2023 Study

The original study looked at IP address data on an IP-by-IP basis, toggling a bit in a bitmap if a given IP was seen at least once. The focus was on coverage: what fraction of all IP addresses had been seen in DNSDB in a fixed time period.

Here, we wanted to know not just if a given IP was seen, but how often addresses from a given `/24` have been seen, as measured by RRset counts. TTL-related effects obviously make it difficult or impossible to make valid distinctions between close count values, but we can at least identify gross differences in count magnitude.

We also strived to use the same tools we offer for customers. Doing so demonstrates what's possible while also ensuring that users can replicate our work if they find it useful. Using customer access methods also serves as a bit of a "torture test," ensuring we find possible issues first!

Technical Considerations and DNSDB Features

Therefore, we choose to check IPv4 addresses in DNSDB API "A" records using `dnsdbq`,[10] just as a regular DNSDB API customer might make bulk DNSDB IP queries for projects of their own. There are a few specific features of DNSDB we will use to help solve some specific challenges with doing this form of large-scale data queries: the `summarize` verb, time-fencing in DNSDB, and the `qdetail` option. We also will need to think carefully about netblocks.

The Summarize Verb

The existence of the `summarize` verb, a DNSDB API query option introduced in October 2019,[11] helps with this work. The summarize verb aggregates query results, replacing the normal "result-by-result" output from DNSDB API with a summary of all results found for each query. For a sample query with output in default presentation format, this looks like:

```
$ dnsdbq -r \*.uoregon.edu -l 0 -V summarize
;; record times: 2010-06-24 03:08:15 .. 2023-05-19 00:46:42
;;   zone times: 2010-04-13 18:39:17 .. 2020-12-31 20:00:03

;; count: 2410673624; num_results: 108297
```

The summarize verb dramatically reduces the volume of output, which is a big help. However, using the summarize verb does impose one noteworthy limitation – we can't use `offset` to "page forward" to get more than 1,000,000 results. Attempts to do so return an error:

[10] https://github.com/dnsdb/dnsdbq
[11] https://www.domaintools.com/resources/blog/new-dnsdb–v-summarize-option-sometimes-less-is-more/

```
$ dnsdbq -i 1.2.3.0/24 -l0 -j -A90d -O1000000 -V summarize
error: only 'lookup' understands offsets

try      dnsdbq -h  for a short description of program usage.
```

Timefencing in DNSDB

DNSDB has data going back to June 2010, so making un-time-fenced queries that go back to the “dawn of DNSDB” would maximize opportunities for us to find IPv4 entries for as many IPs as possible. However, we’re interested in what’s happening NOW, and we think most of you are, too, so we used DNSDB’s time-fencing functionality[12] to limit results to just those seen at least sometime within the last 90 days.[13] Time-fencing is an important part of ensuring that we’re not just “coasting” on results that may have been seen long ago.

The Qdetail Option

There is another handy option for us to use: the `dnsdbq -T qdetail` option to explicitly link details of a query with its resultant output.

`dnsdbq` with `-V summarize` has previously returned JSON Lines-formatted results with no “read back” of what a given result pertains to. If you’re running `dnsdbq -V summarize` interactively, this is not a problem – the query you just made would be followed by the corresponding results, so the relationship between query and output is self-evident.

But now imagine running literally millions of `dnsdbq -V summarize` queries. The need for a clean way to be able to connect the received output with the associated query quickly becomes pretty obvious. Fortunately, the `dnsdbq` maintainers agreed to add a new `-T qdetail` option that reports the query details that resulted in the associated output. To make this concrete, compare (lines manually wrapped for display here):

```
$ dnsdbq -i 128.223.0.0/16 -A90d -V summarize -l0 -j \
-T datefix
{"count":1285767497,"num_results":15128, "time_first":"2010-06-24
03:08:15",
"time_last":"2023-05-18 22:16:54",
"zone_time_first":"2019-12-01 23:47:00",
"zone_time_last":"2023-05-17 23:16:10"}

$ dnsdbq -i 128.223.0.0/16 -A90d -V summarize -l0 -j \
-T datefix,qdetail
{"count":1285767497,"num_results":15128, "time_first":"2010-06-24
03:08:15",
"time_last":"2023-05-18 22:16:54",
"zone_time_first":"2019-12-01 23:47:00",
"zone_time_last":"2023-05-17 23:16:10",
"_dnsdbq":{"descr":"rdata/ip/128.223.0.0,16",
```

[12] https://www.domaintools.com/resources/blog/farsights-dnsdb-time-fencing-a-post-attack-time-machine/
[13] The 90 day period runs back from the time each query began to run, rather than for a “fixed” or “hardcoded” 90 day interval. All queries were run during the month of May 2023 which means the data we analyzed had results seen during February, March, April, or May 2023.

```
"after":"2023-02-17 22:19:38",
"limit":0,"gravel":false,"complete":false}}
```

The new `_dnsdbq` JSON element in the output adds just the context we needed! If you do lots of `dnsdbq` queries with JSON Lines output, we think you'll like this new functionality, too.

Netblocks

What size netblocks should we Query? Scrutinizing all of the Internet's IPv4 address space at an IP-by-IP basis would require making 2^{32} = 4.29 billion DNSDB queries. That's a lot of queries.

We could try making queries for `/16`'s or even `/8`'s, but remember, we can only count at most one million results for any netblock we summarize. Virtually all `/8`'s, most `/16`'s, and even some `/24`'s will have total counts over a million. When that happens, we won't know if the total count is actually just 1,000,001, 2,000,000, 10,000,000, or even more.

Ultimately, pragmatically, we decided to make DNSDB API queries at a `/24` level of granularity, the same level of granularity reported in the 2017 study. Querying for `/24`'s will "just" require 2^{32}/256 or ~ 16.7 million DNSDB API queries. That's obviously still a substantial number of queries, but it is a doable number as the completion of this study demonstrates.

What about the "Special Purpose" or "Reserved" netblocks? We could have potentially reduced our workload by excluding things like `0.0.0.0/8` (normally referred to as the "This Network" range), `127.0.0.0/8` (the "Loopback" range), `224.0.0.0/4` (IP Multicast space), and `240.0.0.0/4` (Class E reserved space), since these special purpose[14] netblocks would normally not be expected to appear in DNSDB "A" records. We knew, however, that references to at least some of those addresses DO show up in DNSDB results, so for completeness sake, we processed the special netblocks just as we did for all the other netblocks.

[14] https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

Collecting the Data We Need

To query A Record from DNSDB for all /24's, we needed to make 256 ∗ 256 ∗ 256 = 16,777,216 queries.

Because DNSDB API users are allowed to run ten concurrent query streams, we created one bash shell script for each `/8`. That means we ended up with 256 such scripts. We selected ten of them to run at any given time.

Each of those streams executes 256 ∗ 256 = 65536 queries, creating one line of results for each query.

Let's look at one of the bash scripts in detail. Here is the script for `184.0.0.0/8`:

```
i=184
length=255
for (j = 0; j <= $length; j++)
do
    for (k = 0; k <= $length; k++)
    do
        dnsdbq -i "$i"."$j"."$k".0/24 -l0 -A90d \
        -V summarize -T qdetail,datefix -j >> \
        "$i"."$j".jsonl
    done
done
```

The above script creates 256 data files as output, `184.0.jsonl` through `184.255.jsonl`, each with 256 results per file. Part of one of those output files looks like, with lines wrapped here for readability:

```
$ more 184.220.jsonl
{"count":36675,"num_results":170, "time_first":"2018-06-27 15:24:46",
"time_last":"2023-05-19 18:55:06",
"_dnsdbq":{"descr":"rdata/ip/184.220.0.0,24",
"after":"2023-02-19 21:25:20","limit":0,
"gravel":false,"complete":false}}
{"count":4669,"num_results":180, "time_first":"2018-06-11 22:54:07",
"time_last":"2023-05-20 20:11:24",
"_dnsdbq":{"descr":"rdata/ip/184.220.1.0,24",
"after":"2023-02-19 21:25:21","limit":0,
"gravel":false,"complete":false}}
{"count":1163,"num_results":181, "time_first":"2018-08-19 17:34:20",
"time_last":"2023-05-20 20:56:23",
"_dnsdbq":{"descr":"rdata/ip/184.220.2.0,24",
"after":"2023-02-19 21:25:21","limit":0,
"gravel":false,"complete":false}} [...]
```

After running our scripts, we had a bunch of data files. We processed the data in two phases. For the first phase, we ran a script to extract the desired fields and concatenate the 256 files for each /8:

```
# 0.*.jsonl
cat 0.*.jsonl | jq -r '"\(._dnsdbq.descr) \(.count)"' | \
sed 's#rdata/ip/##' | sed 's#,24##' > 0.summary
# 1.*.jsonl
cat 1.*.jsonl | jq -r '"\(._dnsdbq.descr) \(.count)"' | \
sed 's#rdata/ip/##' | sed 's#,24##' > 1.summary
# 2.*.jsonl
cat 2.*.jsonl | jq -r '"\(._dnsdbq.descr) \(.count)"' | \
sed 's#rdata/ip/##' | sed 's#,24##' > 2.summary
# 3.*.jsonl
cat 3.*.jsonl | jq -r '"\(._dnsdbq.descr) \(.count)"' | \
sed 's#rdata/ip/##' | sed 's#,24##' > 3.summary
[...]
```

After running those bash scripts, we run the second phase: concatenate the summary files and confirm that we've got the right number of results:

```
$ cat *.summary > all-results.dat
$ wc -l all-results.dat
16777216 all-results.dat
```

Beginning to Analyze Our New Data: the "Heavy Hitters"

Now that we have all of this data, we started by performing some exploratory data analysis to understand what is here. One obvious question: which entries have the most counts? Let's begin by finding our top /24 entries. To improve readability, we've manually added 'thousands' separators, ASN data, and associated company names.

```
$ awk '{print $2 " " $1}' < all-results.dat | sort -nr
175,692,274,099 50.31.142.0           <-- AS22075 (Server Central Network)
175,692,195,196 70.42.32.0            <-- AS22075 (Unitas Global)
174,912,269,493 64.202.112.0          <-- AS22075 (Server Central Network)
174,819,025,428 64.74.236.0           <-- AS22075 (Unitas Global)
160,041,097,671 173.245.59.0          <-- AS13335 (Cloudflare)
159,399,334,560 173.245.58.0          <-- AS13335 (Cloudflare)
155,054,730,917 204.79.197.0          <-- AS8068  (Microsoft)
140,675,283,476 205.251.193.0         <-- AS16509 (Amazon)
140,143,288,176 13.107.21.0           <-- AS8068  (Microsoft)
135,918,083,749 205.251.192.0         <-- AS16509 (Amazon)
130,332,336,002 62.115.252.0          <-- AS1299  (Akamai)
120,673,005,428 205.251.195.0         <-- AS16509 (Amazon)
107,575,472,550 194.58.117.0          <-- AS197695 (Reg.ru)
97,747,468,286 13.107.42.0            <-- AS8068  (Microsoft)
96,198,075,331 176.99.13.0            <-- AS197695 (Reg.ru)
94,550,323,166 205.251.194.0          <-- AS16509 (Amazon)
90,823,124,913 205.251.197.0          <-- AS16509 (Amazon)
90,641,466,193 108.162.193.0          <-- AS13335 (Cloudflare)
90,585,994,546 172.64.33.0            <-- AS13335 (Cloudflare)
90,190,400,734 108.162.192.0          <-- AS13335 (Cloudflare)
90,154,755,550 172.64.32.0            <-- AS13335 (Cloudflare)
88,636,898,115 69.28.143.0            <-- AS22822 (Limelight Networks)
84,870,678,684 162.251.82.0           <-- AS13335 (Cloudflare)
```

```
84,059,876,414 205.251.199.0          <-- AS16509 (Amazon)
82,251,353,513 205.251.198.0          <-- AS16509 (Amazon)
77,880,978,575 162.159.24.0           <-- AS13335 (Cloudflare)
70,455,069,599 62.115.253.0           <-- AS1299  (Arelion Sweden AB)
70,060,035,407 52.113.194.0           <-- AS8068  (Microsoft)
69,312,053,145 205.251.196.0          <-- AS16509 (Amazon)
68,833,992,264 2.16.103.0             <-- AS20940 (Akamai)
67,810,970,068 45.57.9.0              <-- AS40027 (Netflix)
67,803,368,522 45.57.8.0              <-- AS40027 (Netflix)
65,062,995,370 162.159.25.0           <-- AS13335 (Cloudflare)
56,448,042,445 88.221.132.0           <-- AS20940 (Akamai)
55,644,520,425 80.239.137.0           <-- AS1299  (Arelion Sweden AB)
55,337,508,929 2.19.204.0             <-- AS20940 (Akamai)
52,897,104,669 162.159.26.0           <-- AS13335 (Cloudflare)
52,366,534,468 162.159.27.0           <-- AS13335 (Cloudflare)
51,105,757,489 66.235.157.0           <-- AS15224 (Adobe)
46,636,680,216 2.19.205.0             <-- AS20940 (Akamai)
45,000,323,867 208.85.46.0            <-- AS40428 (Pandora Media)
44,936,990,690 208.85.42.0            <-- AS40428 (Pandora Media)
44,872,404,628 68.142.254.0           <-- AS7280  (Oath Holdings)
44,152,423,900 68.180.130.0           <-- AS10880 (Oath Holdings)
42,510,721,898 192.5.6.0              <-- AS396559 (Verisign)
42,404,466,204 192.31.80.0            <-- AS396559 (Verisign)
[...]
```

What the heck is driving those counts into the billions? Given those exceptionally large counts, we naturally wondered whether there were particular domains associated with those values. Now that we know "where to look", we dug in by making non-summarized follow-on queries. The companies hosting the ASNs we're going to look at are:

- Zemanta
- Cloudflare
- Microsoft
- Amazon
- Akamai
- Reg.ru
- Limelight Networks
- Arelion Sweden AB (actually "Akamai by a different name")

For each one, we will see what DNSDB has to say about where the counts are coming from.

Zemanta

For example, consider **50.31.142.0/24**, **70.42.32.0/24**, **64.202.112.0/24**, and **64.74.236.0/24**, all blocks associated with AS22075. When we dig in on those blocks, we find those IPs are dominated by an ad tracking company's domains. Looking at the first of those (the others are similar):

```
$ dnsdbq -i 50.31.142.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,477,927,842      [commas added manually here and below]
```

```
zemanta-nychi.zemanta.com. A 50.31.142.191

;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,477,927,842
zemanta-nychi.zemanta.com. A 50.31.142.159

;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,477,927,842
zemanta-nychi.zemanta.com. A 50.31.142.95

;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,477,927,842
zemanta-nychi.zemanta.com. A 50.31.142.63

;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,477,927,842
zemanta-nychi.zemanta.com. A 50.31.142.31

;; record times: 2021-10-13 11:50:34 .. 2023-05-30 00:19:55 (~1y
~228d)
;; count: 22,459,468,193
zemanta-nychi.zemanta.com. A 50.31.142.127

;; record times: 2021-12-16 07:01:16 .. 2023-05-30 00:19:55 (~1y
~164d)
;; count: 21,012,572,989
zemanta-nychi.zemanta.com. A 50.31.142.255

;; record times: 2021-12-16 07:01:16 .. 2023-05-30 00:19:55 (~1y
~164d)
;; count: 21,012,572,989
zemanta-nychi.zemanta.com. A 50.31.142.223 [remaining hits all below
12 million]
```

Zemanta is interesting in that if you check its FQDN in the live DNS, you'll see that it returns **32** dotted quads, an unusually large RRset:

```
$ dig zemanta-nychi.zemanta.com
[snip]
;; ANSWER SECTION:
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.127
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.63
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.95
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.223
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.191
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.31
```

```
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.255
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.63
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.255
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.63
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.223
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.191
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.31
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.191
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.223
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.159
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.95
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.127
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.95
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.223
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.159
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.31
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.127
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.255
zemanta-nychi.zemanta.com. 120 IN A 64.202.112.255
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.95
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.127
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.191
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.63
zemanta-nychi.zemanta.com. 120 IN A 50.31.142.31
zemanta-nychi.zemanta.com. 120 IN A 64.74.236.159
zemanta-nychi.zemanta.com. 120 IN A 70.42.32.159
[snip]
```

Cloudflare

If we go beyond the four blocks associated with AS22075, are there other domains that dominate other “mega-count” netblocks?

Let’s look at some AS13335 prefixes: **173.234.59.0/24**, **173.245.58.0/24**, **108.162.193.0/24**, **172.64.33.0/24**, **108.162.192.0/24**, and **172.64.32.0/24**. These IPs all show heavy counts associated with Cloudflare’s own nameserver domains. Here’s an excerpt for one of those blocks:

```
$ dnsdbq -i 173.245.58.0/24 -l0 -A90d -S -k count
;; record times: 2014-03-24 17:13:33 .. 2023-05-29 23:11:50 (~9y ~68d)
;; count: 5,424,711,872   [commas added manually here and below]
ns2.digitalocean.com.  A  173.245.59.41

;; record times: 2011-06-07 23:48:07 .. 2023-05-29 19:16:47 (~11y ~358d)
;; count: 1,843,185,664
```

```
pete.ns.cloudflare.com.  A  173.245.59.136

;; record times: 2011-06-07 22:24:59 .. 2023-05-29 20:24:59 (~11y ~358d)
;; count: 1,761,788,257
cody.ns.cloudflare.com.  A  173.245.59.107

;; record times: 2014-07-18 02:44:47 .. 2023-05-29 23:54:01 (~8y ~317d)
;; count: 1,688,978,297
terry.ns.cloudflare.com.  A  173.245.59.237

;; record times: 2011-06-08 00:07:47 .. 2023-05-30 00:31:00 (~11y ~359d)
;; count: 1,685,951,435
woz.ns.cloudflare.com.  A  173.245.59.150

;; record times: 2011-06-07 22:15:11 .. 2023-05-29 23:27:52 (~11y ~359d)
;; count: 1,559,428,724
andy.ns.cloudflare.com.  A  173.245.59.101

;; record times: 2011-06-07 23:49:17 .. 2023-05-29 22:41:11 (~11y ~358d)
;; count: 1,510,086,221
ray.ns.cloudflare.com.  A  173.245.59.138

;; record times: 2011-06-07 22:59:25 .. 2023-05-30 00:24:21 (~11y ~359d)
;; count: 1,508,887,124
ivan.ns.cloudflare.com.  A  173.245.59.120

;; record times: 2014-07-16 22:34:09 .. 2023-05-29 23:40:39 (~8y ~319d)
;; count: 1,498,105,529
peyton.ns.cloudflare.com.  A  173.245.59.221

;; record times: 2011-06-08 00:04:34 .. 2023-05-30 01:00:00 (~11y ~359d)
;; count: 1,465,210,675
sid.ns.cloudflare.com.  A  173.245.59.143

;; record times: 2011-06-07 23:49:47 .. 2023-05-29 21:39:59 (~11y ~358d)
;; count: 1,414,346,564
rick.ns.cloudflare.com.  A  173.245.59.139

;; record times: 2011-06-07 23:04:38 .. 2023-05-29 20:43:33 (~11y ~358d)
;; count: 1,387,345,985
jim.ns.cloudflare.com.  A  173.245.59.125

;; record times: 2011-06-07 23:02:12 .. 2023-05-29 22:12:22 (~11y ~358d)
;; count: 1,383,289,805
jay.ns.cloudflare.com.  A  173.245.59.123
```

```
;; record times: 2011-06-07 23:32:20 .. 2023-05-30 00:19:02 (~11y ~359d)
;; count: 1,361,286,241
lee.ns.cloudflare.com.  A  173.245.59.129

;; record times: 2011-06-07 22:14:50 .. 2023-05-29 22:34:46 (~11y ~359d)
;; count: 1,347,306,104
art.ns.cloudflare.com.  A  173.245.59.102

;; record times: 2011-06-07 23:44:11 .. 2023-05-29 19:04:08 (~11y ~358d)
;; count: 1,324,281,124
noah.ns.cloudflare.com.  A  173.245.59.133
;; record times: 2011-06-07 22:30:48 .. 2023-05-29 21:20:47 (~11y ~358d)
;; count: 1,309,889,431
duke.ns.cloudflare.com.  A  173.245.59.110

;; record times: 2011-06-07 23:44:28 .. 2023-05-29 23:32:56 (~11y ~358d)
;; count: 1,307,754,547
max.ns.cloudflare.com.  A  173.245.59.132

;; record times: 2011-06-07 22:59:39 .. 2023-05-30 00:34:05 (~11y ~359d)
;; count: 1,283,104,311
ian.ns.cloudflare.com.  A  173.245.59.118

;; record times: 2011-06-07 22:56:49 .. 2023-05-30 00:45:57 (~11y ~359d)
;; count: 1,272,159,416
igor.ns.cloudflare.com.  A  173.245.59.119

;; record times: 2011-06-07 22:37:26 .. 2023-05-30 00:22:59 (~11y ~359d)
;; count: 1,255,139,791
eric.ns.cloudflare.com.  A  173.245.59.112

;; record times: 2011-06-07 22:55:38 .. 2023-05-30 00:01:12 (~11y ~359d)
;; count: 1,253,509,117
hank.ns.cloudflare.com.  A  173.245.59.116

;; record times: 2011-06-07 22:18:55 .. 2023-05-29 19:55:49 (~11y ~358d)
;; count: 1,243,584,159
brad.ns.cloudflare.com.  A  173.245.59.105

;; record times: 2011-06-07 22:48:17 .. 2023-05-30 00:25:37 (~11y ~359d)
;; count: 1,242,112,108
gabe.ns.cloudflare.com.  A  173.245.59.114
[etc]
```

Other “mega-count” Cloudflare blocks have 3rd-party (presumably customer) nameservers. This includes **162.251.82.0/24**, **162.159.24.0/24**, **162.159.25.0/24**, **162.159.26.0/24** and **162.159.27.0/24**. For instance:

```
$ dnsdbq -i 162.251.82.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2015-08-24 11:55:49 .. 2023-05-30 02:55:17 (~7y ~280d)
;; count: 337438571
ns1.whois.com.  A  162.251.82.122

;; record times: 2015-08-24 11:55:49 .. 2023-05-30 02:55:17 (~7y ~280d)
;; count: 337438543
ns1.whois.com.  A  162.251.82.123

;; record times: 2015-08-24 11:55:49 .. 2023-05-30 02:55:17 (~7y ~280d)
;; count: 337438524
ns1.whois.com.  A  162.251.82.250

;; record times: 2015-08-24 11:55:49 .. 2023-05-30 02:55:17 (~7y ~280d)
;; count: 337438434
ns1.whois.com.  A  162.251.82.251

;; record times: 2015-08-24 11:58:52 .. 2023-05-29 22:14:37 (~7y ~280d)
;; count: 337240247
ns2.whois.com.  A  162.251.82.120

;; record times: 2015-08-24 11:58:52 .. 2023-05-29 22:14:37 (~7y ~280d)
;; count: 337240162
ns2.whois.com.  A  162.251.82.121

;; record times: 2015-08-24 11:58:52 .. 2023-05-29 22:14:37 (~7y ~280d)
;; count: 337240085
ns2.whois.com.  A  162.251.82.248

;; record times: 2015-08-24 11:58:52 .. 2023-05-29 22:14:37 (~7y ~280d)
;; count: 337239990
ns2.whois.com.  A  162.251.82.249

;; record times: 2015-06-23 14:03:51 .. 2023-05-29 22:44:13 (~7y ~342d)
;; count: 330163659
ns3.whois.com.  A  162.251.82.118

;; record times: 2015-06-23 13:49:58 .. 2023-05-29 22:44:13 (~7y ~342d)
;; count: 330163632
ns3.whois.com.  A  162.251.82.119
;; record times: 2015-06-23 13:49:58 .. 2023-05-29 22:44:13 (~7y ~342d)
;; count: 330163557
ns3.whois.com.  A  162.251.82.246

;; record times: 2015-06-23 13:47:20 .. 2023-05-29 22:44:13 (~7y ~342d)
;; count: 330163456
ns3.whois.com.  A  162.251.82.247

[etc]
```

Microsoft

Moving on to **Microsoft's 204.79.197.0/24**, we again see a block dominated by just a couple of domains. In this case, a-msedge.net and edge.bing.com plus a few other top Microsoft brands appear on this list:

```
$ dnsdbq -i 204.79.197.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2017-08-07 23:53:13 .. 2023-05-29 22:29:59 (~5y ~295d)
;; count: 127,714,163,745
dual-a-0001.a-msedge.net.  A  204.79.197.200

;; record times: 2014-05-09 02:34:23 .. 2023-05-29 22:30:55 (~9y ~22d)
;; count: 19,813,437,096
a-0003.a-msedge.net.  A  204.79.197.203

;; record times: 2014-03-11 18:30:04 .. 2023-05-29 22:33:53 (~9y ~81d)
;; count: 3,603,345,675
a-0001.a-msedge.net.  A  204.79.197.200

;; record times: 2014-06-17 18:38:24 .. 2023-05-29 23:32:43 (~8y ~348d)
;; count: 1,467,345,649
a-0011.a-msedge.net.  A  204.79.197.213

;; record times: 2017-08-08 00:03:40 .. 2023-05-29 22:31:19 (~5y ~295d)
;; count: 731,753,040
ipv4-a-0001.a-msedge.net.  A  204.79.197.200

;; record times: 2012-08-17 22:28:22 .. 2023-05-30 00:39:27 (~10y ~287d)
;; count: 567,862,176
any.edge.bing.com.  A  204.79.197.200

;; record times: 2014-03-08 05:10:30 .. 2023-05-29 22:45:47 (~9y ~84d)
;; count: 559,637,276
ns1.a-msedge.net.  A  204.79.197.1

;; record times: 2014-09-15 20:23:35 .. 2023-05-29 18:47:30 (~8y ~257d)
;; count: 547,145,694
a-0009.a-msedge.net.  A  204.79.197.211

;; record times: 2015-08-15 10:40:47 .. 2023-05-29 22:53:13 (~7y ~289d)
;; count: 313,002,571
a-0010.a-msedge.net.  A  204.79.197.212

;; record times: 2014-05-22 05:24:31 .. 2023-05-29 22:50:49 (~9y ~9d)
;; count: 309,950,264
a-0006.a-msedge.net.  A  204.79.197.208

;; record times: 2014-04-24 15:21:48 .. 2023-05-29 22:40:55 (~9y ~37d)
;; count: 299,315,050
a-0002.a-msedge.net.  A  204.79.197.201

;; record times: 2014-05-29 02:15:58 .. 2023-05-29 18:50:07 (~9y ~2d)
;; count: 295,571,051
a-0008.a-msedge.net.  A  204.79.197.210

;; record times: 2015-08-17 09:14:33 .. 2023-05-29 22:43:43 (~7y ~287d)
;; count: 271,798,035
a-0005.a-msedge.net.  A  204.79.197.204

;; record times: 2014-06-24 17:32:06 .. 2023-05-29 18:50:13 (~8y ~341d)
;; count: 260,918,161
a-0007.a-msedge.net.  A  204.79.197.209

;; record times: 2012-07-13 15:43:55 .. 2023-05-29 22:48:57 (~10y ~322d)
;; count: 208,024,771
msedge.net.  A  204.79.197.197

;; record times: 2012-11-03 14:27:16 .. 2023-05-29 22:50:59 (~10y ~209d)
;; count: 164,804,006
www.msedge.net.  A  204.79.197.197

;; record times: 2012-05-01 23:31:29 .. 2023-05-29 16:32:23 (~11y ~29d)
;; count: 144,614,407
ns1.msedge.net.  A  204.79.197.1

;; record times: 2012-05-01 23:31:29 .. 2023-05-29 23:22:05 (~11y ~29d)
;; count: 143,080,940
ns2.msedge.net.  A  204.79.197.2

;; record times: 2015-01-30 19:05:53 .. 2023-05-29 18:35:59 (~8y ~120d)
;; count: 89,091,064
```

```
ns2.a-msedge.net.  A  204.79.197.2

;; record times: 2015-04-24 18:27:48 .. 2023-05-29 22:49:58 (~8y ~37d)
;; count: 66,829,563
a-0004.a-msedge.net.  A  204.79.197.206

;; record times: 2014-09-30 05:11:13 .. 2023-05-30 00:26:14 (~8y ~243d)
;; count: 29,939,019
a-0012.a-msedge.net.  A  204.79.197.214

;; record times: 2013-02-24 17:06:12 .. 2023-05-29 21:12:31 (~10y ~96d)
;; count: 24,726,160
bing.com.  A  204.79.197.200

;; record times: 2014-01-03 19:39:33 .. 2023-05-30 00:47:36 (~9y ~148d)
;; count: 24,343,065
explicit.any.edge.bing.com.  A  204.79.197.201

;; record times: 2017-11-08 17:46:54 .. 2023-05-29 17:41:46 (~5y ~202d)
;; count: 17,692,945
live.com.  A  204.79.197.212

;; record times: 2017-11-08 17:32:19 .. 2023-05-29 17:52:51 (~5y ~203d)
;; count: 11,796,049
hotmail.com.  A  204.79.197.212

[etc]
```

Amazon

AWS is obviously very well known as a cloud services provider. When we look at **205.251.193.0/24** will we see Amazon's own infrastructure, or a client's domain dominating traffic counts? Let's see:

```
;; record times: 2016-09-19 22:26:12 .. 2023-05-30 00:14:48 (~6y ~253d)
;; count: 3,382,737,486
ns4-b2.warnerbros.com.  A  205.251.193.169

;; record times: 2018-09-24 16:47:07 .. 2023-05-29 18:30:51 (~4y ~248d)
;; count: 1,568,589,720
ns3a.immunet.com.  A  205.251.193.54

;; record times: 2010-12-08 03:21:31 .. 2023-05-29 19:42:29 (~12y ~175d)
;; count: 1,062,598,405
ns-386.awsdns-48.com.  A  205.251.193.130
```

```
;; record times: 2010-11-13 03:01:45 .. 2023-05-30 00:56:13 (~12y ~200d)
;; count: 753,792,504
g-ns-352.awsdns-32.co.uk.  A  205.251.193.96

;; record times: 2010-12-14 02:38:55 .. 2023-05-29 21:51:01 (~12y ~169d)
;; count: 718,061,151
ns-421.awsdns-52.com.  A  205.251.193.165

;; record times: 2010-12-07 02:25:01 .. 2023-05-30 01:35:57 (~12y ~176d)
;; count: 714,695,587
g-ns-380.awsdns-60.co.uk.  A  205.251.193.124

;; record times: 2010-11-23 03:15:44 .. 2023-05-30 00:21:58 (~12y ~190d)
;; count: 679,168,004
g-ns-339.awsdns-19.co.uk.  A  205.251.193.83

;; record times: 2010-12-08 02:05:00 .. 2023-05-29 23:21:18 (~12y ~175d)
;; count: 671,524,350
g-ns-375.awsdns-55.co.uk.  A  205.251.193.119

;; record times: 2018-02-27 23:22:20 .. 2023-05-29 17:52:29 (~5y ~91d)
;; count: 667,593,392
b.r10.mopub.com.  A  205.251.193.181

;; record times: 2010-12-08 03:51:30 .. 2023-05-30 00:54:46 (~12y ~175d)
;; count: 646,368,806
g-ns-333.awsdns-13.co.uk.  A  205.251.193.77

;; record times: 2010-11-04 21:47:49 .. 2023-05-29 23:04:39 (~12y ~209d)
;; count: 643,995,691
g-ns-356.awsdns-36.co.uk.  A  205.251.193.100

;; record times: 2010-12-09 17:19:33 .. 2023-05-30 01:22:28 (~12y ~174d)
;; count: 634,104,891
g-ns-376.awsdns-56.co.uk.  A  205.251.193.120

;; record times: 2010-12-06 17:17:42 .. 2023-05-29 18:17:09 (~12y ~177d)
;; count: 633,915,954
g-ns-327.awsdns-07.co.uk.  A  205.251.193.71

;; record times: 2010-12-07 03:01:03 .. 2023-05-29 20:06:05 (~12y ~176d)
;; count: 627,712,371
g-ns-349.awsdns-29.co.uk.  A  205.251.193.93
```

```
;; record times: 2010-12-06 17:43:39 .. 2023-05-30 00:56:38 (~12y ~177d)
;; count: 615,259,769
g-ns-359.awsdns-39.co.uk.  A  205.251.193.103

;; record times: 2010-11-10 03:23:36 .. 2023-05-30 00:55:36 (~12y ~203d)
;; count: 593,810,047
g-ns-361.awsdns-41.co.uk.  A  205.251.193.105

;; record times: 2010-12-06 16:42:16 .. 2023-05-29 22:33:01 (~12y ~177d)
;; count: 587,682,940
g-ns-363.awsdns-43.co.uk.  A  205.251.193.107

;; record times: 2010-12-06 23:55:30 .. 2023-05-29 20:48:34 (~12y ~176d)
;; count: 581,991,757
g-ns-379.awsdns-59.co.uk.  A  205.251.193.123

;; record times: 2010-11-05 02:59:18 .. 2023-05-29 18:26:18 (~12y ~208d)
;; count: 580,364,915
g-ns-461.awsdns-12.net.  A  205.251.193.205

;; record times: 2010-12-31 03:24:23 .. 2023-05-30 01:01:11 (~12y ~152d)
;; count: 576,602,272
g-ns-455.awsdns-06.net.  A  205.251.193.199

;; record times: 2010-12-09 03:27:14 .. 2023-05-29 21:26:43 (~12y ~174d)
;; count: 576,216,280
g-ns-330.awsdns-10.co.uk.  A  205.251.193.74

;; record times: 2010-12-14 21:41:47 .. 2023-05-30 01:13:29 (~12y ~169d)
;; count: 574,920,877
g-ns-322.awsdns-02.co.uk.  A  205.251.193.66

;; record times: 2010-12-07 03:15:15 .. 2023-05-30 00:48:13 (~12y ~176d)
;; count: 569,080,595
g-ns-345.awsdns-25.co.uk.  A  205.251.193.89

;; record times: 2010-12-06 18:03:01 .. 2023-05-29 20:35:38 (~12y ~177d)
;; count: 568,048,750
g-ns-321.awsdns-01.co.uk.  A  205.251.193.65

;; record times: 2010-11-05 02:59:34 .. 2023-05-29 22:36:53 (~12y ~208d)
;; count: 566,862,680
g-ns-459.awsdns-10.net.  A  205.251.193.203
[etc]
```

Akamai

How about what's driving **Akamai's** high ranking for **62.115.252.0/24**? Looks to be their own domains.

```
$ dnsdbq -i 62.115.252.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2020-09-10 12:16:06 .. 2023-05-30 01:39:34 (~2y ~261d)
;; count: 1,377,545,133
e28622.a.akamaiedge.net.  A  62.115.252.33

;; record times: 2020-09-10 12:16:04 .. 2023-05-29 22:59:51 (~2y ~261d)
;; count: 1,376,393,679
e28622.a.akamaiedge.net.  A  62.115.252.11

;; record times: 2020-09-10 12:16:04 .. 2023-05-29 22:52:27 (~2y ~261d)
;; count: 1,375,778,423
e28622.a.akamaiedge.net.  A  62.115.252.25

;; record times: 2020-09-10 12:16:05 .. 2023-05-30 01:36:19 (~2y ~261d)
;; count: 1,375,710,046
e28622.a.akamaiedge.net.  A  62.115.252.48

;; record times: 2020-09-10 12:16:04 .. 2023-05-30 00:25:40 (~2y ~261d)
;; count: 1,375,557,895
e28622.a.akamaiedge.net.  A  62.115.252.32

;; record times: 2020-09-10 12:16:04 .. 2023-05-30 00:25:40 (~2y ~261d)
;; count: 1,373,572,073
e28622.a.akamaiedge.net.  A  62.115.252.24

;; record times: 2020-09-10 12:16:05 .. 2023-05-30 01:36:19 (~2y ~261d)
;; count: 1,372,580,060
e28622.a.akamaiedge.net.  A  62.115.252.50

;; record times: 2020-09-10 12:16:06 .. 2023-05-30 01:39:34 (~2y ~261d)
;; count: 1,371,925,410
e28622.a.akamaiedge.net.  A  62.115.252.42

;; record times: 2020-09-10 12:16:04 .. 2023-05-29 22:51:59 (~2y ~261d)
;; count: 1,371,822,652
e28622.a.akamaiedge.net.  A  62.115.252.73

;; record times: 2020-09-10 12:16:04 .. 2023-05-30 00:24:58 (~2y ~261d)
;; count: 1,371,296,147
e28622.a.akamaiedge.net.  A  62.115.252.49
```

```
;; record times: 2020-09-10 12:16:04 .. 2023-05-30 00:24:58 (~2y ~261d)
;; count: 1,371,016,034
e28622.a.akamaiedge.net.  A  62.115.252.9

;; record times: 2020-09-10 12:16:04 .. 2023-05-30 00:25:40 (~2y ~261d)
;; count: 1,370,614,674
e28622.a.akamaiedge.net.  A  62.115.252.19
[etc]
```

Reg.ru

Reg.ru’s 194.58.117.0/24 is another block that’s definitely dominated by Reg.ru name-servers among other things:

```
$ dnsdbq -i 194.58.117.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2015-11-02 10:02:40 .. 2023-05-29 23:13:12 (~7y ~210d)
;; count: 13,040,026,463
ns2.reg.ru.  A  194.58.117.16

;; record times: 2015-11-02 10:01:58 .. 2023-05-29 23:13:12 (~7y ~210d)
;; count: 13,040,026,406
ns2.reg.ru.  A  194.58.117.18

;; record times: 2015-11-02 10:02:20 .. 2023-05-29 23:13:12 (~7y ~210d)
;; count: 13,040,026,391
ns2.reg.ru.  A  194.58.117.12

;; record times: 2015-11-02 10:01:39 .. 2023-05-29 23:13:12 (~7y ~210d)
;; count: 13,040,026,374
ns2.reg.ru.  A  194.58.117.14

;; record times: 2015-11-02 09:57:28 .. 2023-05-30 00:14:32 (~7y ~210d)
;; count: 13,037,758,719
ns1.reg.ru.  A  194.58.117.15

;; record times: 2015-11-02 10:00:48 .. 2023-05-30 00:14:32 (~7y ~210d)
;; count: 13,037,758,578
ns1.reg.ru.  A  194.58.117.17

;; record times: 2015-11-02 09:56:26 .. 2023-05-30 00:14:32 (~7y ~210d)
;; count: 13,037,758,413
ns1.reg.ru.  A  194.58.117.13

;; record times: 2015-11-02 09:56:04 .. 2023-05-30 00:14:32 (~7y ~210d)
;; count: 13,037,752,156
ns1.reg.ru.  A  194.58.117.11
[etc]
```

Limelight Networks

How about **69.28.143.0/24**? Again, this is another netblock that's dominated by nameserver-related domains:

```
$ dnsdbq -i 69.28.143.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2010-06-24 03:06:59 .. 2023-05-30 00:01:09 (~12y ~342d)
;; count: 20,337,085,078
dns12.llnwd.net.  A  69.28.143.12

;; record times: 2010-06-24 03:06:59 .. 2023-05-30 00:01:09 (~12y ~342d)
;; count: 20,319,782,254
dns14.llnwd.net.  A  69.28.143.14

;; record times: 2010-06-24 03:06:59 .. 2023-05-30 00:01:09 (~12y ~342d)
;; count: 20,290,203,066
dns13.llnwd.net.  A  69.28.143.13

;; record times: 2010-06-24 03:06:59 .. 2023-05-30 00:01:09 (~12y ~342d)
;; count: 20,290,133,550
dns11.llnwd.net.  A  69.28.143.11

;; record times: 2011-11-08 00:17:33 .. 2023-05-30 00:33:46 (~11y ~206d)
;; count: 1,026,444,977
dns23.llnwd.net.  A  69.28.143.23

;; record times: 2011-12-08 17:31:49 .. 2023-05-30 00:34:15 (~11y ~175d)
;; count: 1,026,430,019
dns26.llnwd.net.  A  69.28.143.26

;; record times: 2011-10-31 06:39:09 .. 2023-05-30 00:48:47 (~11y ~213d)
;; count: 1,013,821,007
dns24.llnwd.net.  A  69.28.143.24

;; record times: 2011-12-08 17:31:49 .. 2023-05-30 00:42:40 (~11y ~175d)
;; count: 1,012,966,634
dns25.llnwd.net.  A  69.28.143.25
[etc]
```

Arelion Sweden AB

One netblock that wasn't associated with a "household name" was **Arelion Sweden AB**'s 62.115.253.0/24 netblock. It turns out that was actually Akamai, but by another name:

```
$ dnsdbq -i 62.115.253.0/24 -l0 -A90d -Tdatefix -S -k count
;; record times: 2021-02-23 14:33:34 .. 2023-05-29 23:52:54 (~2y ~95d)
;; count: 541,237,599
a1931.dscgi3.akamai.net.  A  62.115.253.186

;; record times: 2021-02-23 14:33:34 .. 2023-05-24 05:30:04 (~2y ~89d)
;; count: 524,396,317
a1931.dscgi3.akamai.net.  A  62.115.253.233

;; record times: 2020-07-03 07:38:28 .. 2023-05-30 01:52:41 (~2y ~330d)
;; count: 363,419,497
a239.gi3.akamai.net.  A  62.115.253.208

;; record times: 2020-07-03 07:38:28 .. 2023-05-30 01:52:41 (~2y ~330d)
;; count: 363,268,054
a239.gi3.akamai.net.  A  62.115.253.203

;; record times: 2020-07-03 07:38:26 .. 2023-05-15 10:30:07 (~2y ~316d)
;; count: 301,703,453
a117.dscd.akamai.net.  A  62.115.253.195

;; record times: 2020-07-03 07:38:26 .. 2023-05-15 10:30:07 (~2y ~316d)
;; count: 301,572,947
a117.dscd.akamai.net.  A  62.115.253.194

;; record times: 2020-07-03 07:38:32 .. 2023-05-15 10:30:06 (~2y ~316d)
;; count: 263,407,468
a27.dscd.akamai.net.  A  62.115.253.216

;; record times: 2020-07-03 07:38:32 .. 2023-05-15 10:30:06 (~2y ~316d)
;; count: 262,324,527
a27.dscd.akamai.net.  A  62.115.253.201
;; record times: 2020-07-03 07:38:26 .. 2023-05-26 09:20:55 (~2y ~327d)
;; count: 217,169,371
a17.d.akamai.net.  A  62.115.253.209

;; record times: 2020-07-03 07:38:26 .. 2023-05-15 10:29:59 (~2y ~316d)
;; count: 216,208,819
a17.d.akamai.net.  A  62.115.253.232
```

```
;; record times: 2020-07-03 07:38:29 .. 2023-05-26 09:20:43 (~2y ~327d)
;; count: 195,327,717
a1488.dscd.akamai.net.  A  62.115.253.224
[etc]
```

We'll let you explore the remaining top-count/"heavy hitter" domains we've exposed if you're interested and you're a Farsight DNSDB customer.

Dealing With the Full Spectrum of Count Values

While we obviously have some very large sums-of-counts with values up to and over 175 billion, we also have many other `/24` blocks where the sum of counts are quite small**.** For example, let's see how many `/24`'s have counts of exactly zero, ≤10, ≤100, or ≤1,000:

```
$ awk '$2 == 0' all-results.dat | wc -l
7597012            <-- 7597012/(256*256*256)*100=45.28%
$ awk '$2 <= 10' all-results.dat | wc -l
8434210            <-- 8434210/(256*256*256)*100=50.27%
$ awk '$2 <= 100' all-results.dat | wc -l
12662810           <-- 12662810/(256*256*256)*100=75.48%
$ awk '$2 <= 1000' all-results.dat | wc -l
13891741           <-- 13891741/(256*256*256)*100=82.80%
```

So at the same time we have some netblocks with truly huge cache miss counts, nearly 83% of all IPv4 netblocks have a count of 1,000 or under.

Why are there so many netblocks with "low" aggregate cache miss counts? Remember that in an effort to be inclusive, we included special purpose netblocks that we normally wouldn't expect to actually see in "A" records, such as 0.0.0.0/8, 10.0.0.0/8, 127.0.0.0/8 plus multicast space (224.0.0.0/4) and class "E" space (240.0.0.0/4). That's a lot of potentially low-traffic `/24`'s. However, there are other "legacy netblocks"[15] that also deserve a "call out," including:

4	Formerly BBN
6	Army Information Center (02/1994)
7	DOD Network Information Center (04/1995)
9	IBM (except for 9.9.9.0/24) (08/1992)
17	Apple (07/1992)
19	Ford (05/1995)
22	DISA (05/1993)
25	UK MOD (08/2005)
26	DISA (05/1995)
28	DSI-North (07/1992)
29	DISA (07/1991)
30	DISA (07/1991)
32	Formerly AT&T Global
33	DLA System Automation Center (01/1991)

[15] https://en.wikipedia.org/wiki/List_of_assigned_/8_IPv4_address_blocks

```
44          ARDC (except for 44.192.0.0/10 which was sold to Amazon)
48          Prudential (05/1995)
53          Mercedes-Benz (10/1993)
55          DOD NIC (04/1995)
56          Formerly US Postal Service
57          Formerly SITA
100         (100.64.0.0/10 is reserved for carrier grade NAT per RFC 6598)
205         DOD (05/1998)
214         DOD (03/1998)
215         DOD (03/1998)
```

These legacy Netblocks are interesting for several reasons. Given their age, with many of them allocated before the birth of HTTP, the IP allocation patterns are different from other more recently allocated blocks. Organizations with access to these blocks have more IPs, so they may not feel the need to place machines behind NATs, for example. Some sets of these blocks have since been resold to others, further highlighting differences in IP usage.

Data Normalization

The broad range of observed counts, ranging from billions to tens, makes it challenging to cleanly represent our data unless we transform it. We decided to tackle this by taking the common (base 10) logarithm of our counts.[16] We then plotted the cumulative distribution function for our count variable:

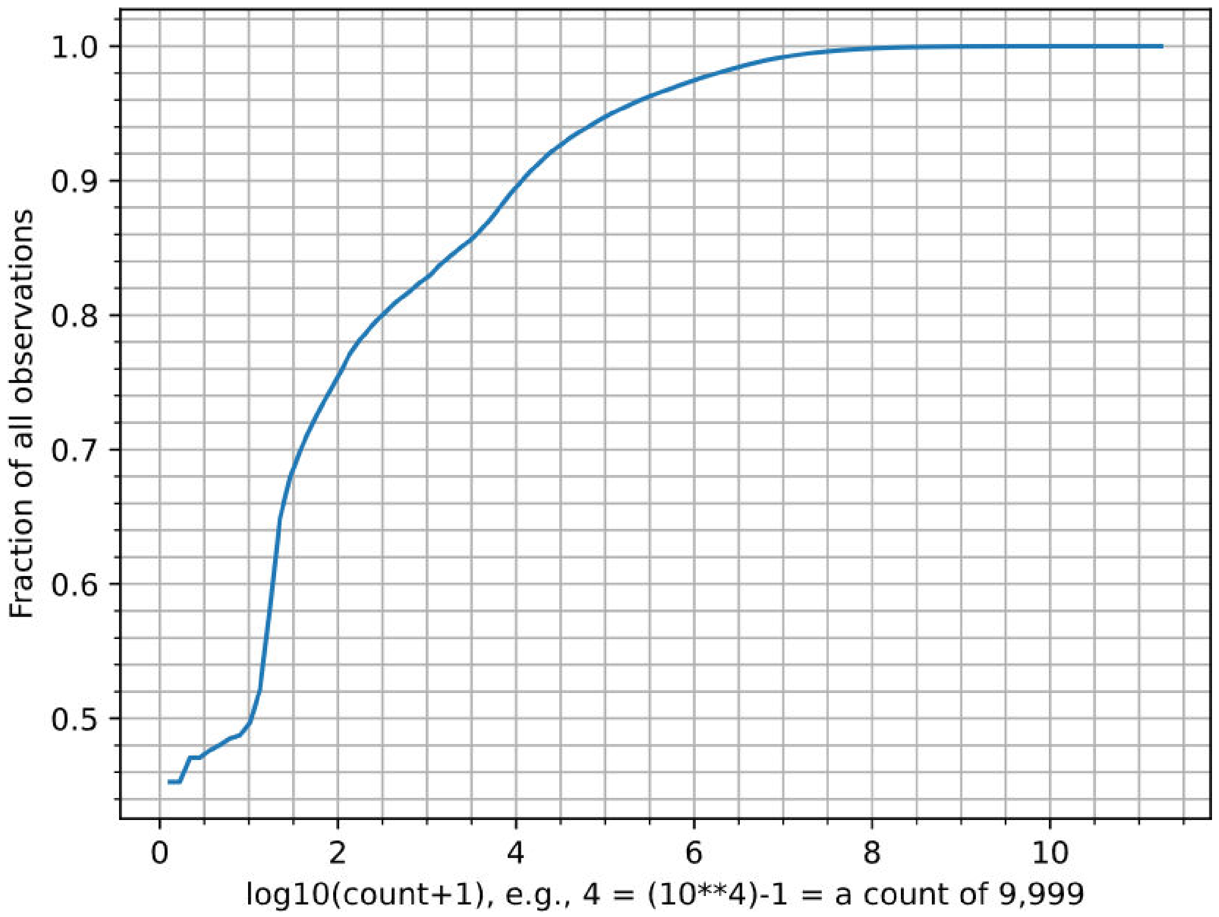
```
$ cat plot_cdf.py
#!/usr/local/bin/python3
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

mycdfplot = 'mycdfplot.pdf'

df = pd.read_csv('all-results-just-counts.dat')
df['log_my_count'] = np.log10(df['my_count']+1)
print(df.describe())

count, bins_count = np.histogram(df.log_my_count, bins=100)
pdf = count / sum(count)
cdf = np.cumsum(pdf)

ax = plt.gca()
ax.plot(bins_count[1:], cdf, label="CDF")
ax.axis('tight')
plt.minorticks_on()
plt.grid(which='both')
plt.xlabel('log10(count), e.g., 4 = 10**4 = count of 10,000\n')
plt.ylabel('Fraction of all observations')
plt.savefig(mycdfplot,dpi=300)
```

[16] Because log10 of zero is undefined, we added 1 to all counts before computing the log.

Running that code, we get the following:

```
            my_count  log_my_count
count  1.677722e+07  1.677722e+07
mean   1.534745e+06  1.350179e+00
std    1.846779e+08  1.738437e+00
min    0.000000e+00  0.000000e+00
25%    0.000000e+00  0.000000e+00
50%    1.000000e+01  1.041393e+00
75%    9.200000e+01  1.968483e+00
max    1.756923e+11  1.124475e+01
```

CDF of log10(count)



This cumulative distribution function over the log10 space of counts. This gives us a way to visualize and reason about this extraordinarily large space--half of the data has a count of 10 or less, and 75% of the data has a count under 100, and on the other extreme end, the max value is over 175 billion. This is a very extreme distribution that is largely sparse, but has some real hot spots. We will use this log10 transform in our visualizations which follow.

Per /8 Violin Plots

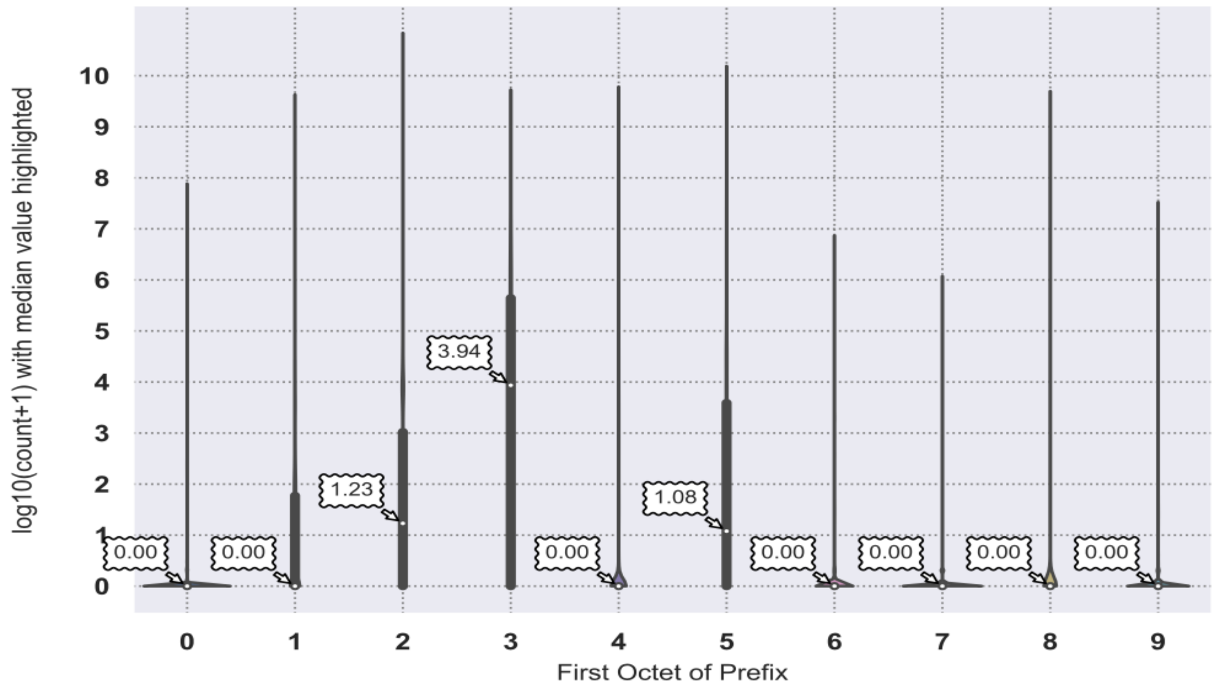
Now that we've found that a log transform should work for the full range of our count values, let's build some violin plots for each `/8`. Violin plots should help us show the maximum, minimum and median values for each of our 256 `/8` prefixes in a way that invites visual comparisons.

Each Violin plot shows the log transformed count along the Y-axis. The X-axis contains each `/8` prefix in numerical order with 10 prefixes per image, with the last image containing the final 6 prefixes. The median value of each prefix is given, and the shape of each graph provides an indication of the distribution of counts for each prefix with a wider shape indicating entries.

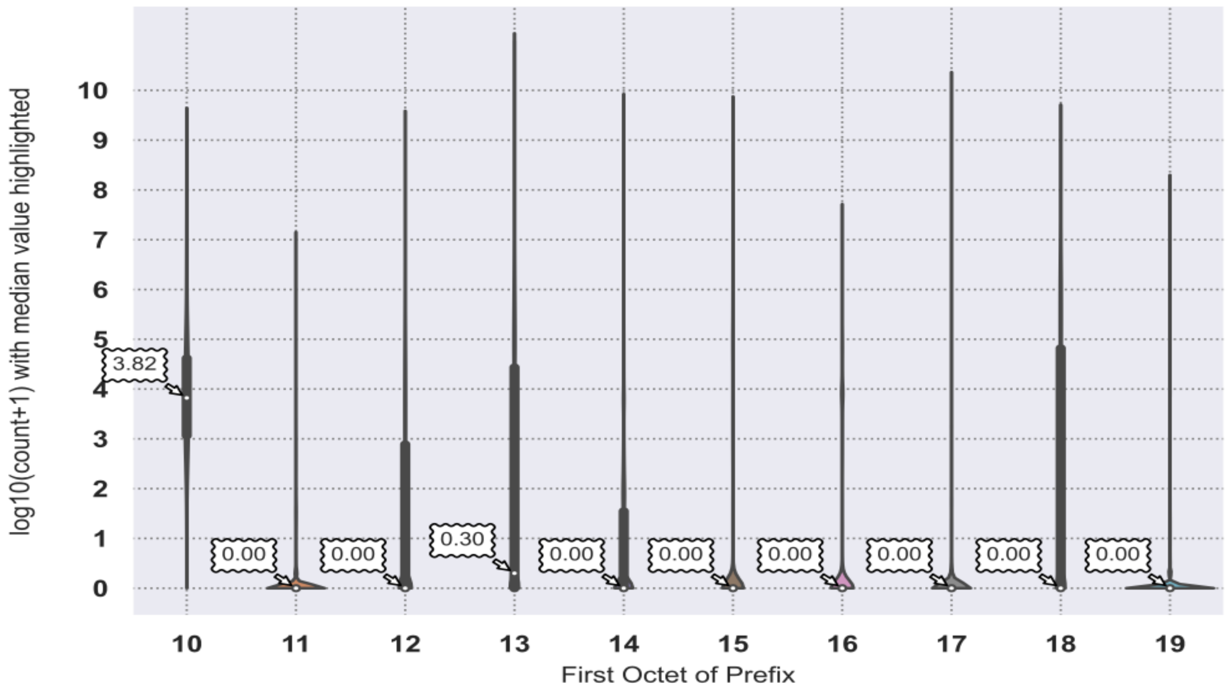
While some of the initial plot panels aren't very visually interesting, they become more so when we hit 60-89, 130-159, and 194-219.

The Violin plot visualizations begin on the next page.

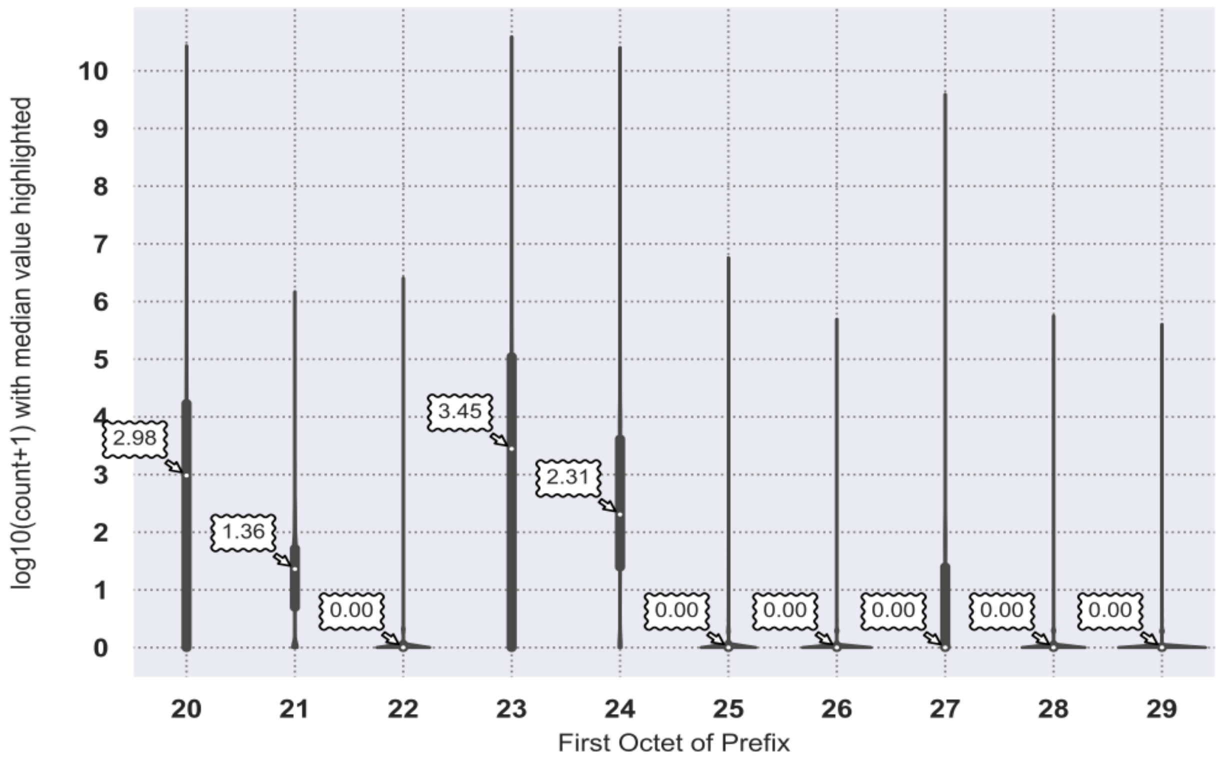
Aggregate count Per /24, Grouped by First Octet



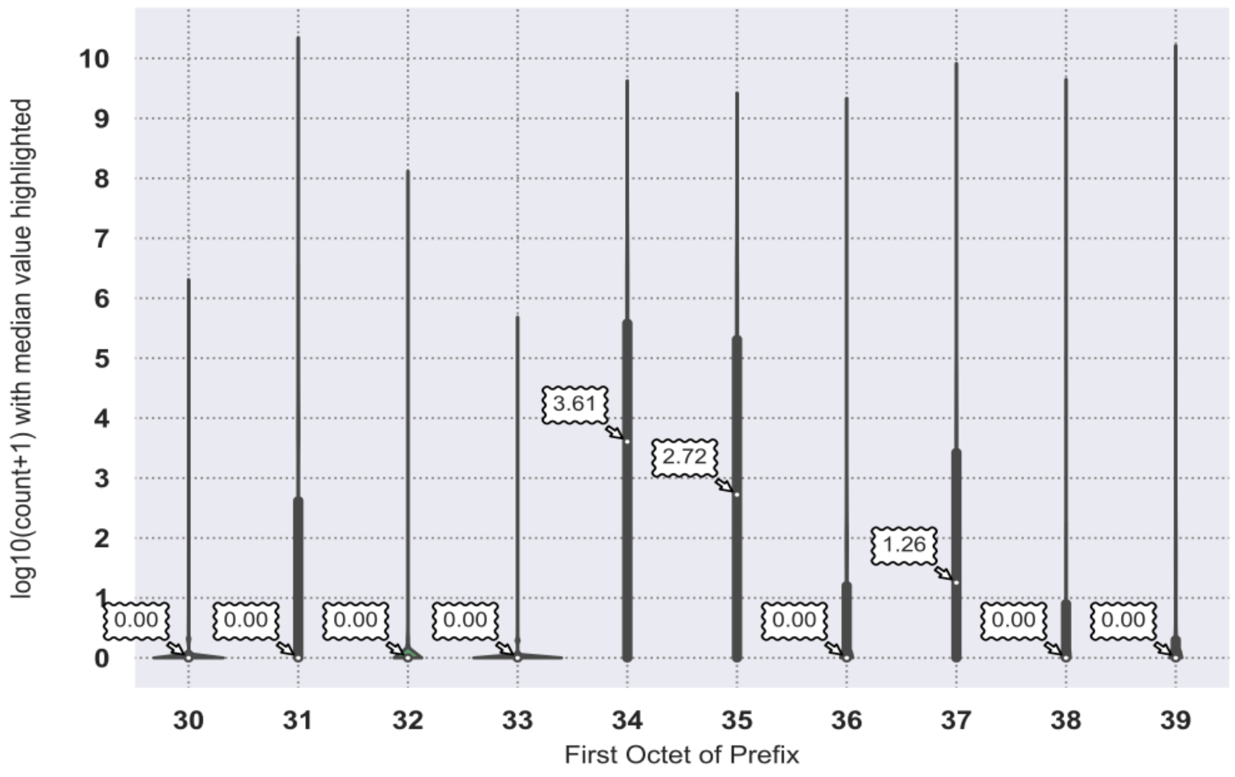
Aggregate count Per /24, Grouped by First Octet



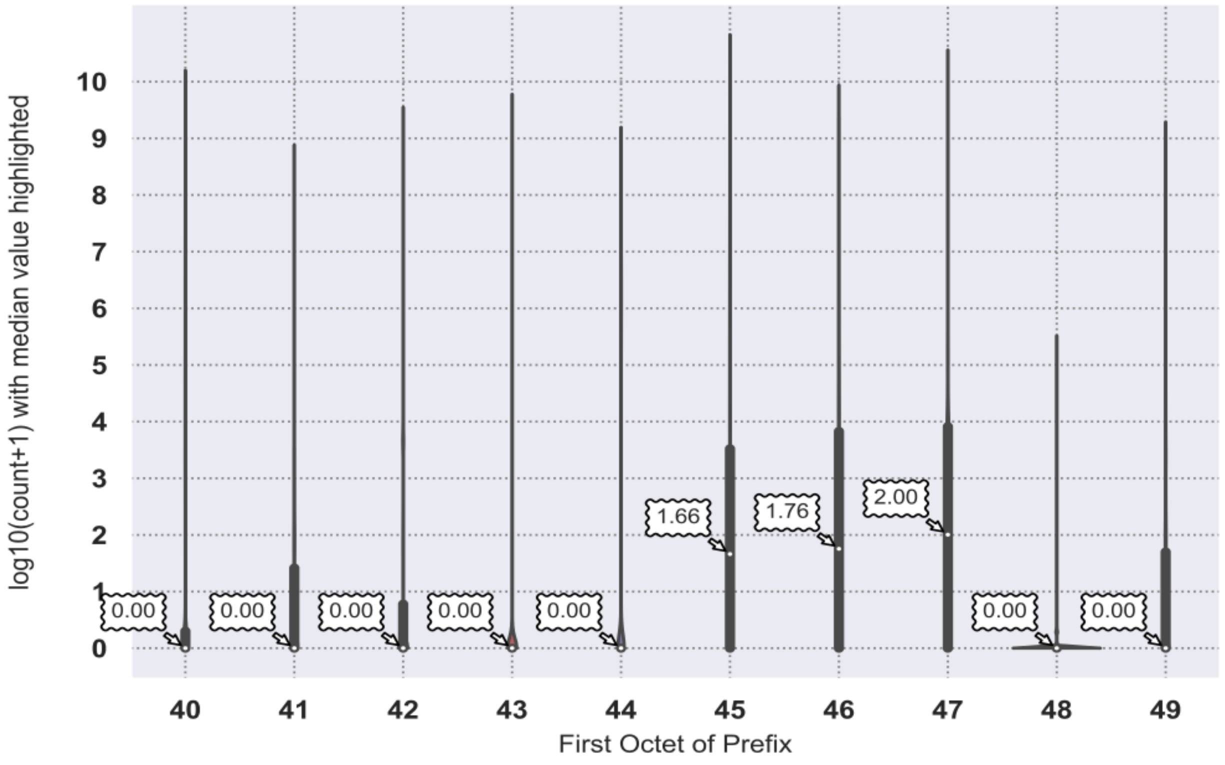
Aggregate count Per /24, Grouped by First Octet



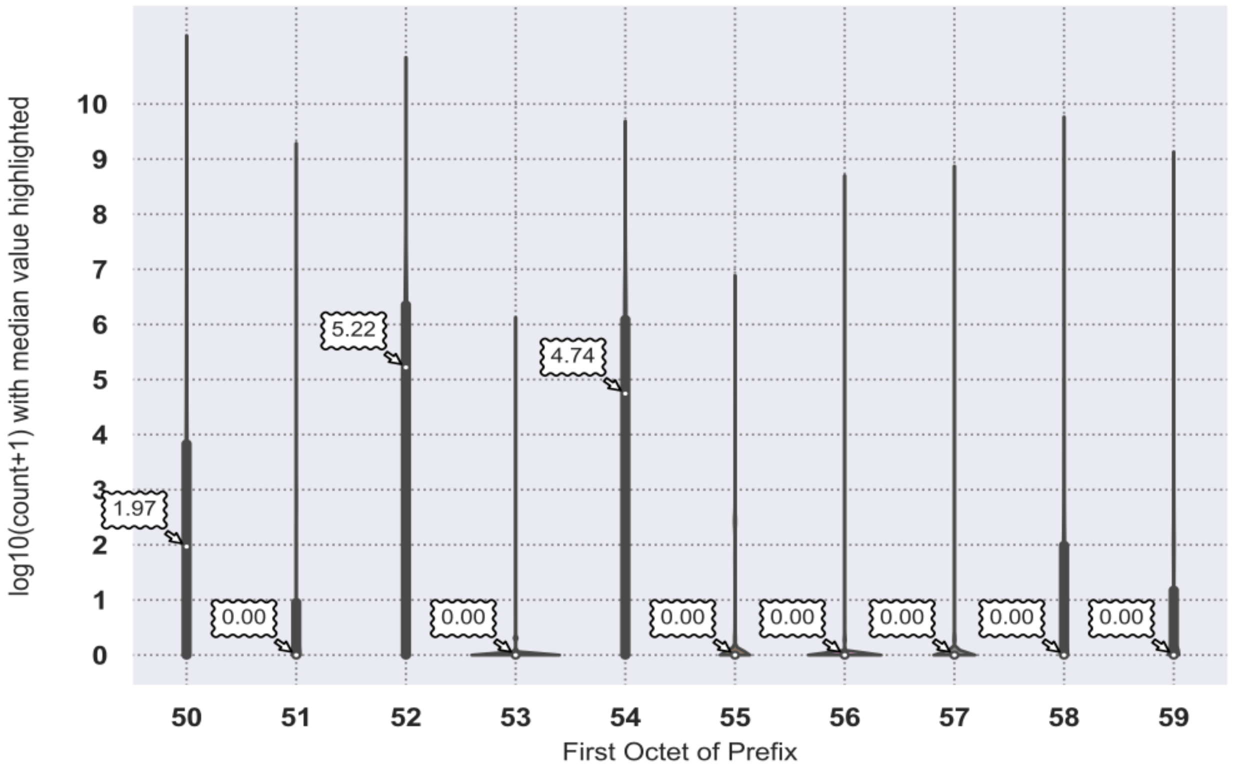
Aggregate count Per /24, Grouped by First Octet



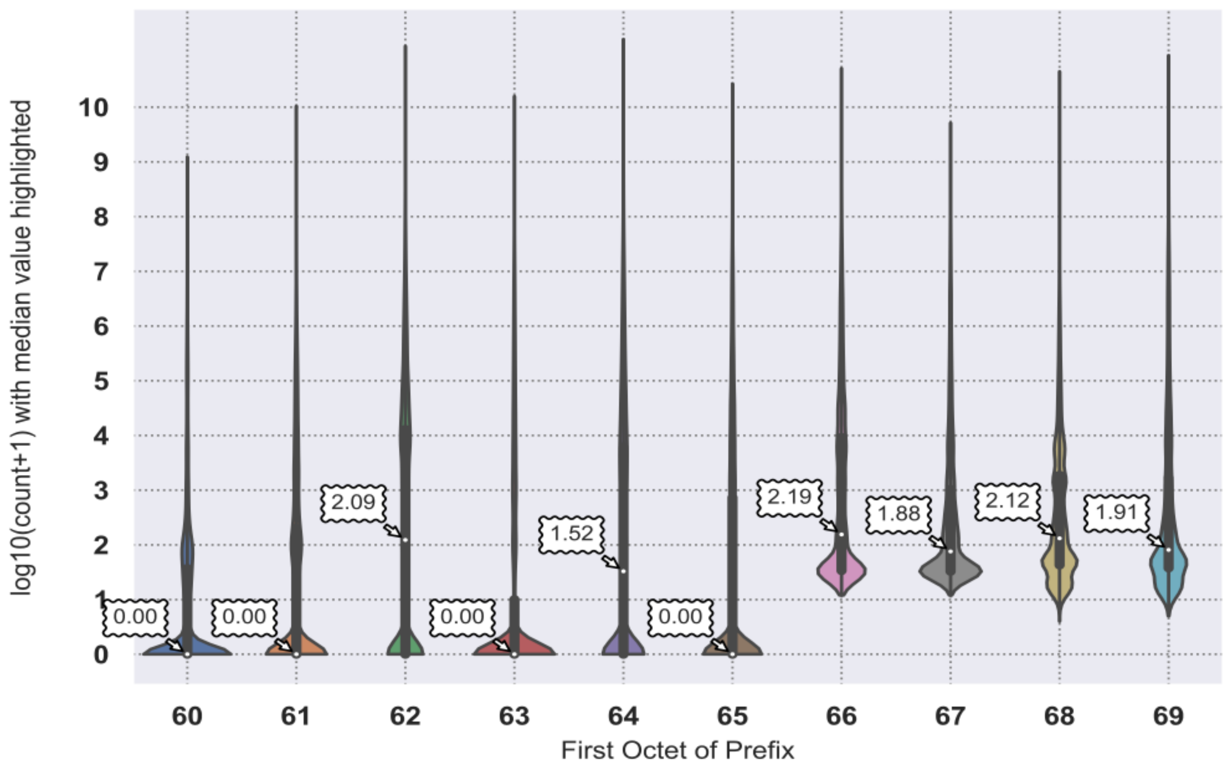
Aggregate count Per /24, Grouped by First Octet



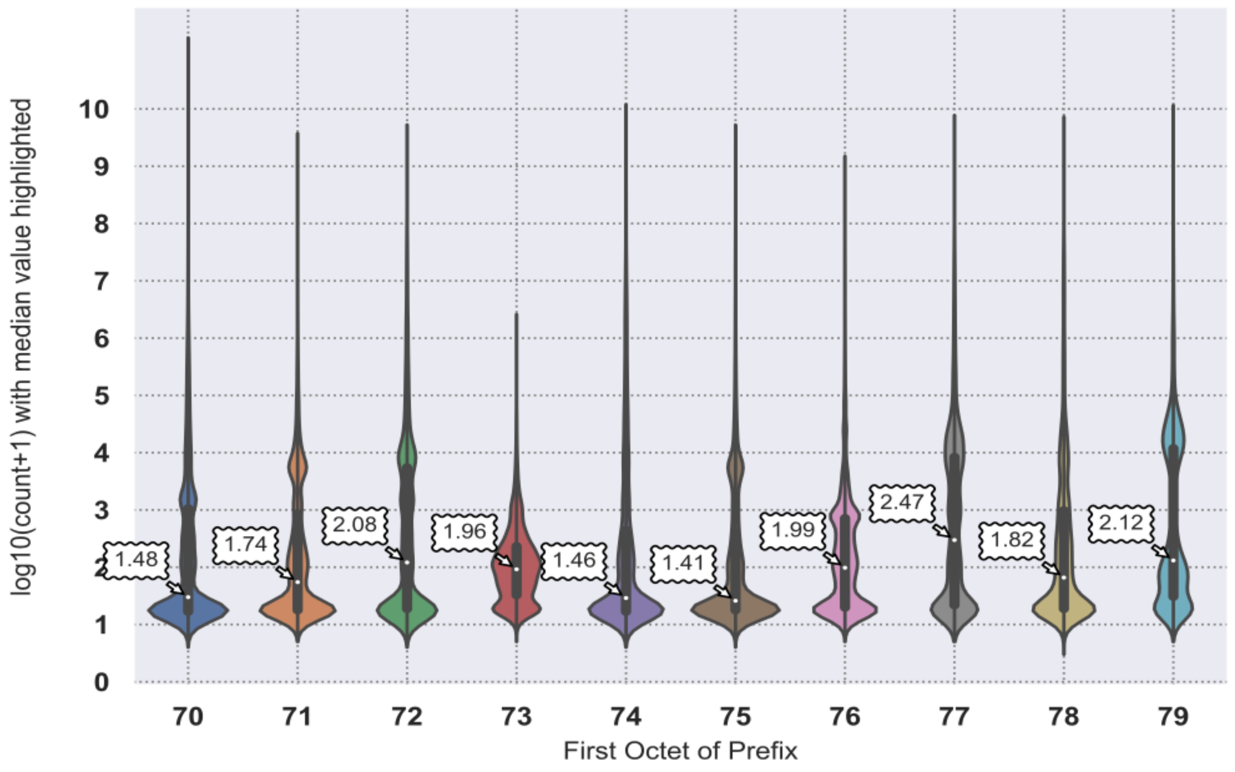
Aggregate count Per /24, Grouped by First Octet



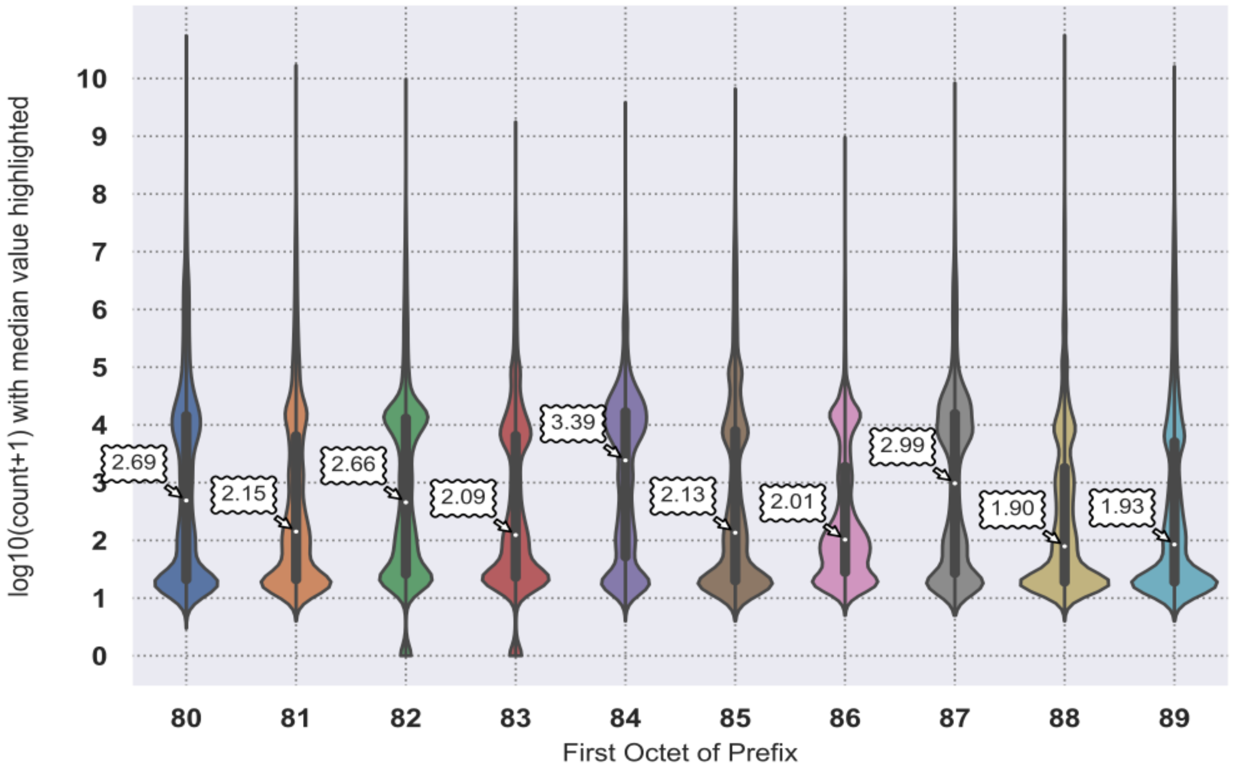
Aggregate count Per /24, Grouped by First Octet



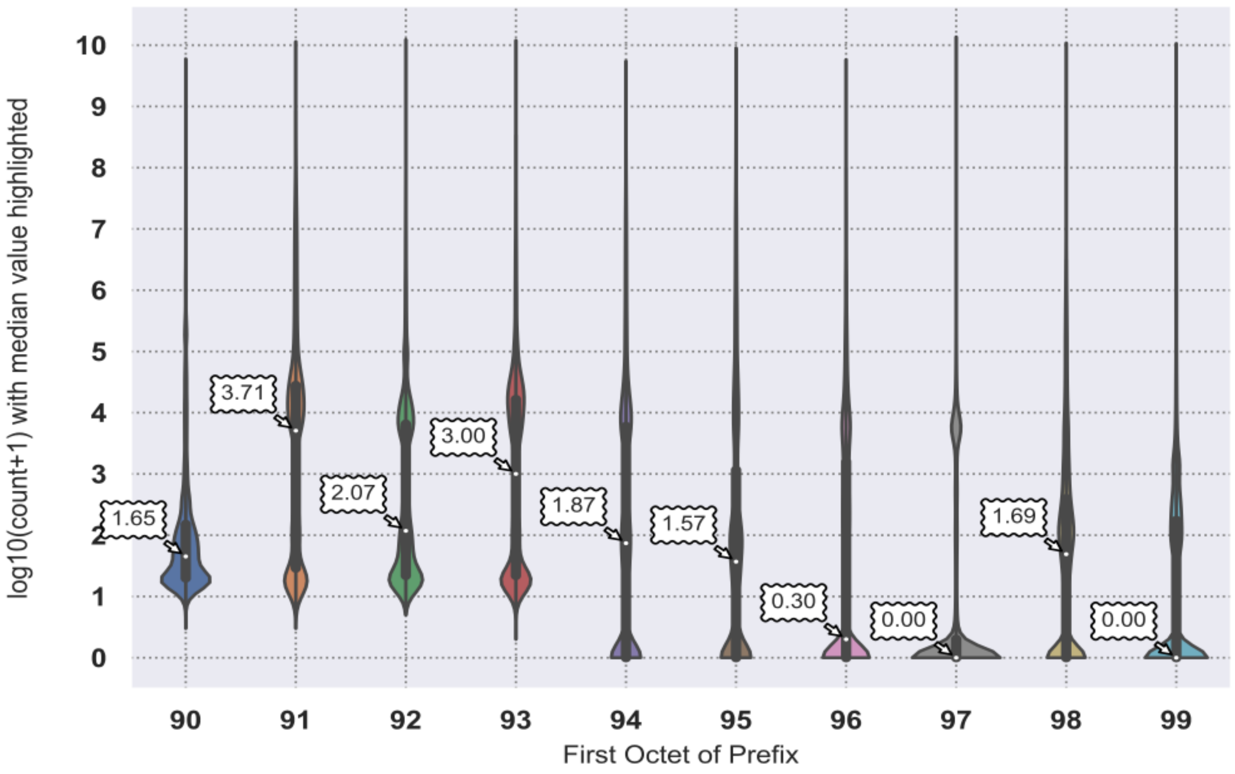
Aggregate count Per /24, Grouped by First Octet



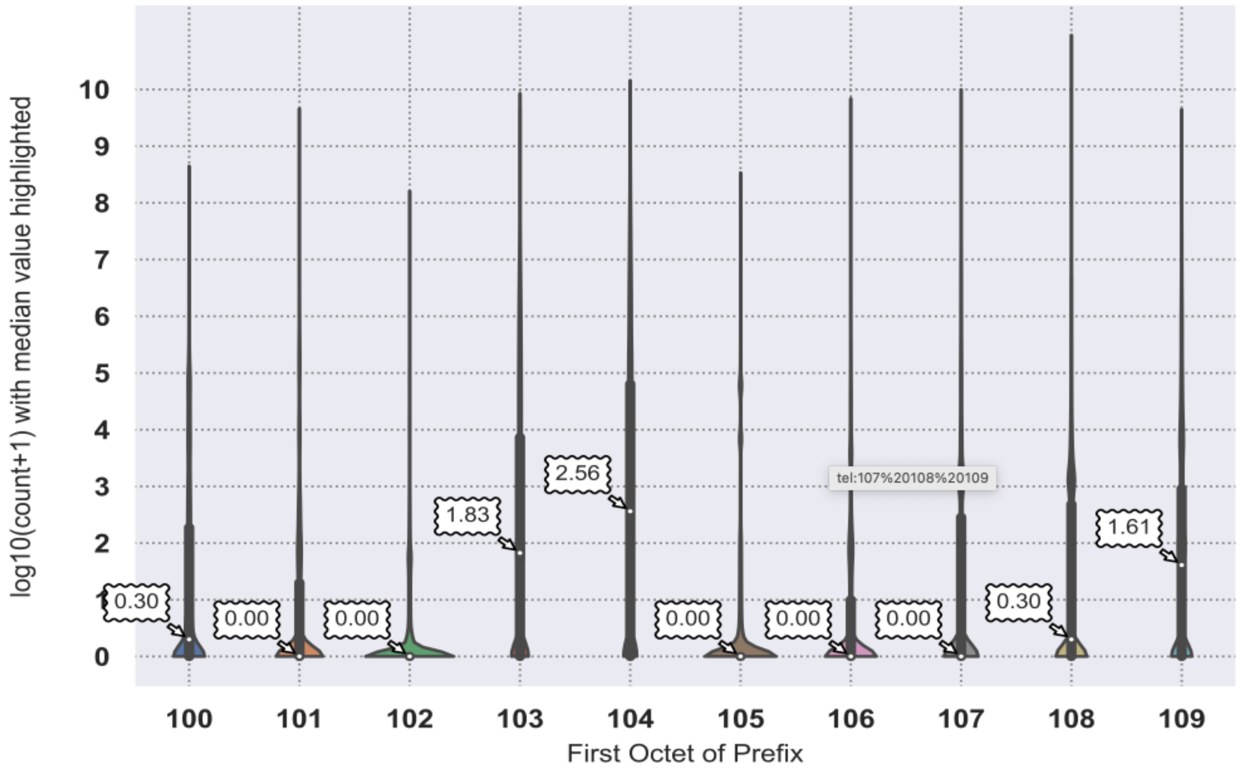
Aggregate count Per /24, Grouped by First Octet



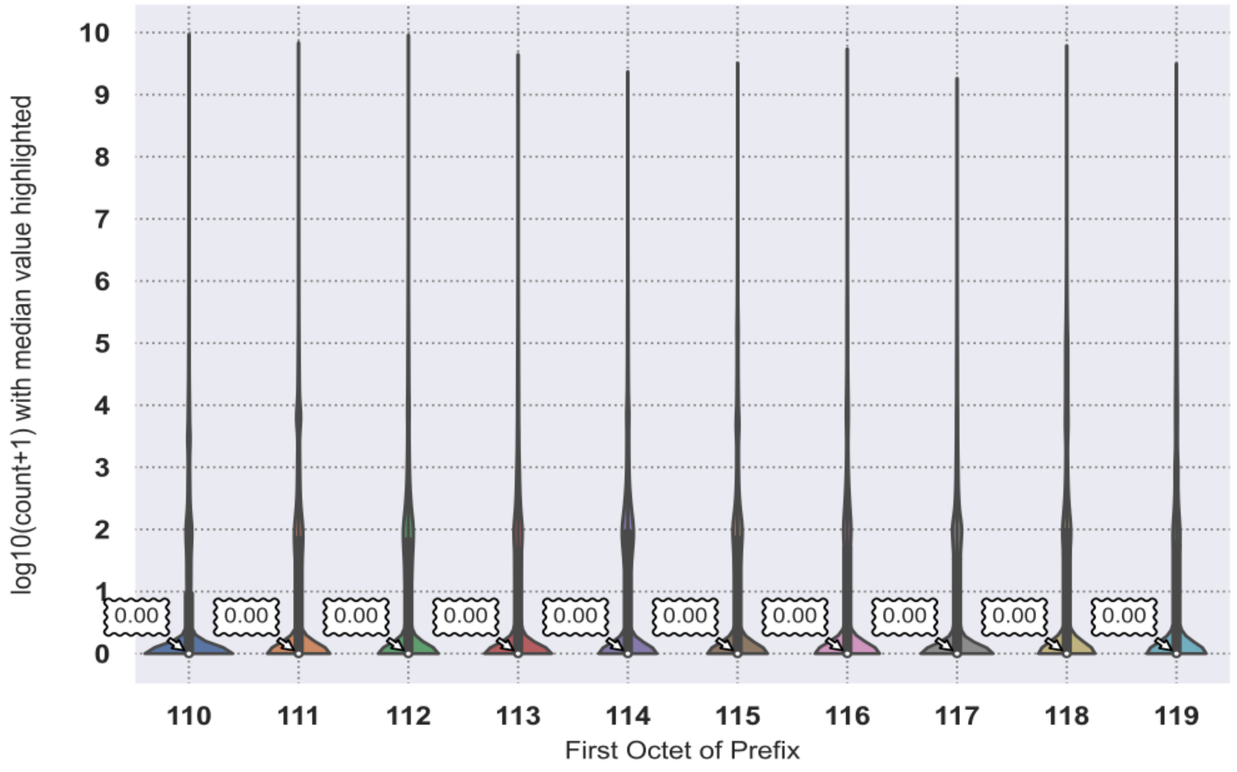
Aggregate count Per /24, Grouped by First Octet



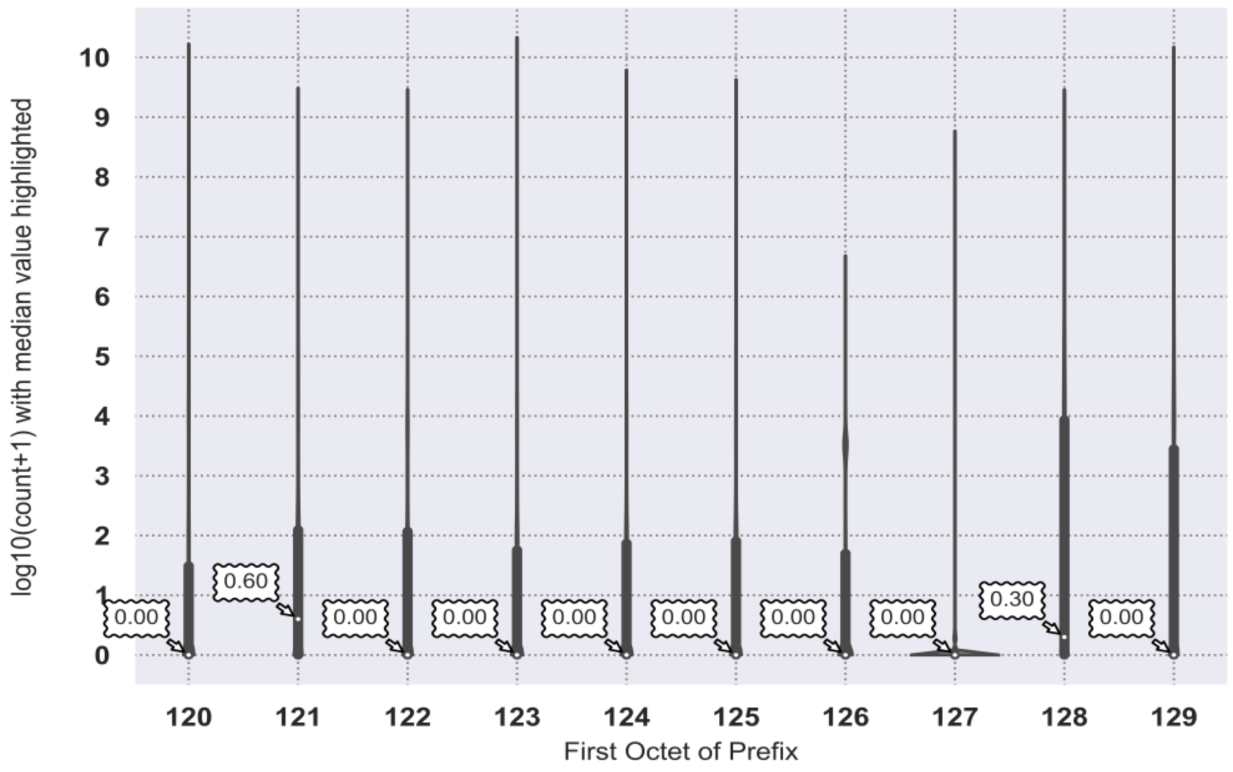
Aggregate count Per /24, Grouped by First Octet



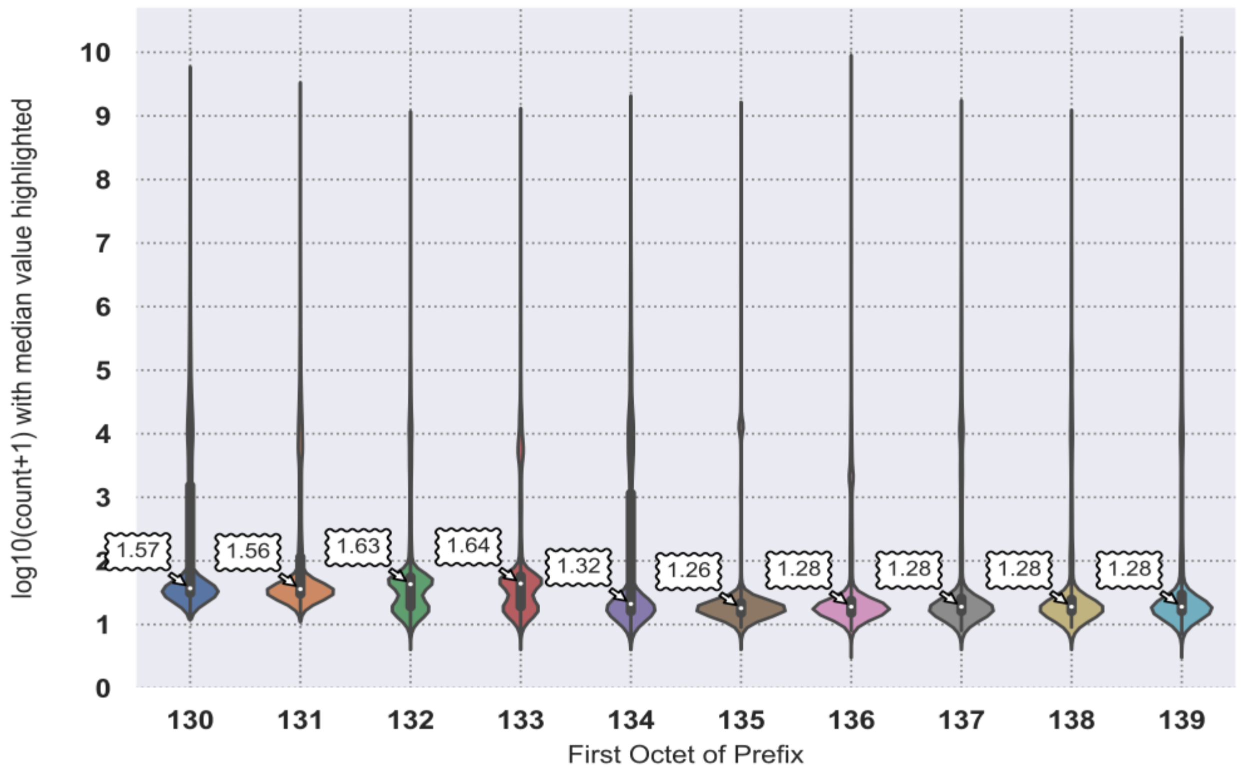
Aggregate count Per /24, Grouped by First Octet



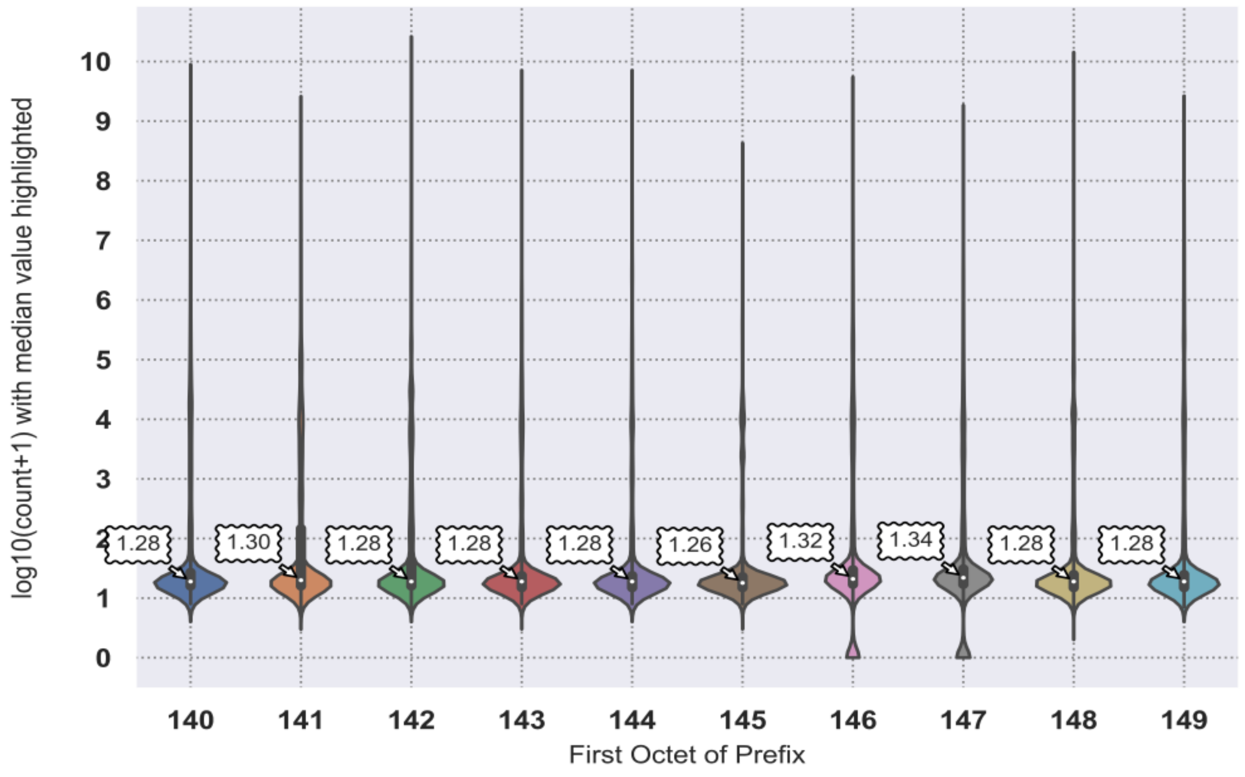
Aggregate count Per /24, Grouped by First Octet



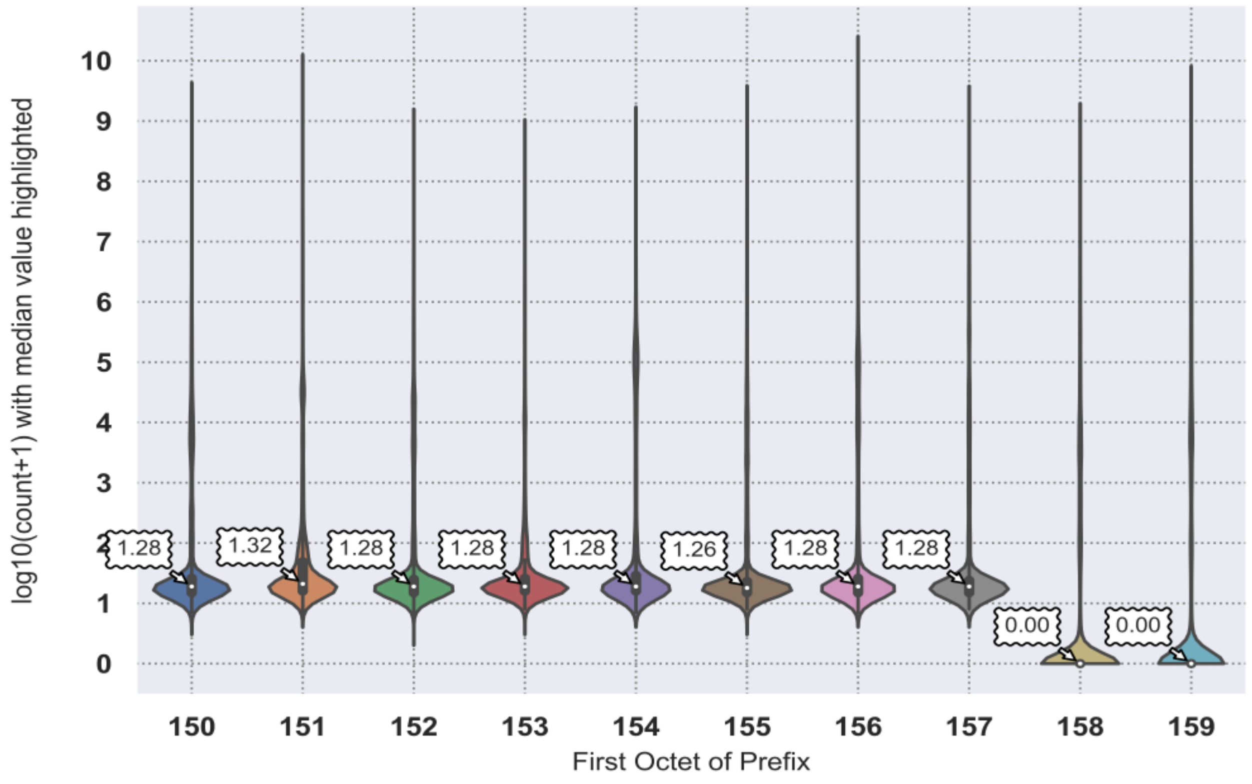
Aggregate count Per /24, Grouped by First Octet



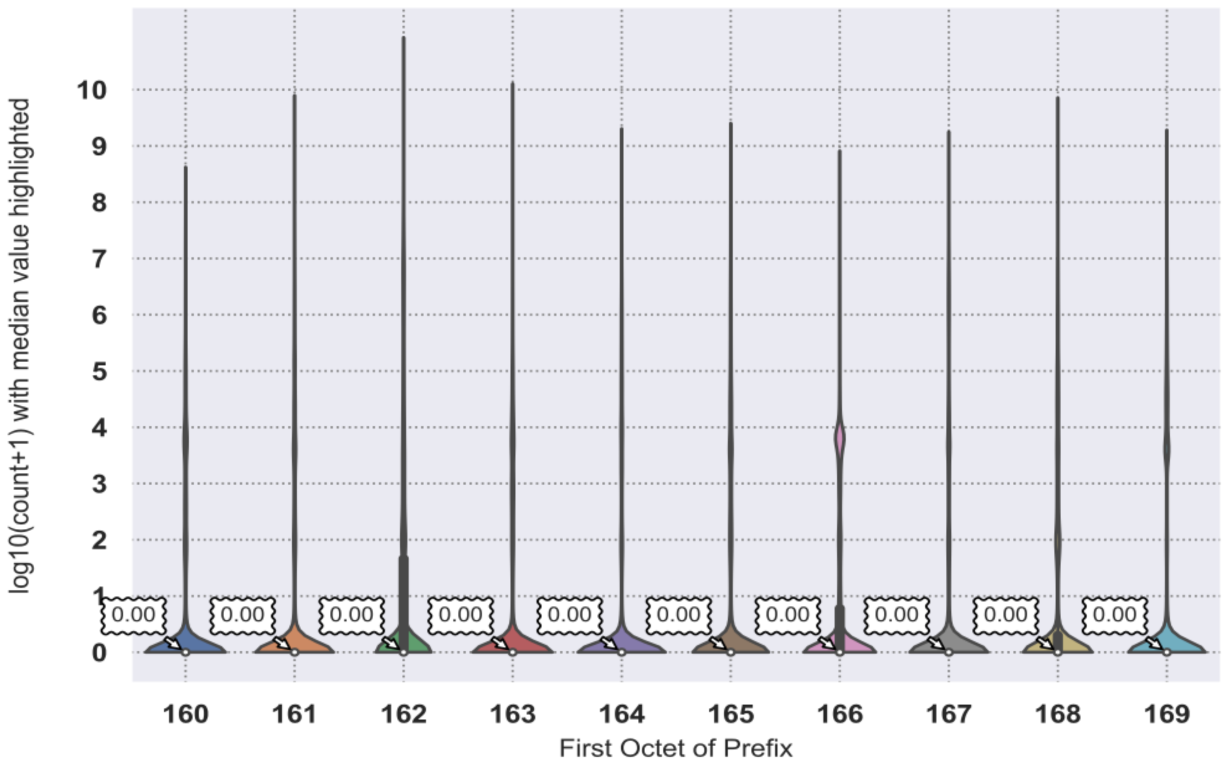
Aggregate count Per /24, Grouped by First Octet



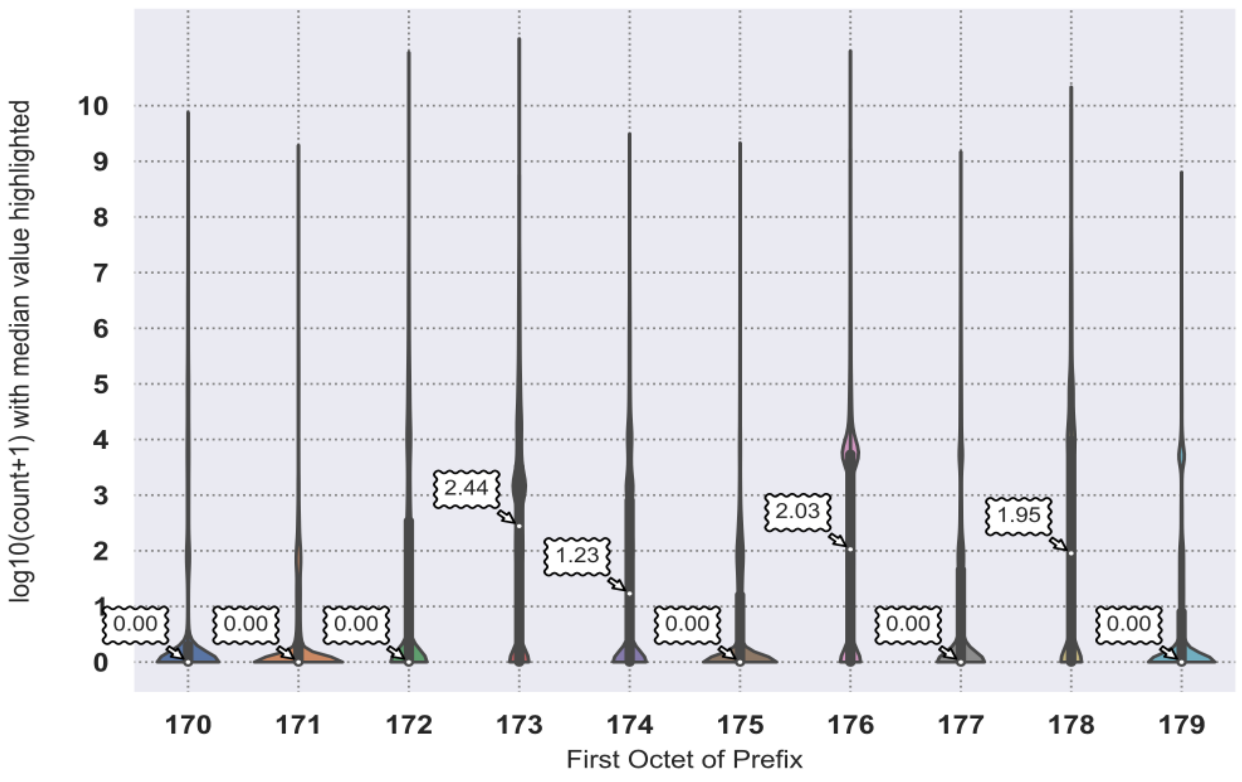
Aggregate count Per /24, Grouped by First Octet



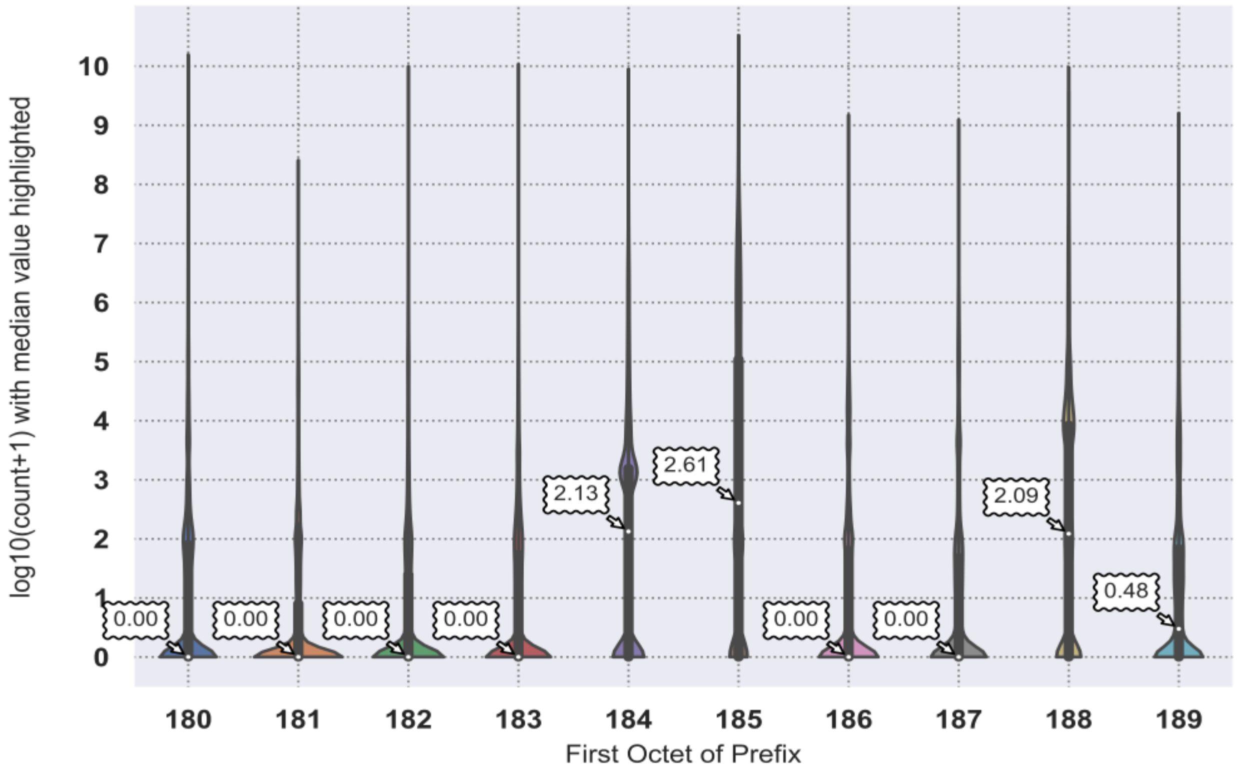
Aggregate count Per /24, Grouped by First Octet



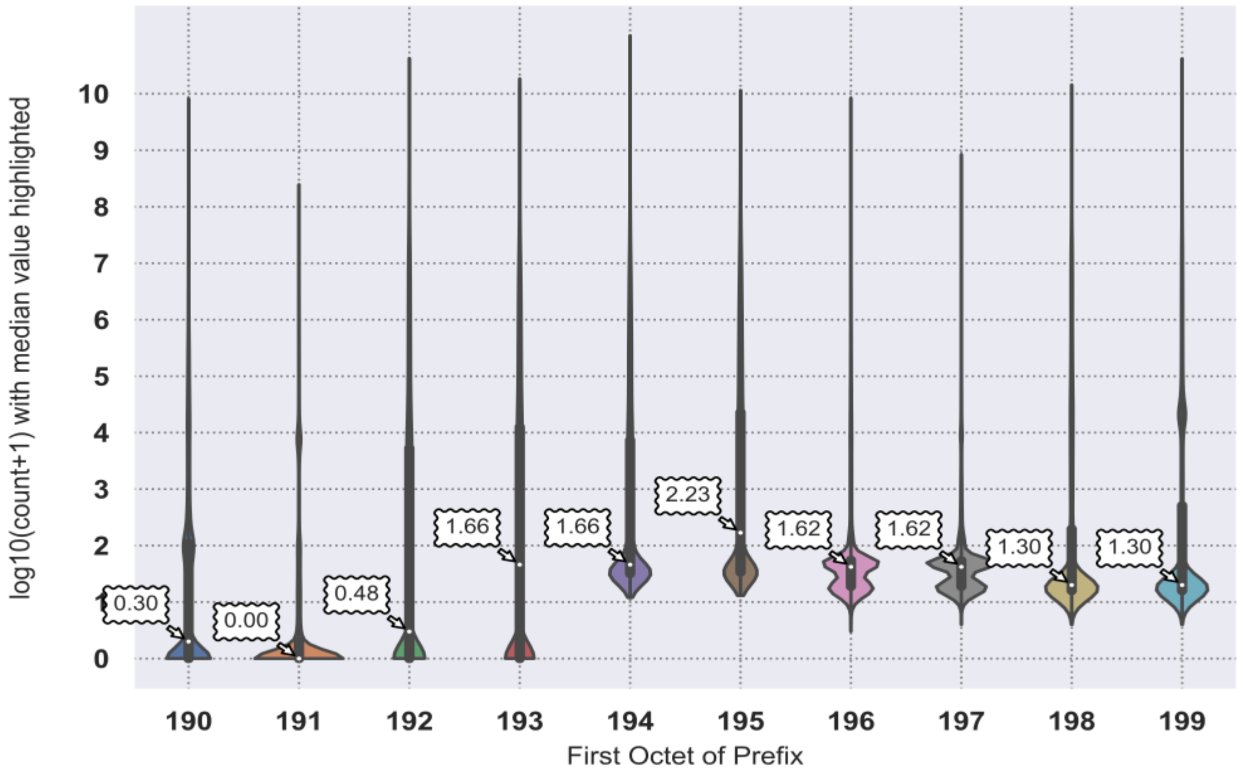
Aggregate count Per /24, Grouped by First Octet



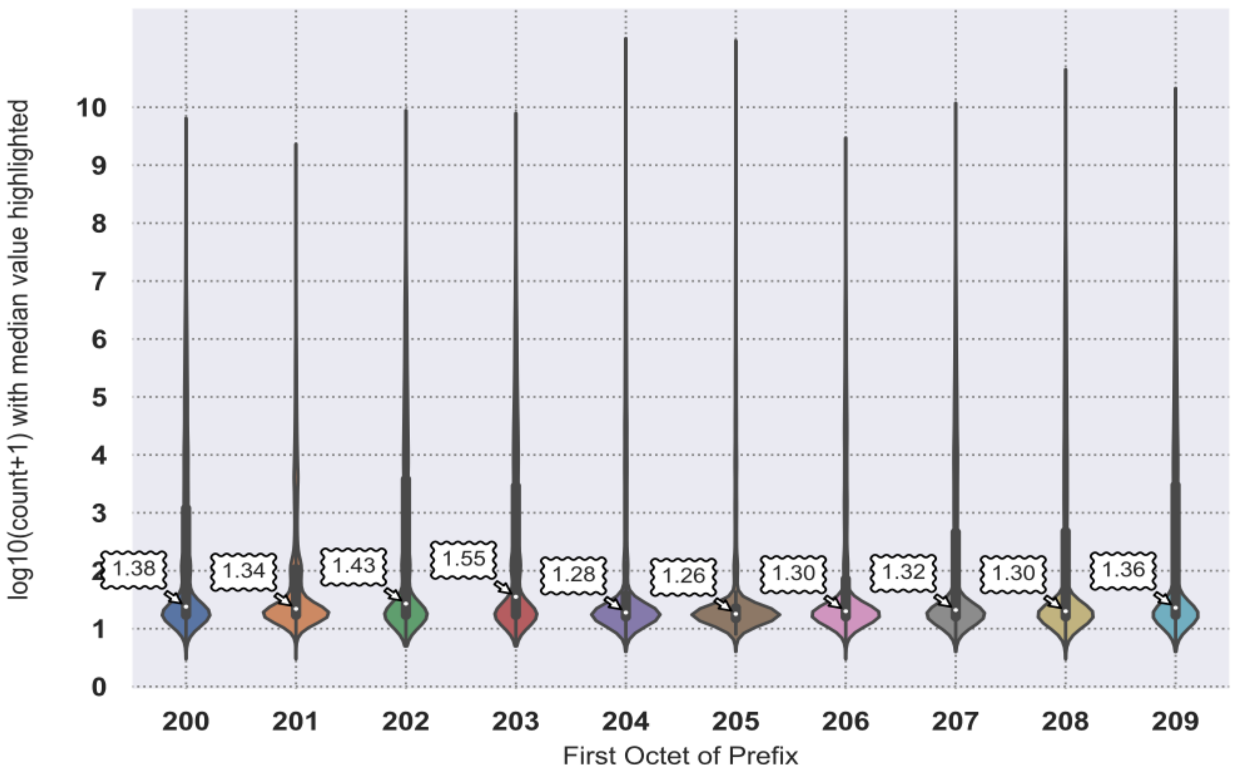
Aggregate count Per /24, Grouped by First Octet



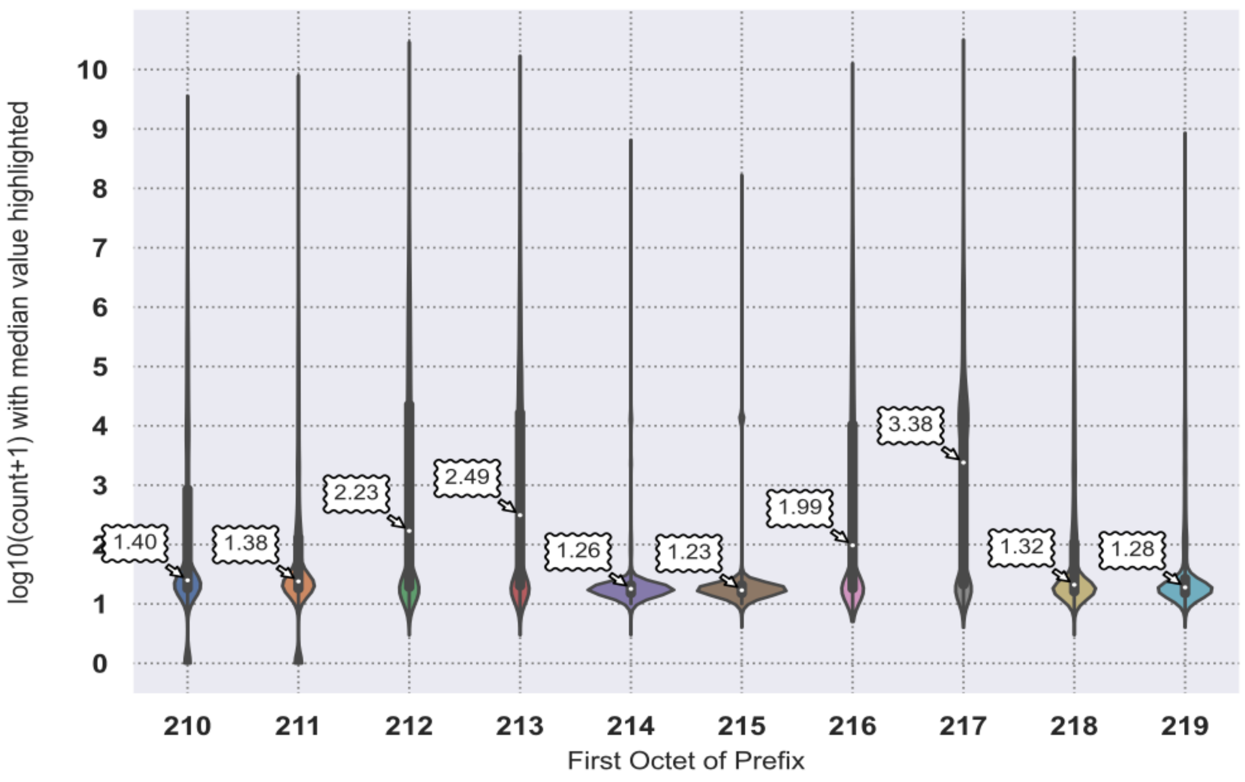
Aggregate count Per /24, Grouped by First Octet



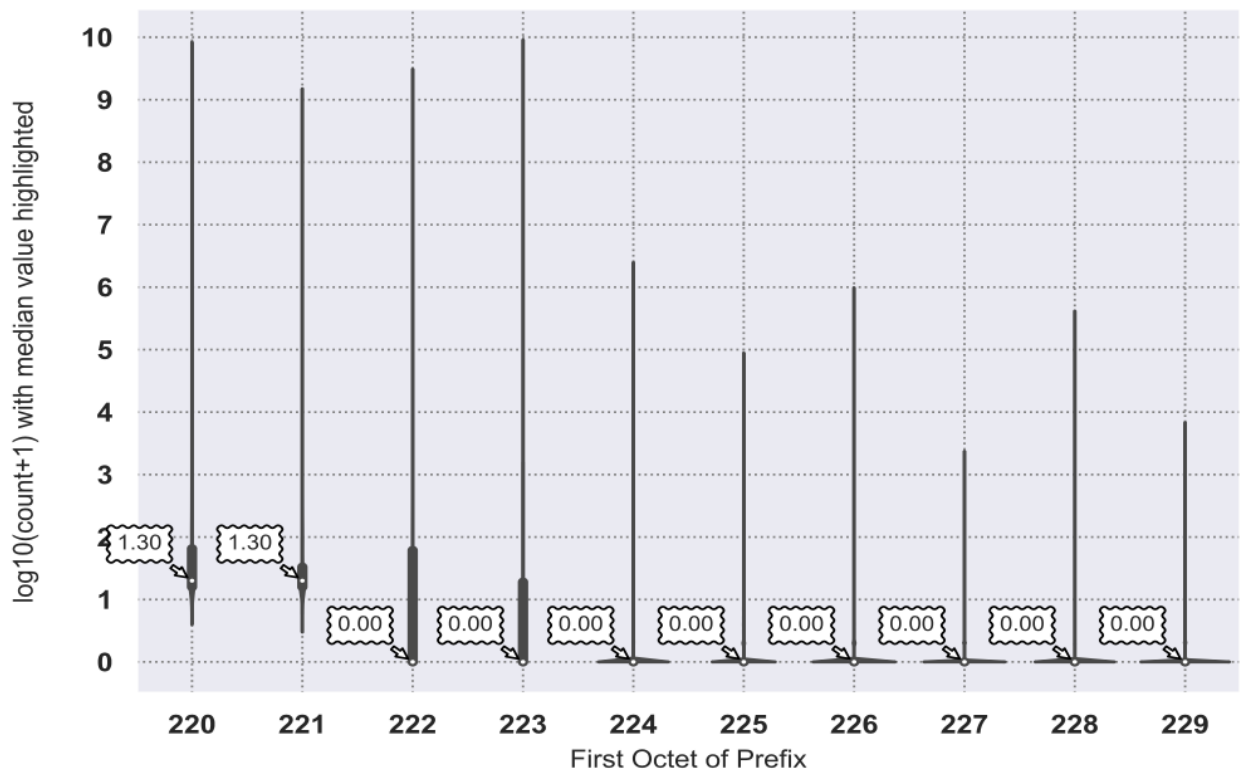
Aggregate count Per /24, Grouped by First Octet



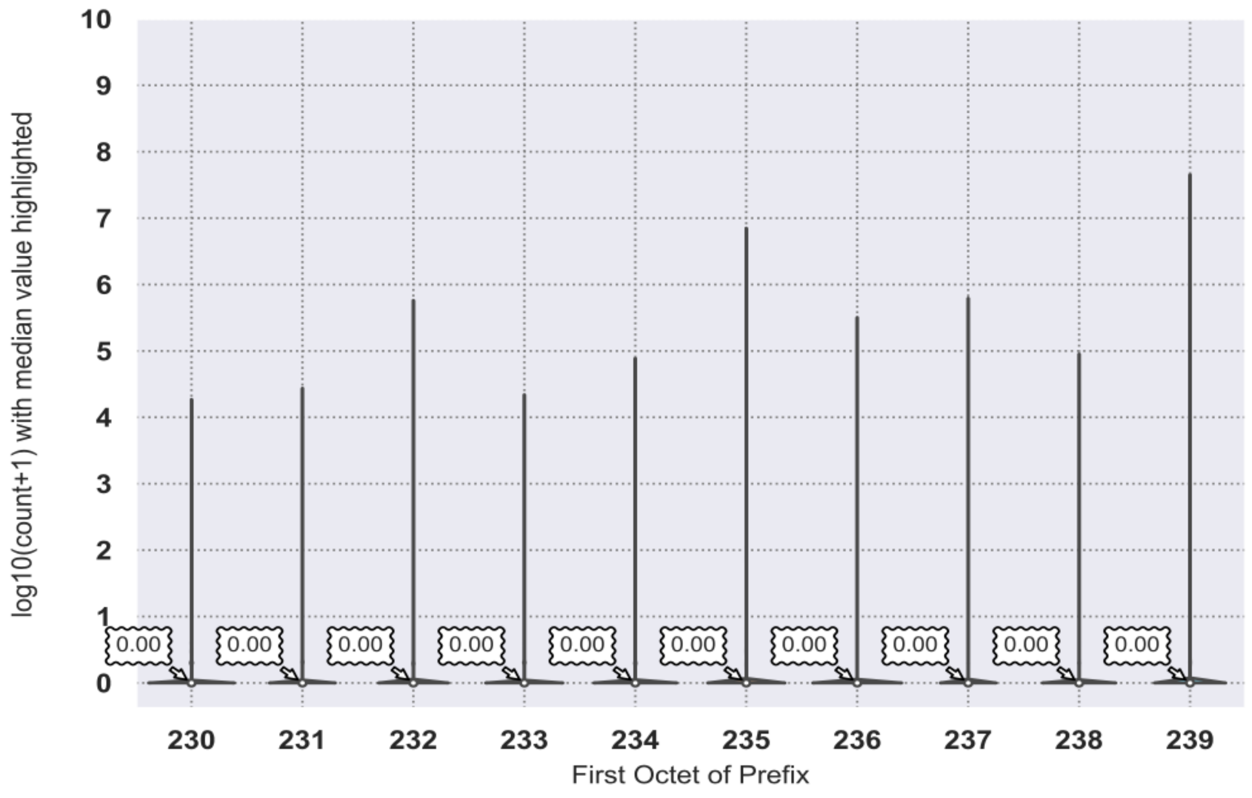
Aggregate count Per /24, Grouped by First Octet



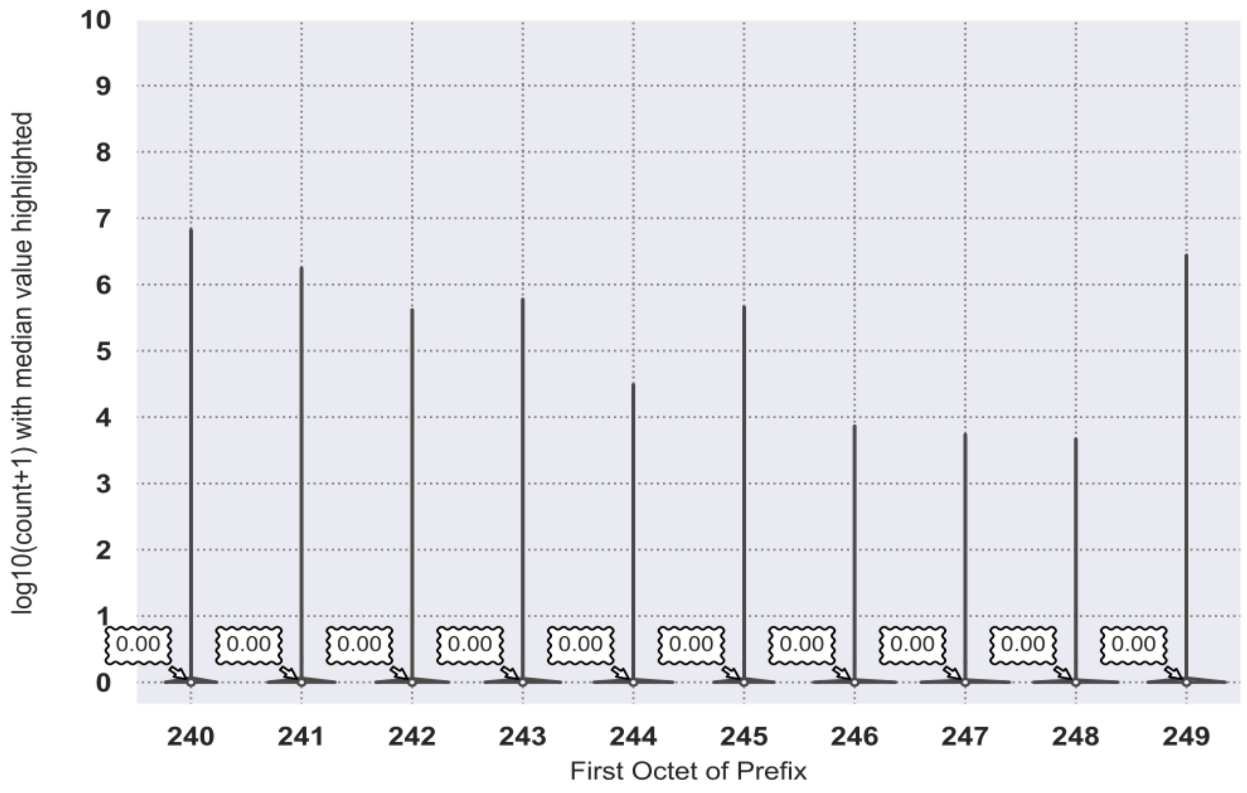
Aggregate count Per /24, Grouped by First Octet



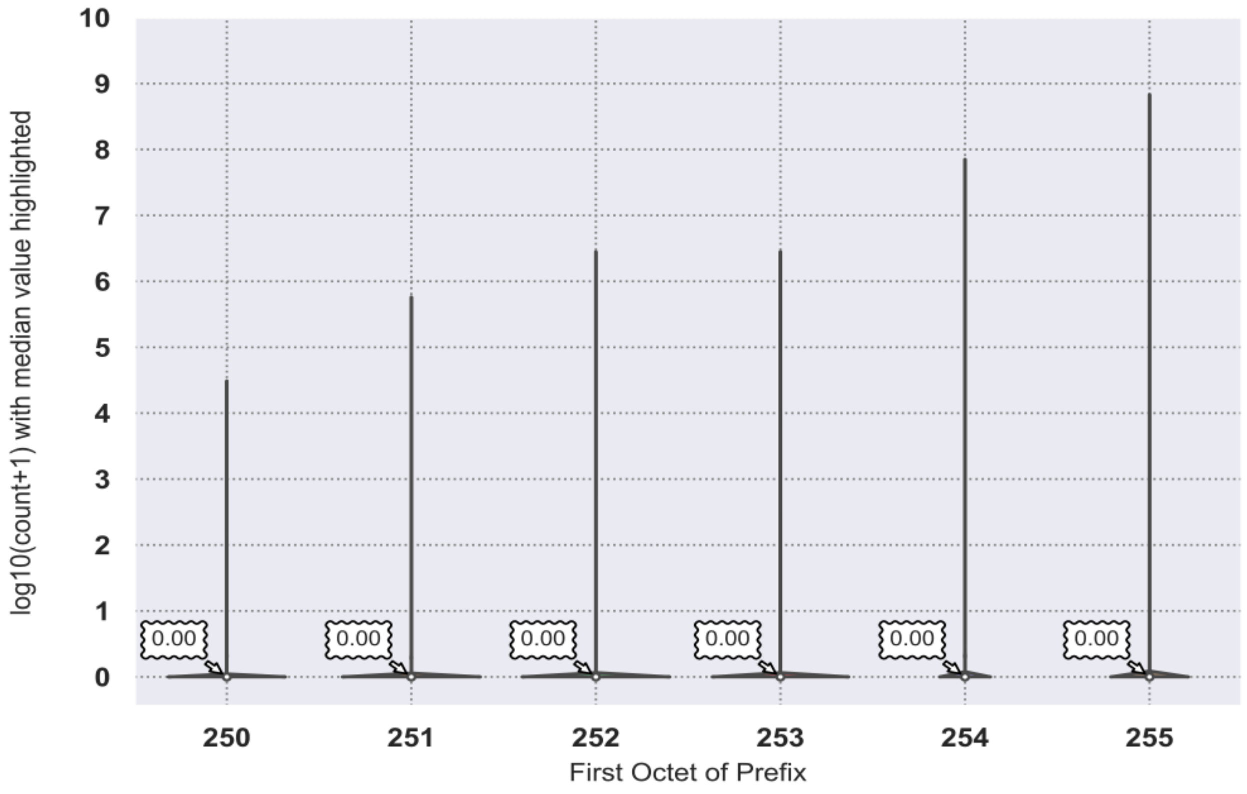
Aggregate count Per /24, Grouped by First Octet



Aggregate count Per /24, Grouped by First Octet



Aggregate count Per /24, Grouped by First Octet



Hilbert Curve “Heatmaps”

Named after the German mathematician David Hilbert, Hilbert Curves[17] are a way to organize ordered data in a continuous space-filling curve. They were first described in 1891, and became a popular way to reason about IP address space thanks in part to Randall Munroe’s XKCD comics.[18]

We now return to the production of the Hilbert Curve “heatmap” shown on the cover of this report. To generate it, we used an updated version of ipv4-heatmap[19] that has support for the viridis[20] color palette. With data in hand, producing the cover graph was as easy as:

```
$ cat all-results.dat | ipv4-heatmap -A 0 -B 40000000000 \
  -a ~/block-labels-2.txt -c 0xE0FFFF
```

Decoding that command:

- `-A 0` means that input data less than or equal to zero wll be set to 1
- `-B 40000000000` means that input data will be logarithmically scaled so that input data greater than or equal to 40 billion will be will be set to 255
- `-a /block-labels-2.txt` specifies the annotations used to label each block. So:

  ```
  0.0.0.0/8        0
  1.0.0.0/8        1
  2.0.0.0/8        2
  3.0.0.0/8        3
  [etc]
  ```

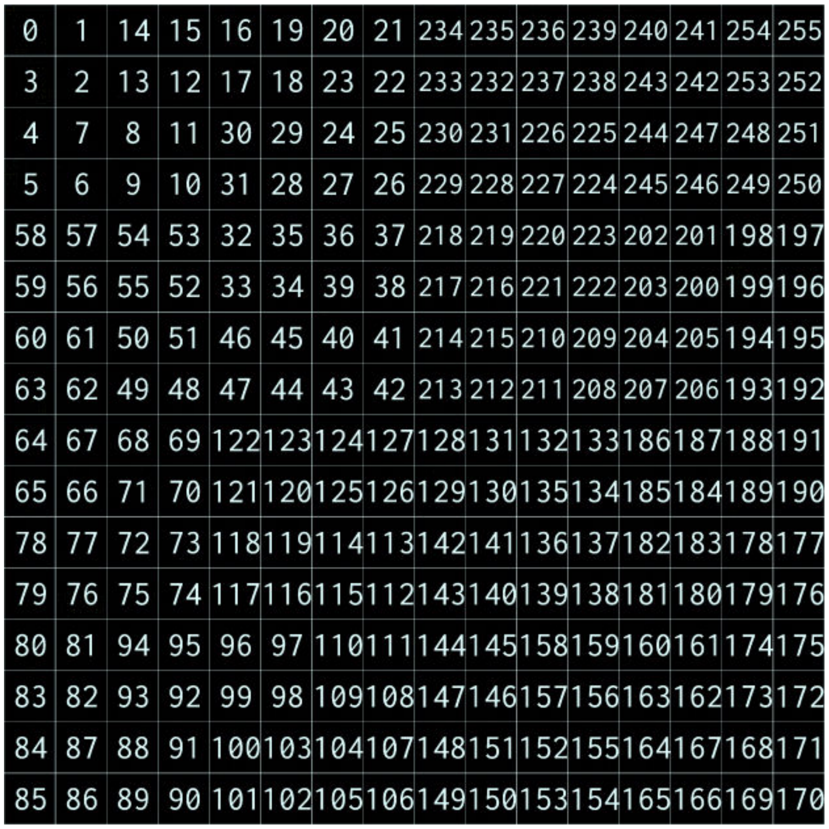
- `-c 0xE0FFFF` specifies the color to be used for those annotations

We recognize that it can be hard to see the 4K individual pixels that comprise that map. For that reason, we’re also supplying “enlarged” (per-`/8`) Hilbert maps for all `/8`’s. We made these with a script that looks like:

```
i=0
length=255
for (( i = 0; i <= $length; i++))
do
    cat "$i".summary | ipv4-heatmap -y "$i".0.0.0/8 -A 0 \
    -B 40000000000 -c 0xE0FFFF
    mv map.png map-8-"$i".png
done
```

Just like the overall summary maps, these (per-`/8`) Hilbert maps are laid out in the following pattern:

[17] https://en.wikipedia.org/wiki/Hilbert_curve
[18] https://xkcd.com/195/
[19] https://github.com/hrbrmstr/ipv4-heatmap
[20] https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

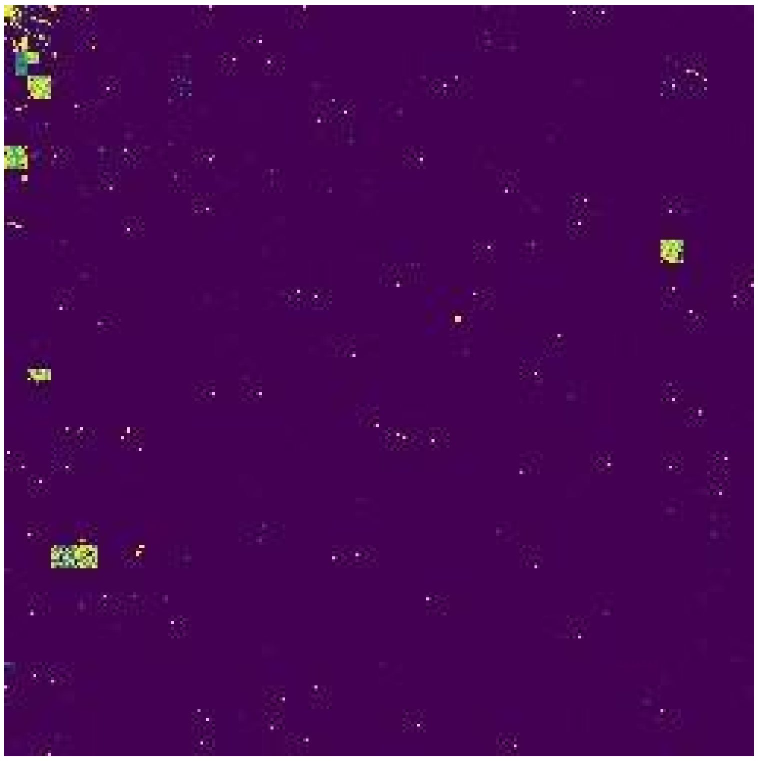


0	1	14	15	16	19	20	21	234	235	236	239	240	241	254	255
3	2	13	12	17	18	23	22	233	232	237	238	243	242	253	252
4	7	8	11	30	29	24	25	230	231	226	225	244	247	248	251
5	6	9	10	31	28	27	26	229	228	227	224	245	246	249	250
58	57	54	53	32	35	36	37	218	219	220	223	202	201	198	197
59	56	55	52	33	34	39	38	217	216	221	222	203	200	199	196
60	61	50	51	46	45	40	41	214	215	210	209	204	205	194	195
63	62	49	48	47	44	43	42	213	212	211	208	207	206	193	192
64	67	68	69	122	123	124	127	128	131	132	133	186	187	188	191
65	66	71	70	121	120	125	126	129	130	135	134	185	184	189	190
78	77	72	73	118	119	114	113	142	141	136	137	182	183	178	177
79	76	75	74	117	116	115	112	143	140	139	138	181	180	179	176
80	81	94	95	96	97	110	111	144	145	158	159	160	161	174	175
83	82	93	92	99	98	109	108	147	146	157	156	163	162	173	172
84	87	88	91	100	103	104	107	148	151	152	155	164	167	168	171
85	86	89	90	101	102	105	106	149	150	153	154	165	166	169	170

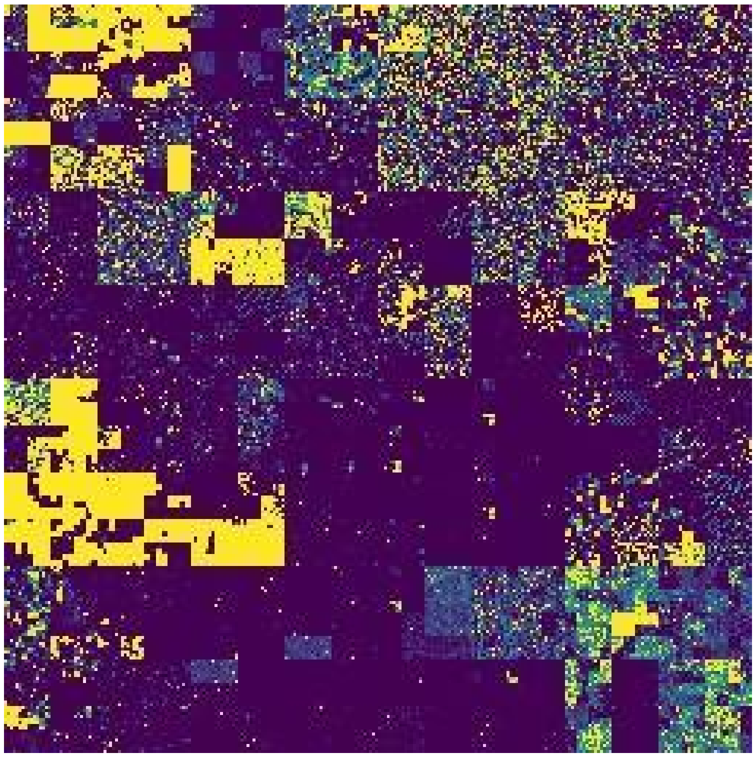
To avoid “stepping on” any pixels, we will **NOT** show those interior cell borders (nor the per-cell interior numbering) on any of the individual `per-/8` graphs.

Let’s now look at those `per-/8` graphs. Note that graphs known to have “special” status of whatever sort will be annotated as such in the title identifying that graph.

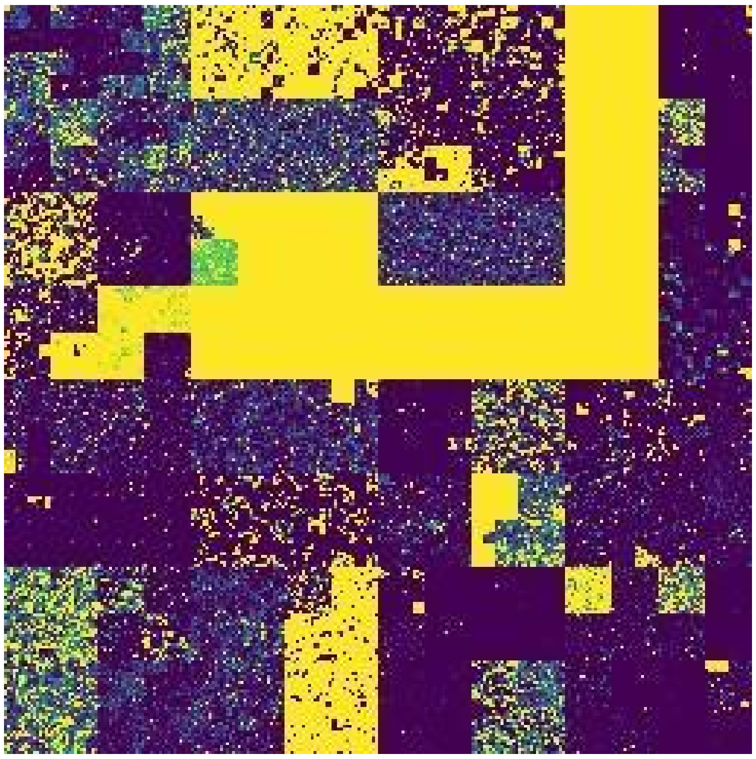
0.0.0.0/8: “This Network” Address Space



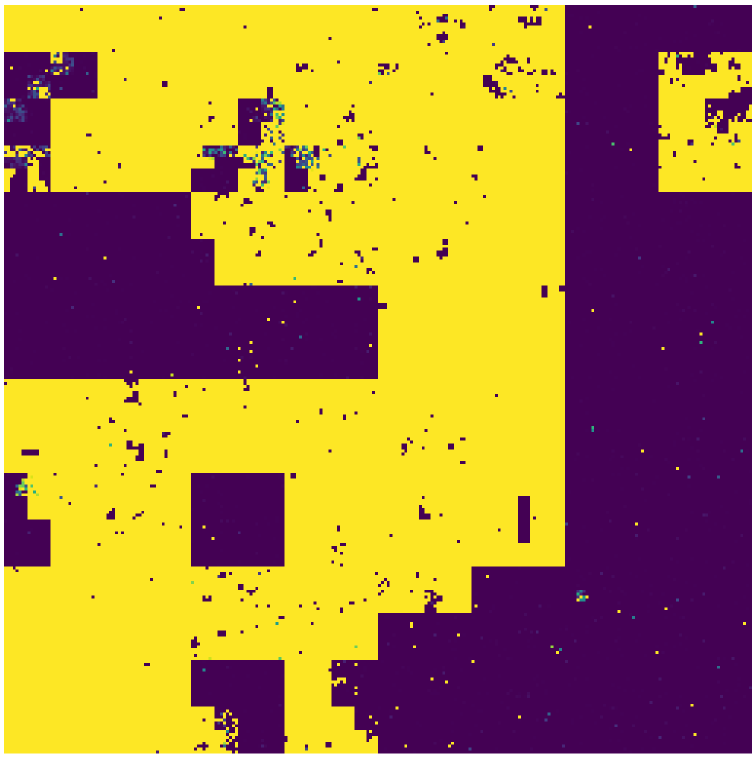
1.0.0.0/8:



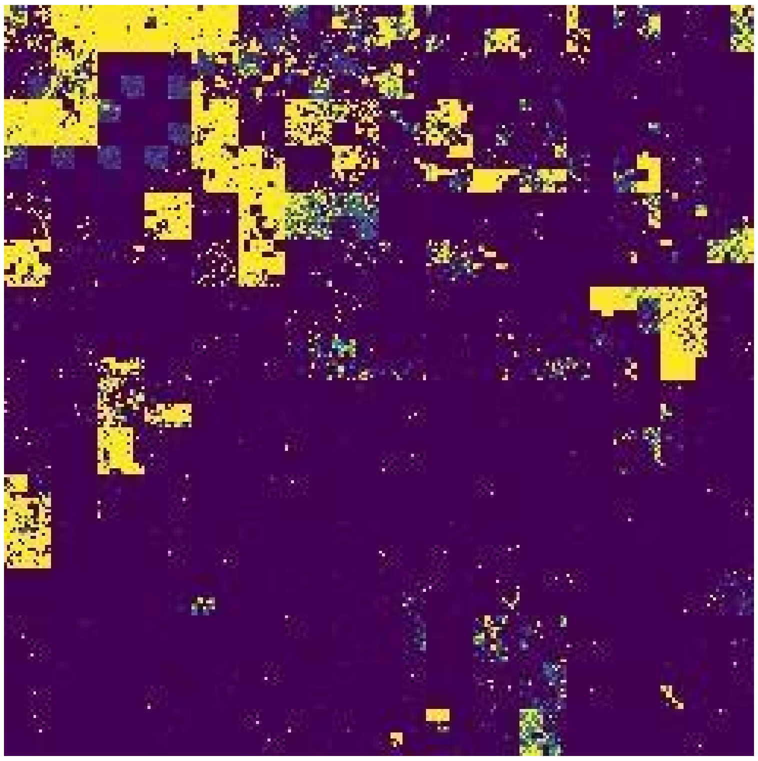
2.0.0.0/8:



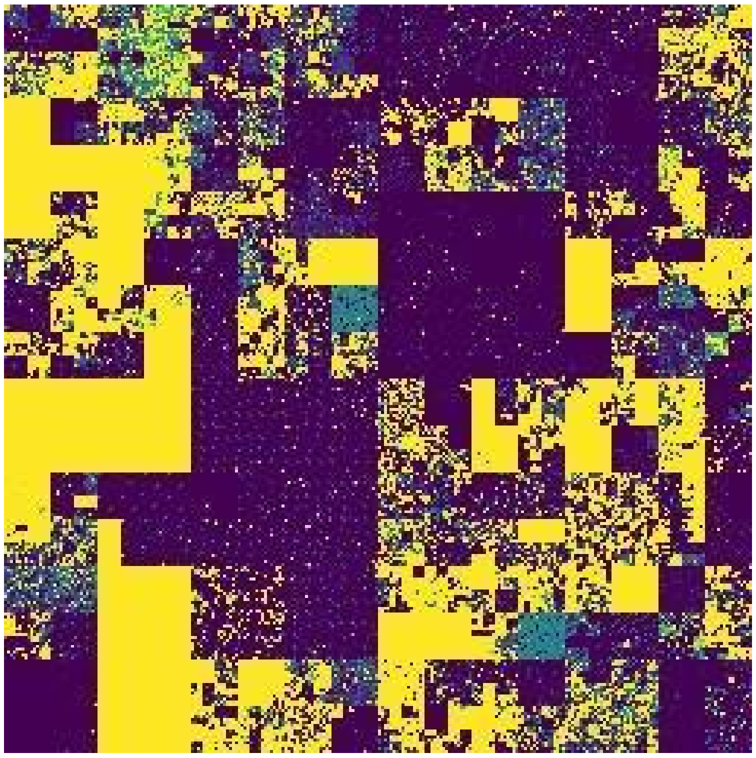
3.0.0.0/8:



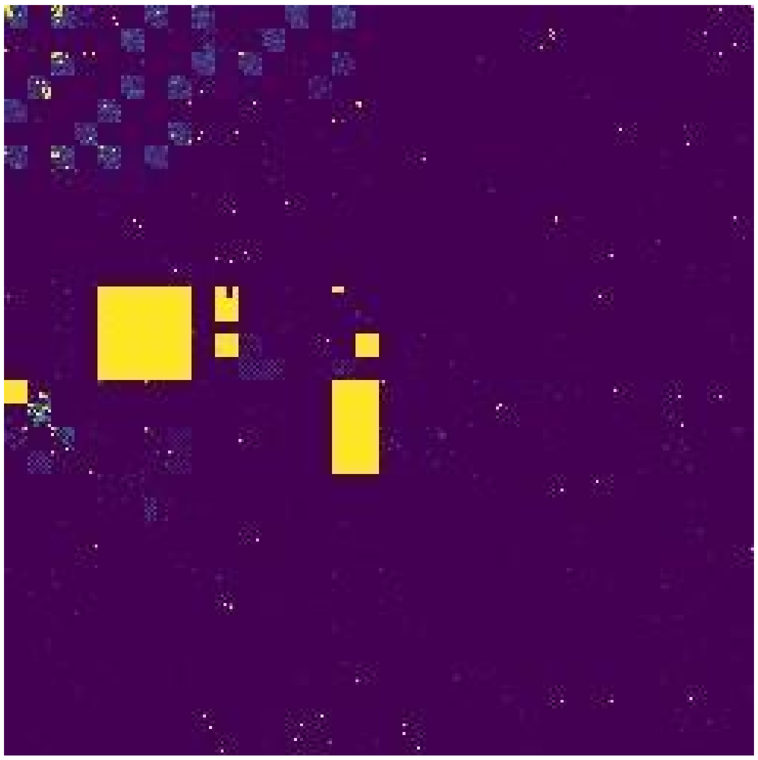
4.0.0.0/8: Formerly BBN



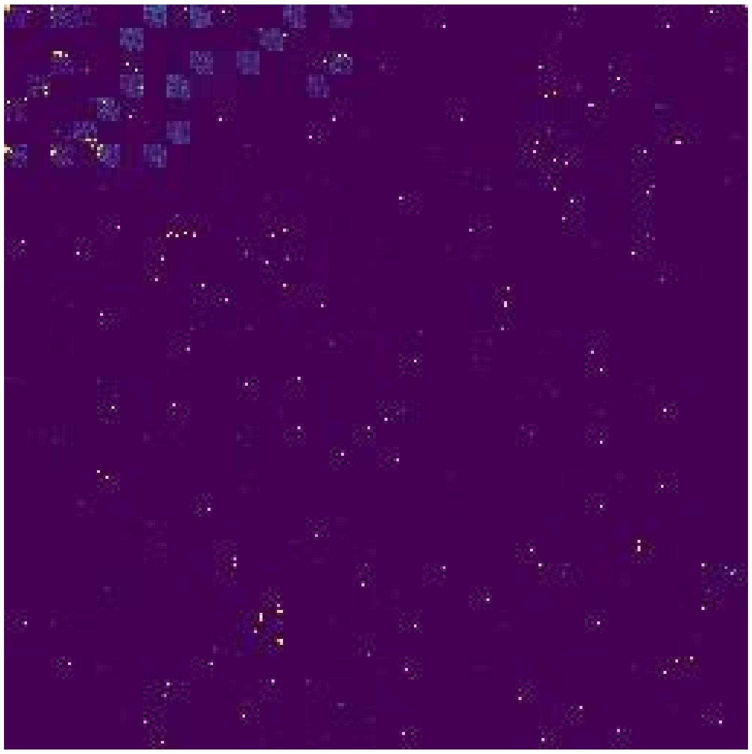
5.0.0.0/8:



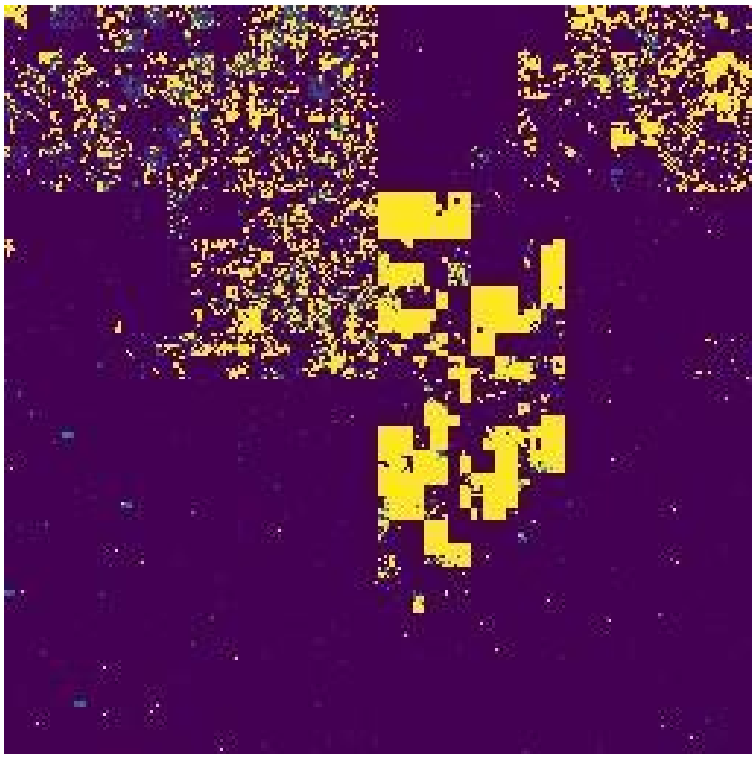
6.0.0.0/8: Army Information Center (02/1994)



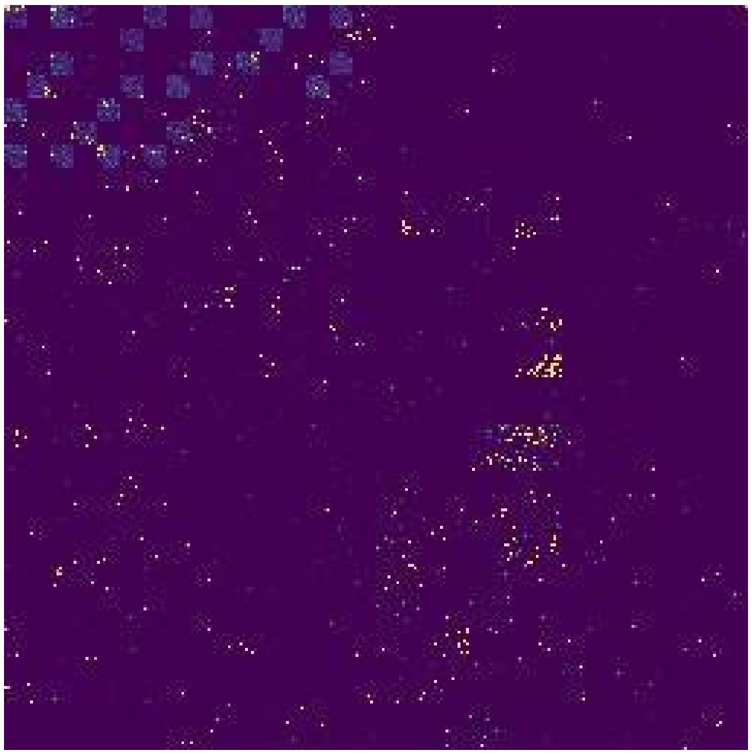
7.0.0.0/8: DOD Network Information Center (04/1995)



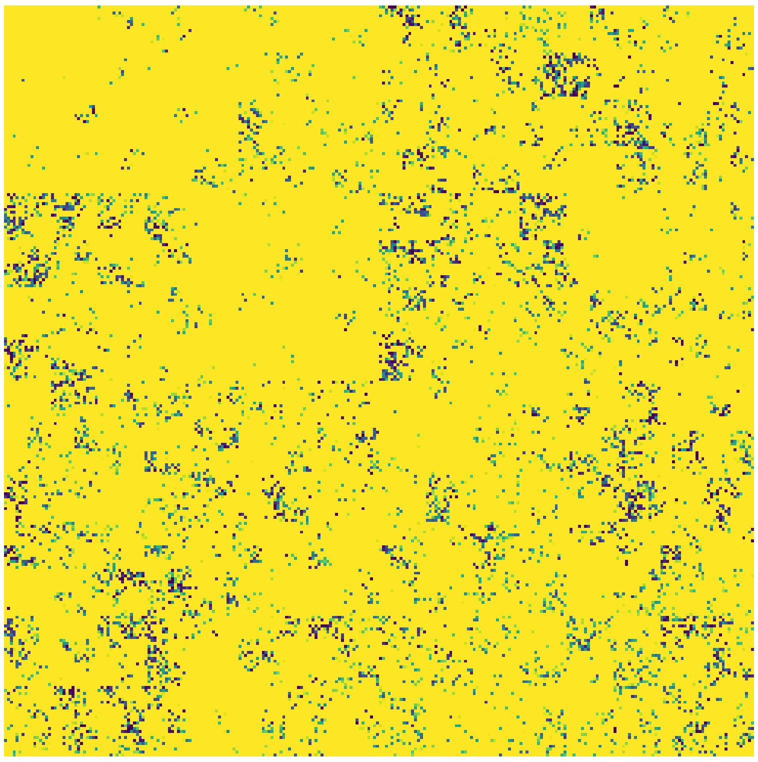
8.0.0.0/8:



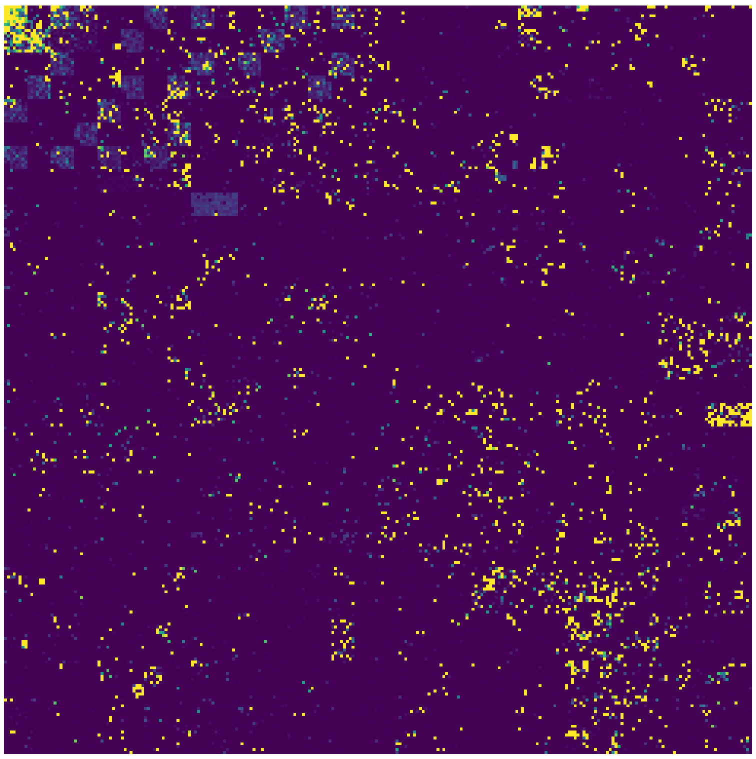
9.0.0.0/8: IBM (except for 9.9.9.0/24) (08/1992)



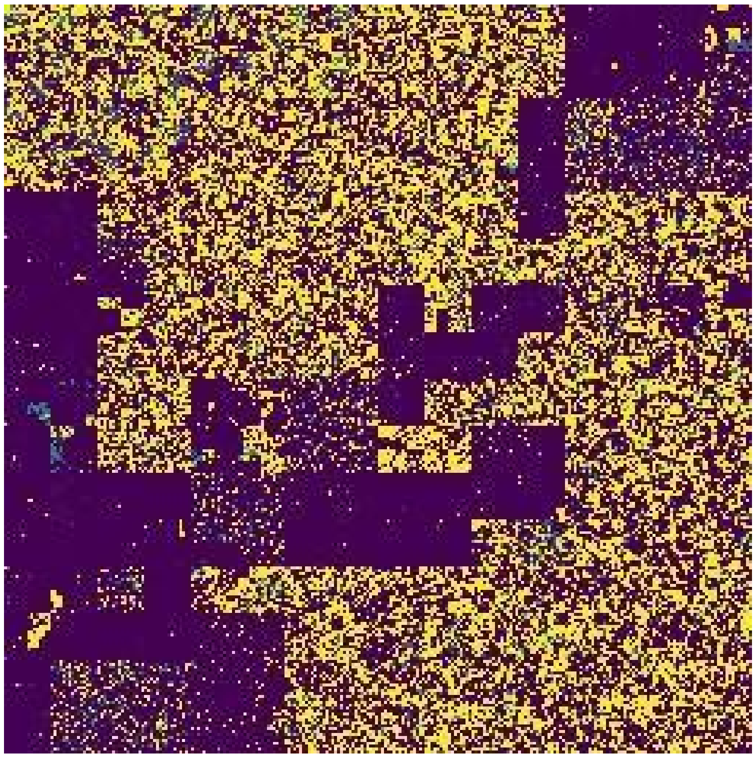
10.0.0.0/8: RFC1918 Private Address Space



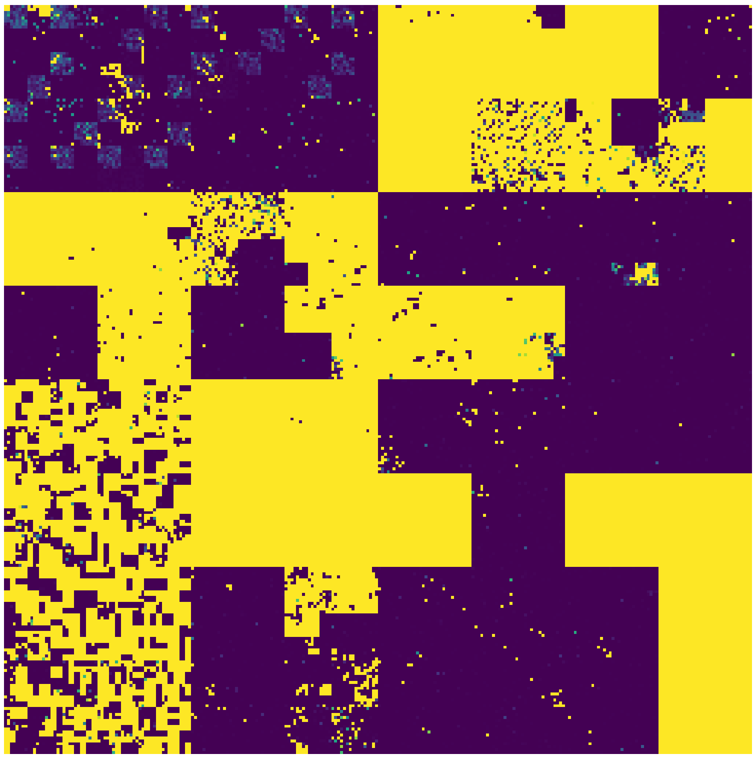
11.0.0.0/8



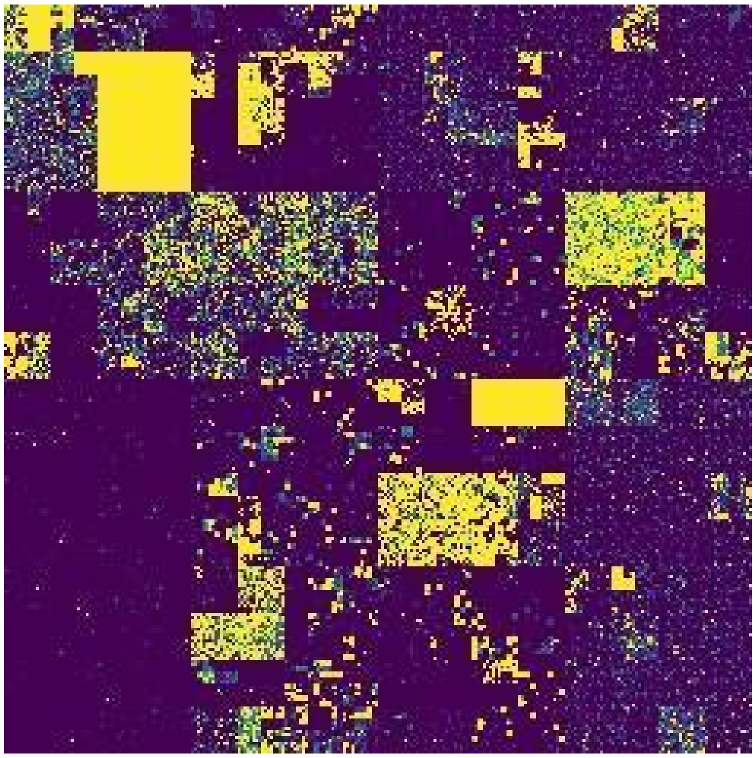
12.0.0.0/8:



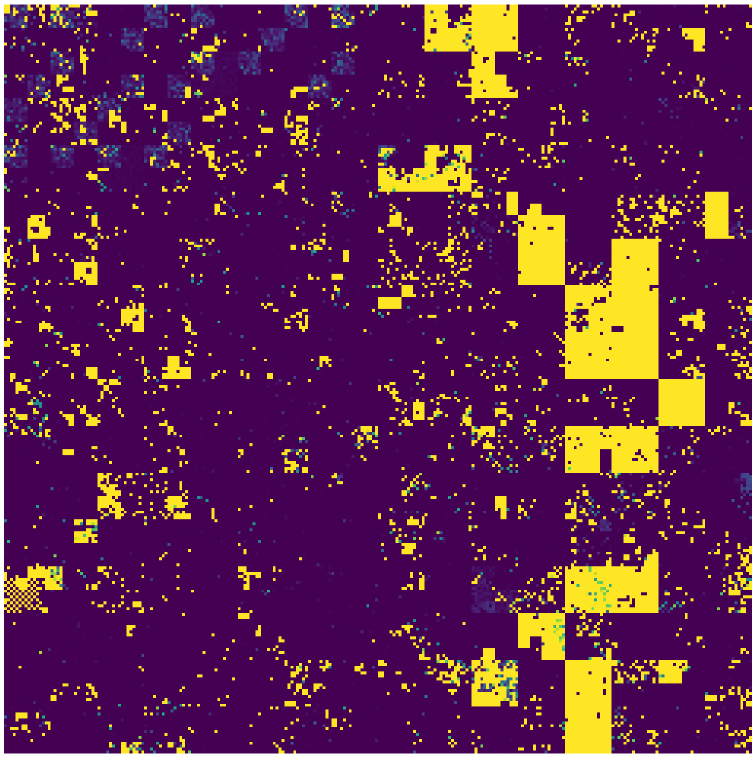
13.0.0.0/8



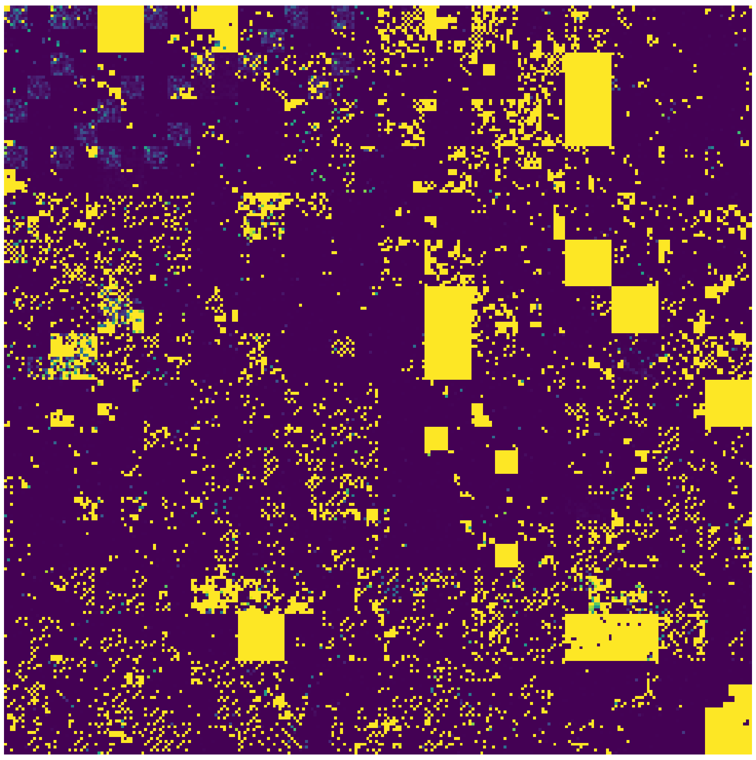
14.0.0.0/8:



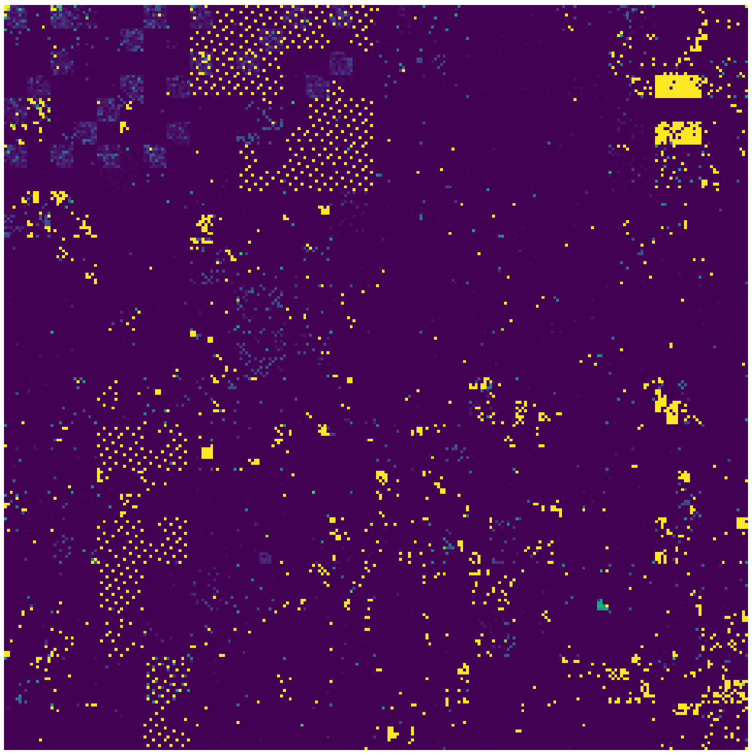
15.0.0.0/8



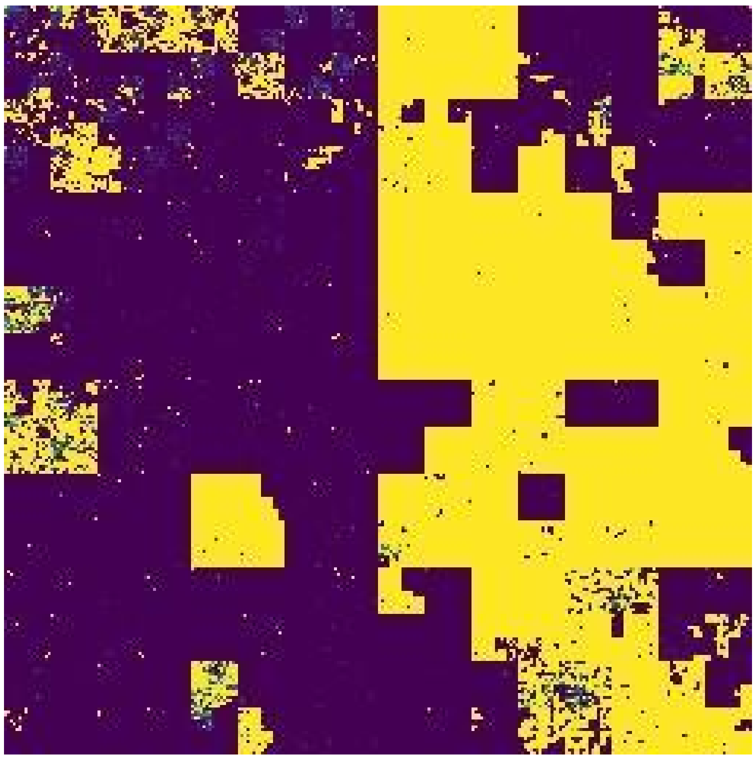
16.0.0.0/8:



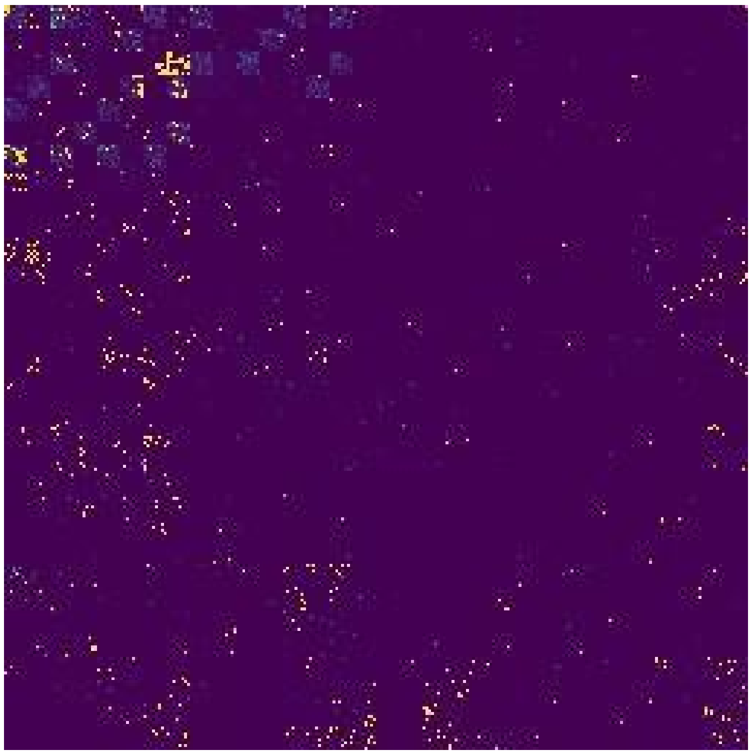
17.0.0.0/8: Apple (07/1992)



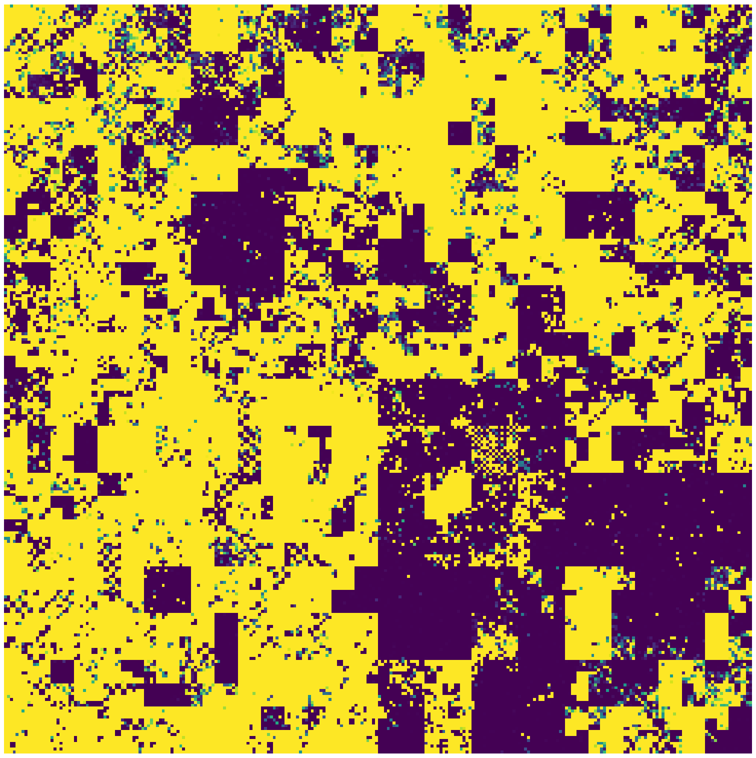
18.0.0.0/8:



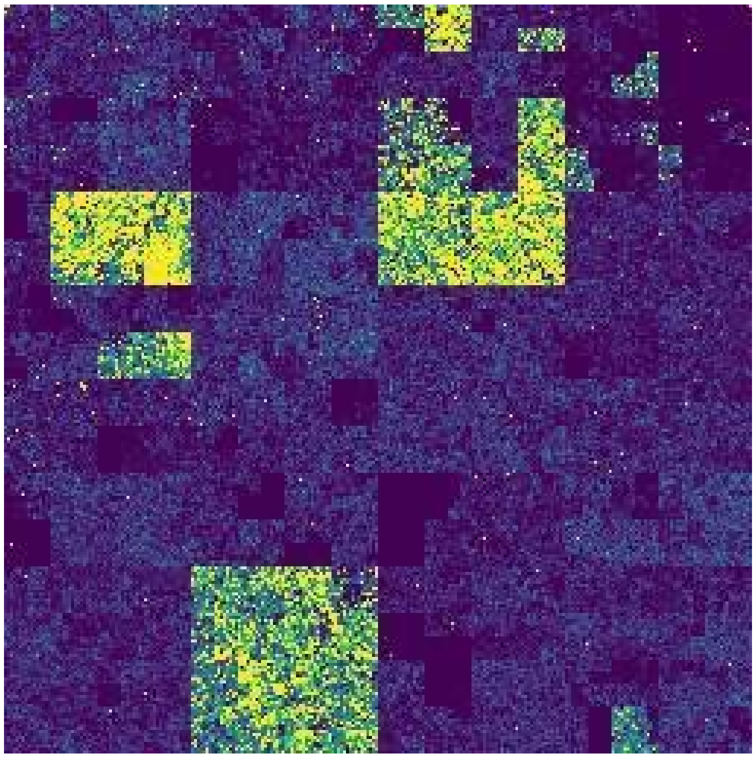
19.0.0.0/8: Ford (05/1995)



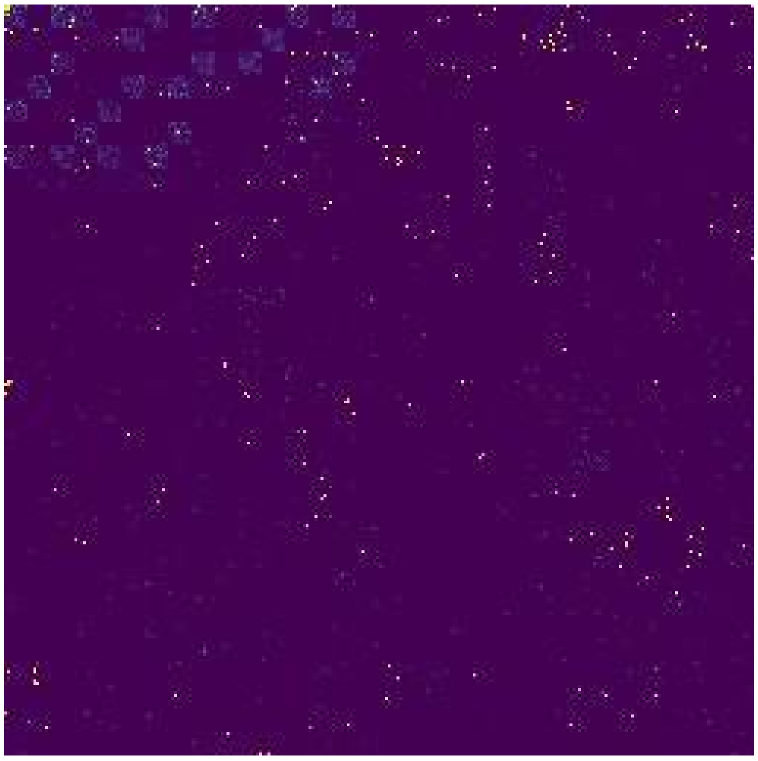
20.0.0.0/8:



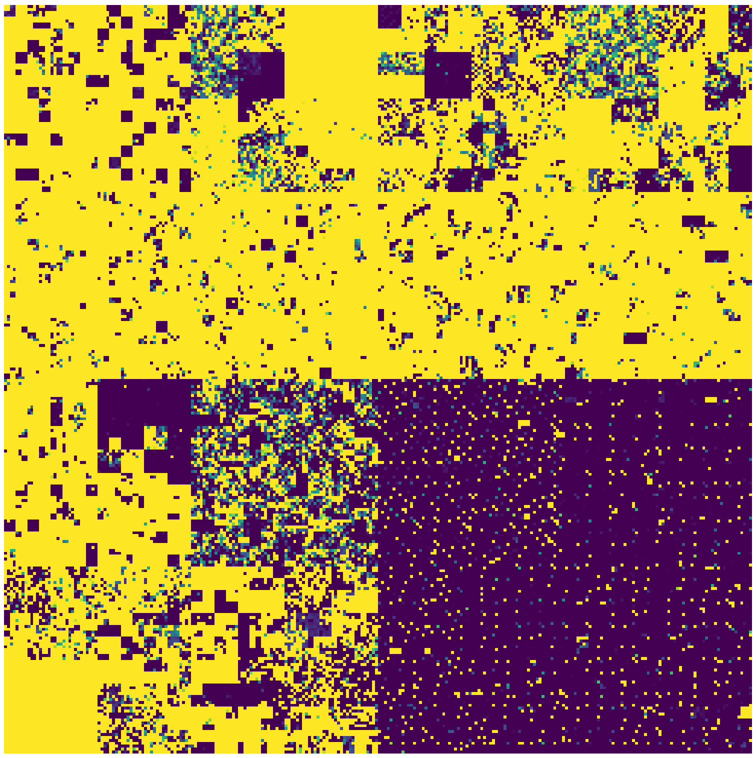
21.0.0.0/8:



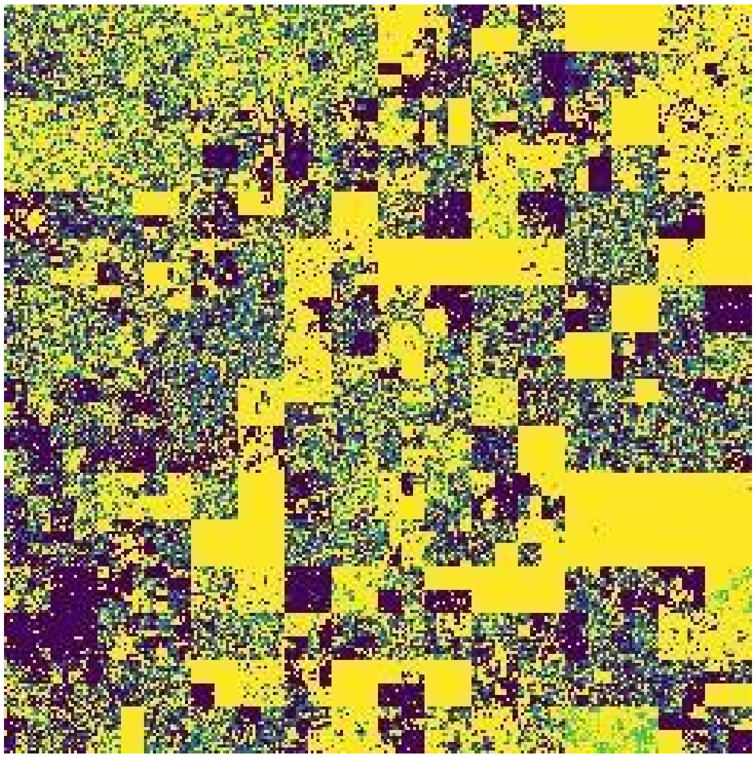
22.0.0.0/8:DISA (05/1993)



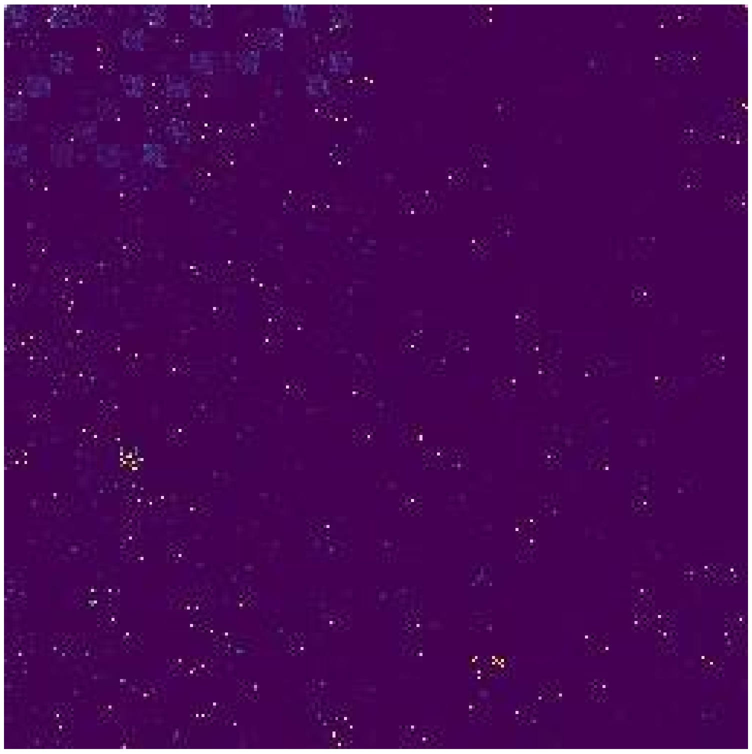
23.0.0.0/8:



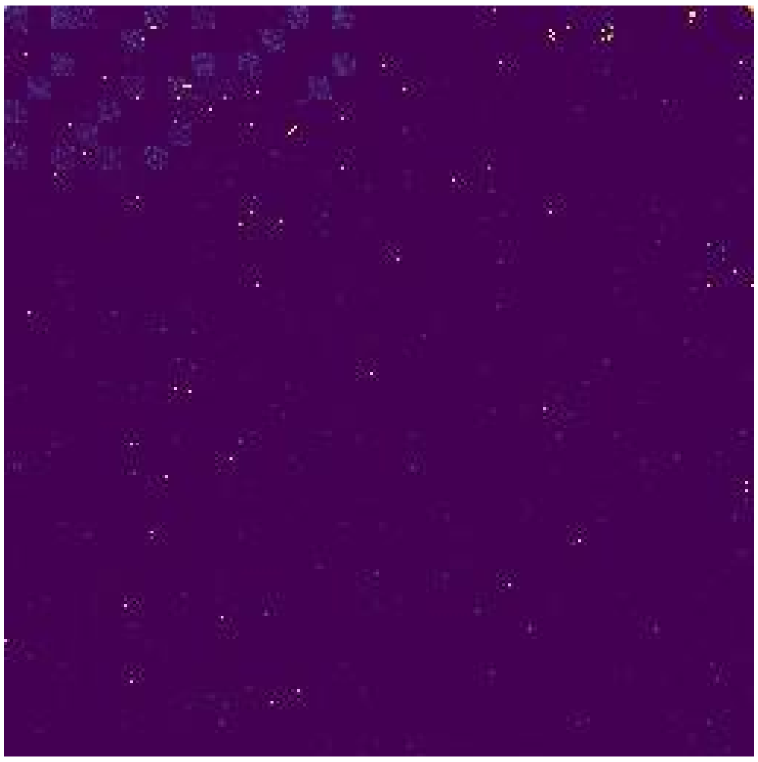
24.0.0.0/8:



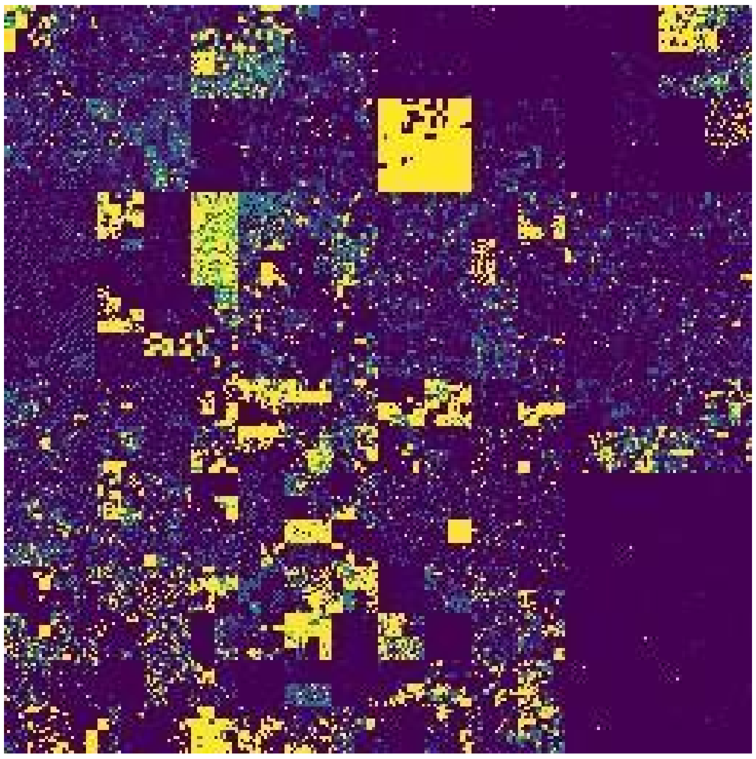
25.0.0.0/8: UK MOD (08/2005)



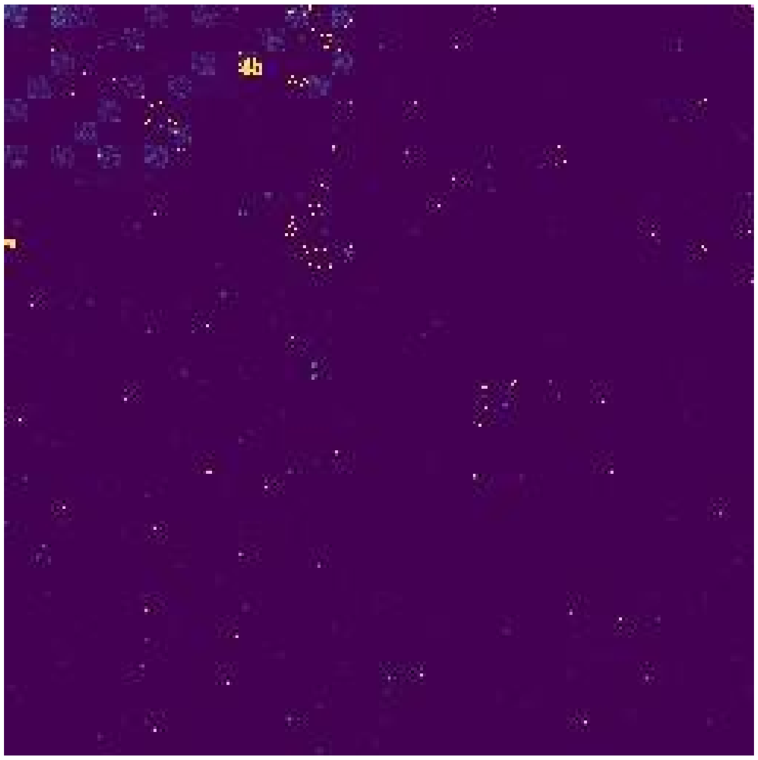
26.0.0.0/8:



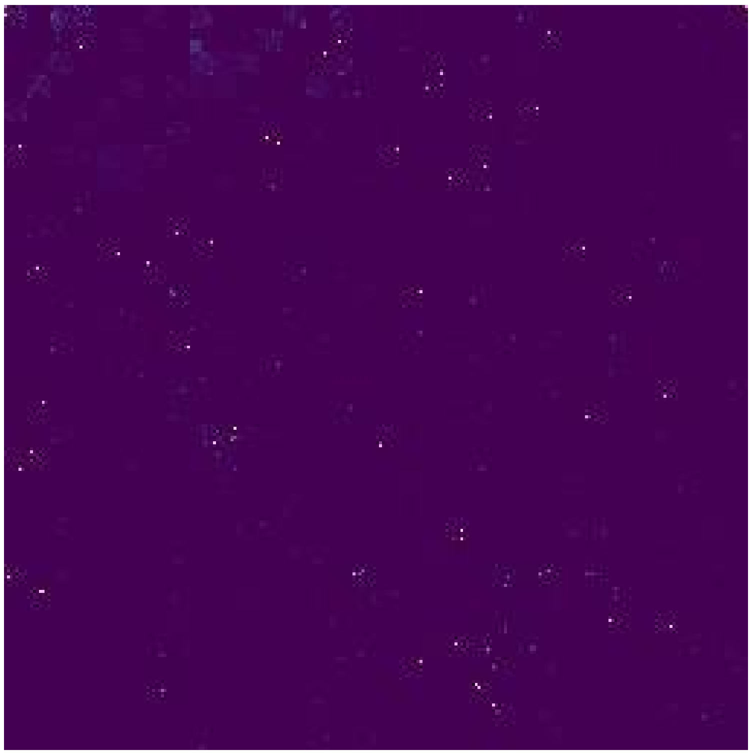
27.0.0.0/8:



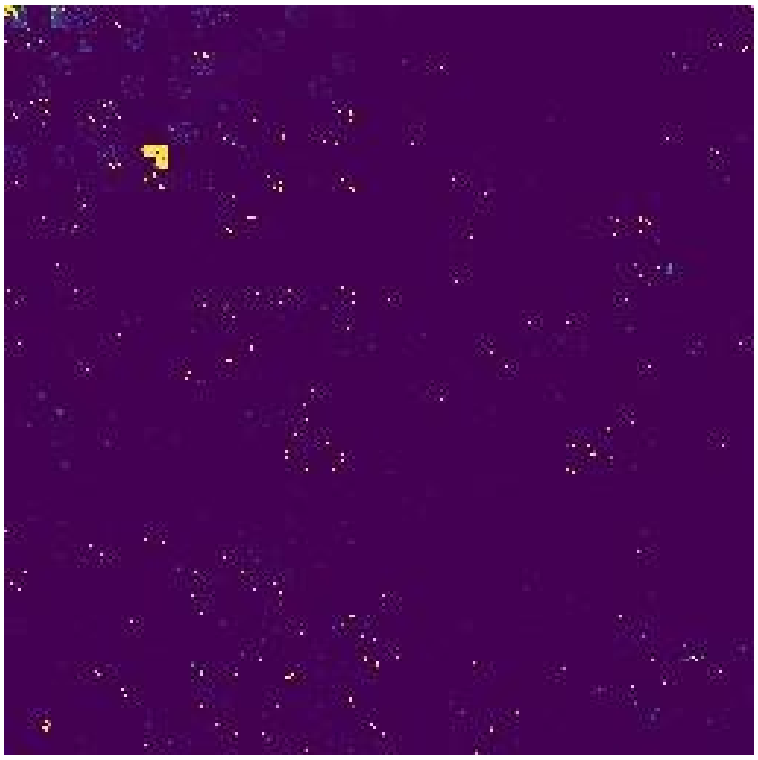
28.0.0.0/8: DSI-North (07/1992)



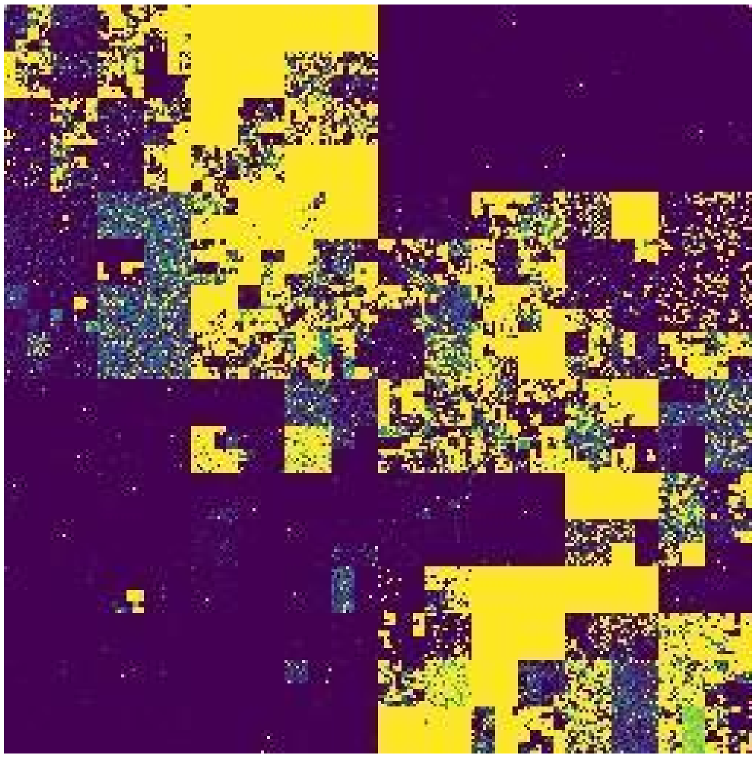
29.0.0.0/8: DISA (07/1991)



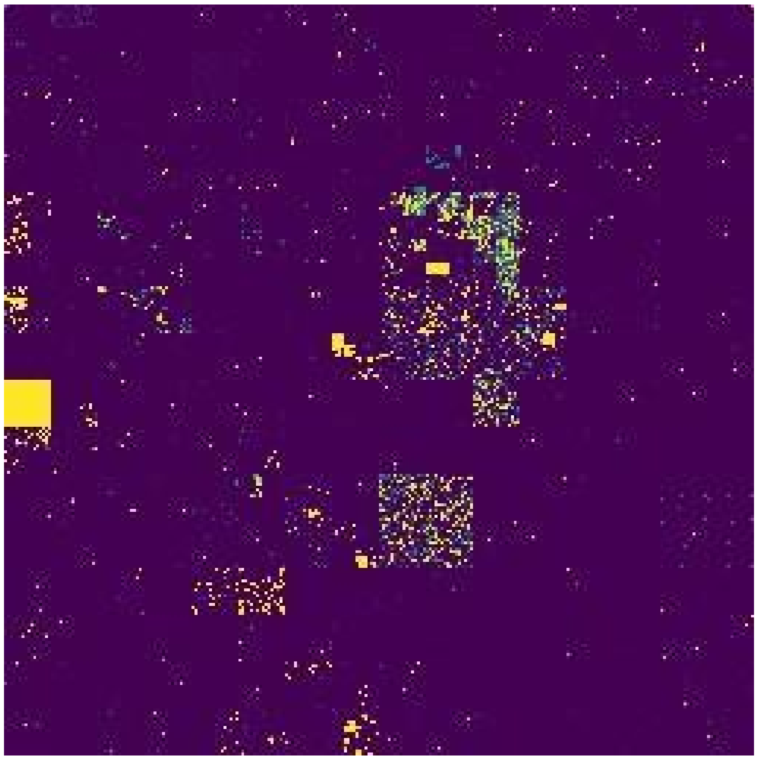
30.0.0.0/8: DISA (07/1991)



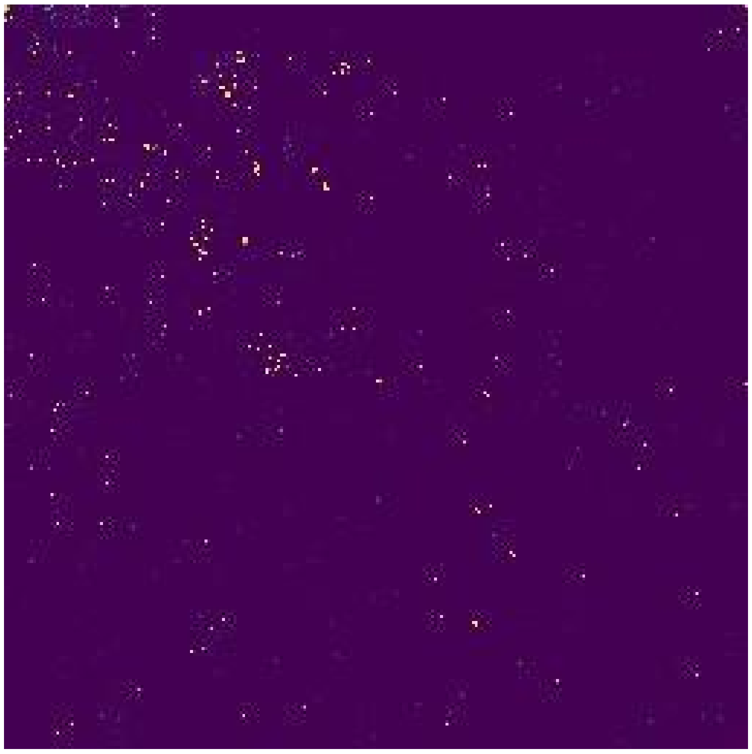
31.0.0.0/8:



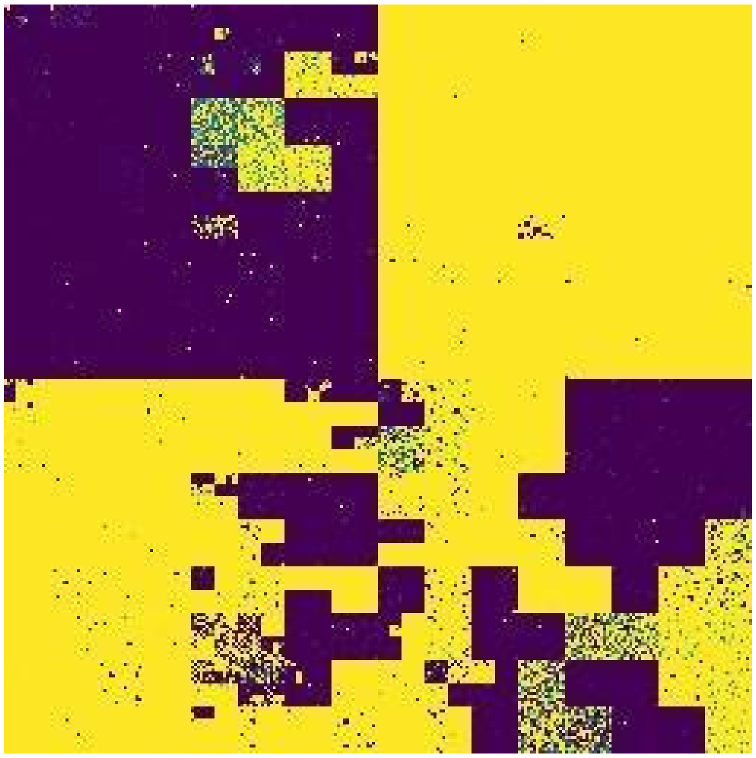
32.0.0.0/8: Formerly AT&T Global



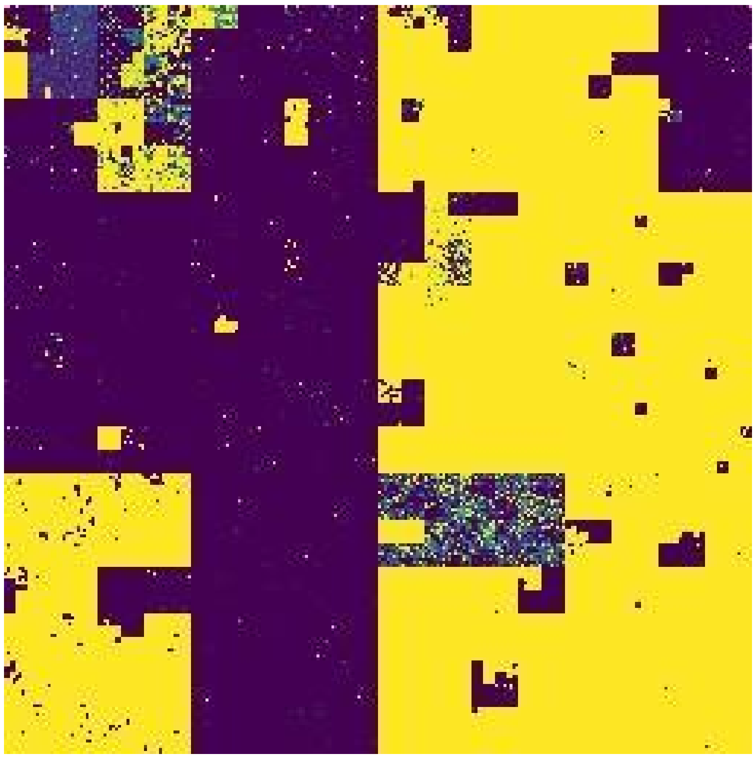
33.0.0.0/8: DLA System Automation Center (01/1991)



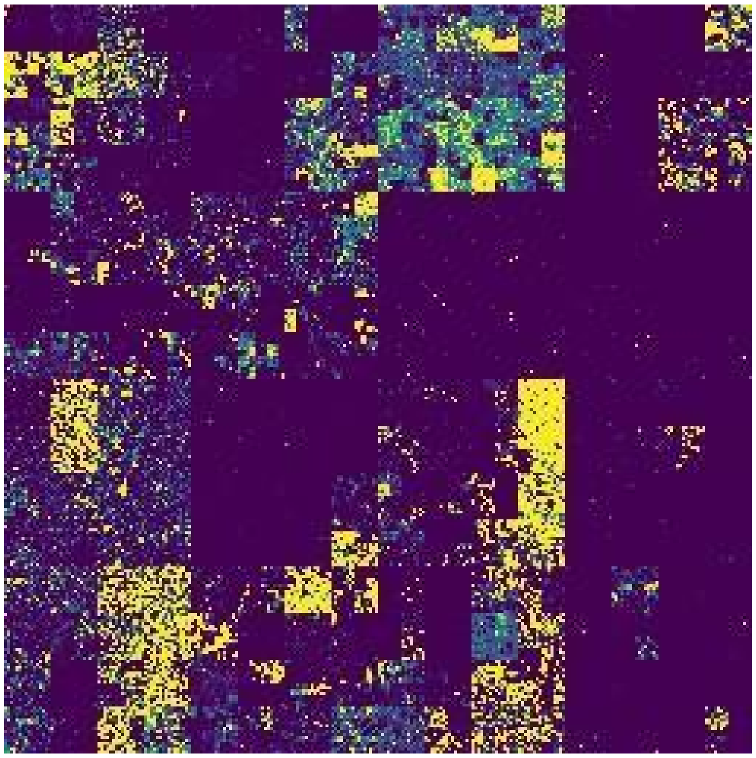
34.0.0.0/8:



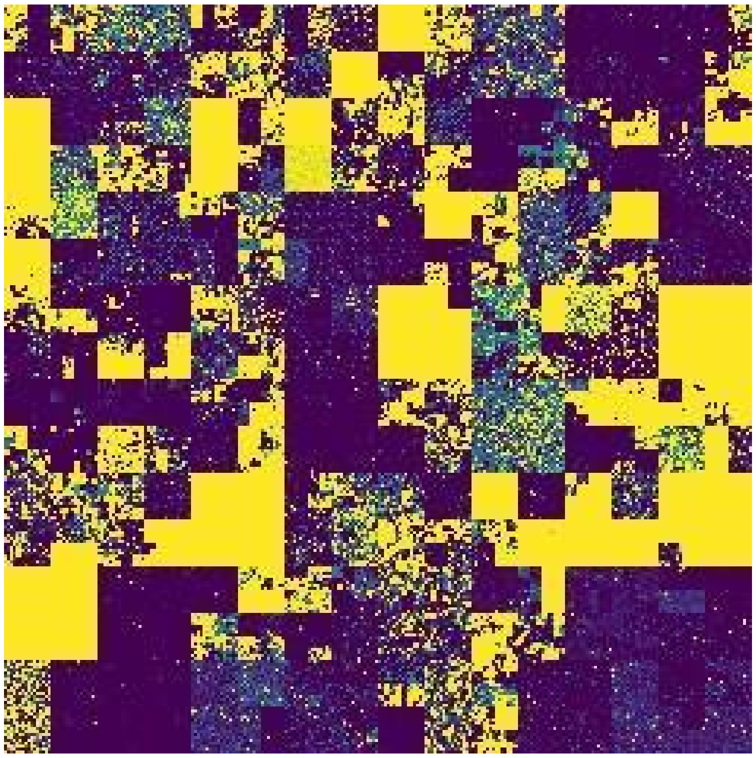
35.0.0.0:



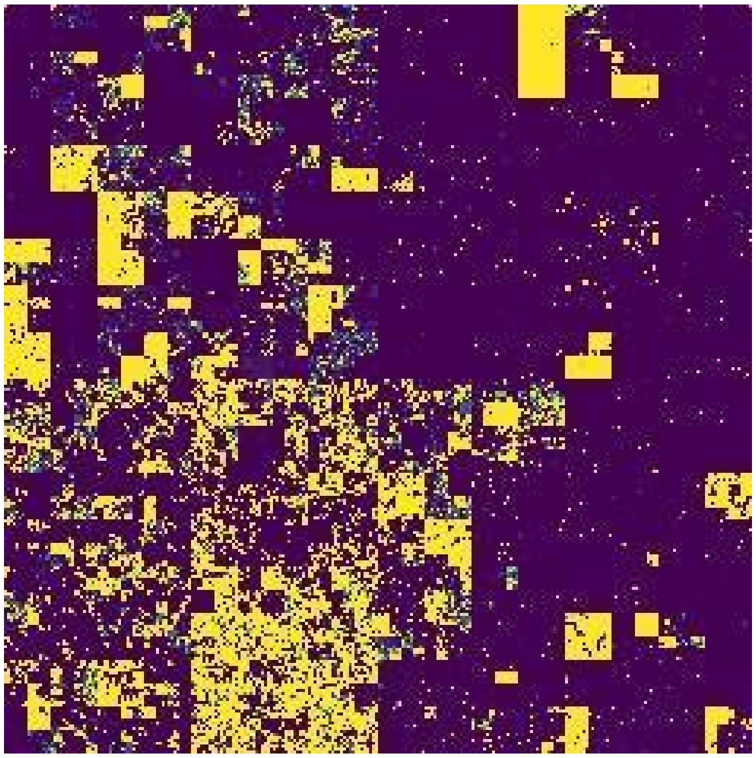
36.0.0.0/8:



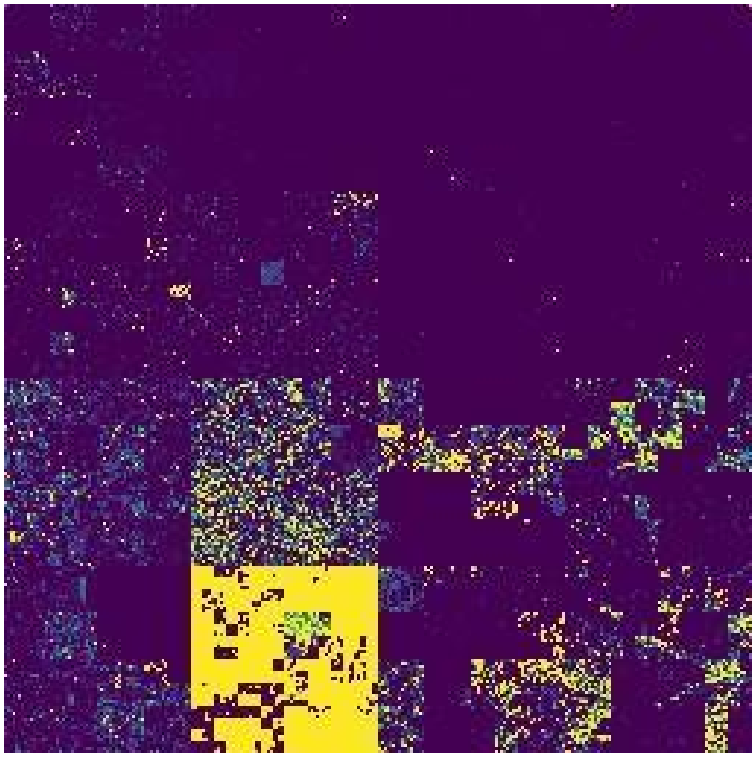
37.0.0.0/8:



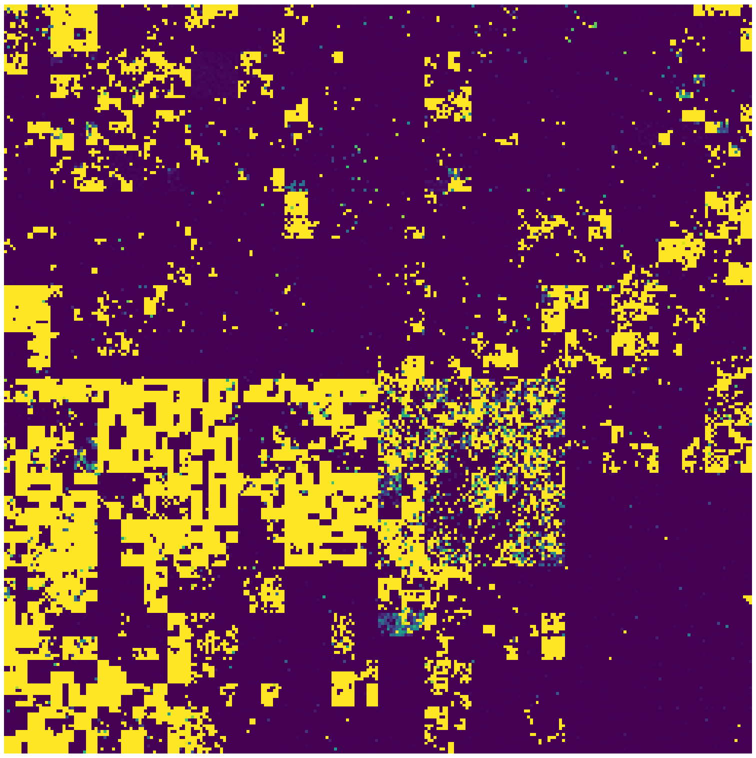
38.0.0.0/8:



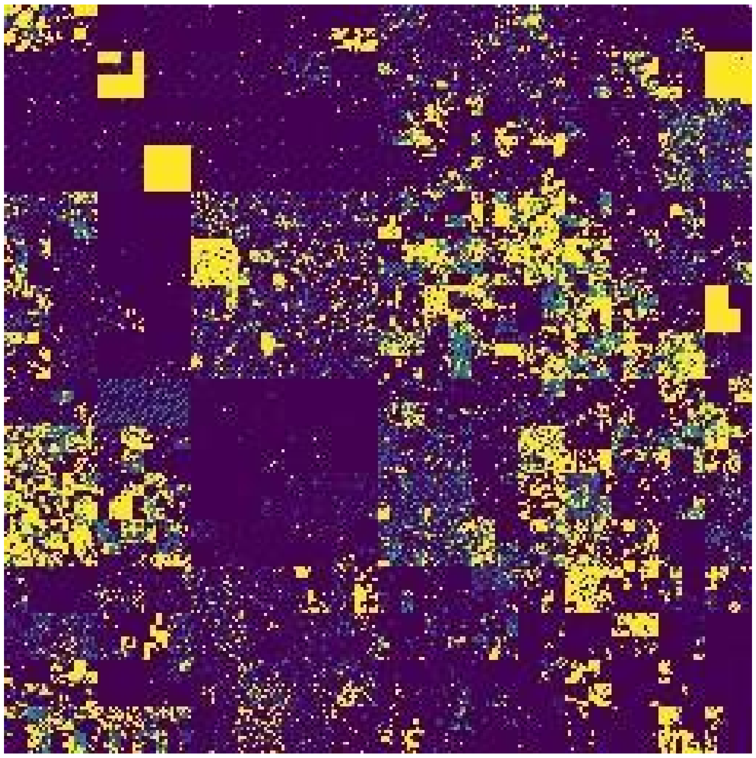
39.0.0.0/8:



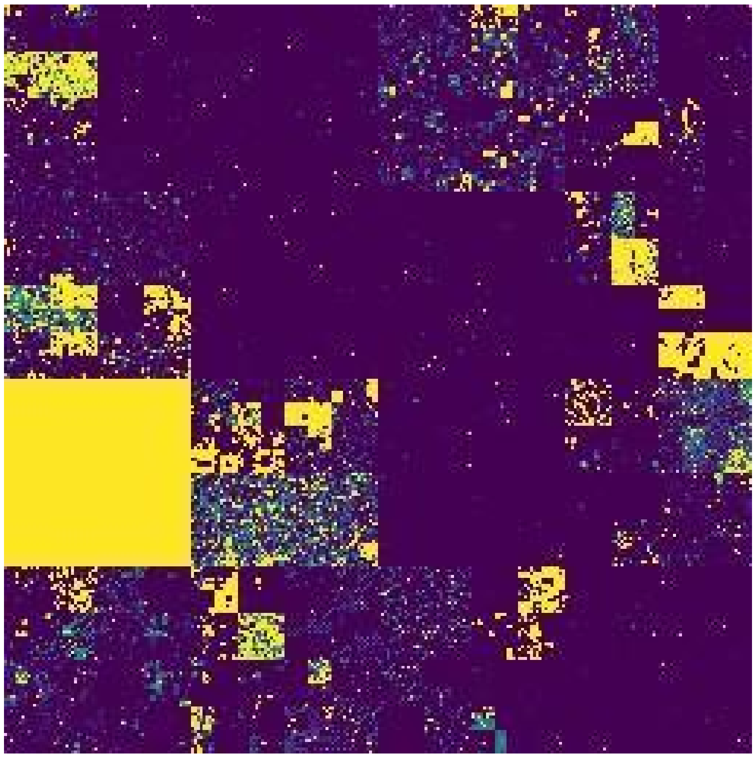
40.0.0.0/8:



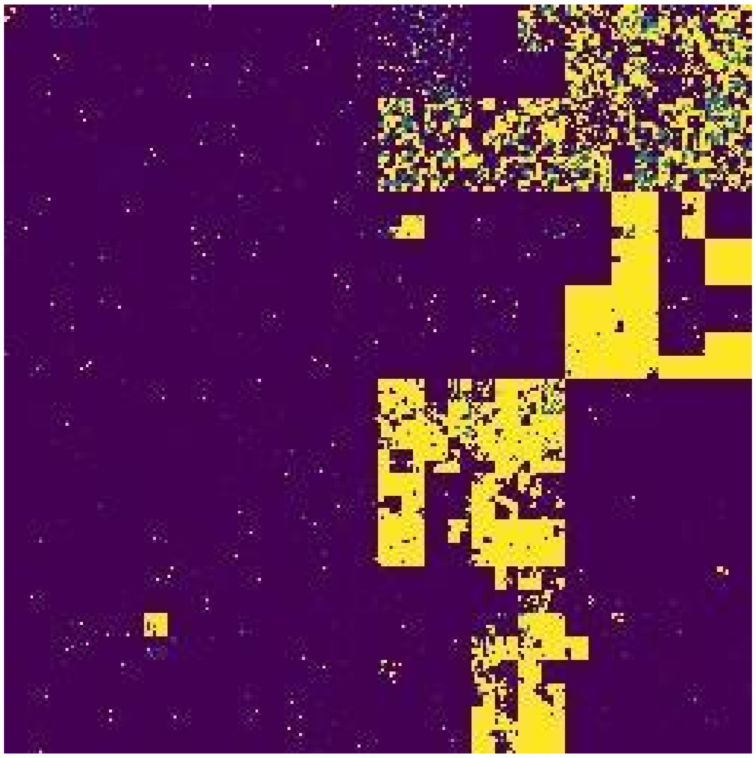
41.0.0.0/8:



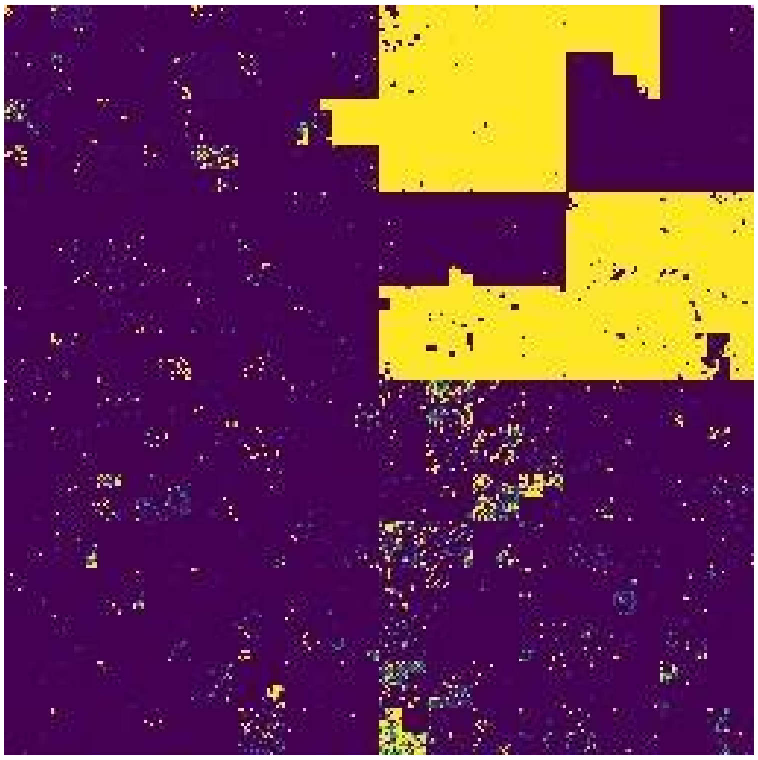
42.0.0.0/8:



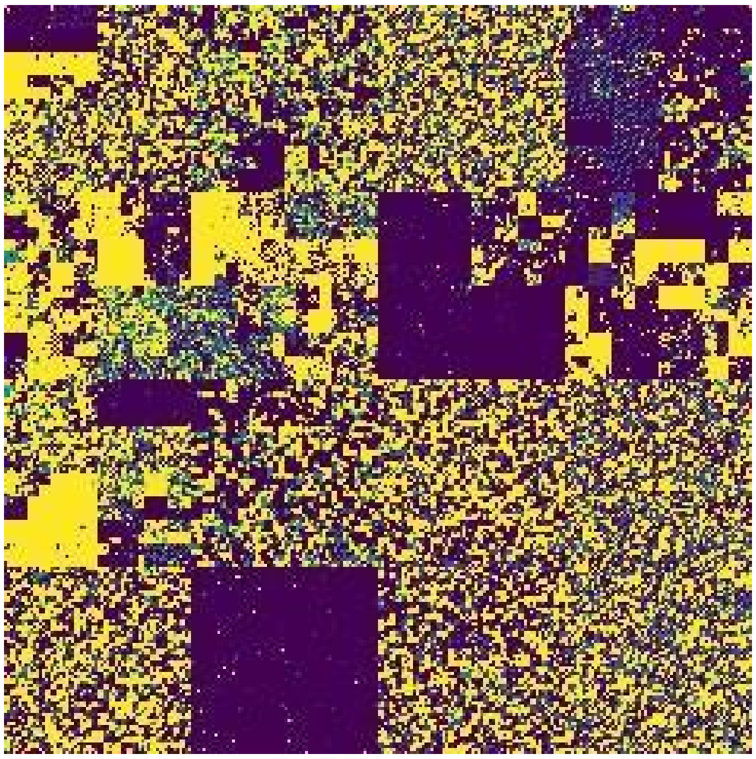
43.0.0.0/8:



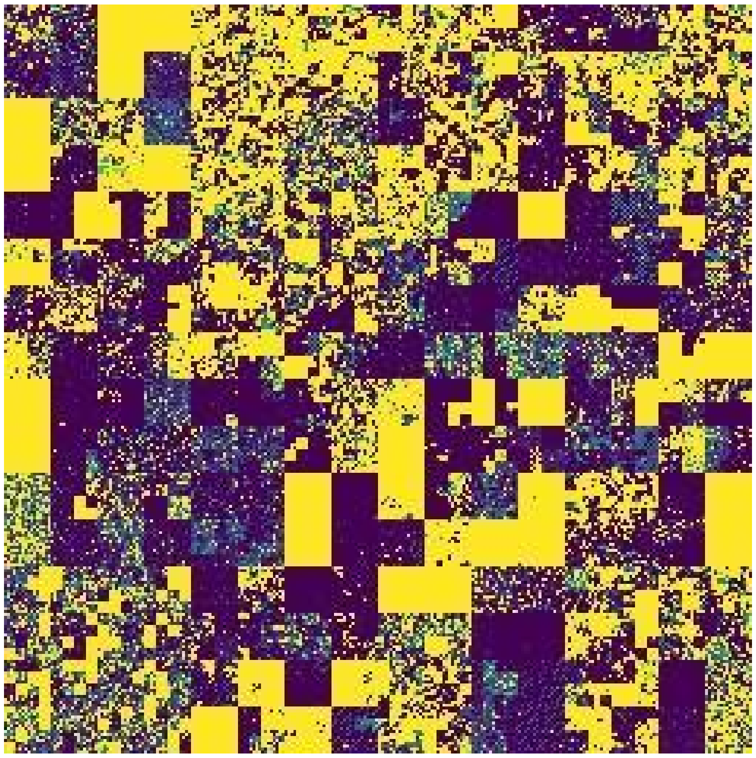
44.0.0.0/8: ARDC (except for 44.192.0.0/10 which was sold to Amazon)



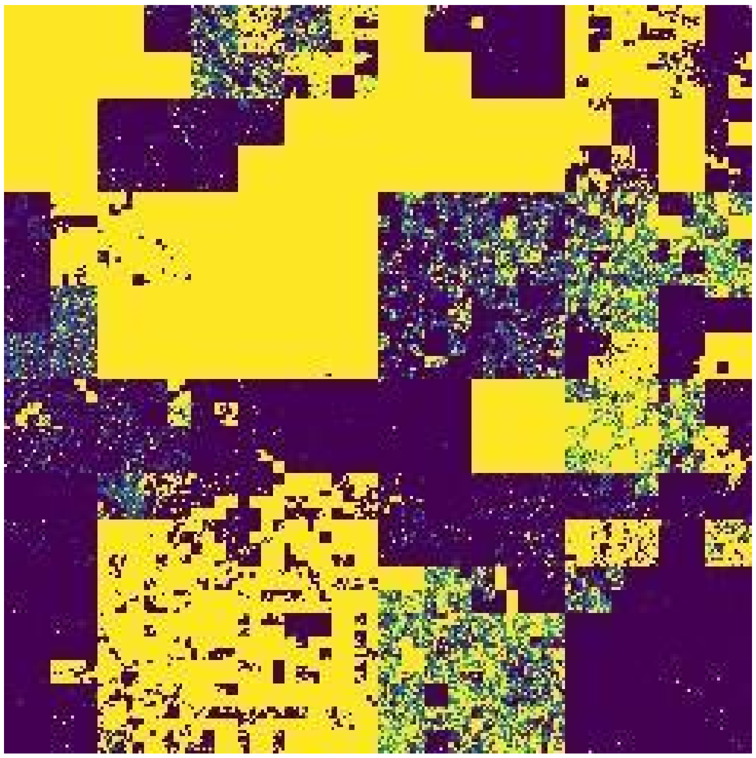
45.0.0.0/8:



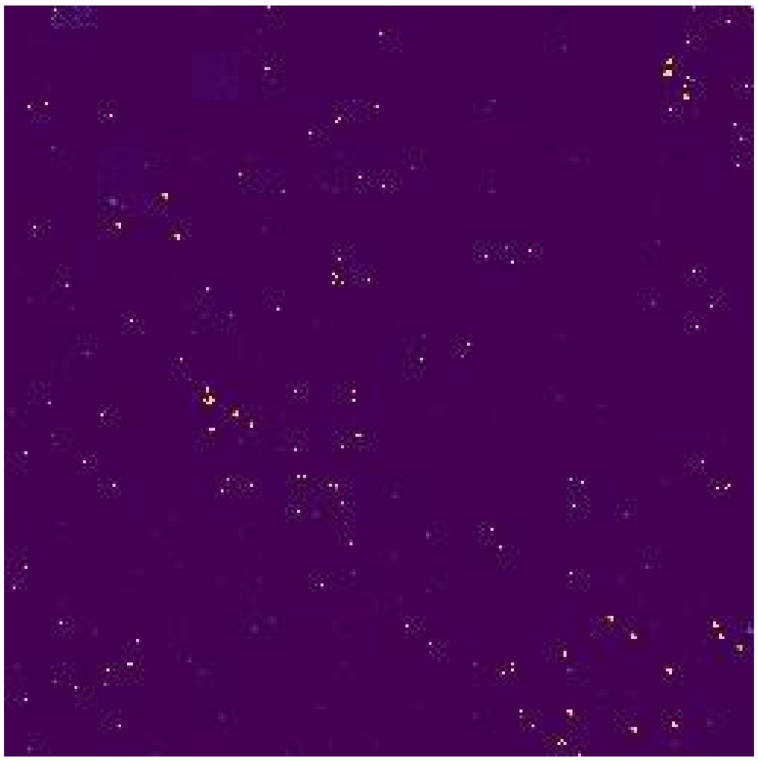
46.0.0.0/8:



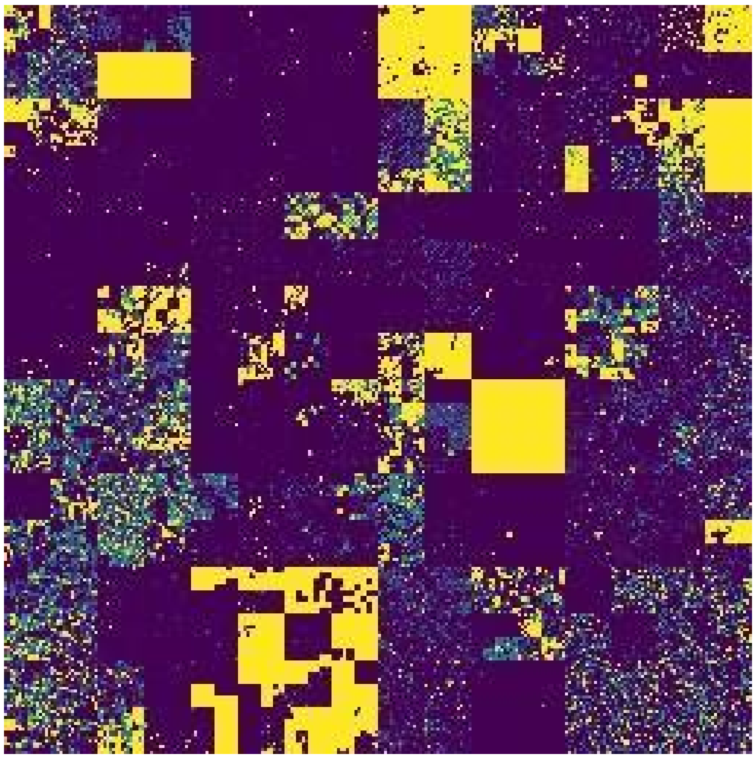
47.0.0.0/8:



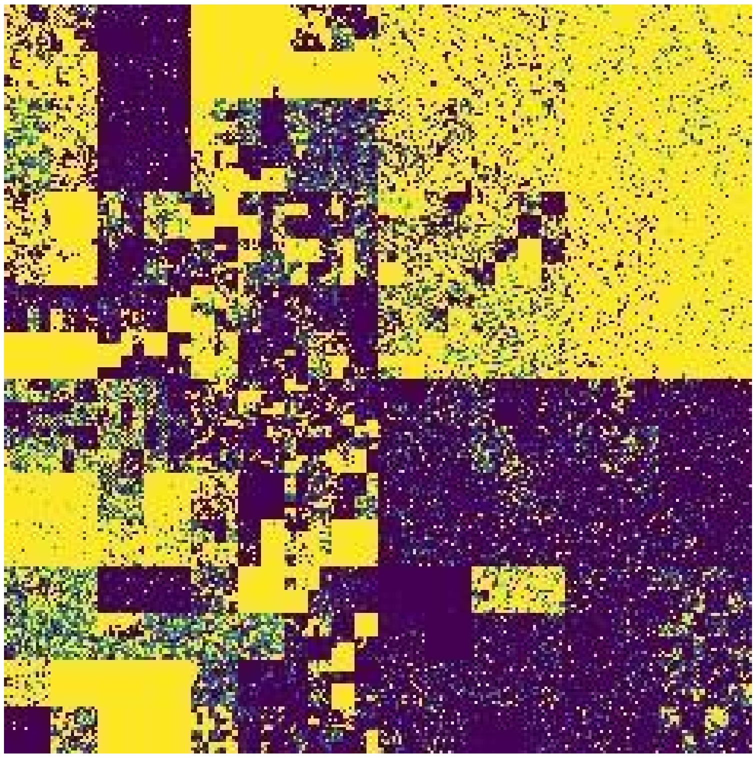
48.0.0.0/8: Prudential (05/1995):



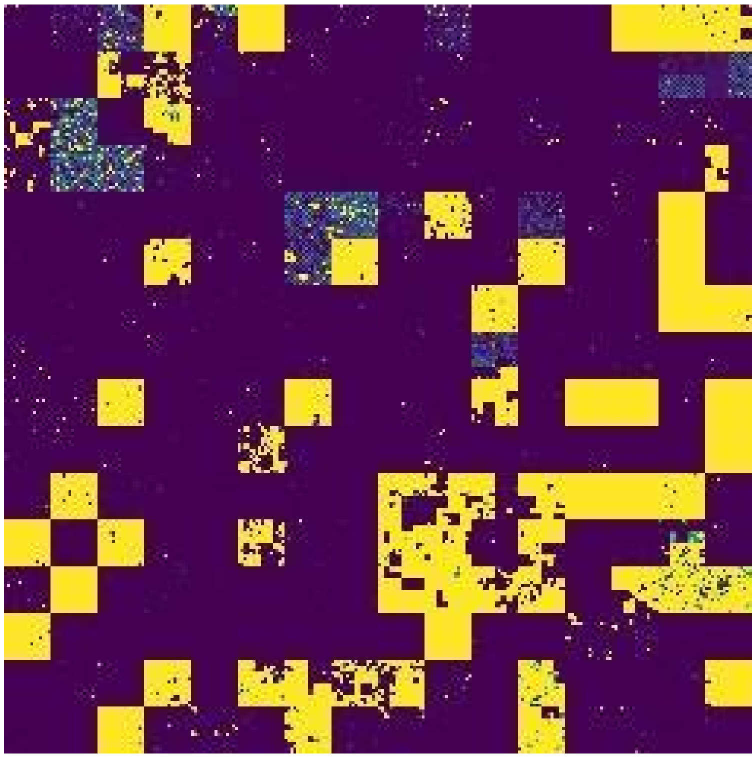
49.0.0.0/8:



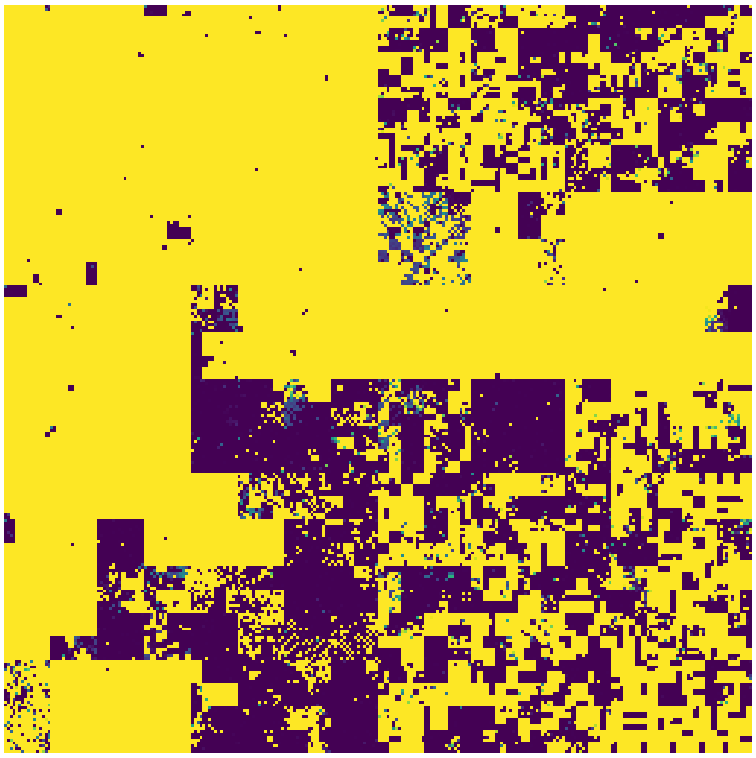
50.0.0.0/8:



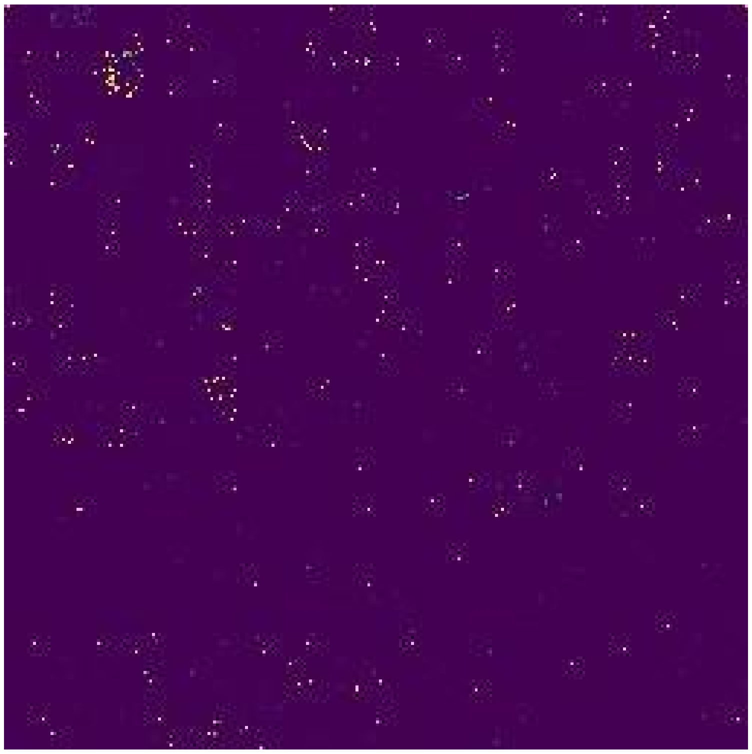
51.0.0.0/8:



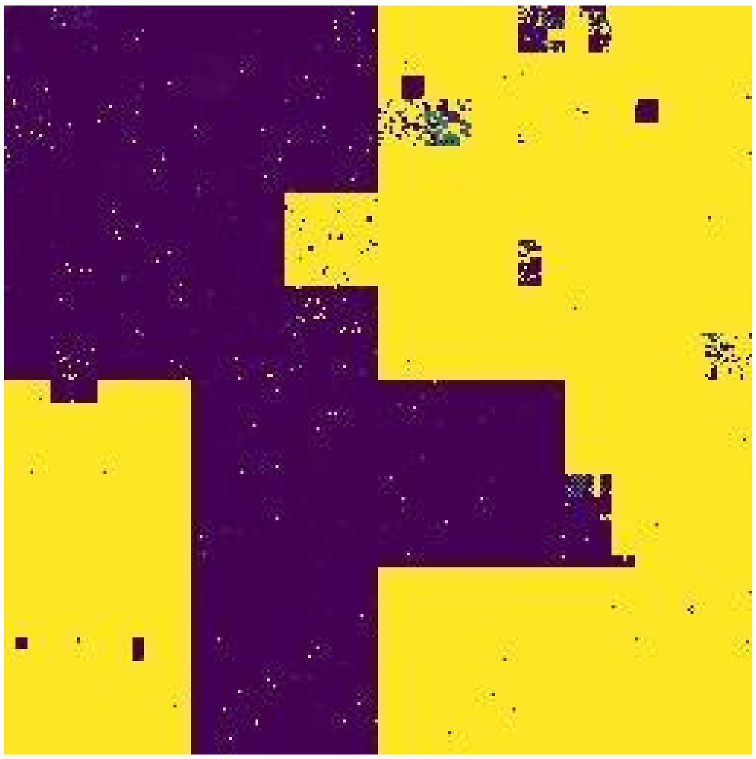
52.0.0.0/8:



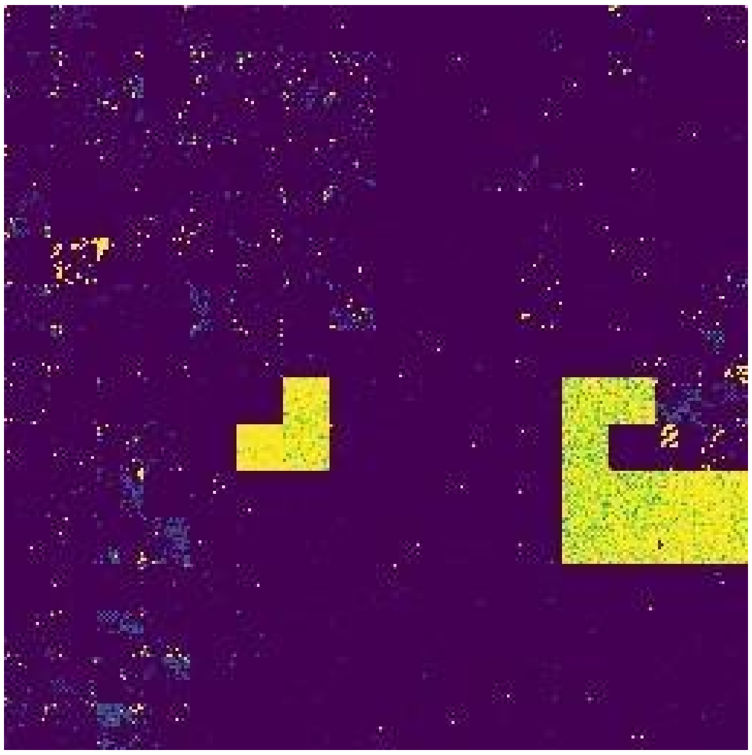
53.0.0.0/8: Mercedes-Benz (10/1993)



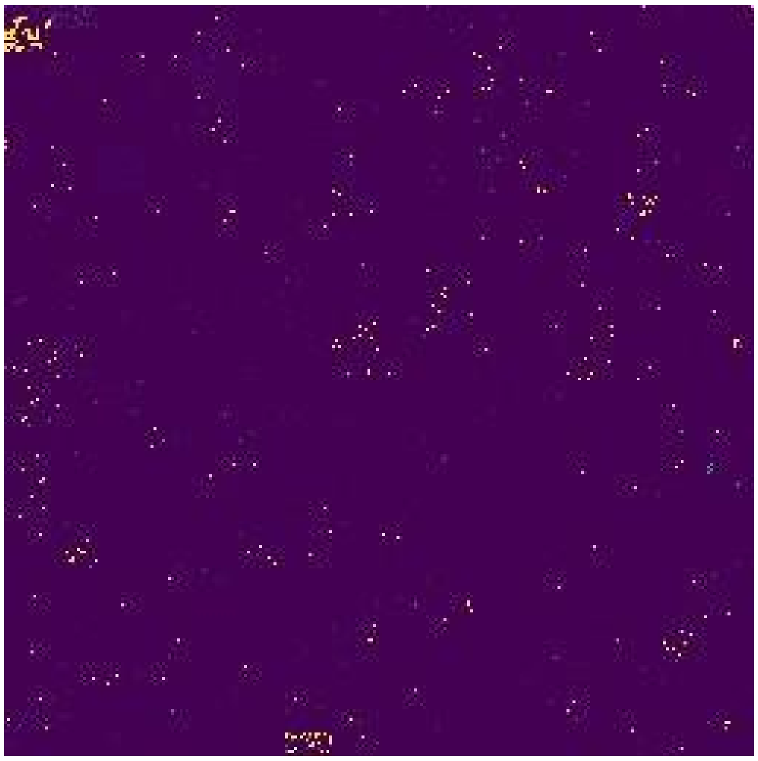
54.0.0.0/8:



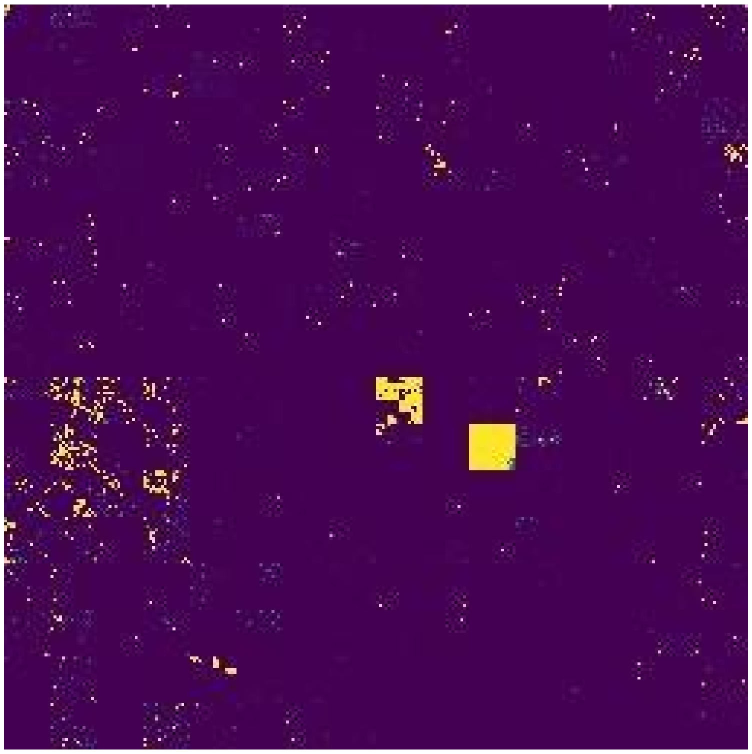
55.0.0.0/8: DOD NIC (04/1995)



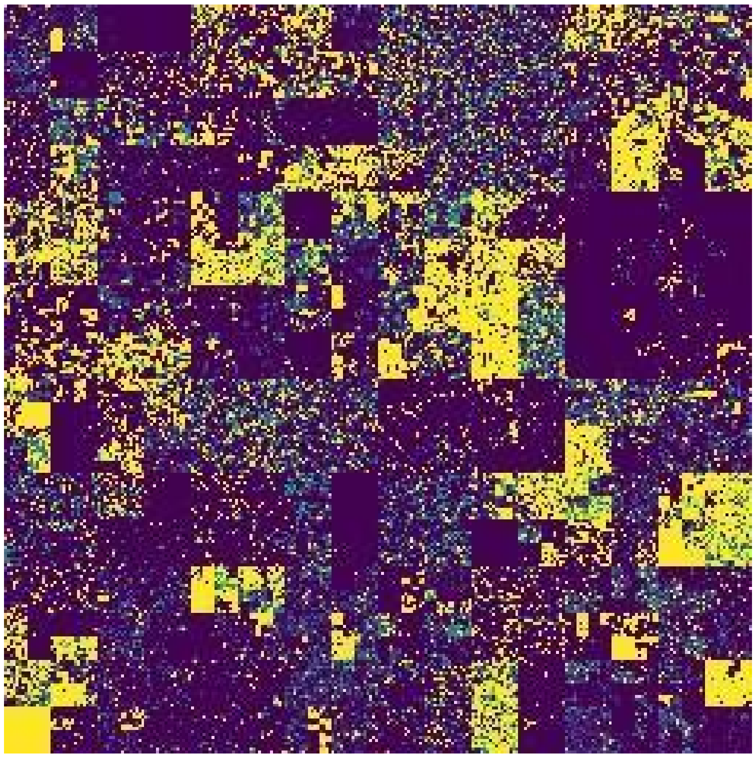
56.0.0.0/8:



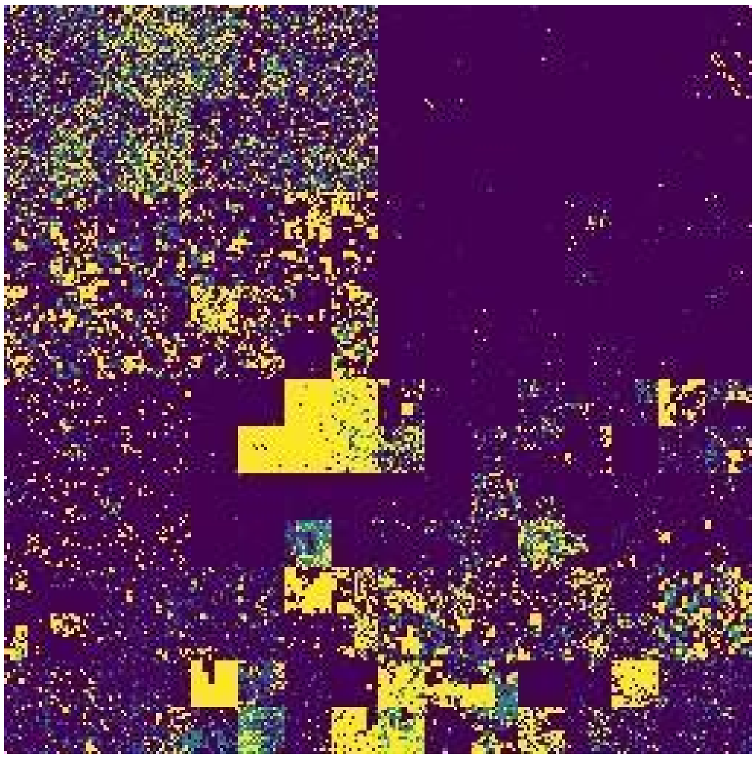
57.0.0.0/8: Formerly SITA



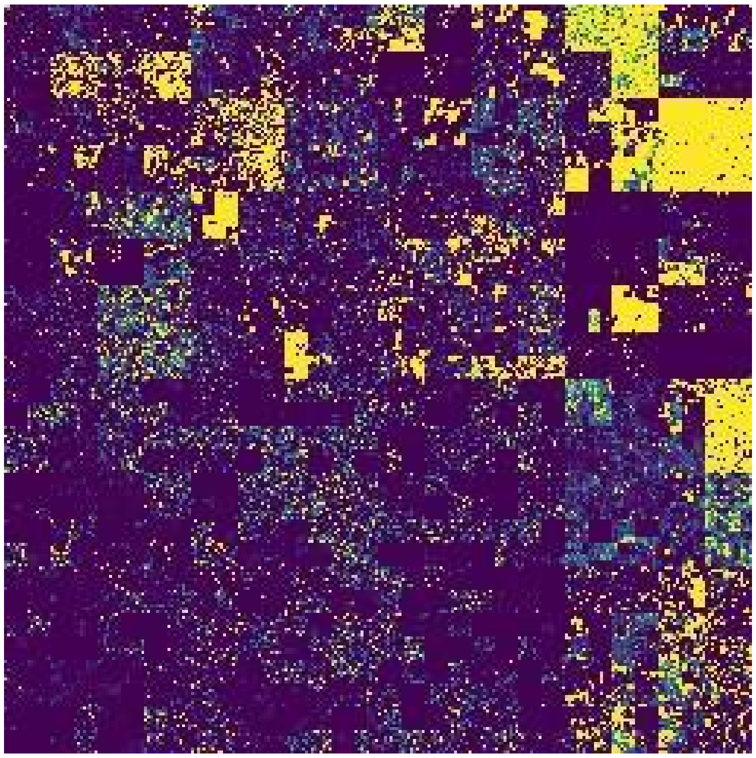
58.0.0.0/8:



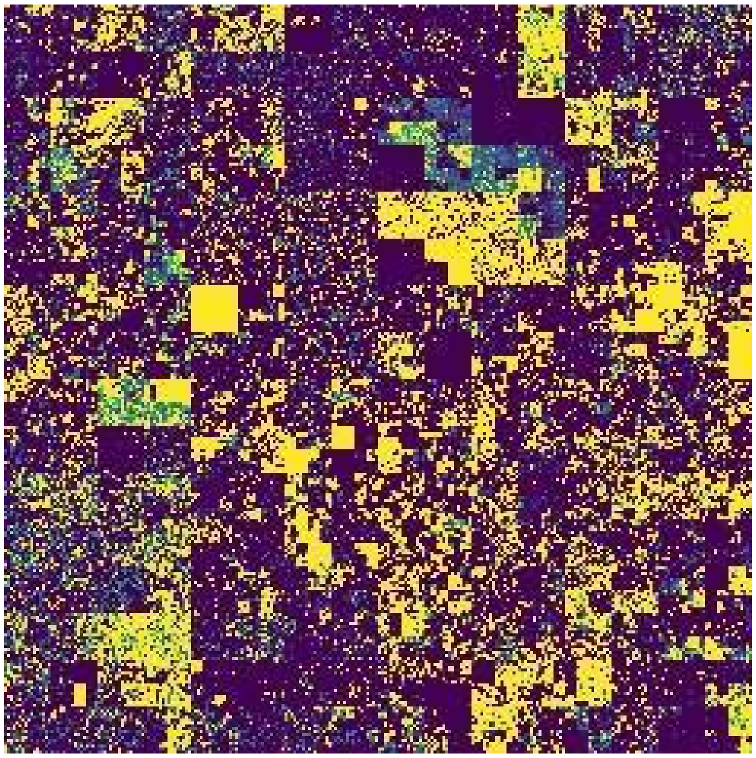
59.0.0.0/8:



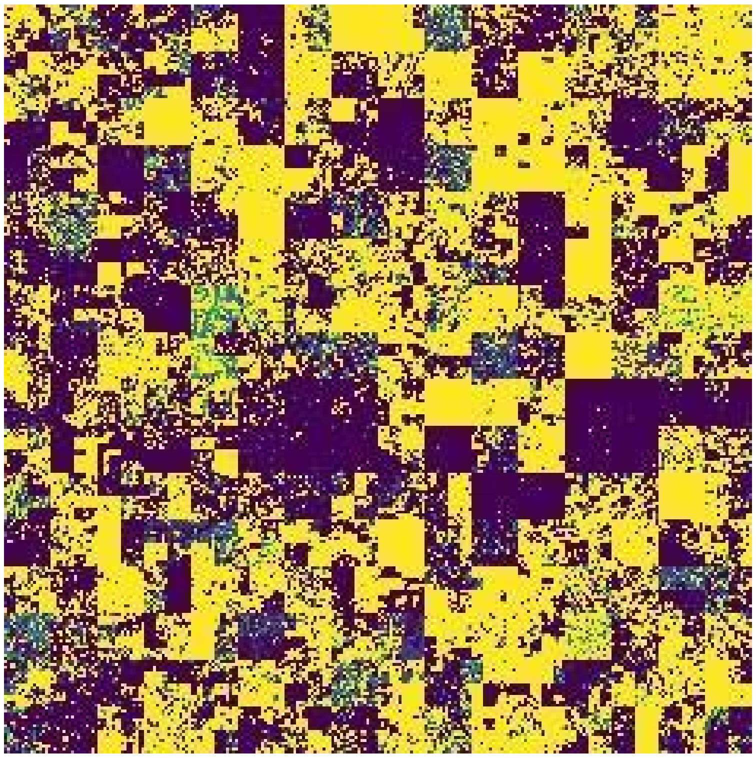
60.0.0.0/8:



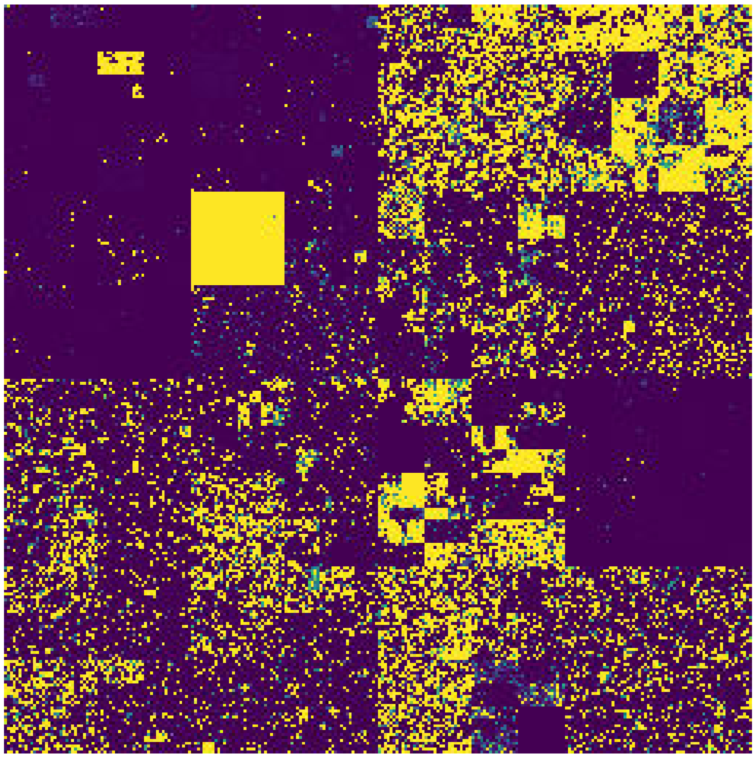
61.0.0.0/8



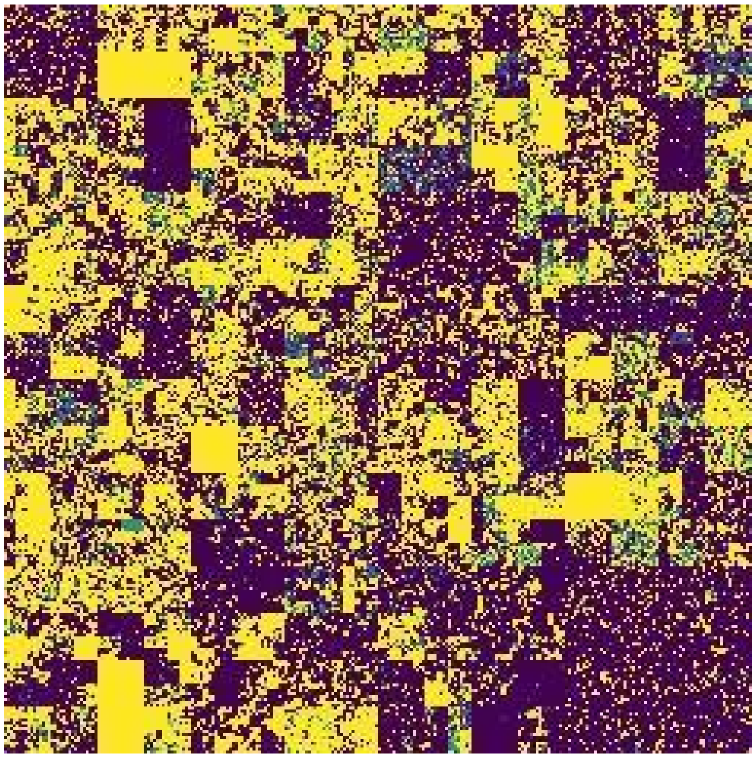
62.0.0.0/8:



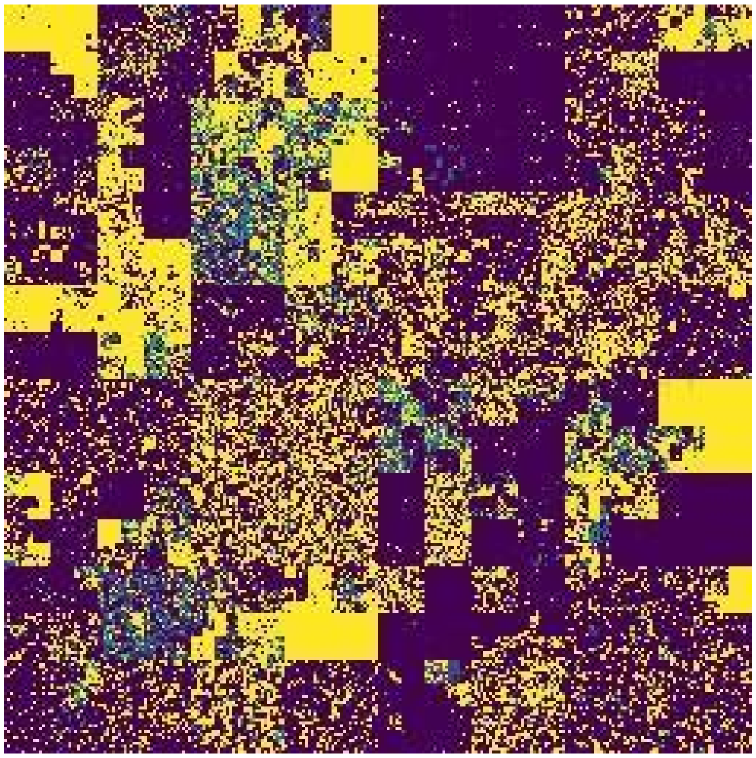
63.0.0.0/8



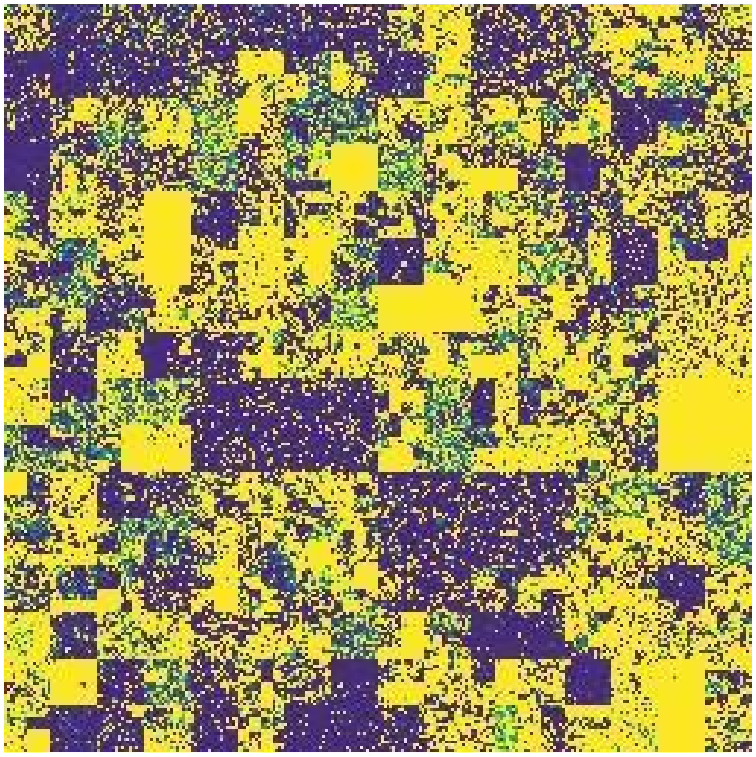
64.0.0.0/8:



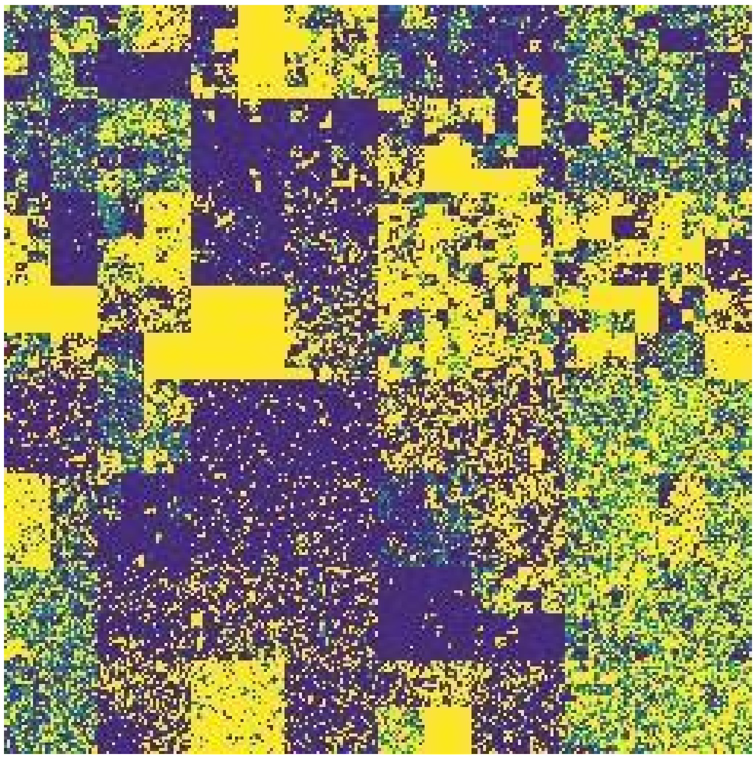
65.0.0.0/8



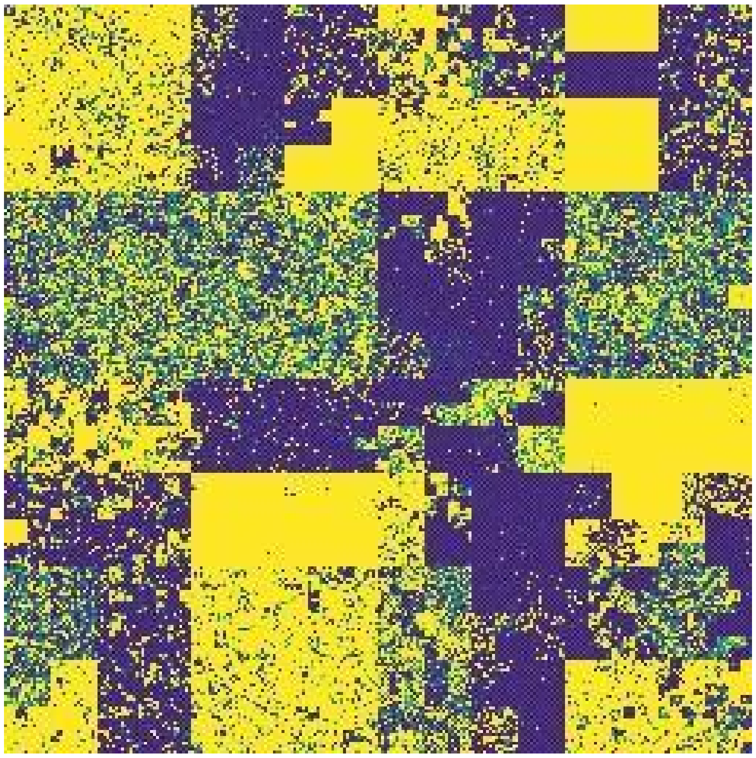
66.0.0.0/8:



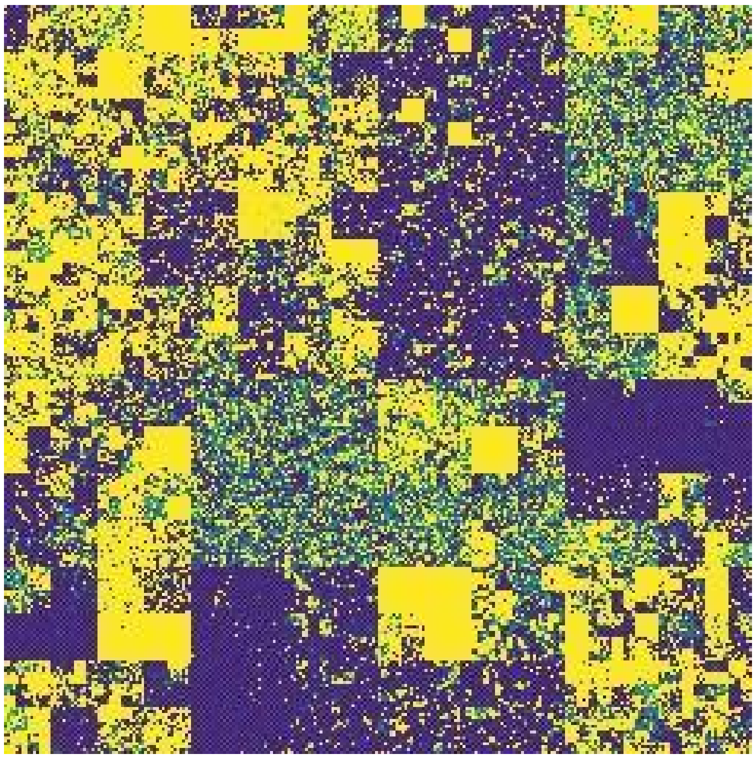
67.0.0.0/8:



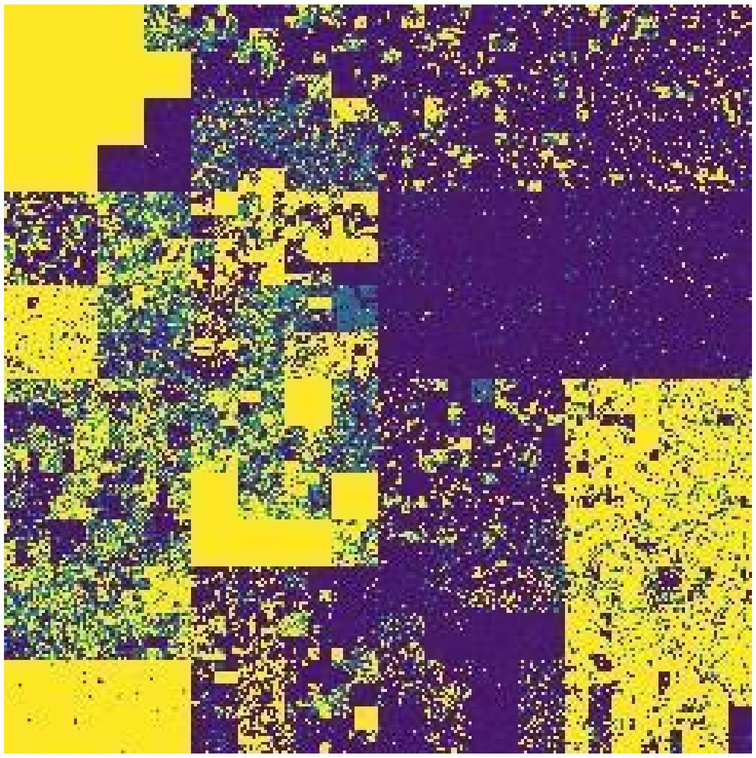
68.0.0.0/8:



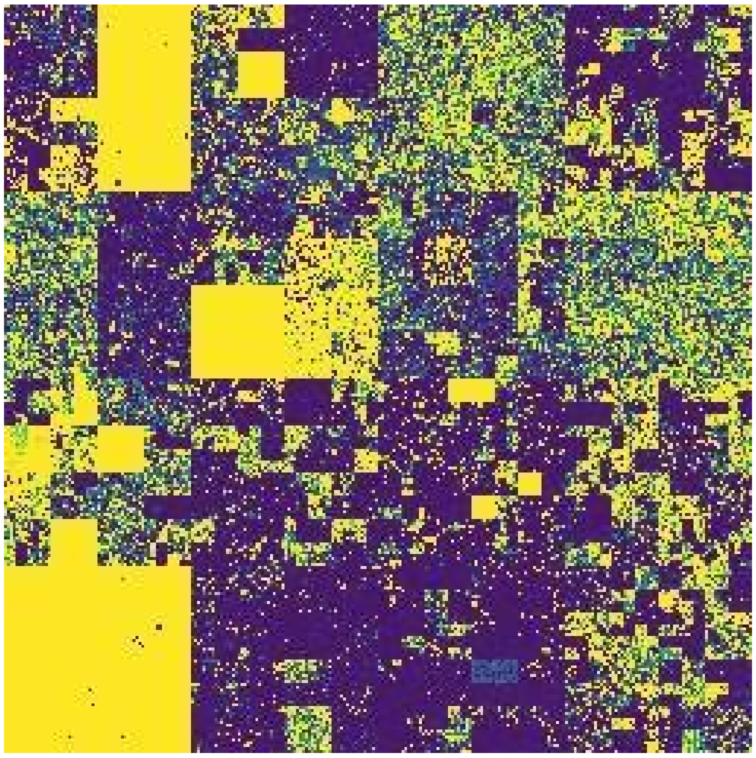
69.0.0.0/8:



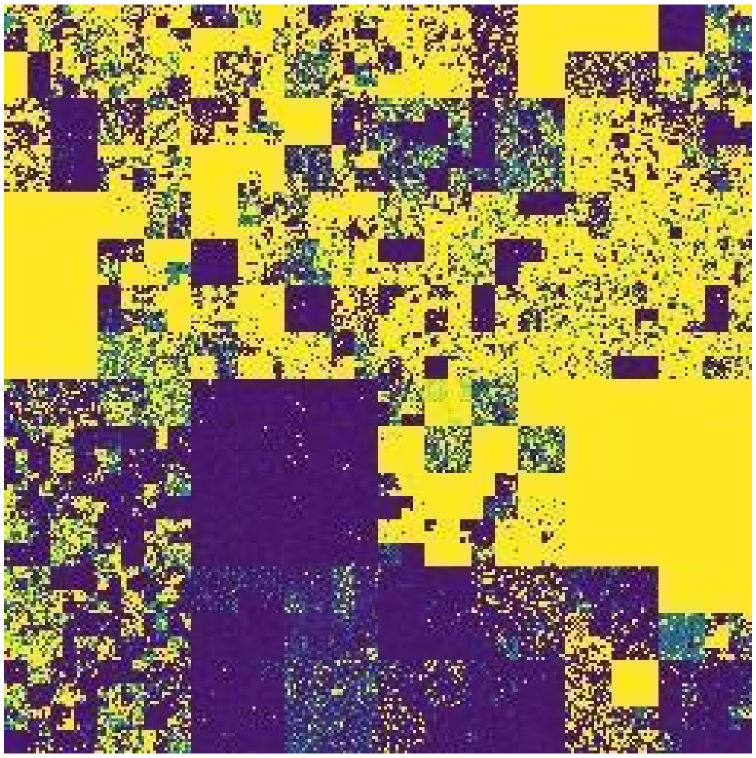
70.0.0.0/8:



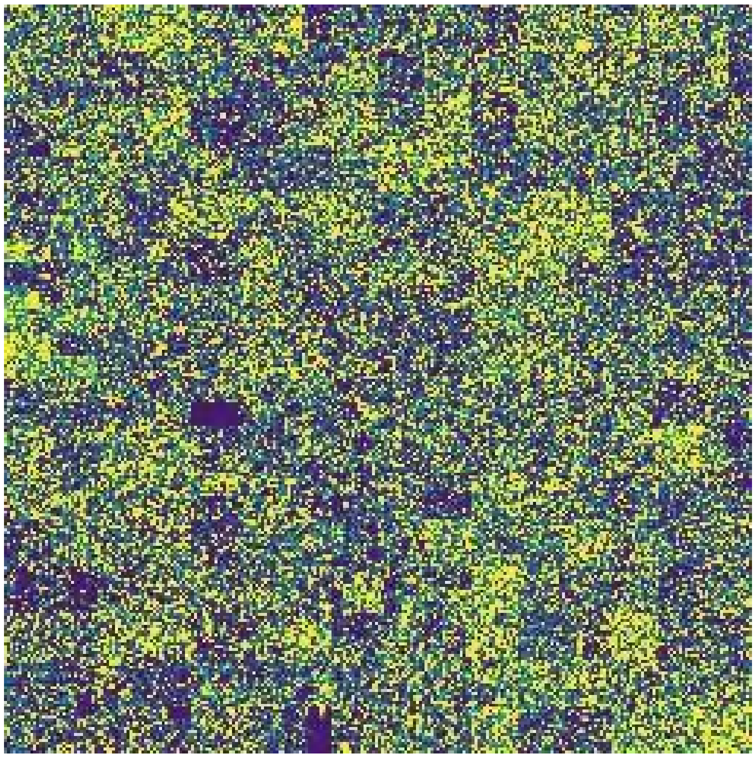
71.0.0.0/8:



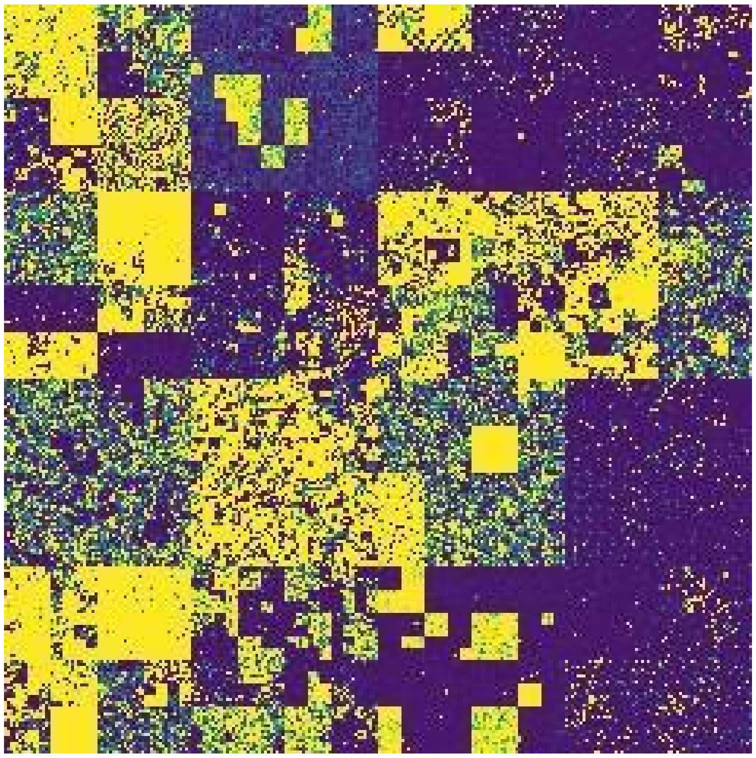
72.0.0.0/8:



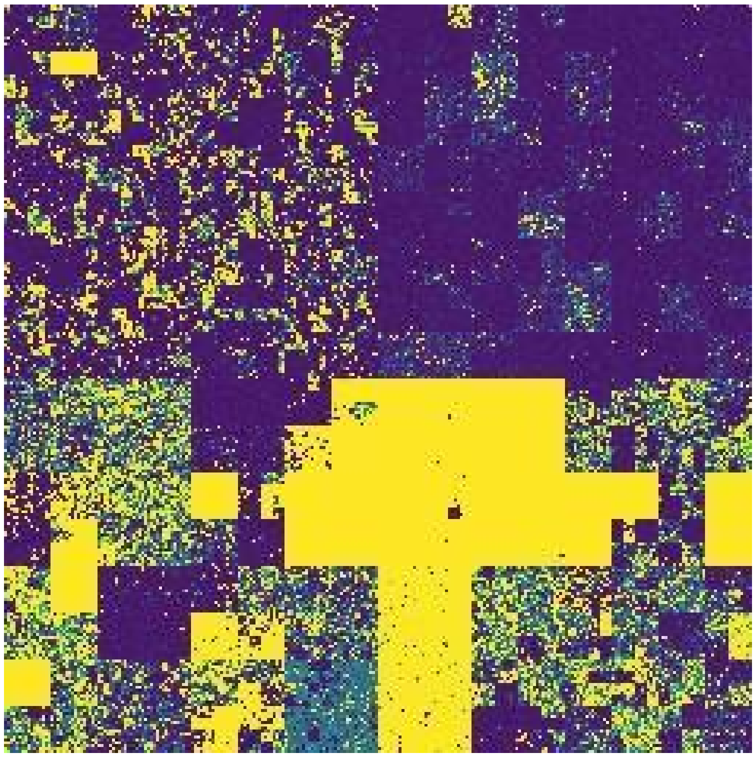
73.0.0.0/8:



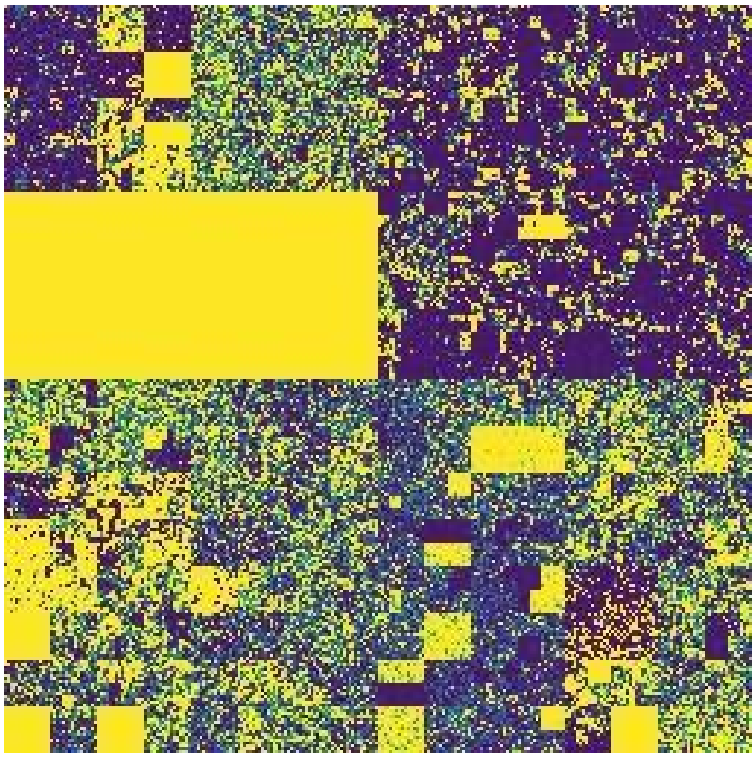
74.0.0.0/8:



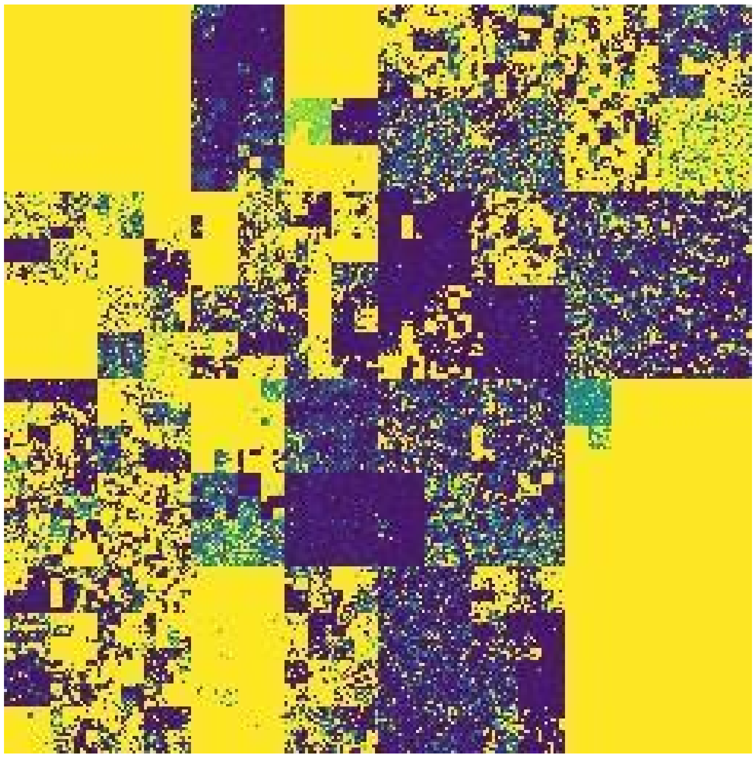
75.0.0.0/8:



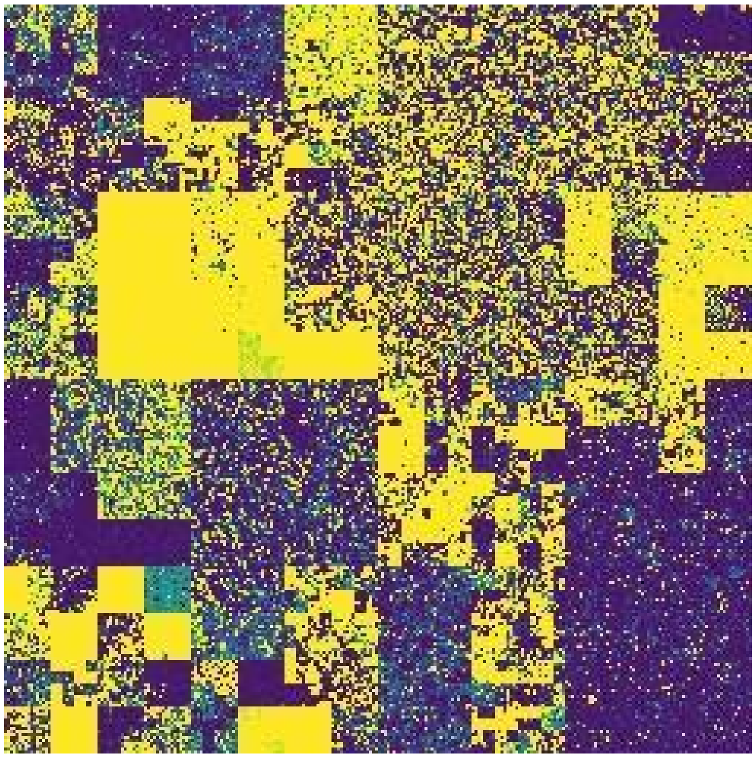
76.0.0.0/8:



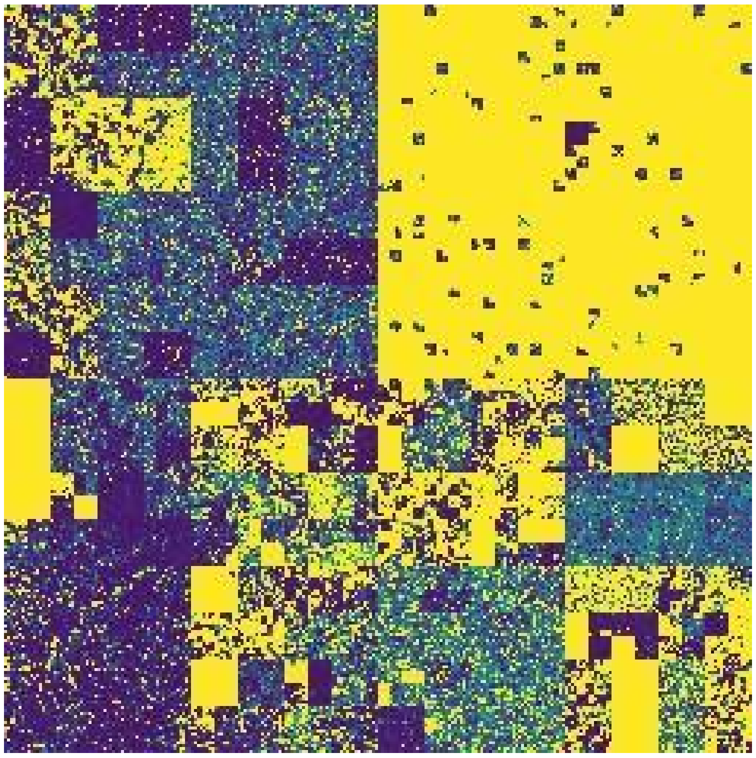
77.0.0.0/8:



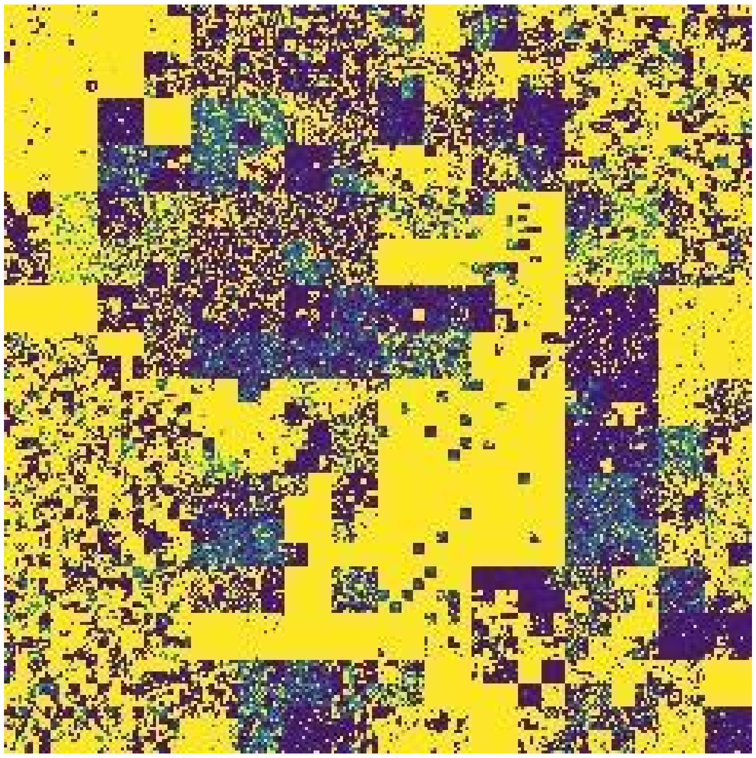
78.0.0.0/8:



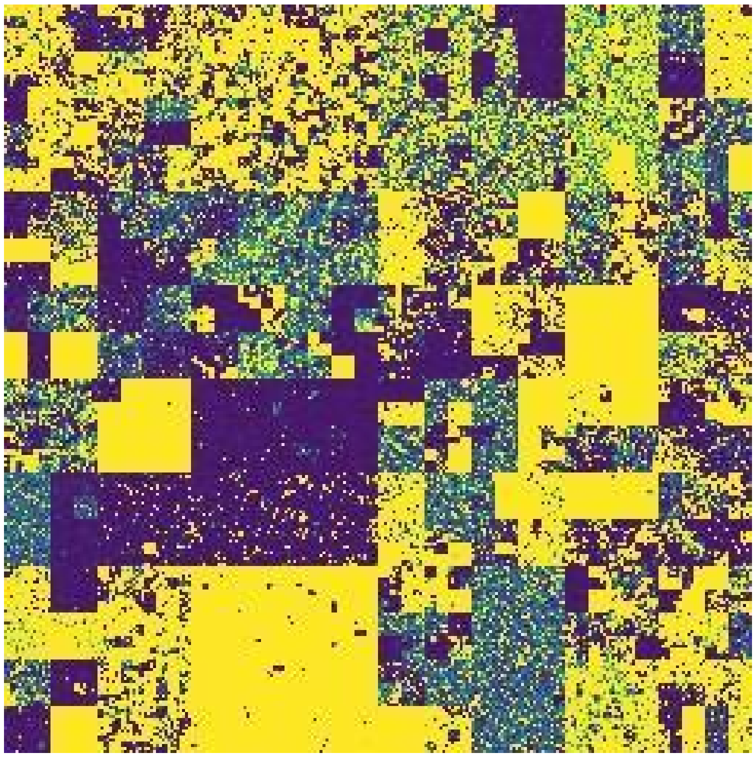
79.0.0.0/8:



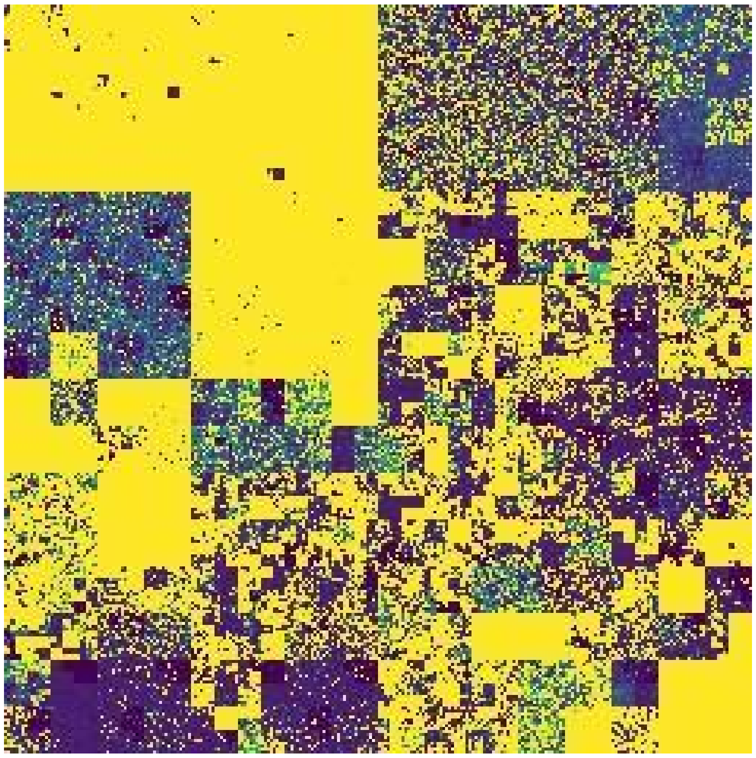
80.0.0.0/8:



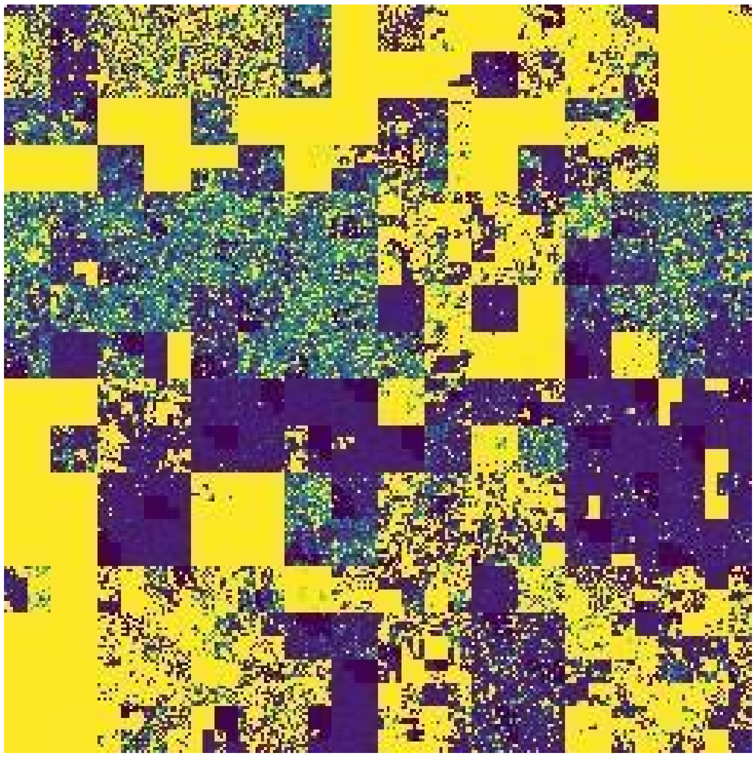
81.0.0.0/8:



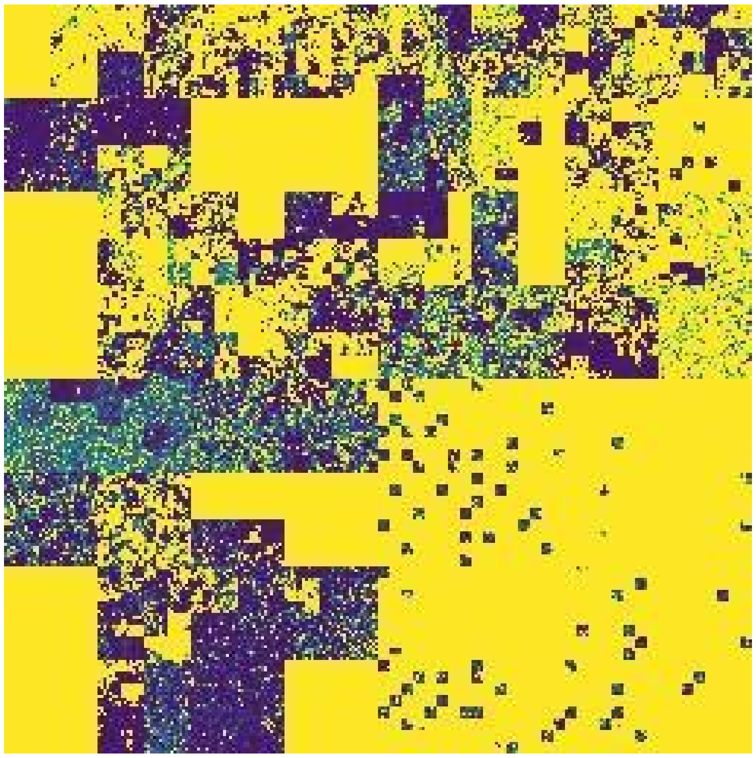
82.0.0.0/8:



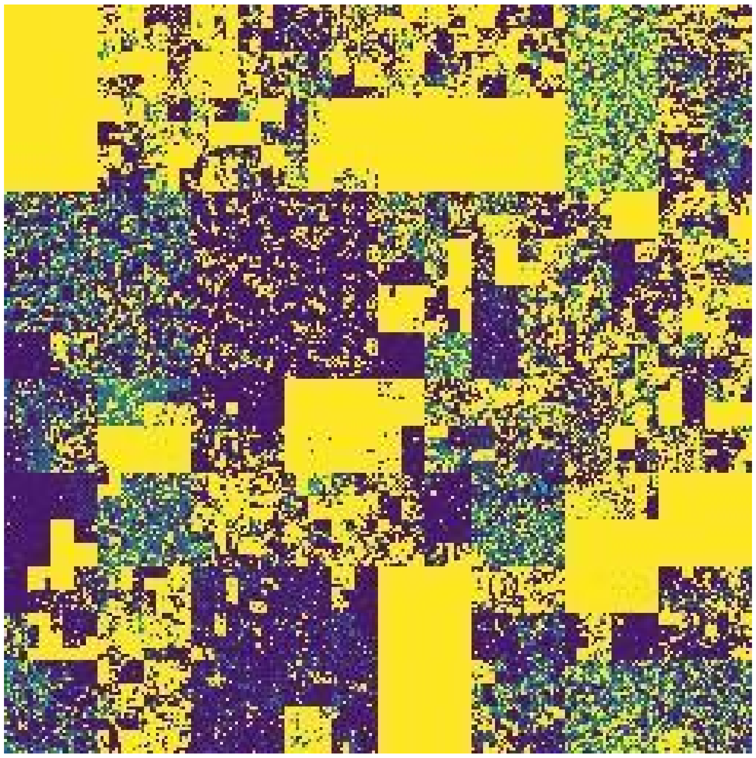
83.0.0.0/8:



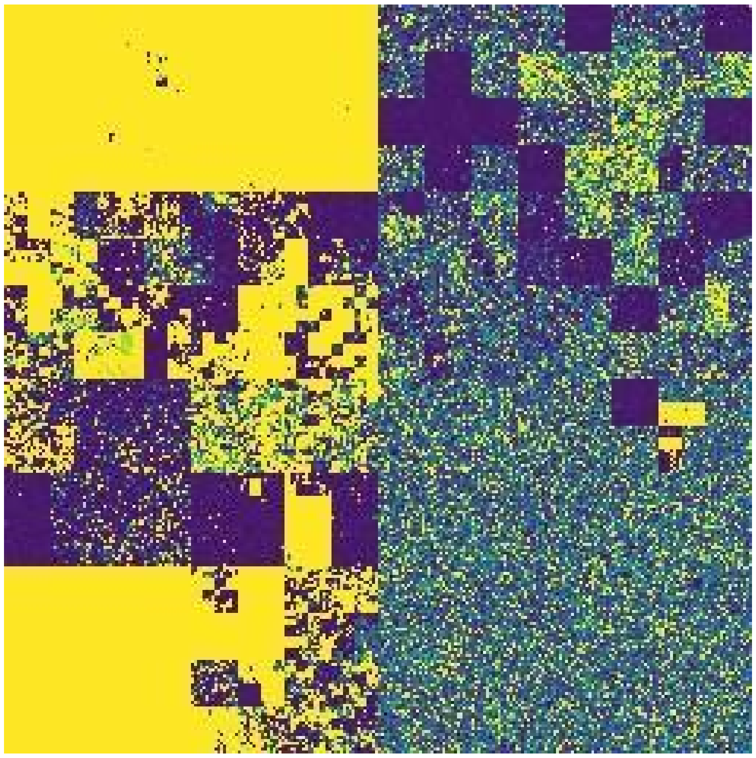
84.0.0.0/8:



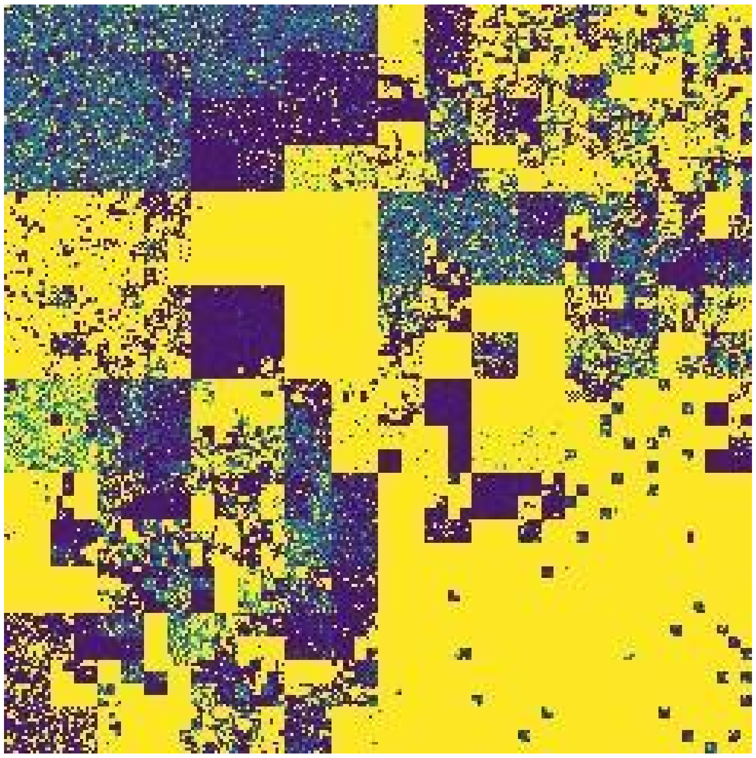
85.0.0.0/8:



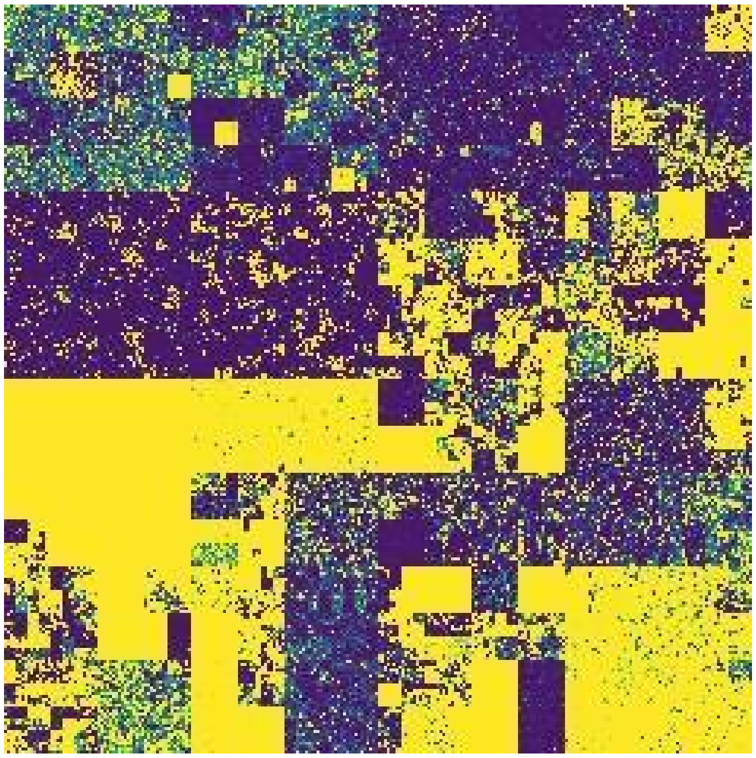
86.0.0.0/8:



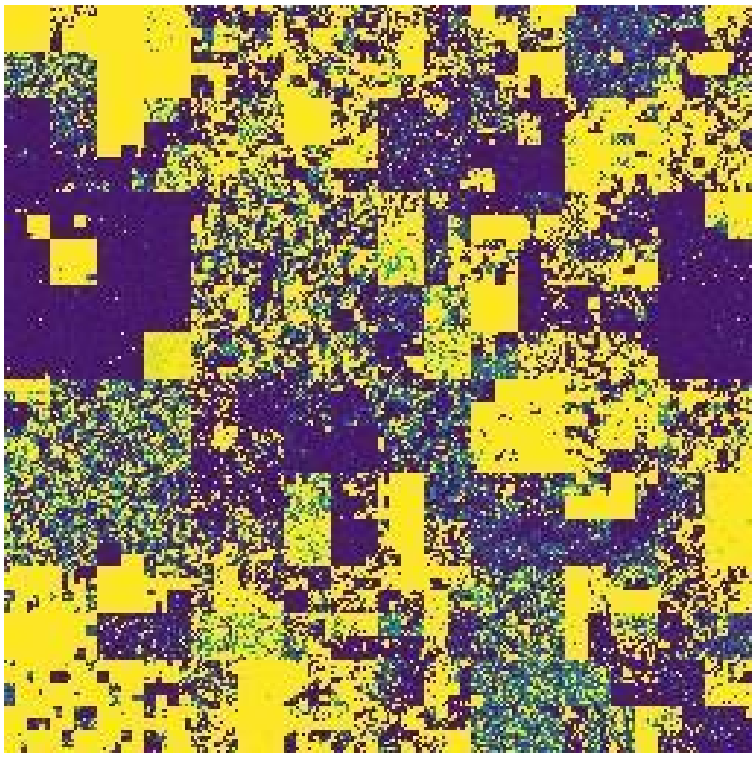
87.0.0.0/8:



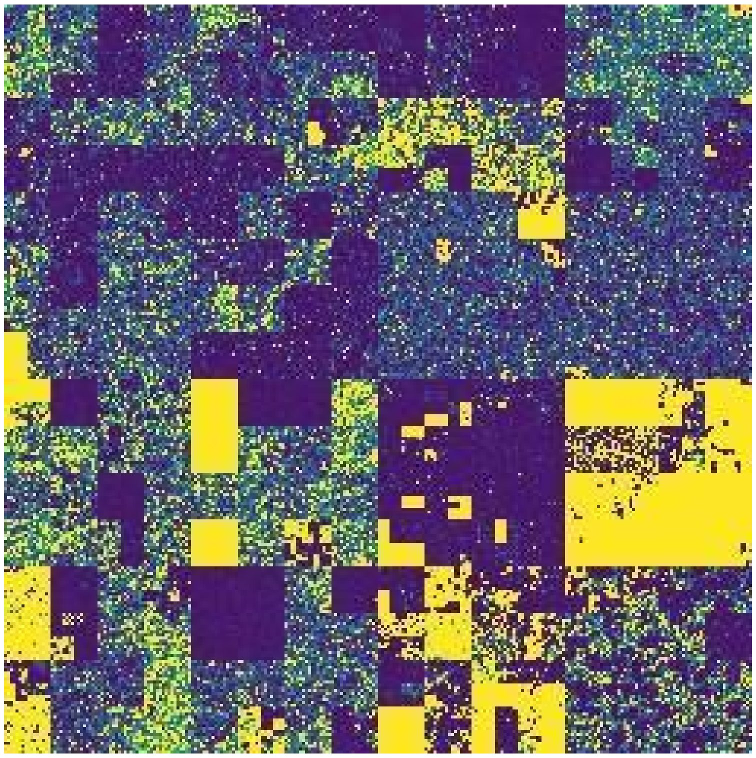
88.0.0.0/8:



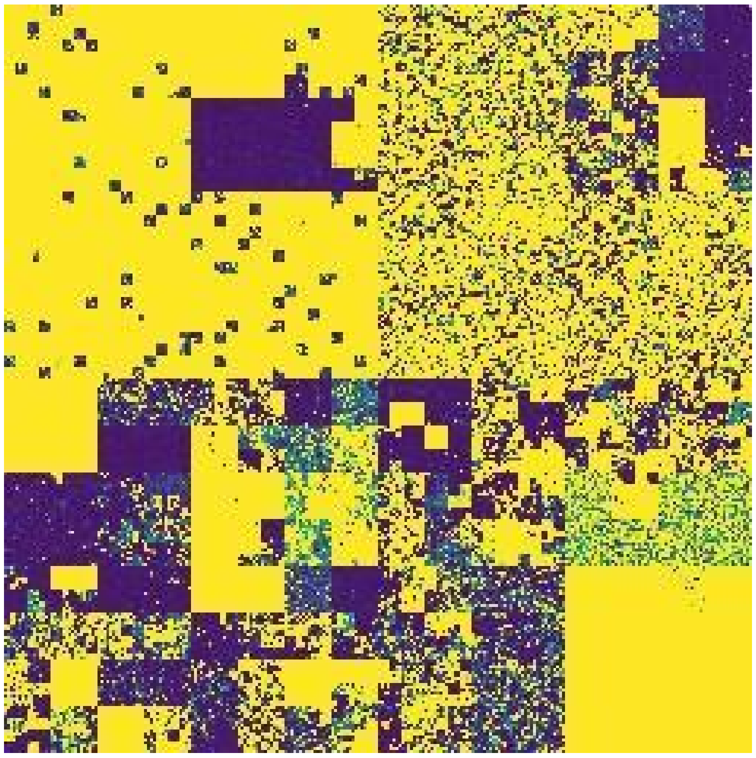
89.0.0.0/8:



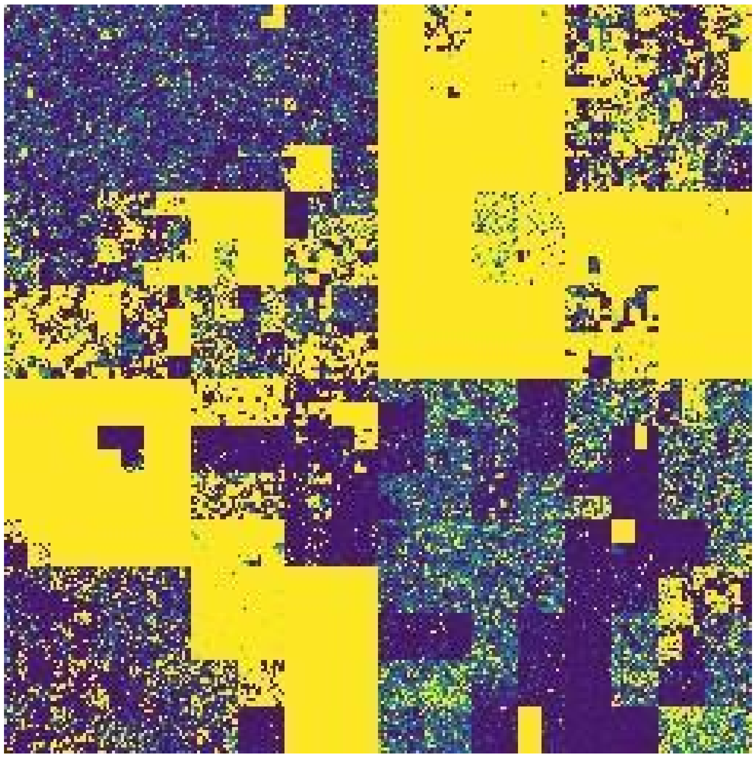
90.0.0.0/8:



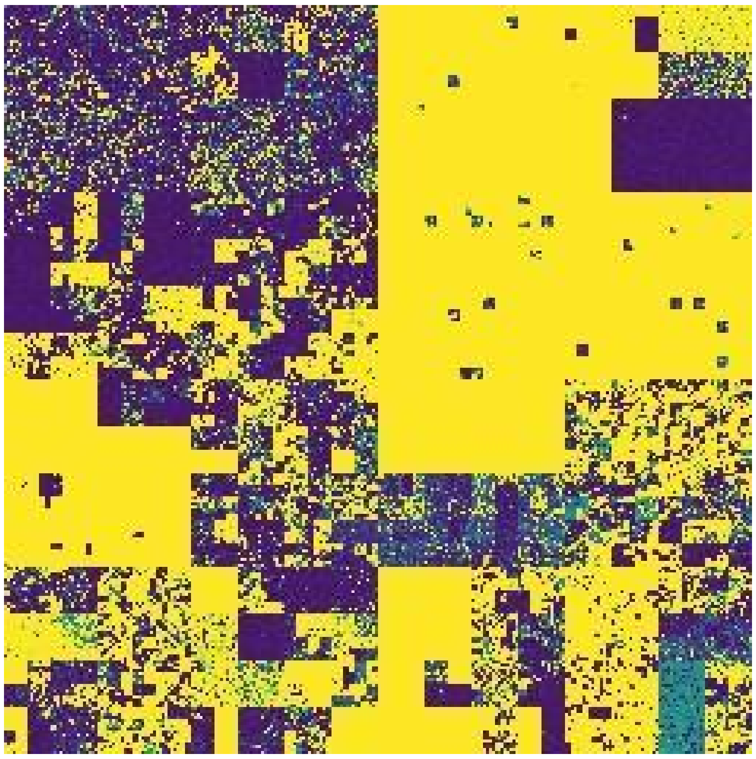
91.0.0.0/8:



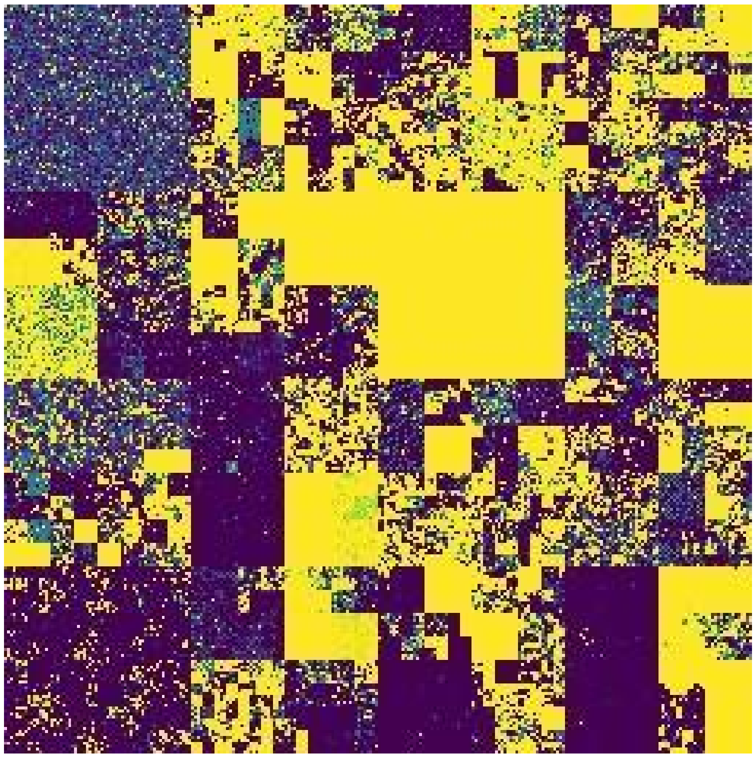
92.0.0.0/8:



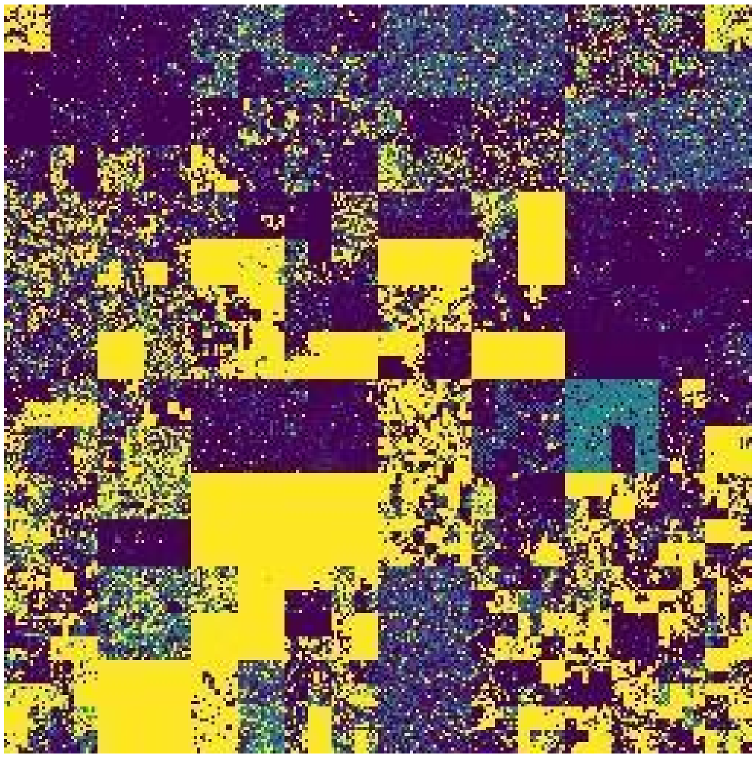
93.0.0.0/8:



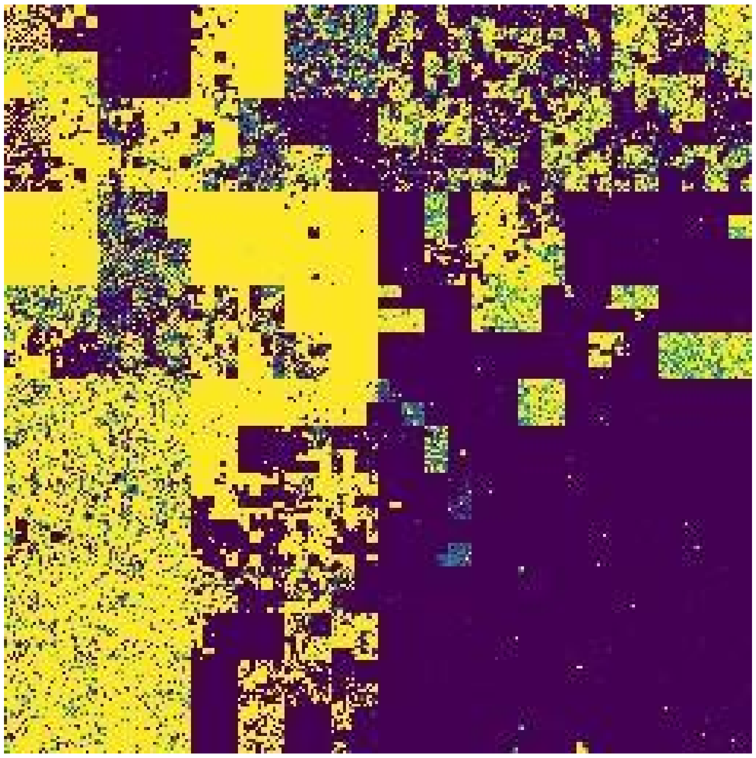
94.0.0.0/8:



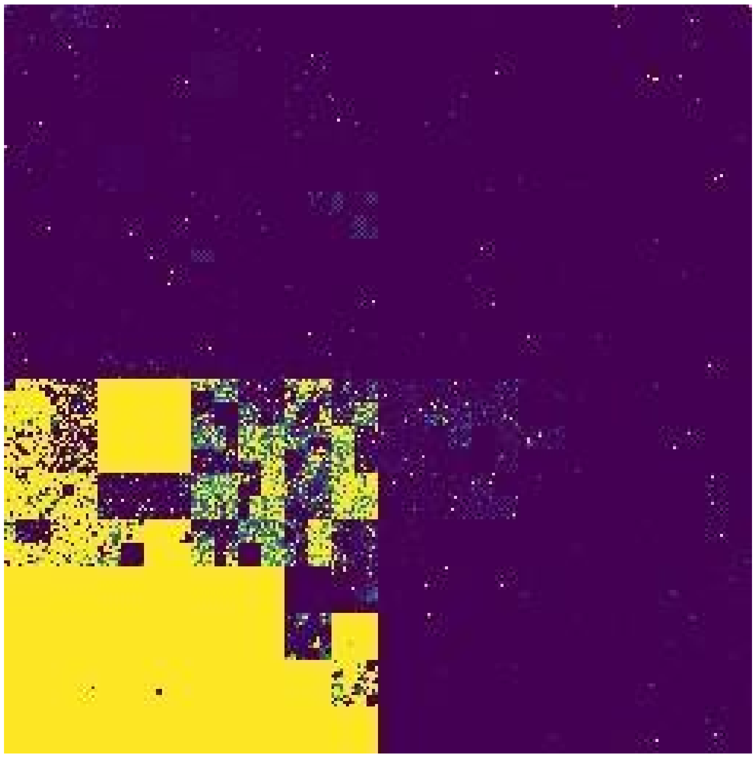
95.0.0.0/8:



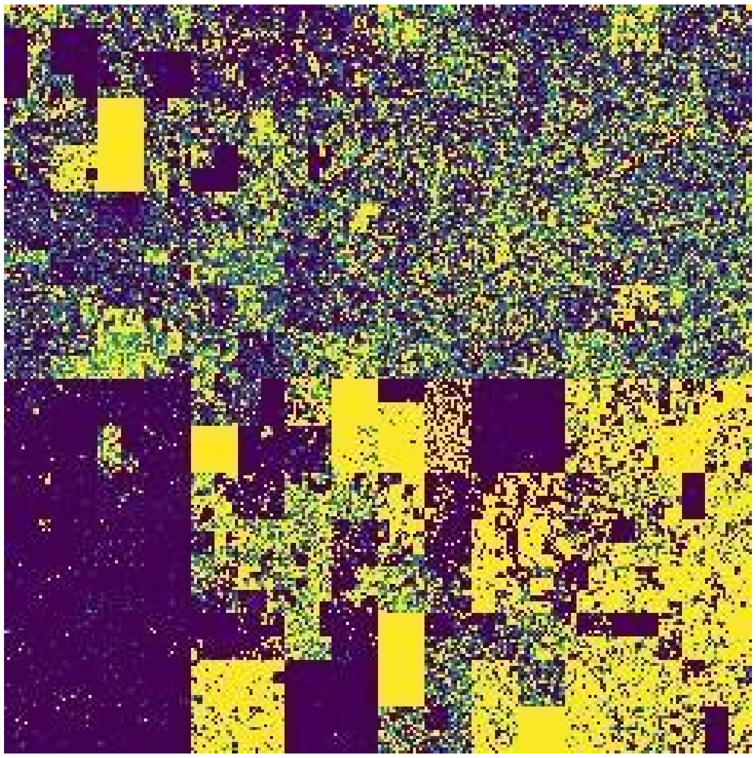
96.0.0.0/8:



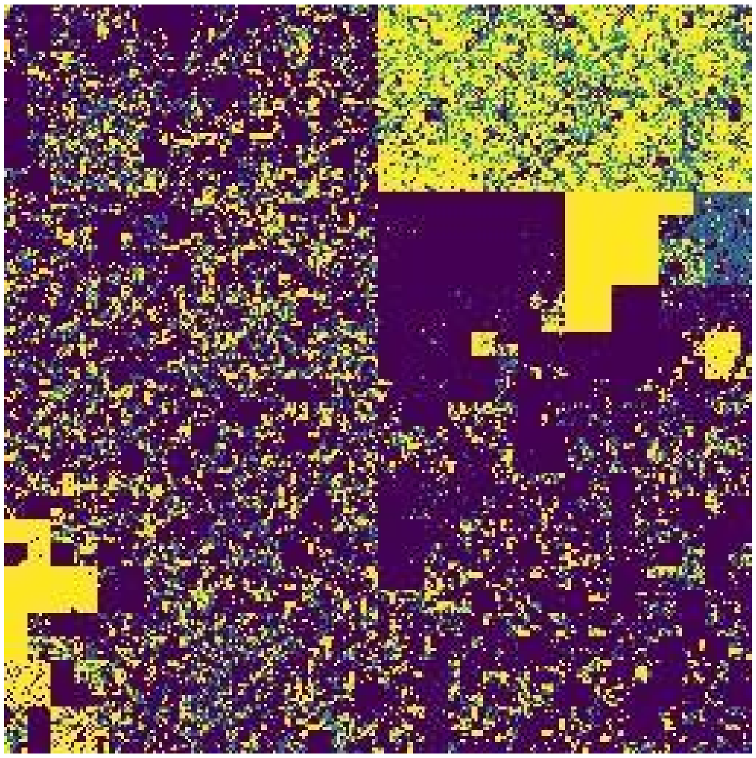
97.0.0.0/8:



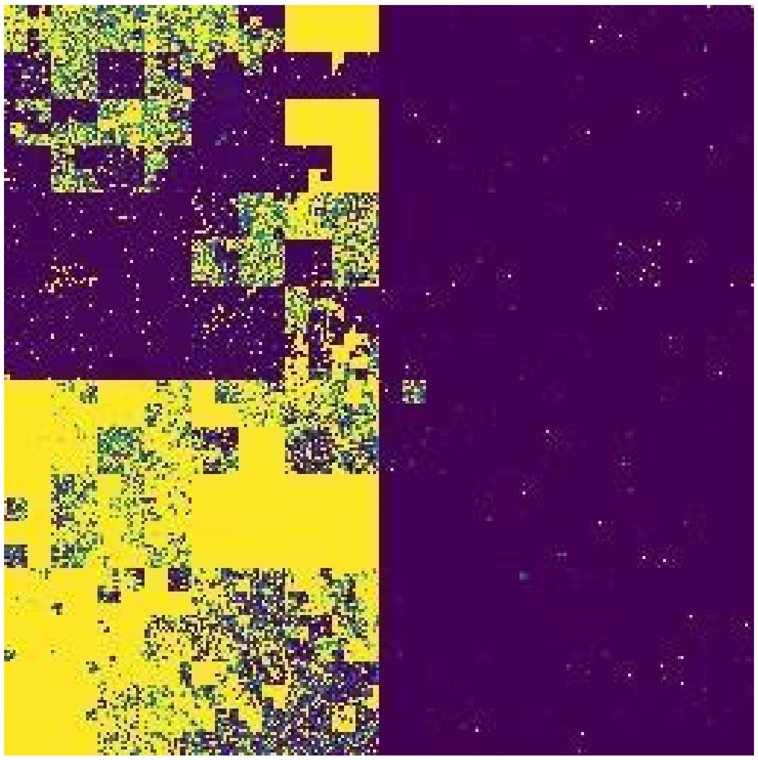
98.0.0.0/8:



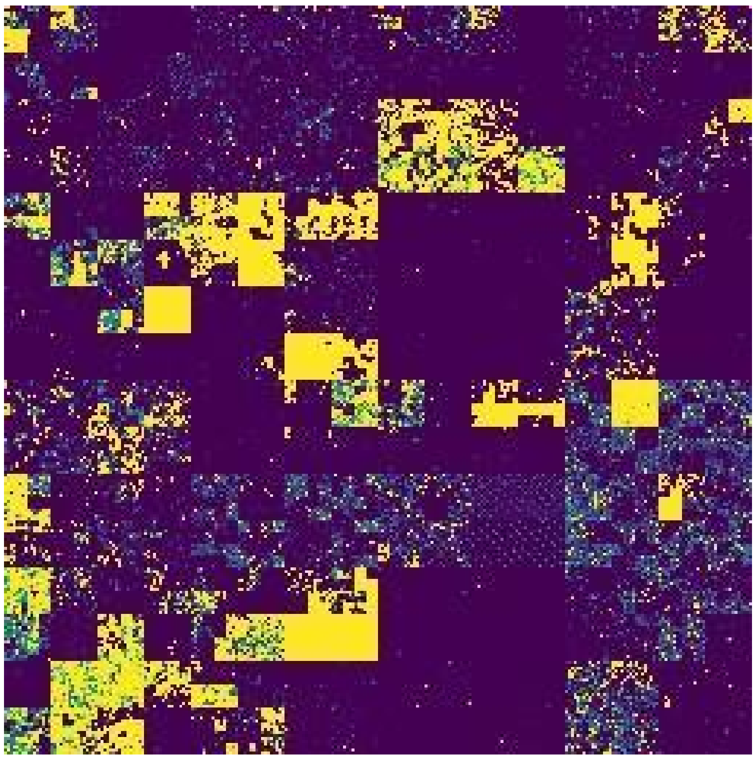
99.0.0.0/8:



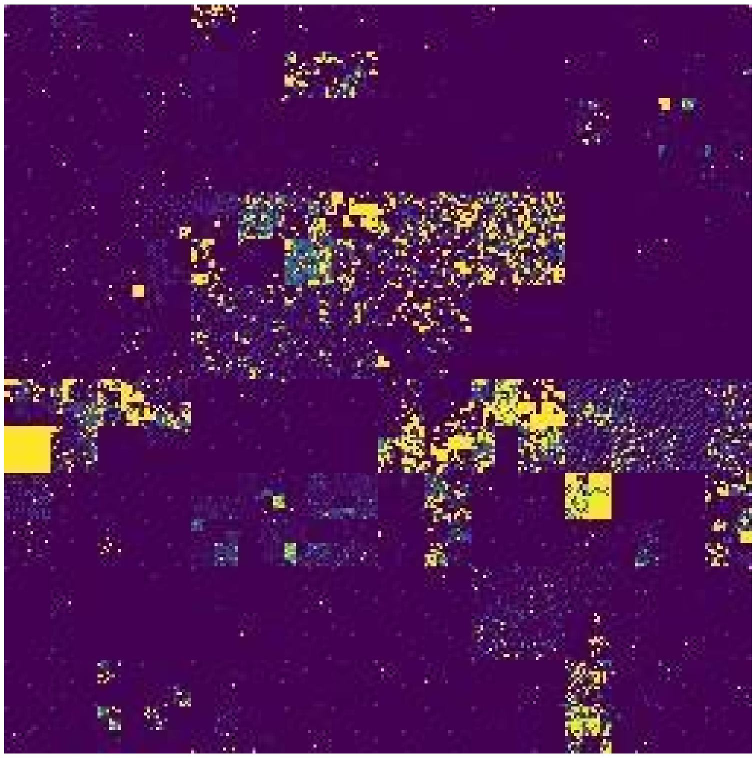
100.0.0.0/8: (100.64.0.0/10 is reserved for carrier grade NAT per RFC 6598)



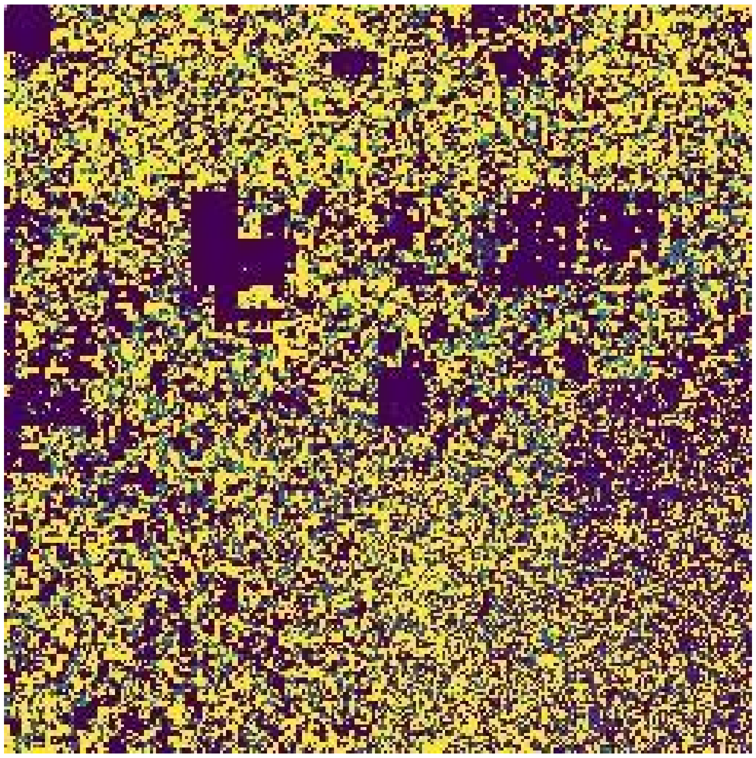
101.0.0.0/8:



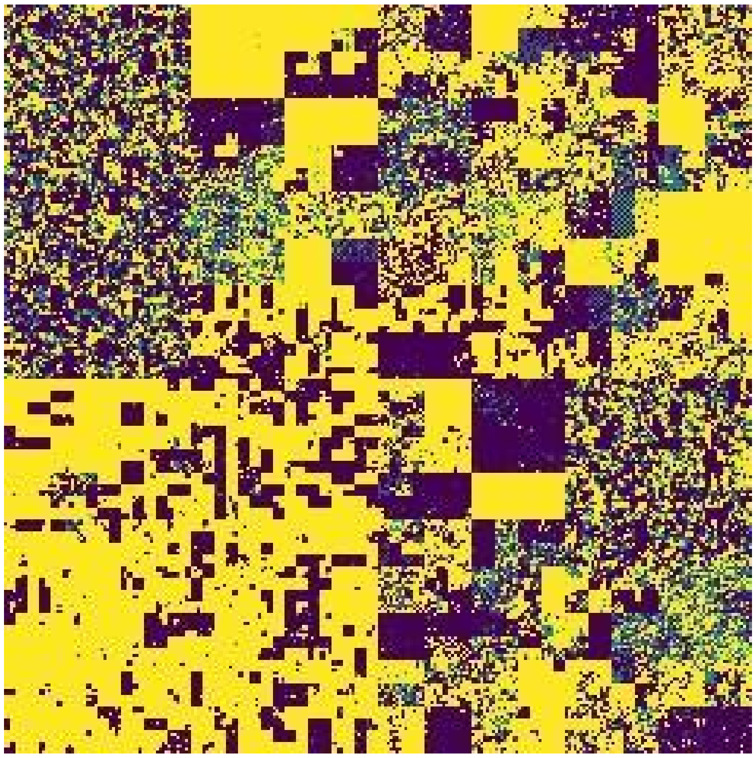
102.0.0.0/8:



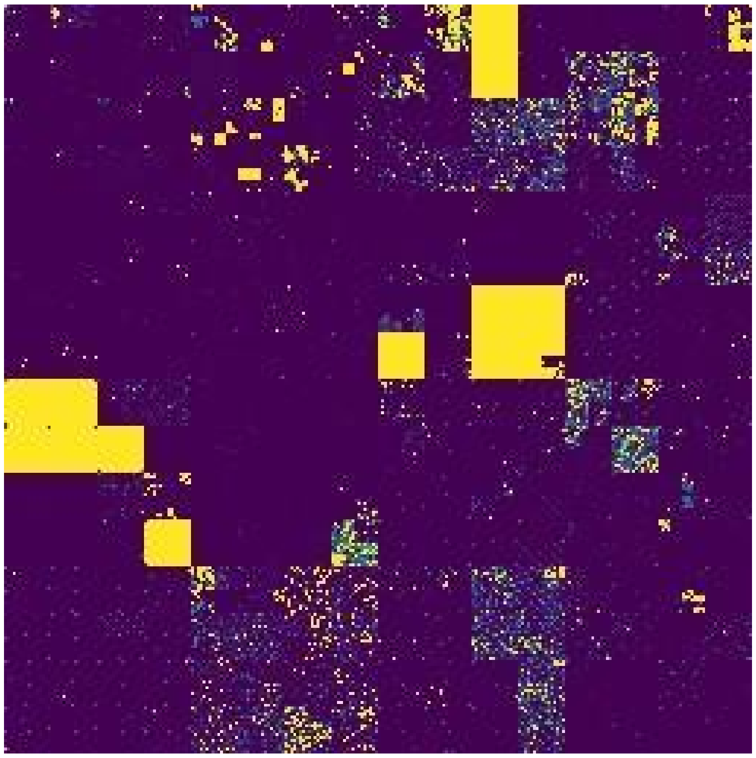
103.0.0.0/8:



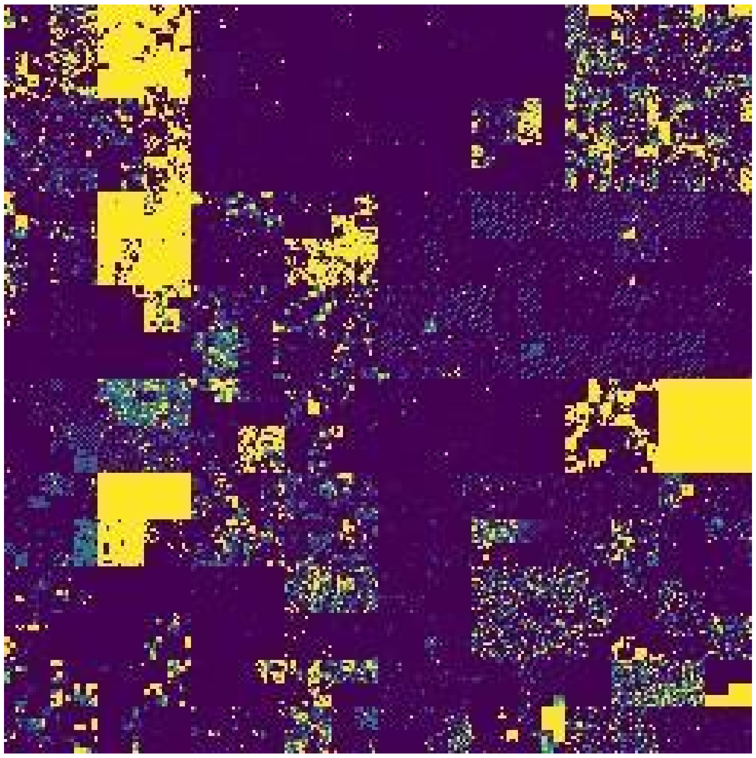
104.0.0.0/8:



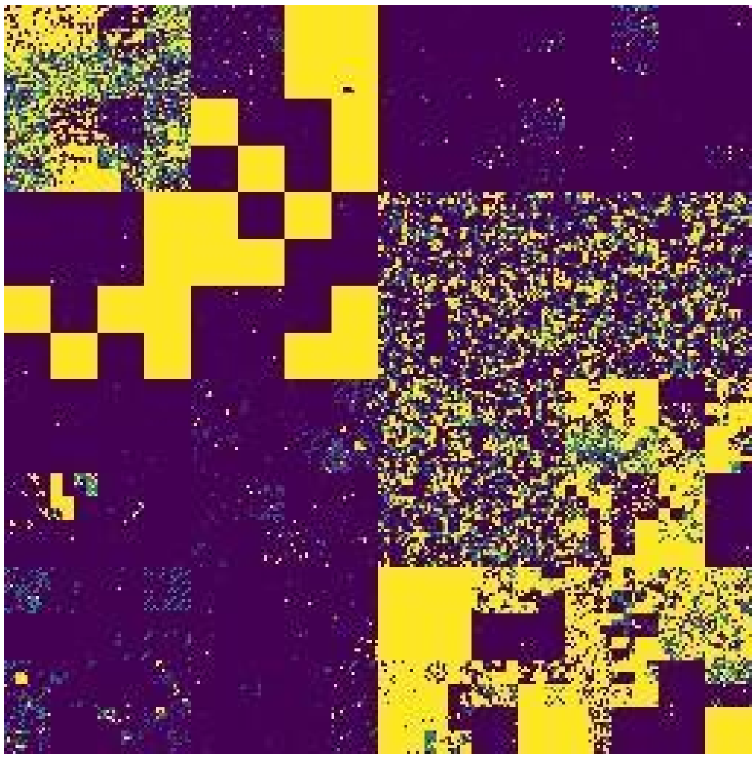
105.0.0.0/8:



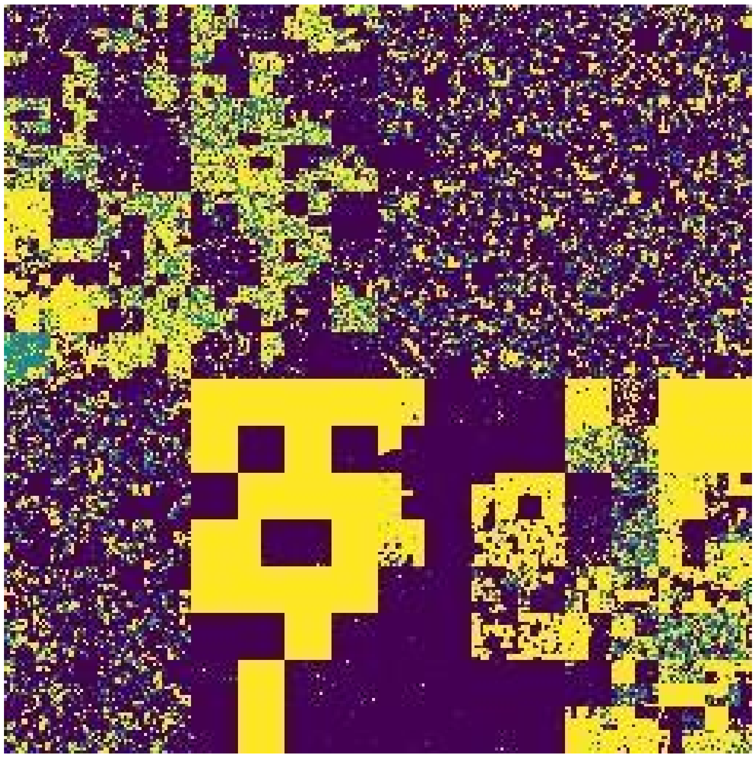
106.0.0.0/8:



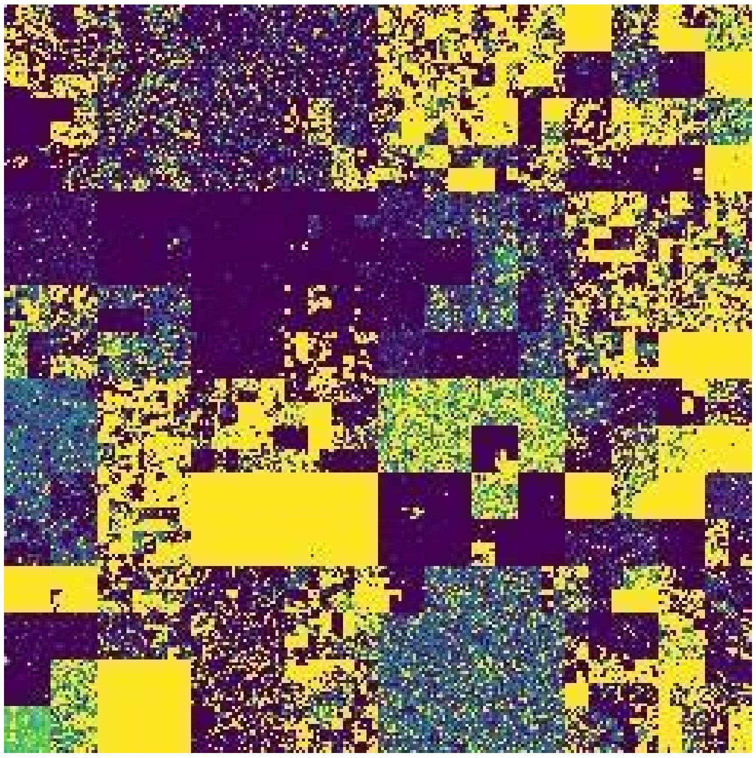
107.0.0.0/8:



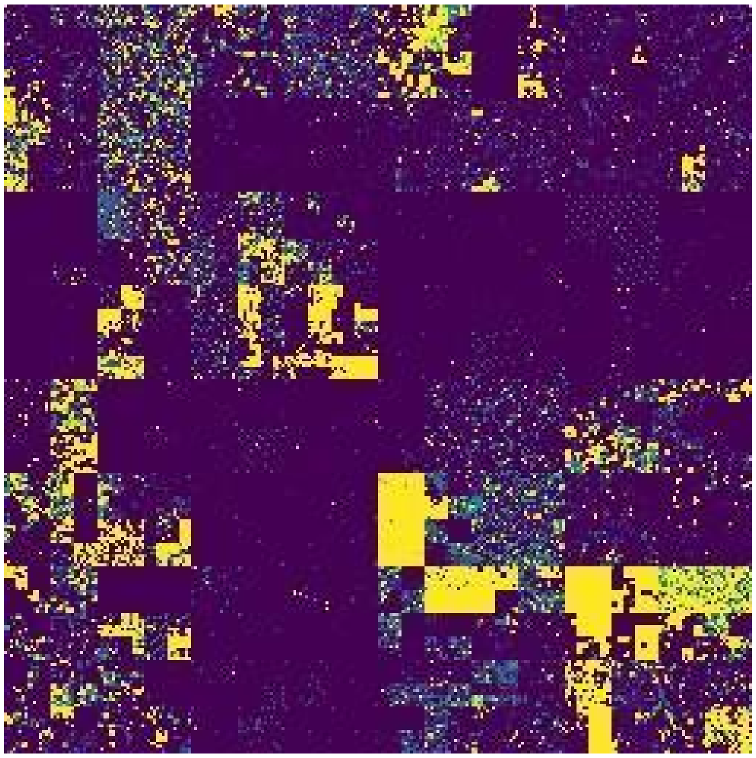
108.0.0.0/8:



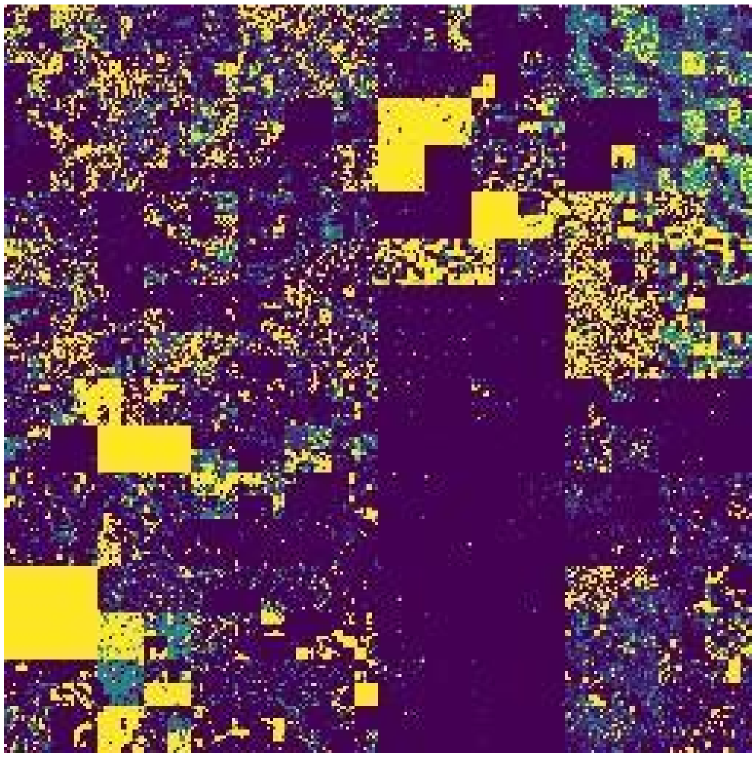
109.0.0.0/8:



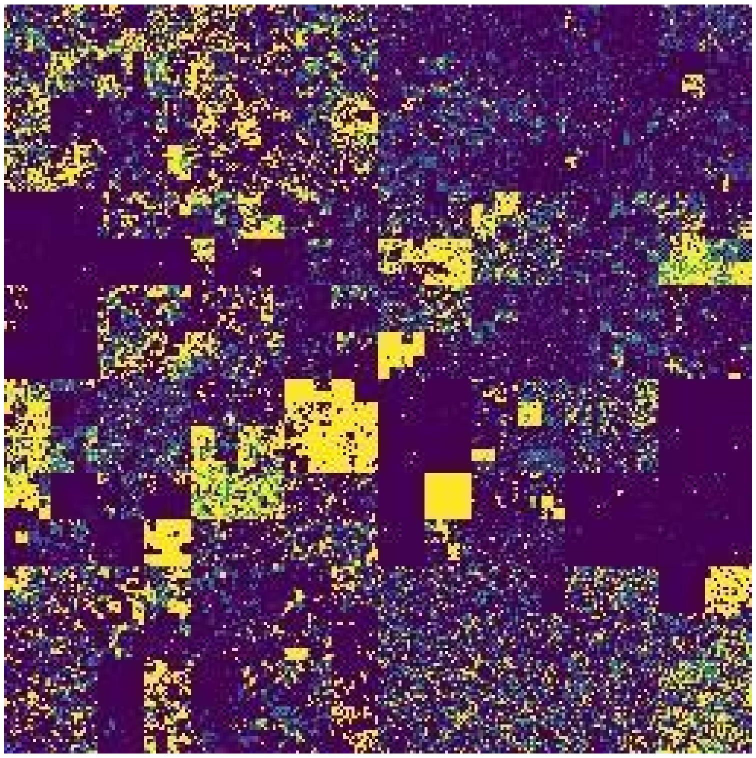
110.0.0.0/8:



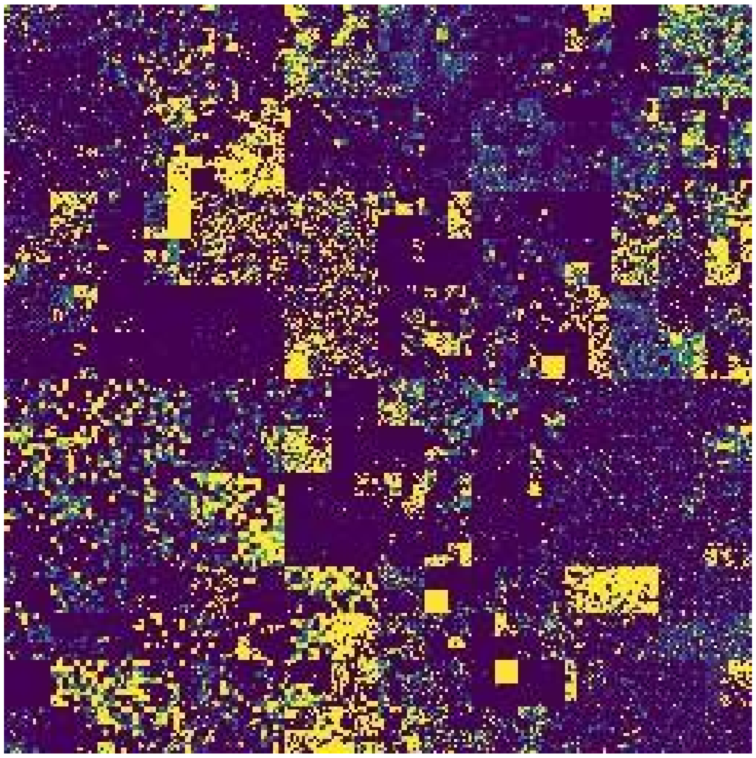
111.0.0.0/8:



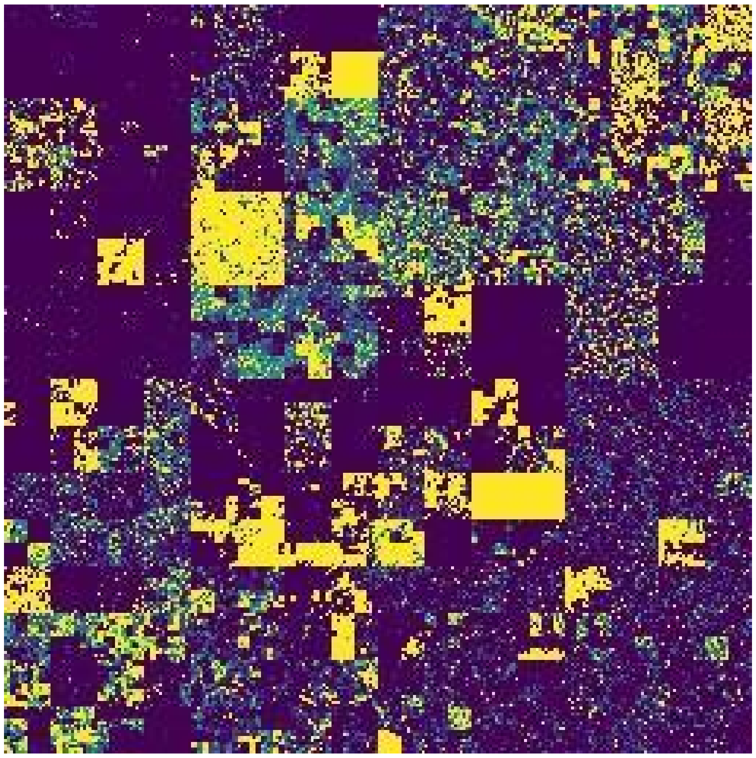
112.0.0.0/8:



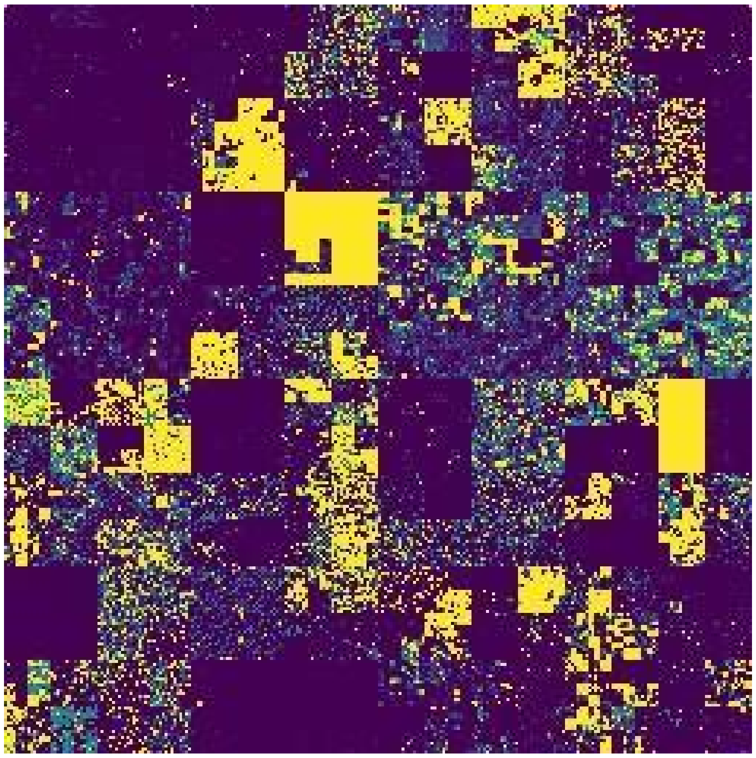
113.0.0.0/8:



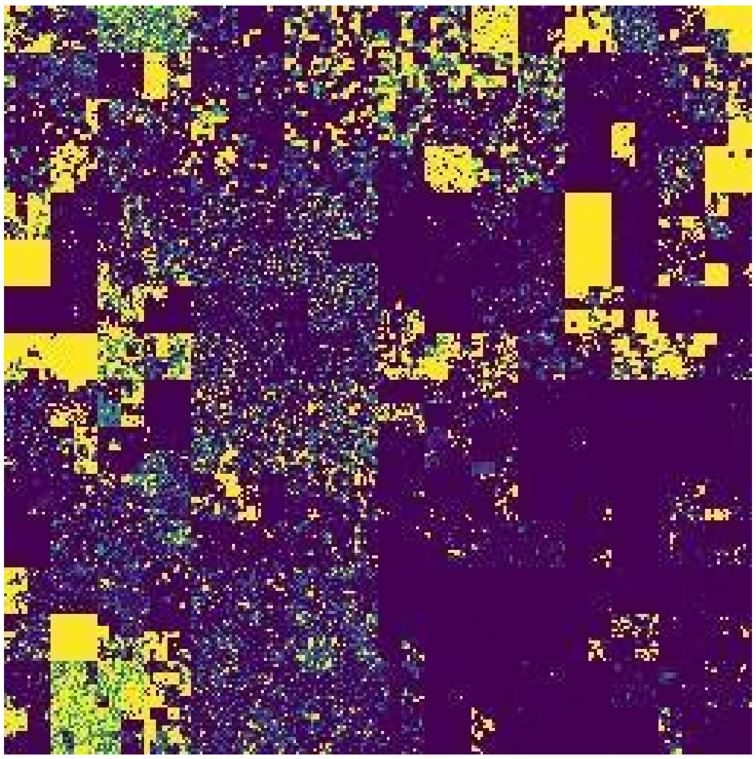
114.0.0.0/8:



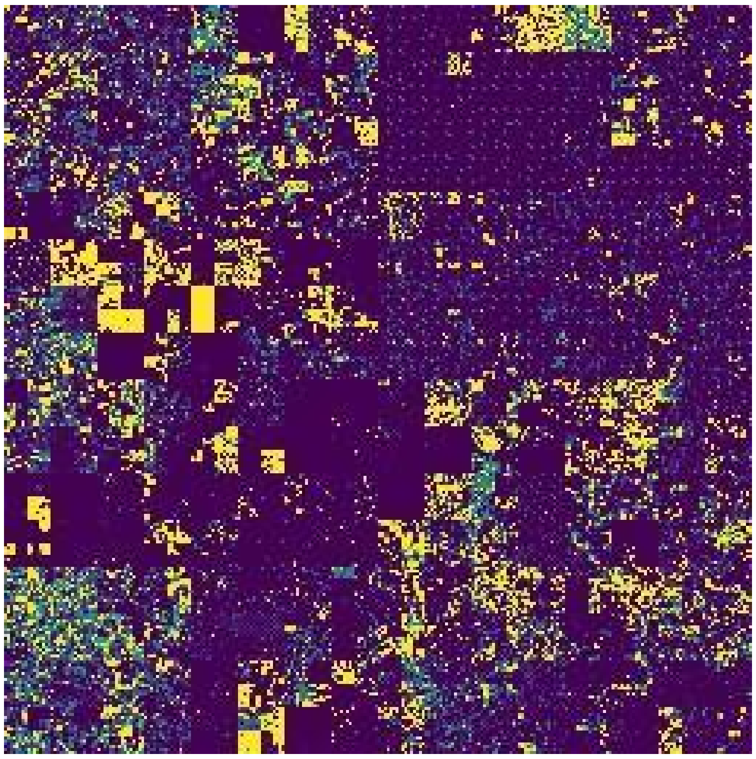
115.0.0.0/8:



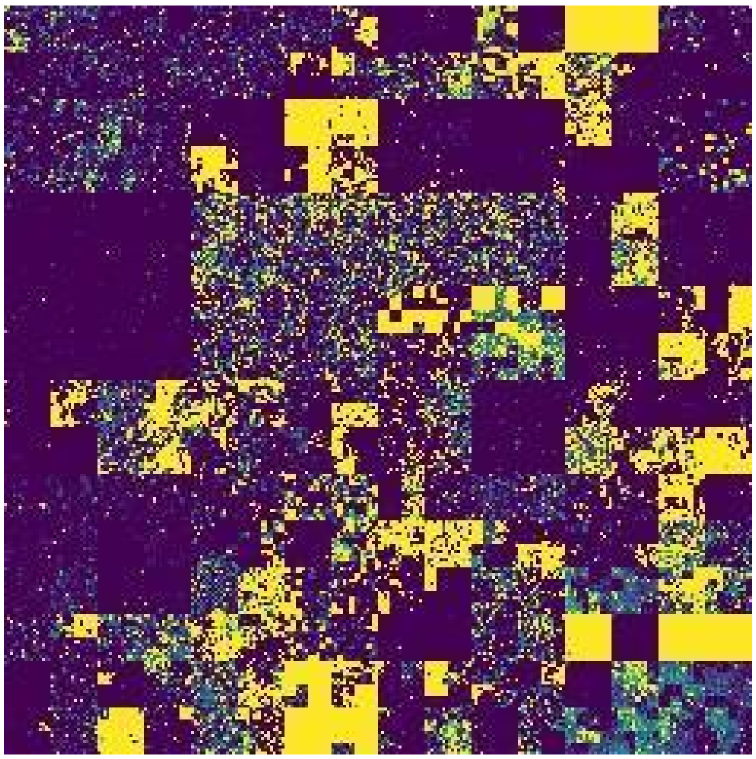
116.0.0.0/8:



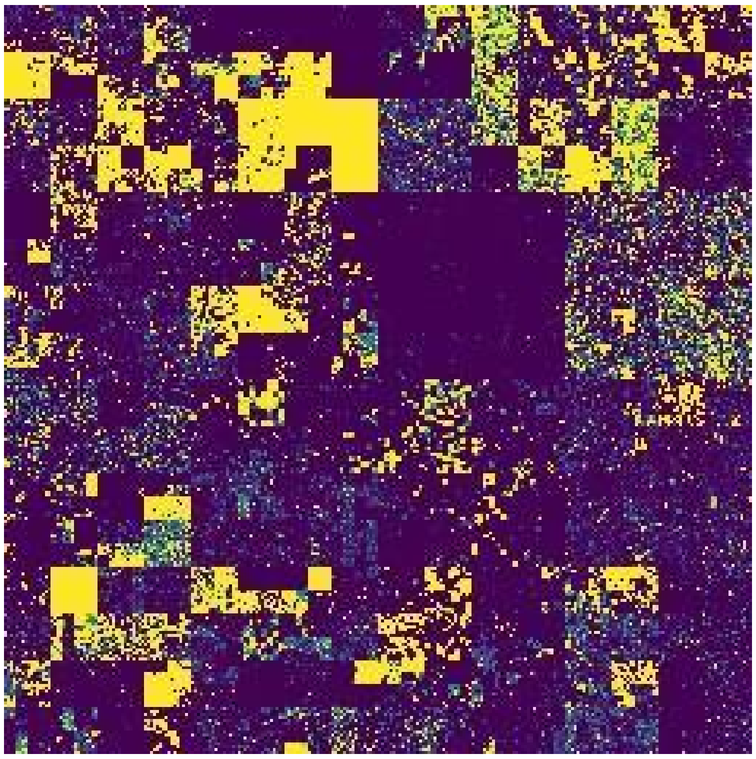
117.0.0.0/8:



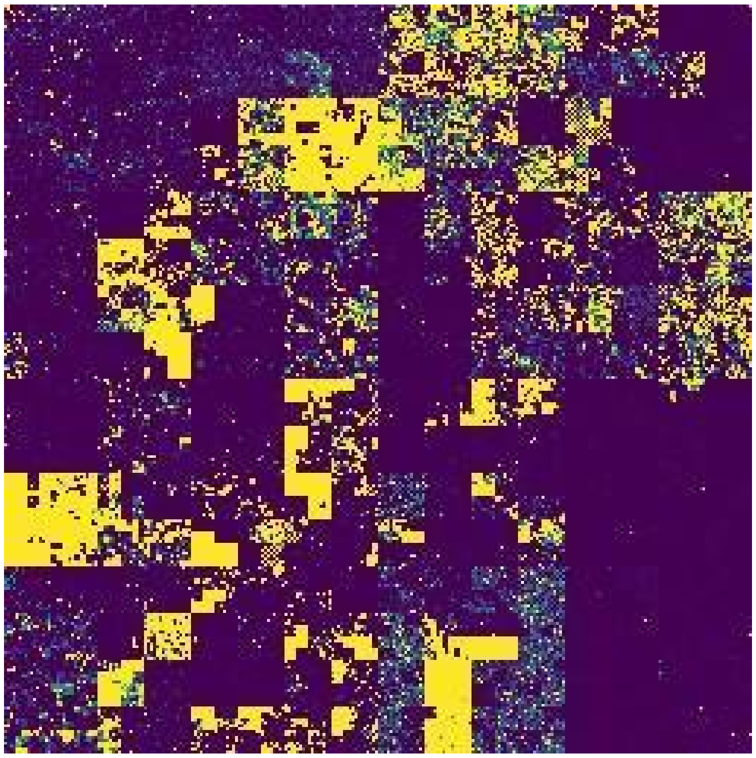
118.0.0.0/8:



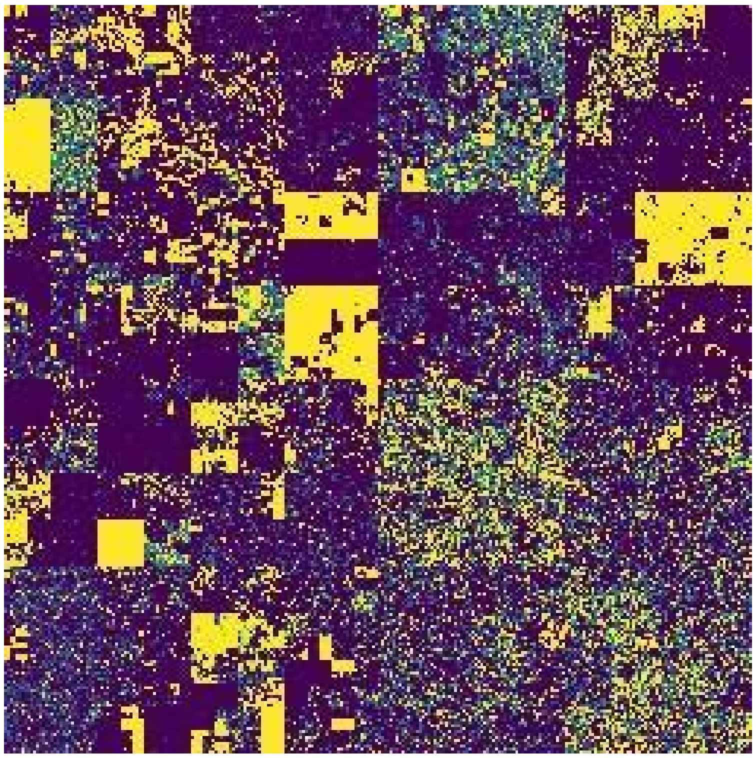
119.0.0.0/8:



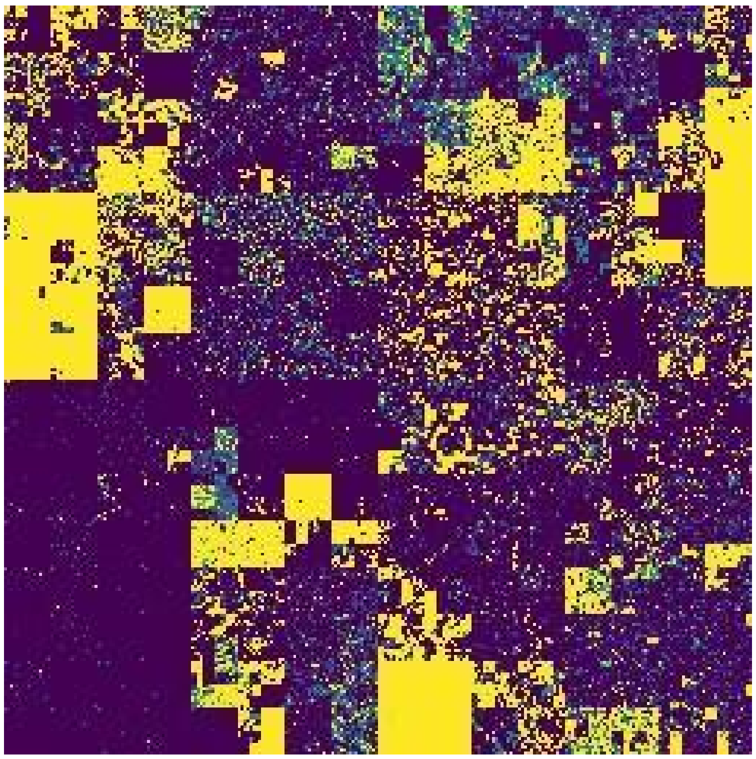
120.0.0.0/8:



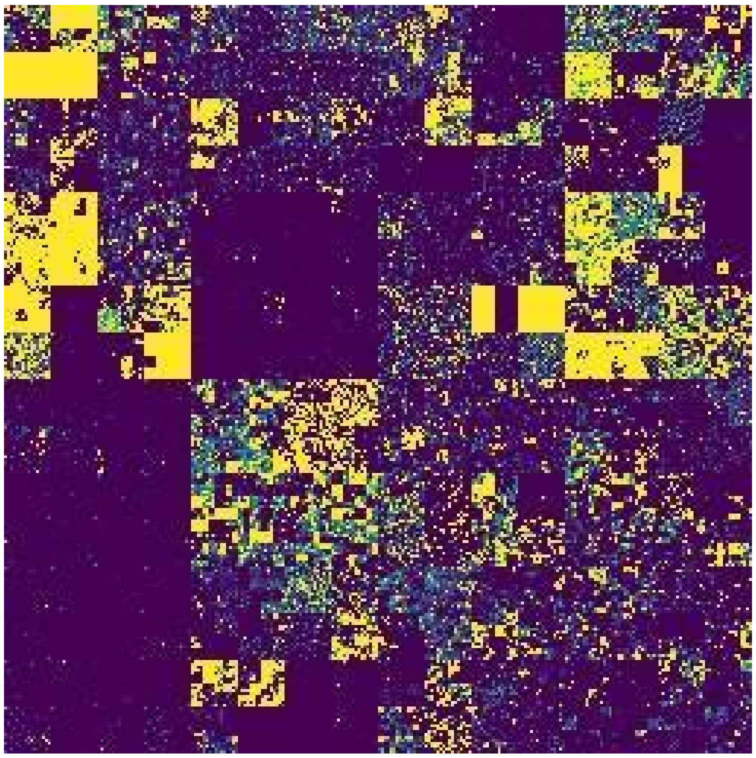
121.0.0.0/8:



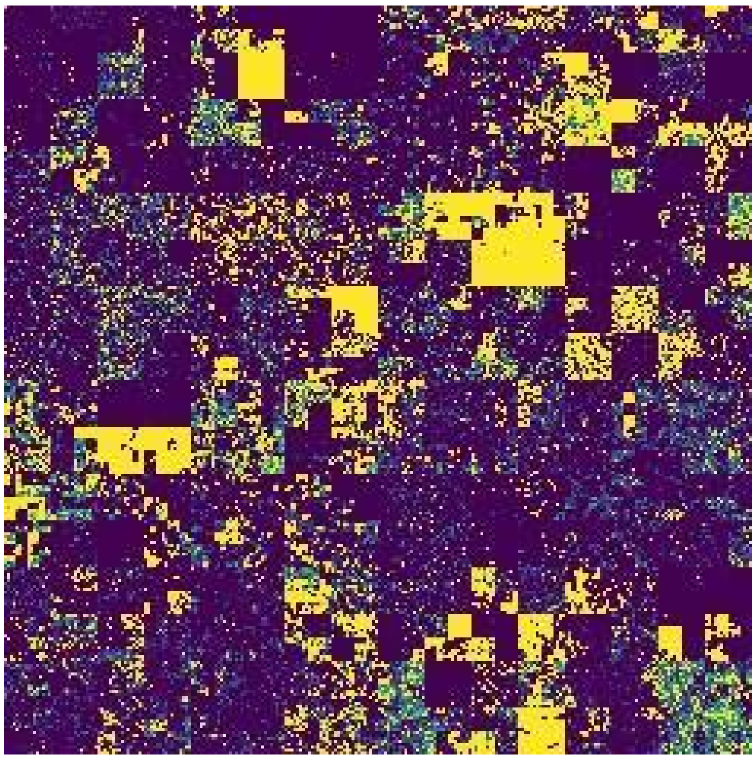
122.0.0.0/8:



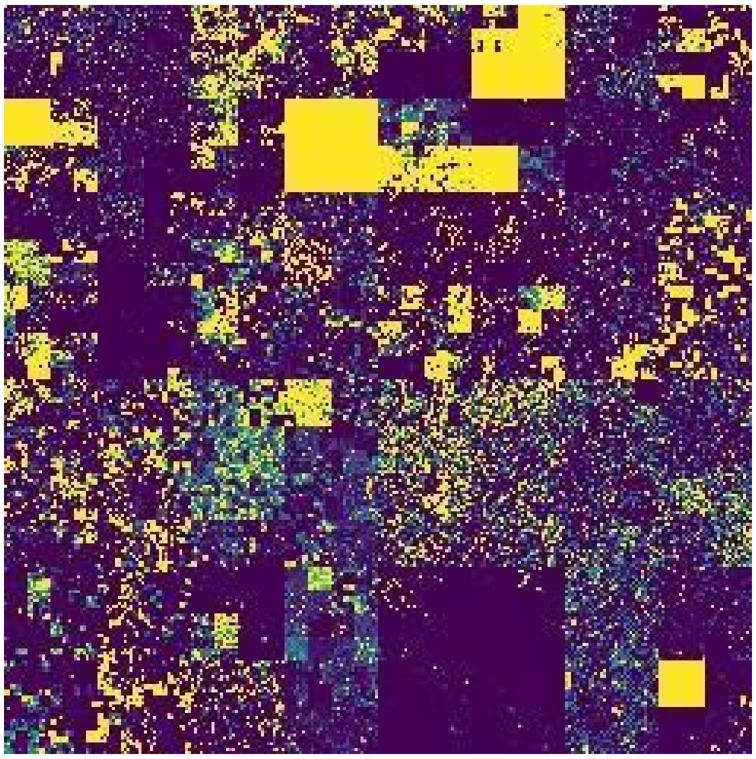
123.0.0.0/8:



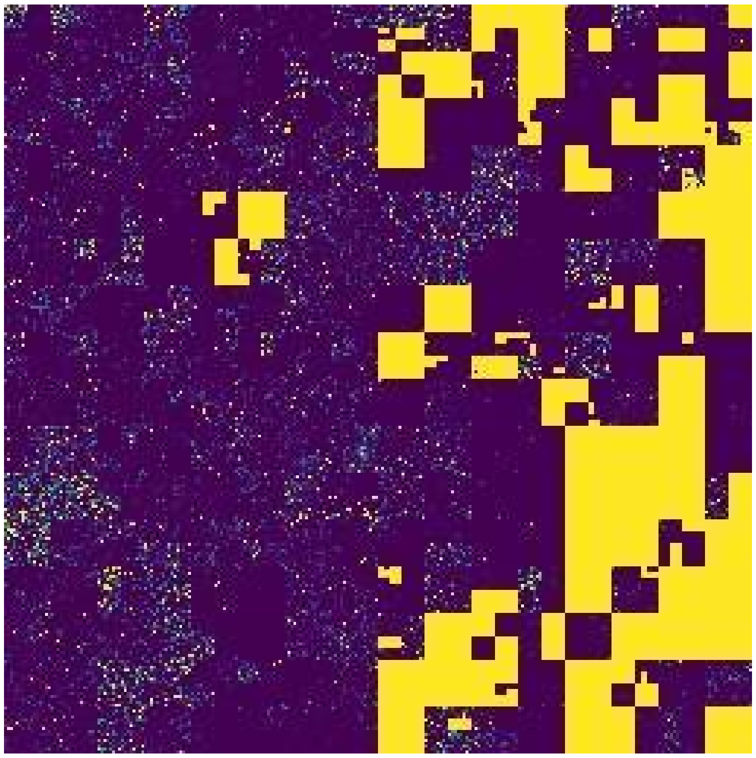
124.0.0.0/8:



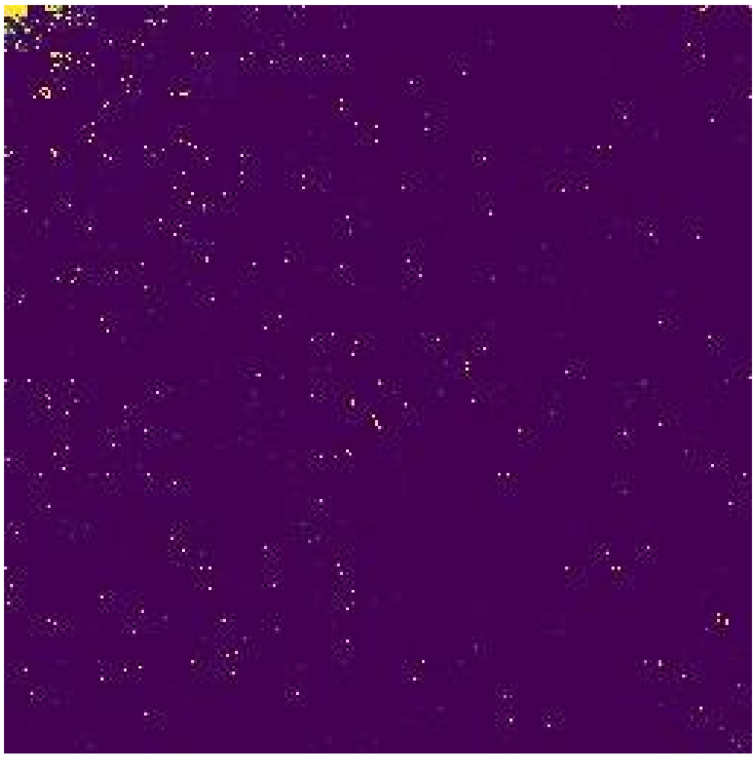
125.0.0.0/8:



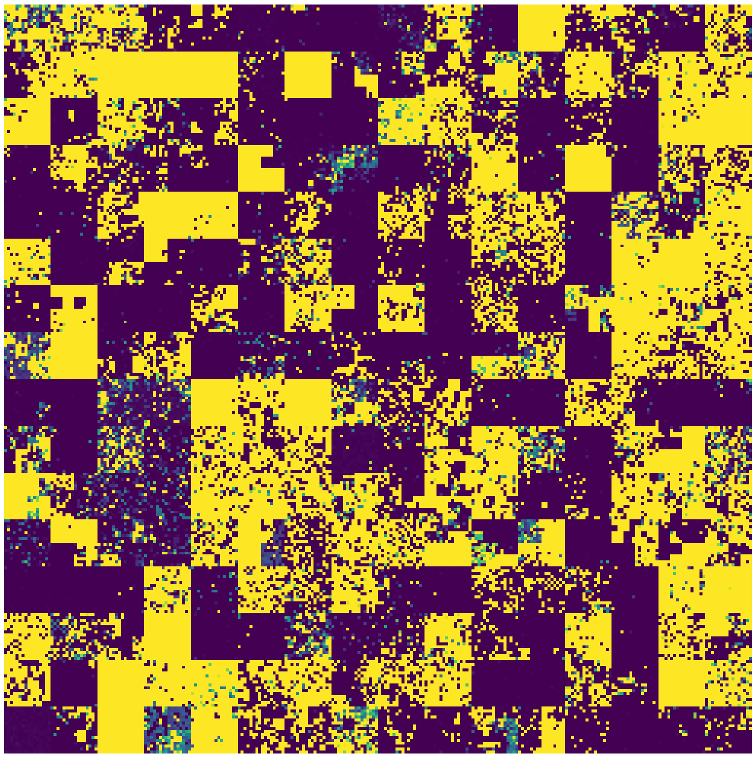
126.0.0.0/8:



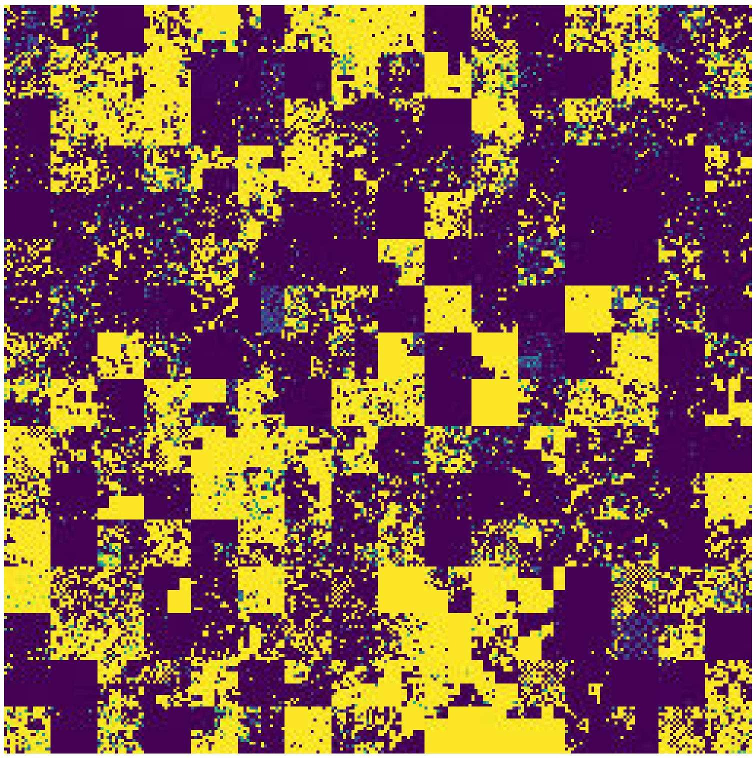
127.0.0.0/8: Localhost



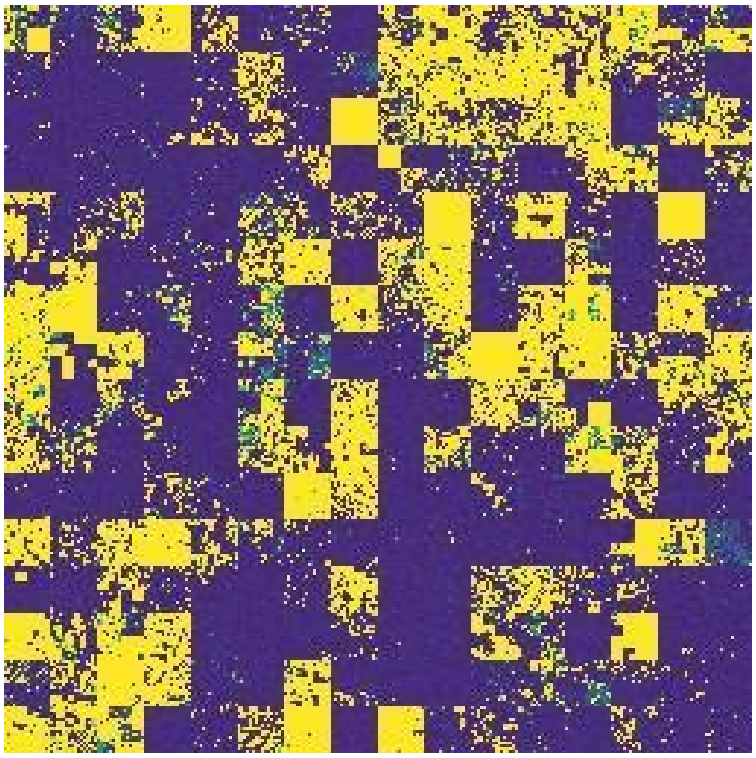
128.0.0.0/8:



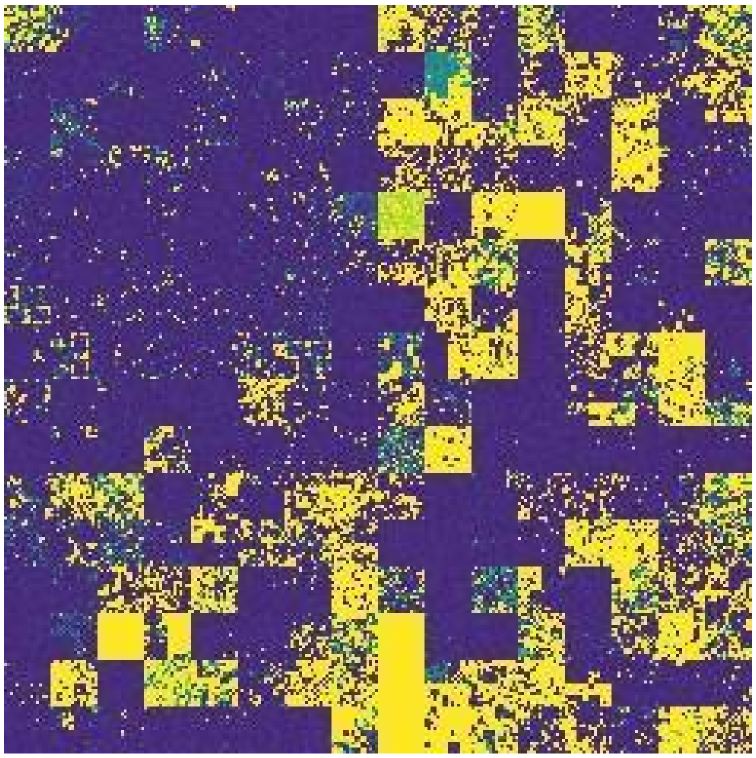
129.0.0.0/8:



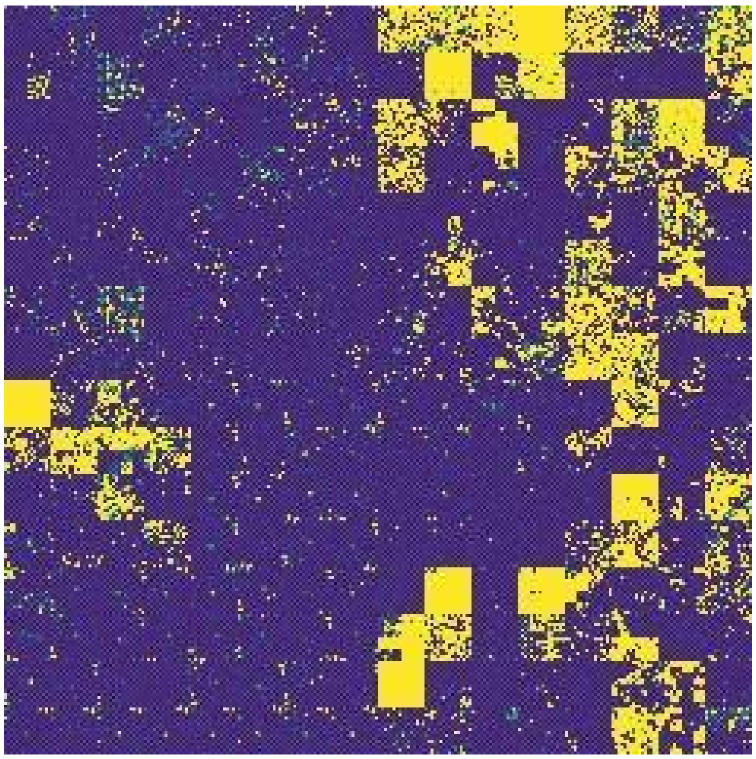
130.0.0.0/8:



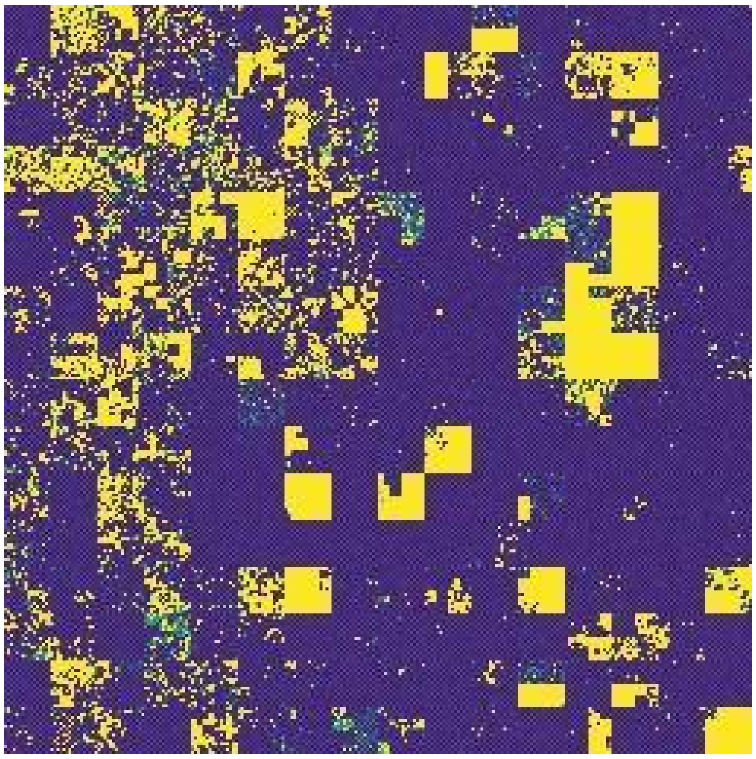
131.0.0.0/8:



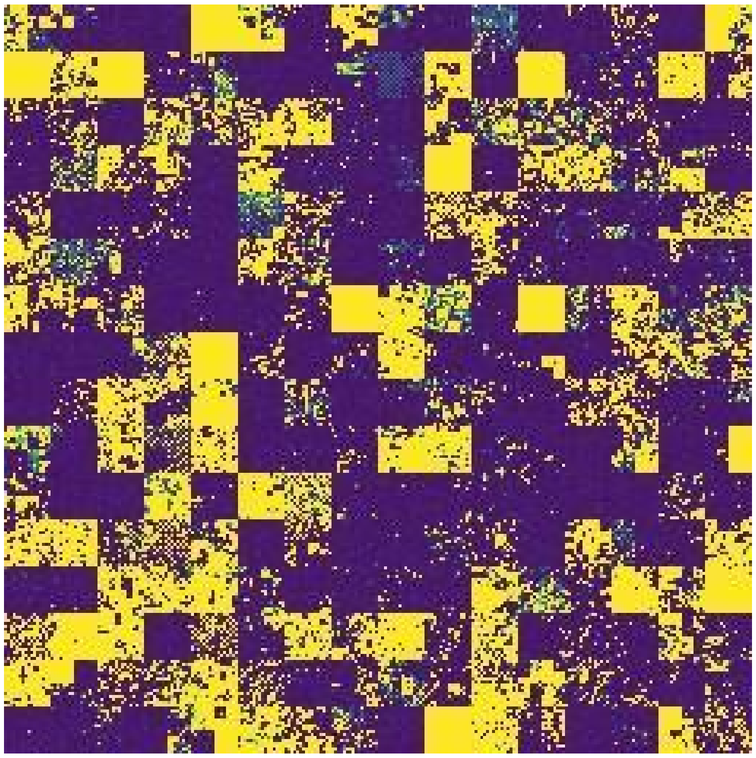
132.0.0.0/8:



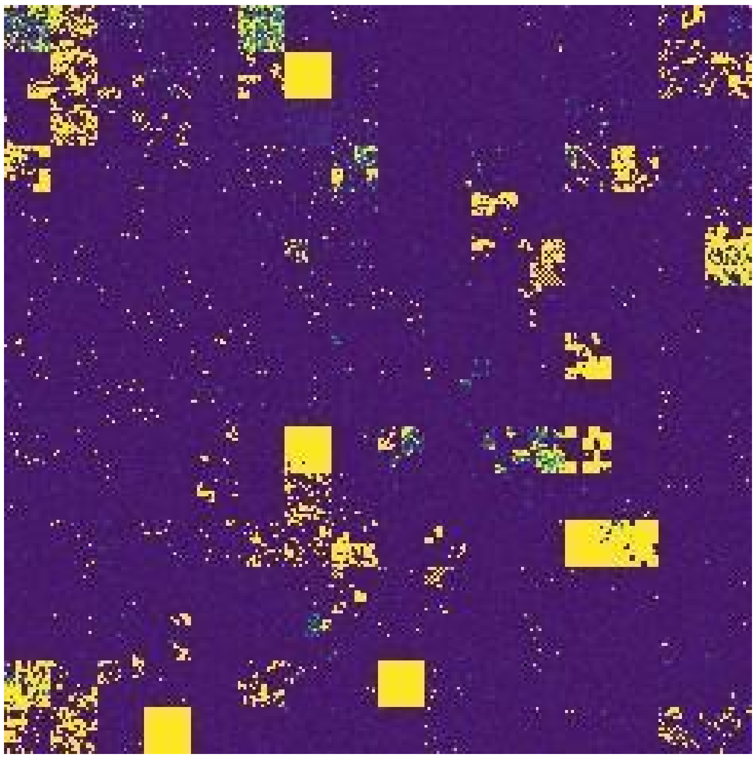
133.0.0.0/8:



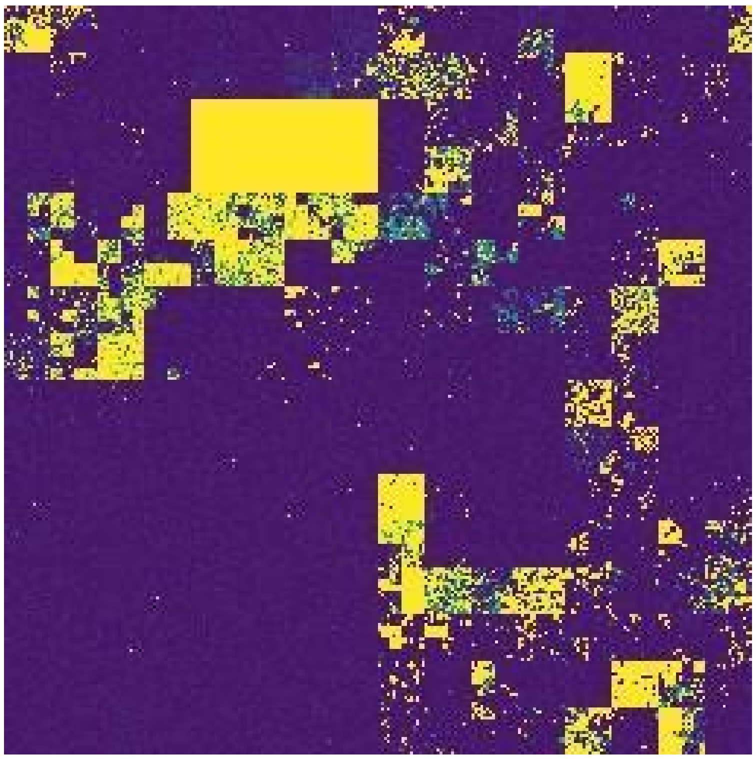
134.0.0.0/8:



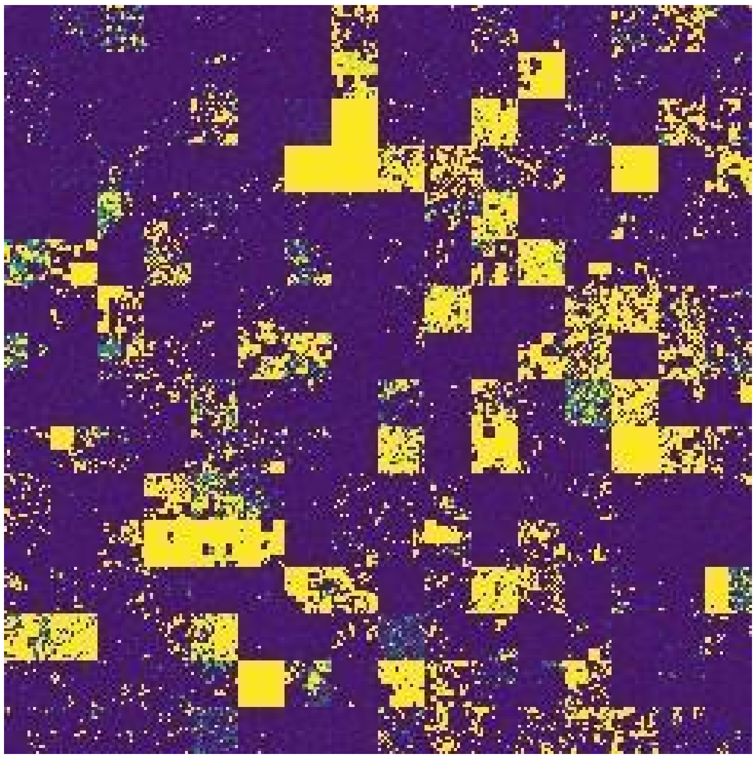
135.0.0.0/8:



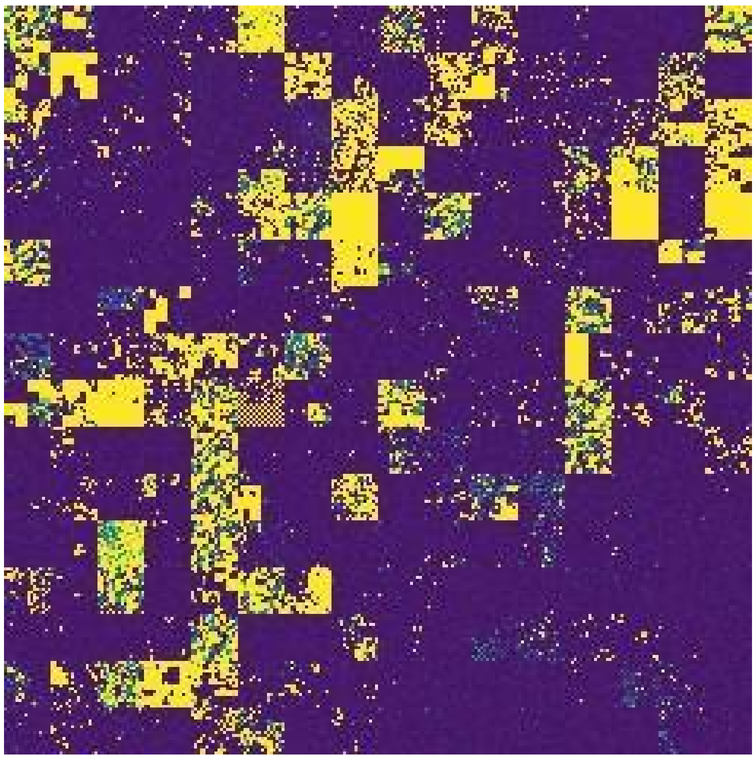
136.0.0.0/8:



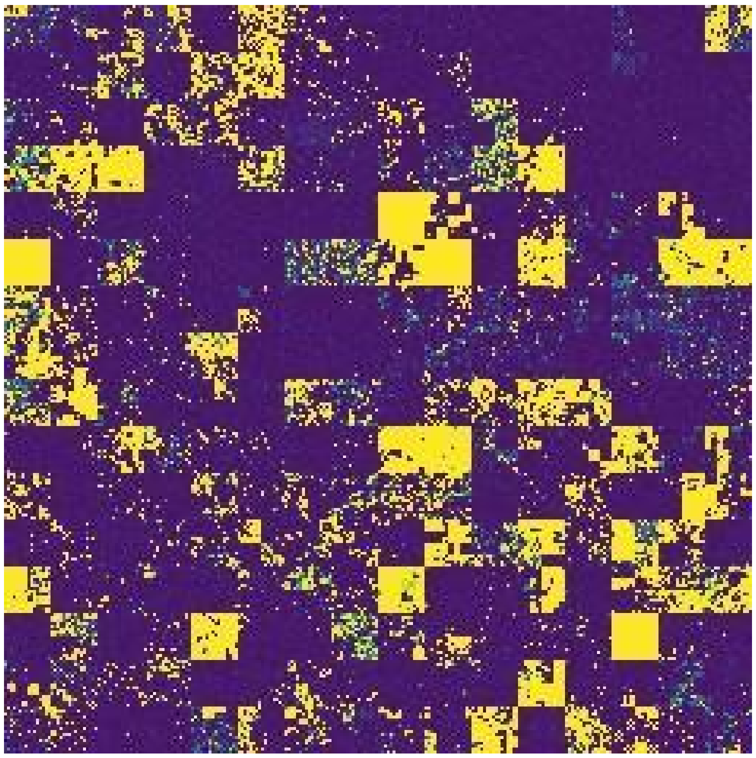
137.0.0.0/8:



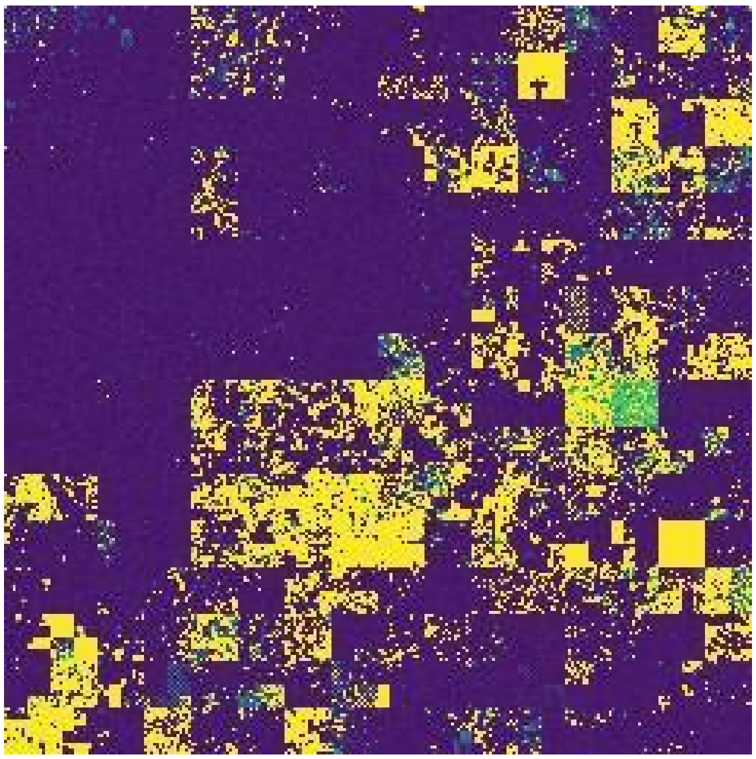
138.0.0.0/8:



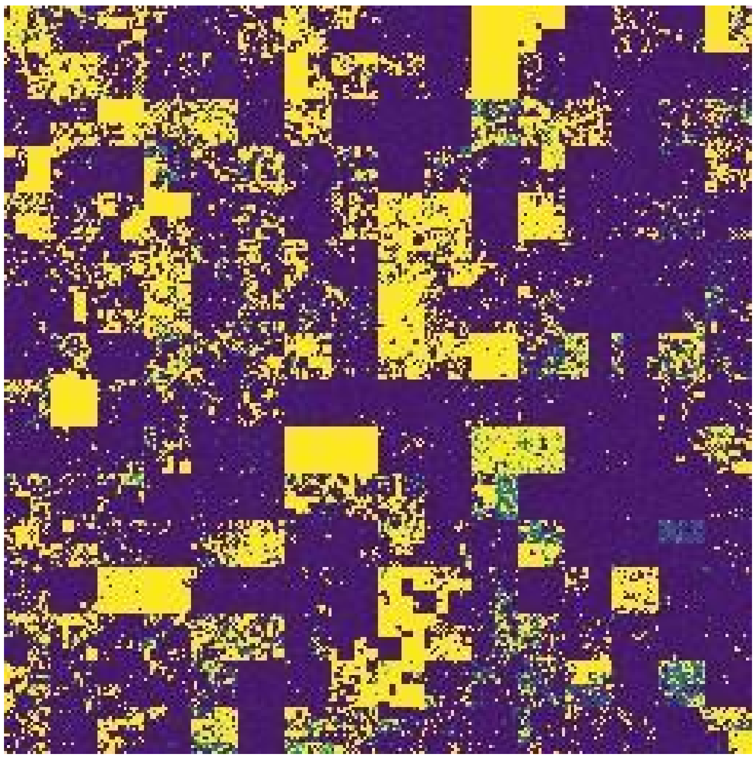
139.0.0.0/8:



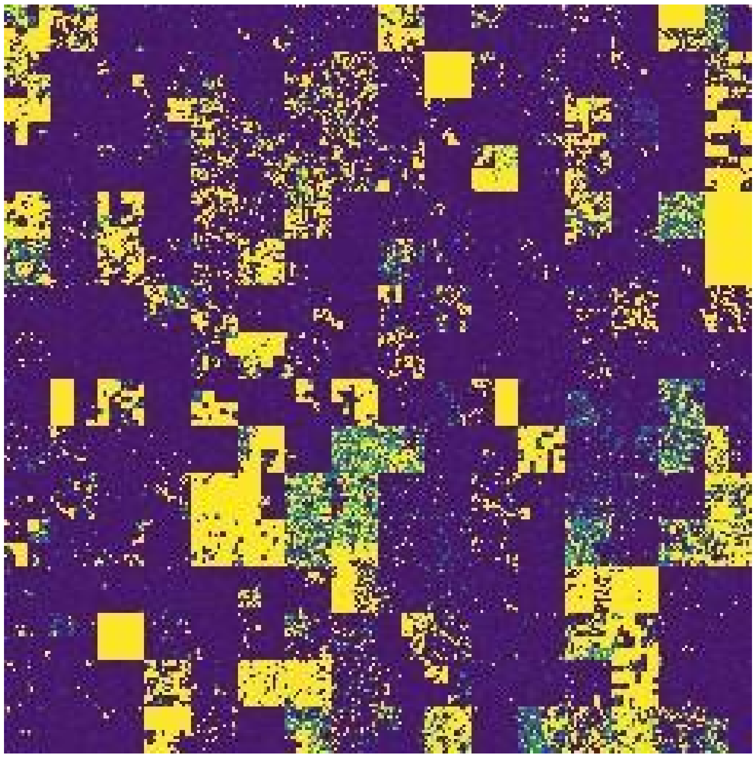
140.0.0.0/8:



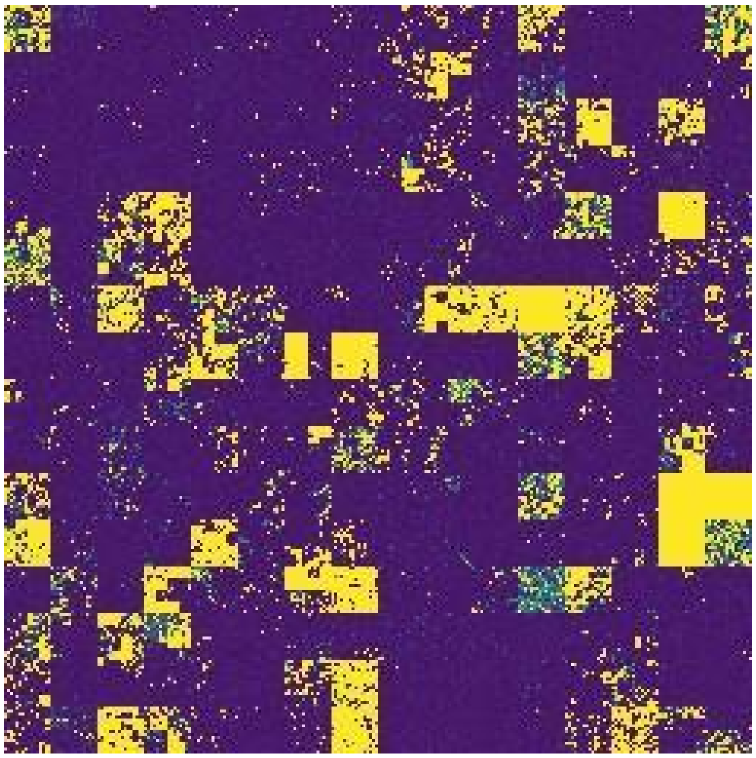
141.0.0.0/8:



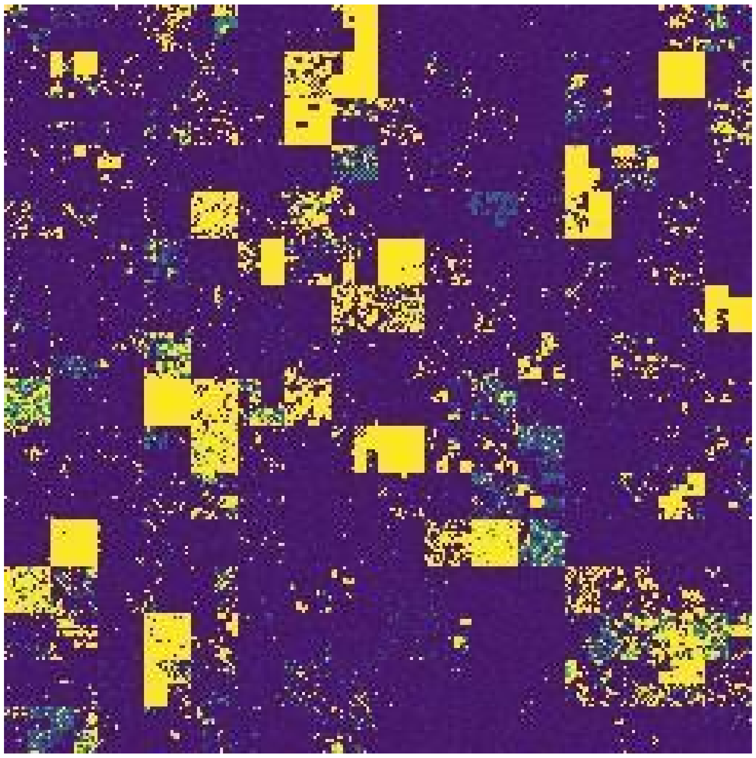
142.0.0.0/8:



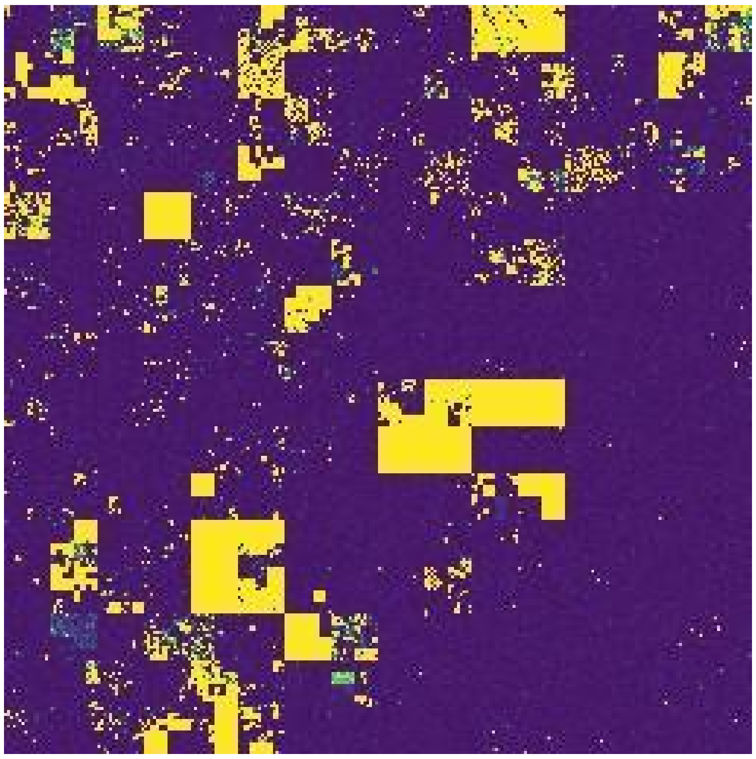
143.0.0.0/8:



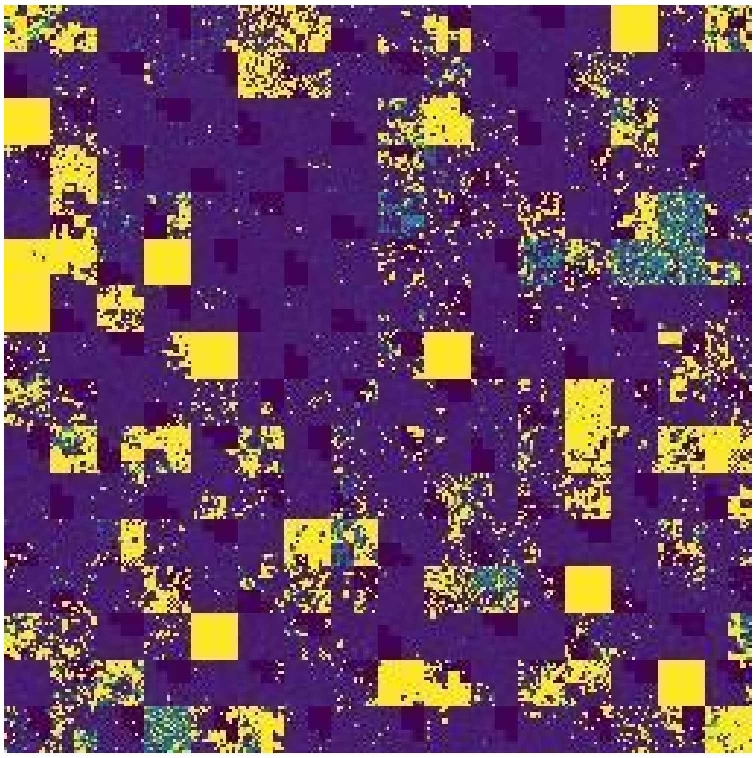
144.0.0.0/8:



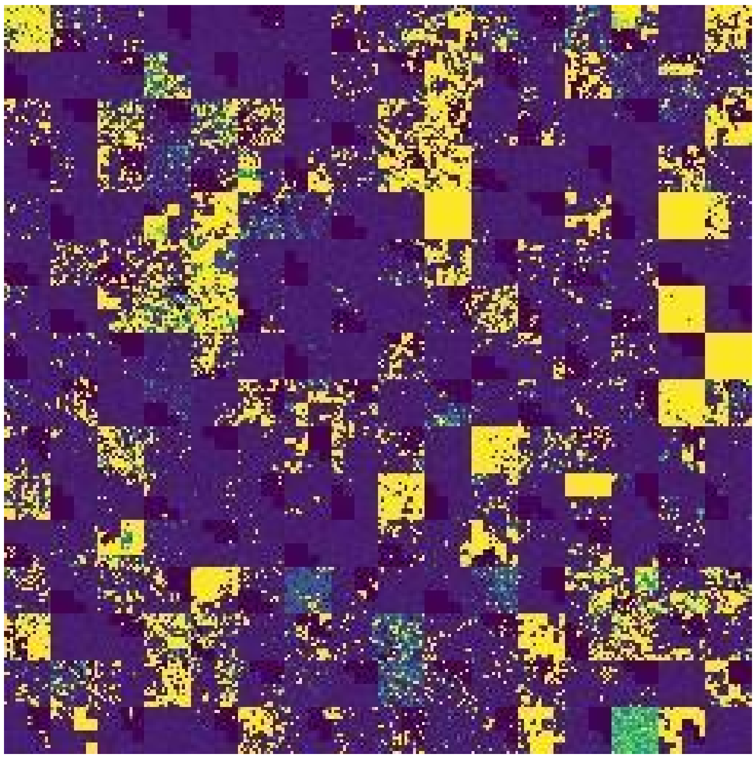
145.0.0.0/8:



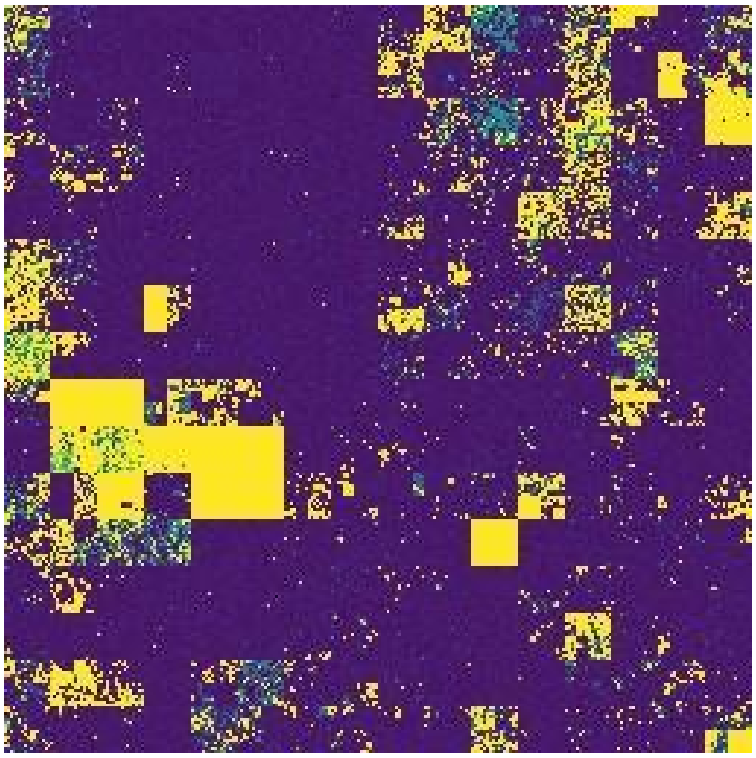
146.0.0.0/8:



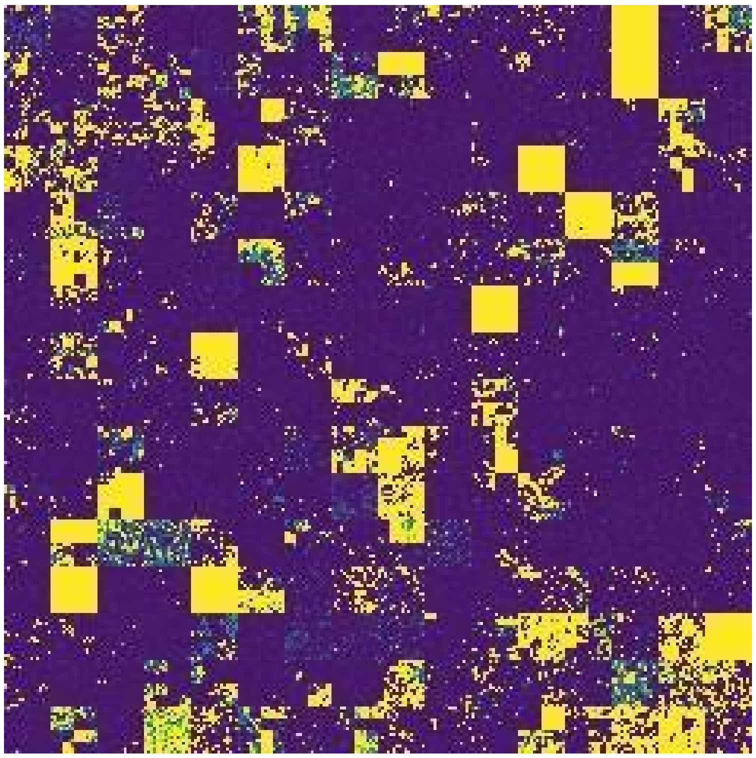
147.0.0.0/8:



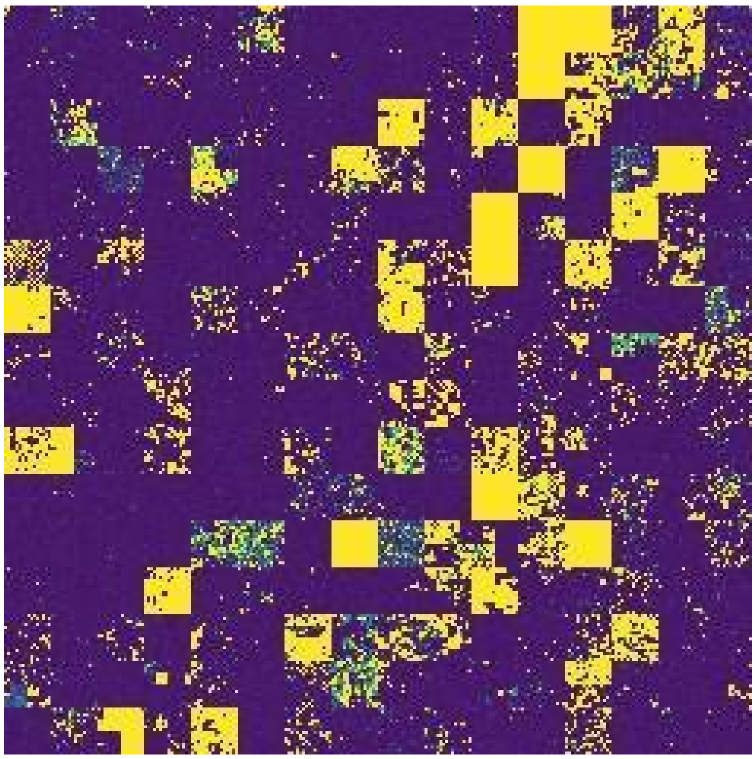
148.0.0.0/8:



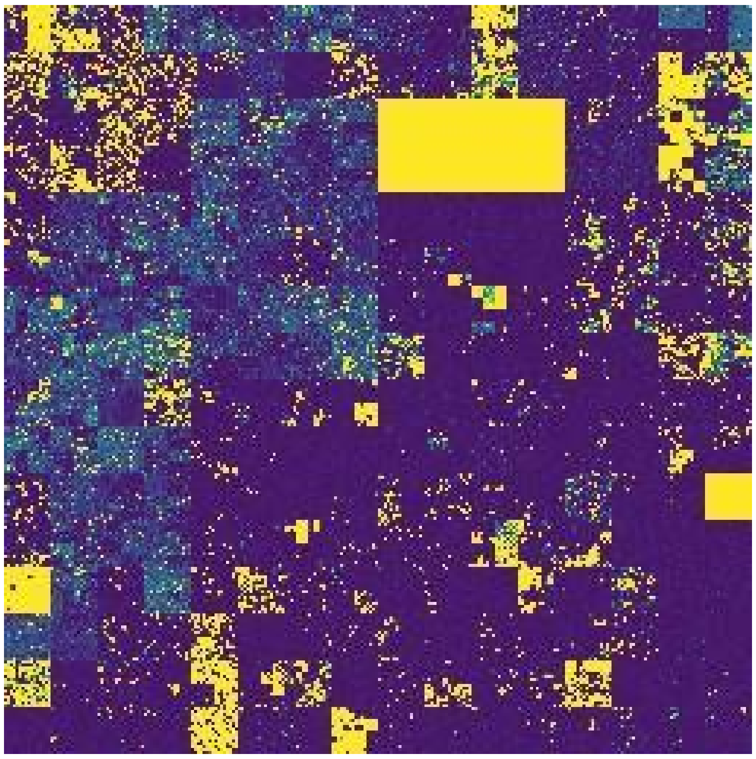
149.0.0.0/8:



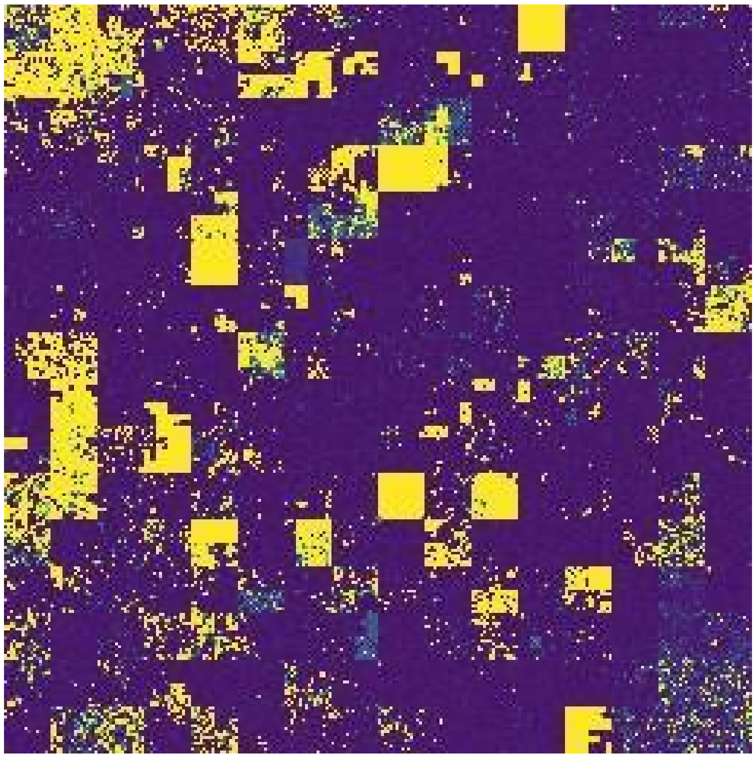
150.0.0.0/8:



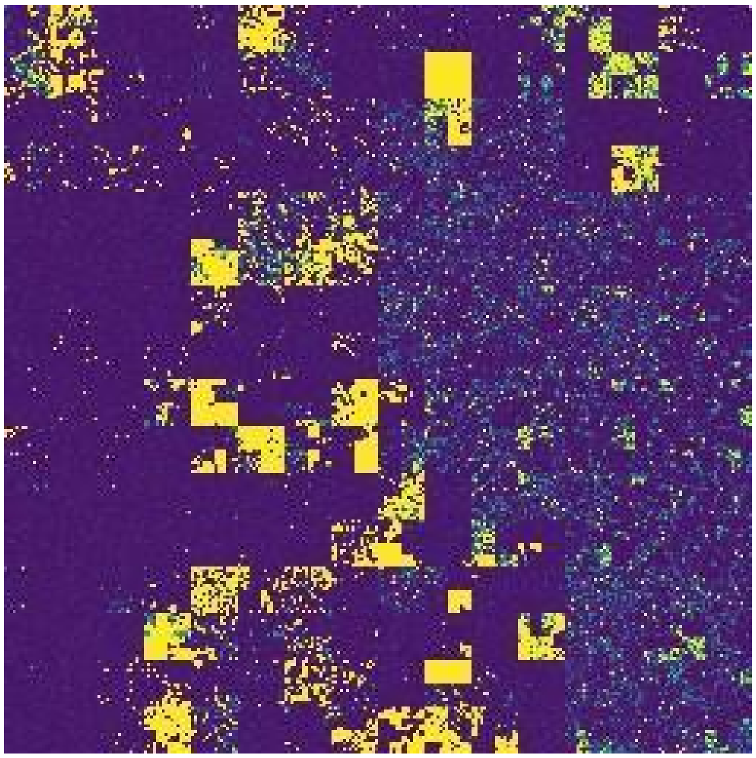
151.0.0.0/8:



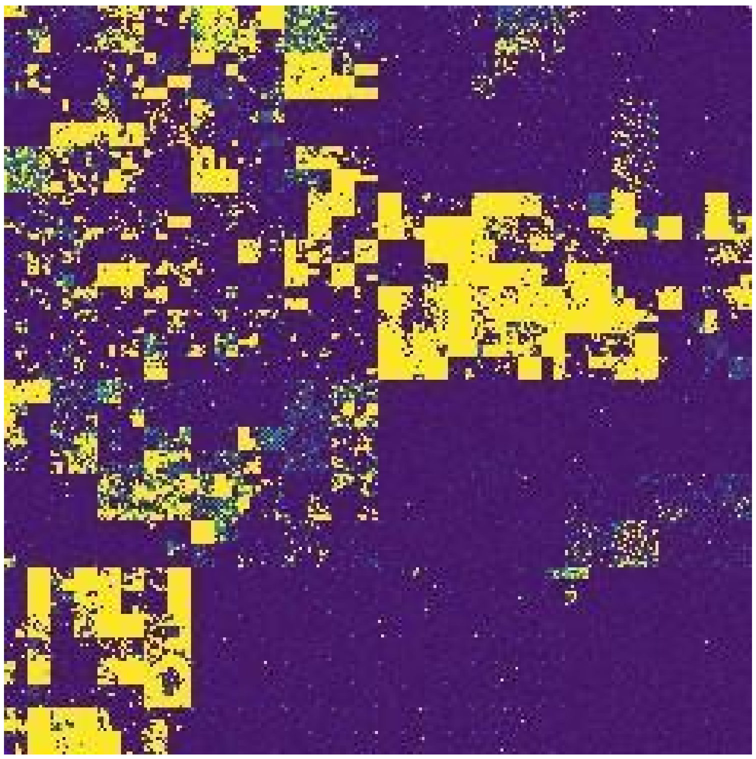
152.0.0.0/8:



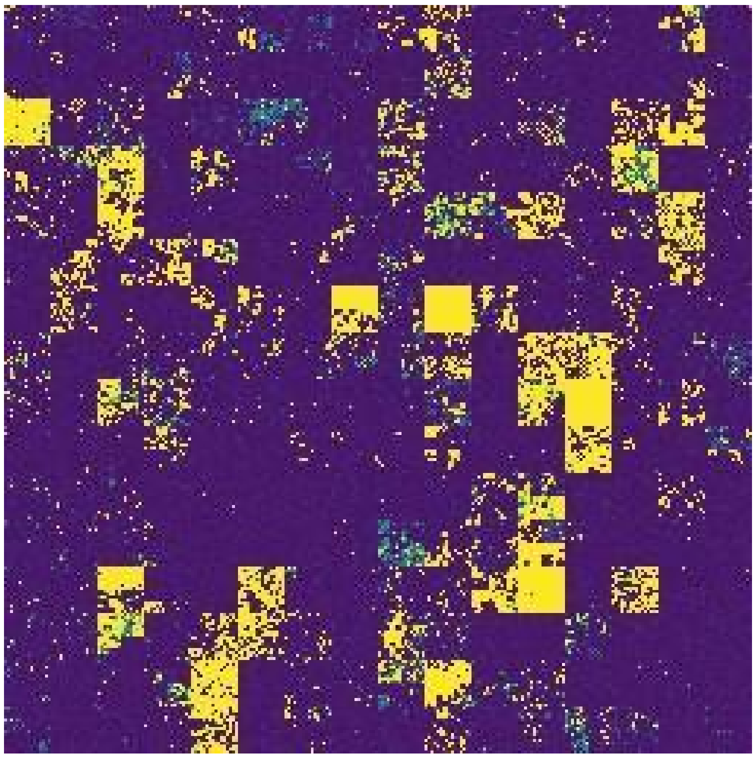
153.0.0.0/8:



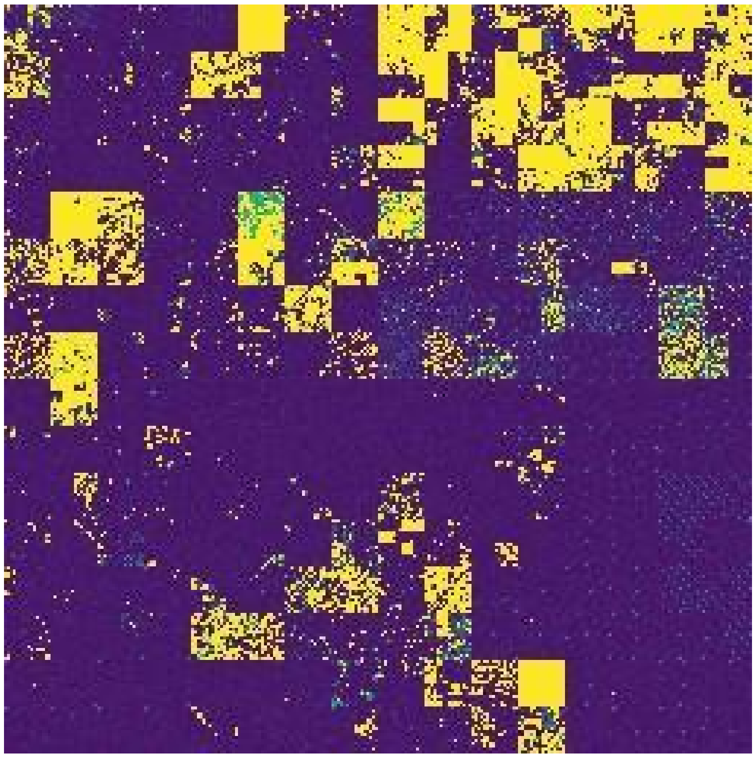
154.0.0.0/8:



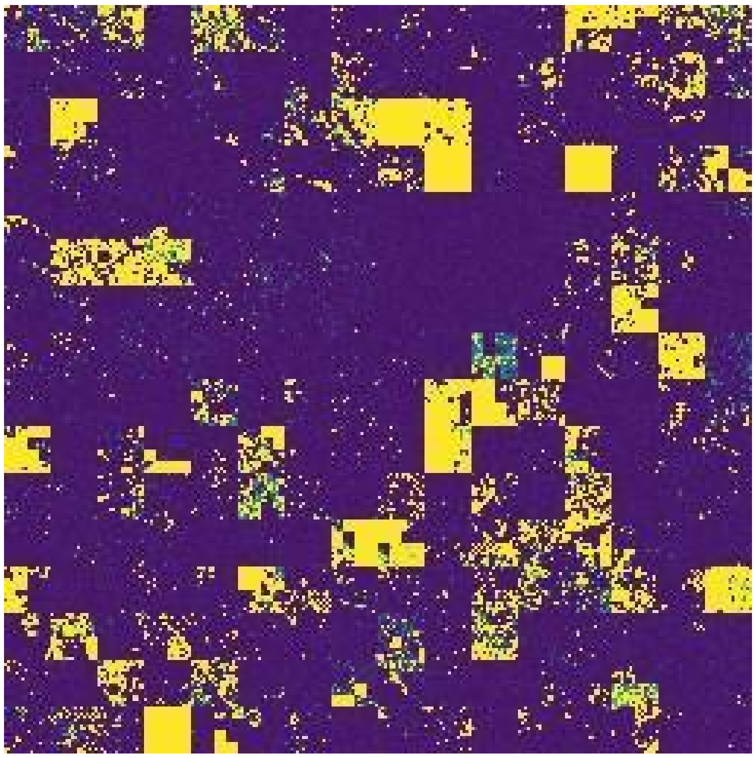
155.0.0.0/8:



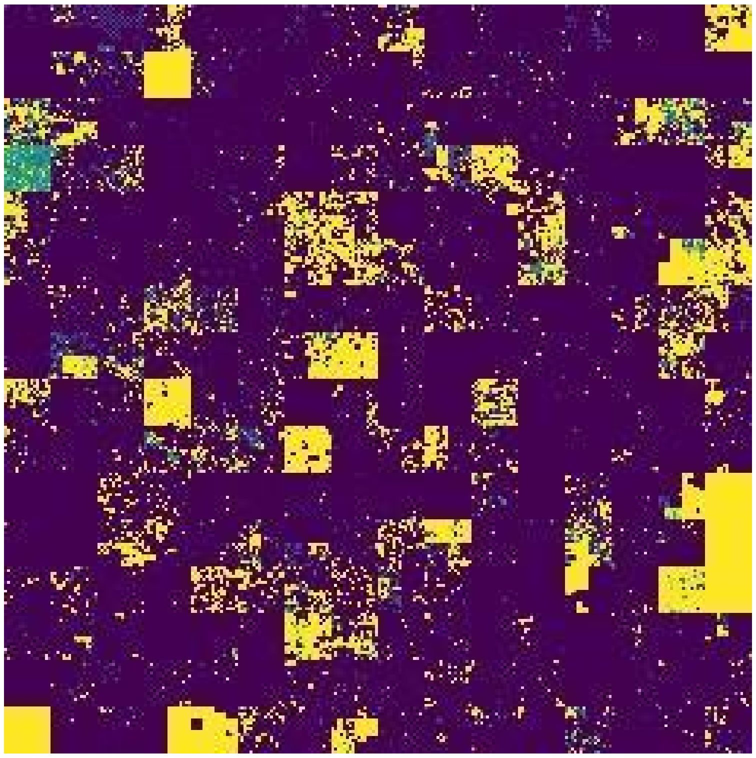
156.0.0.0/8:



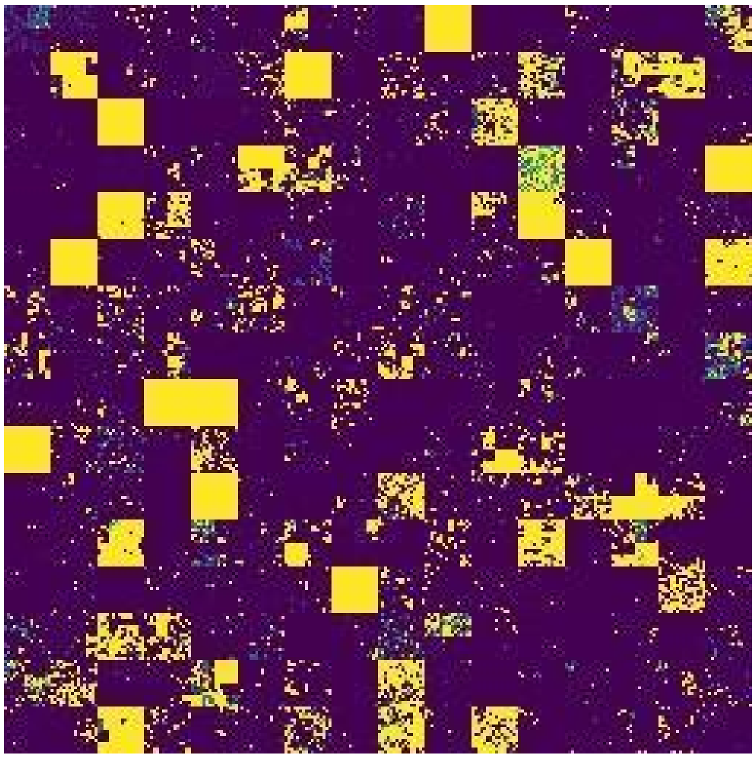
157.0.0.0/8:



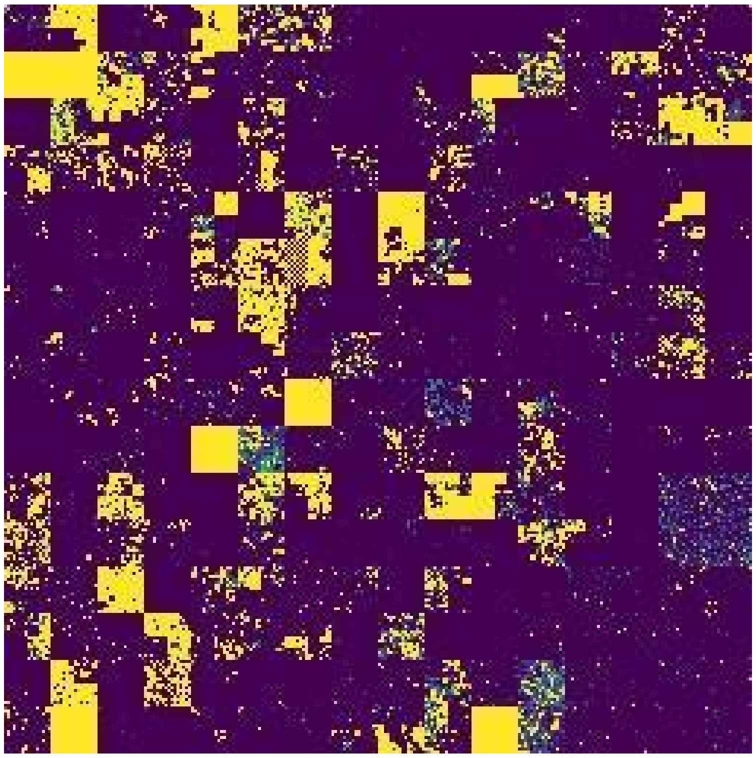
158.0.0.0/8:



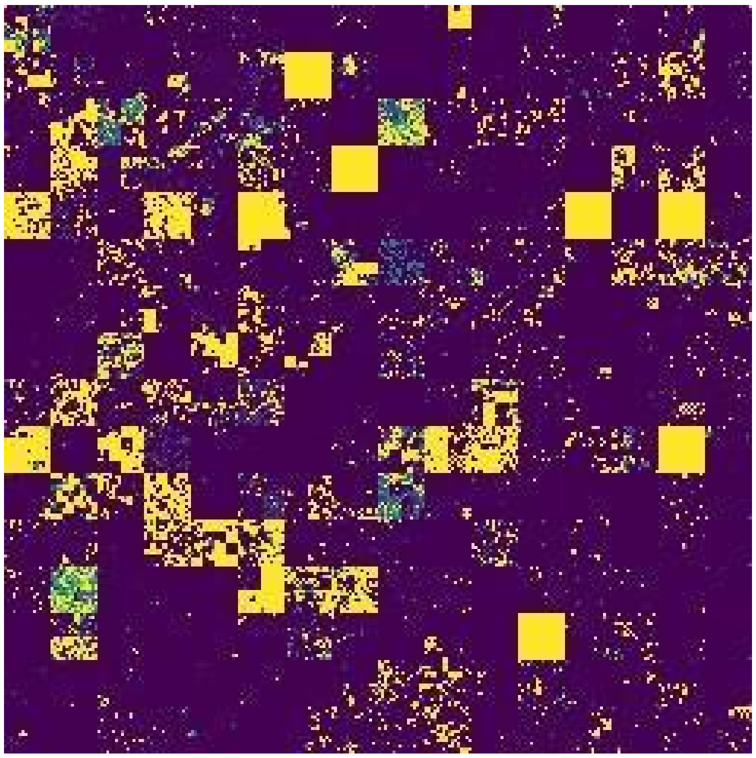
159.0.0.0/8:



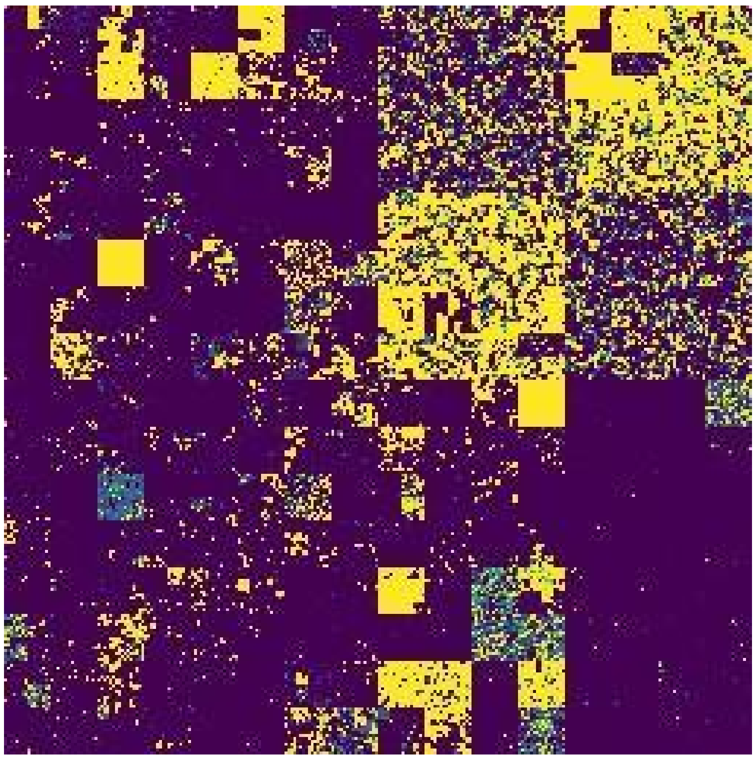
160.0.0.0/8:



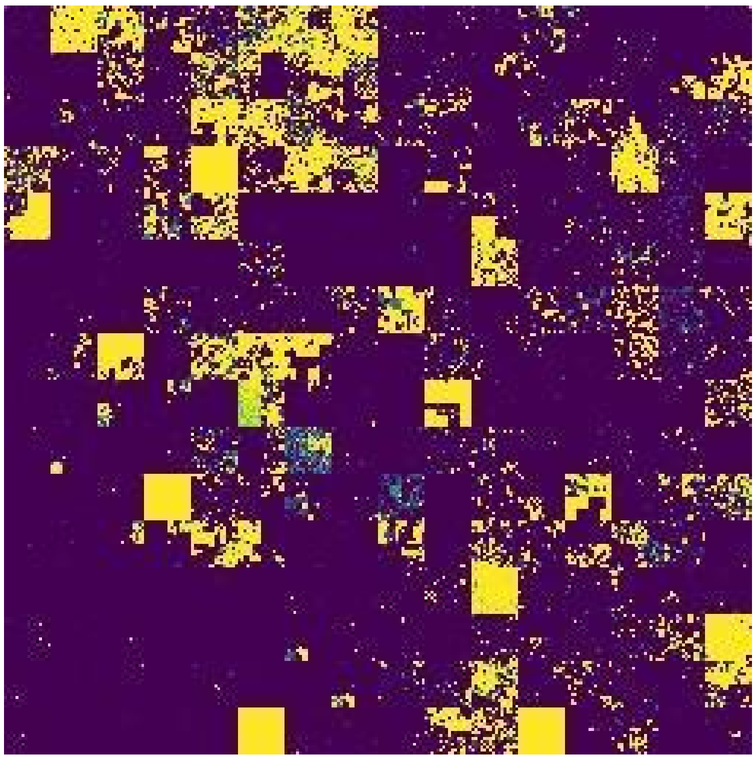
161.0.0.0/8:



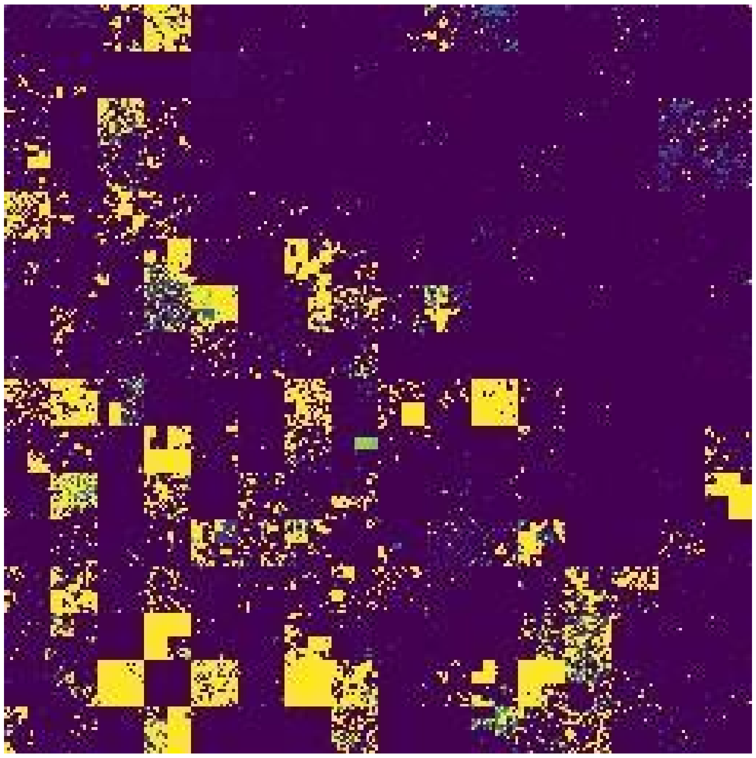
162.0.0.0/8:



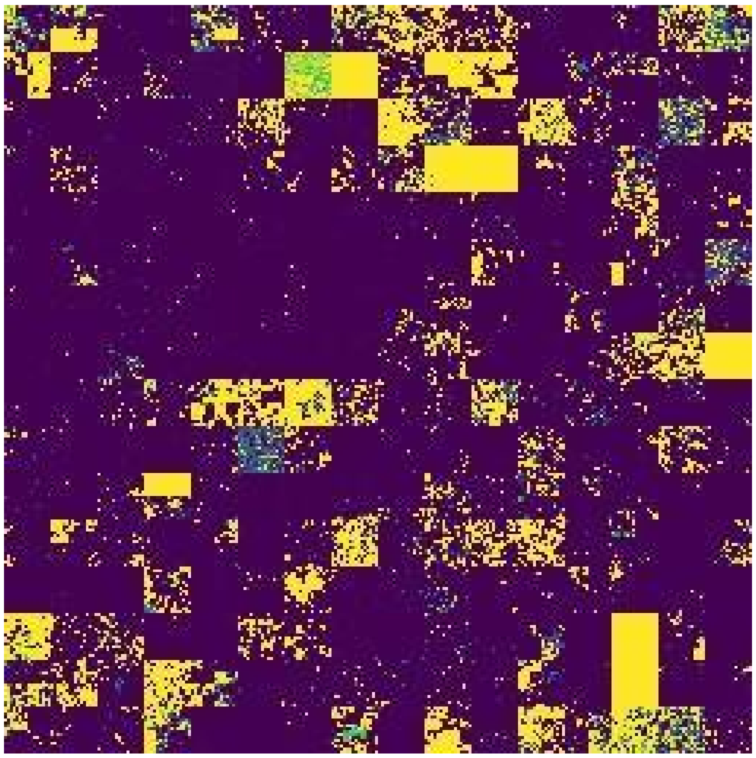
163.0.0.0/8:



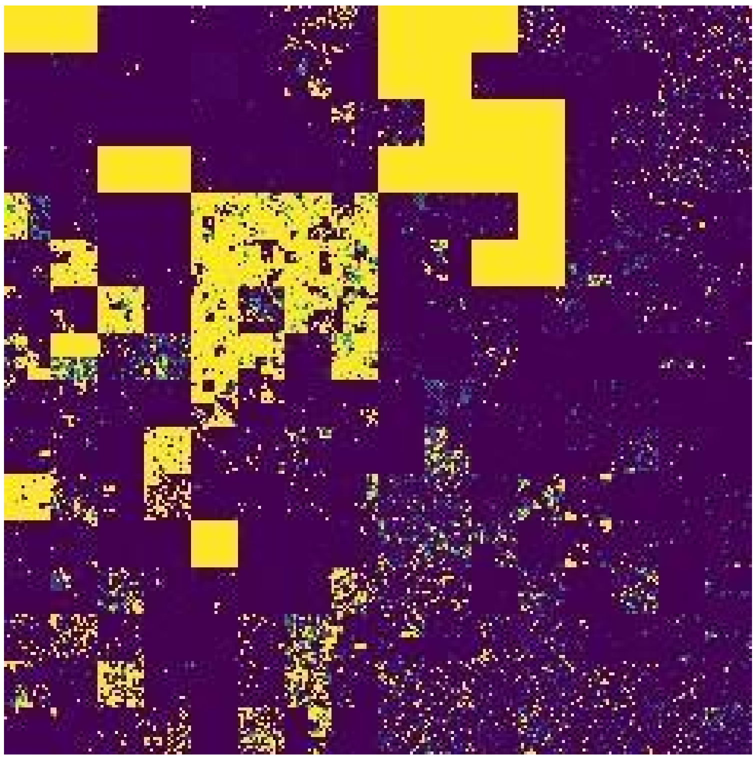
164.0.0.0/8:



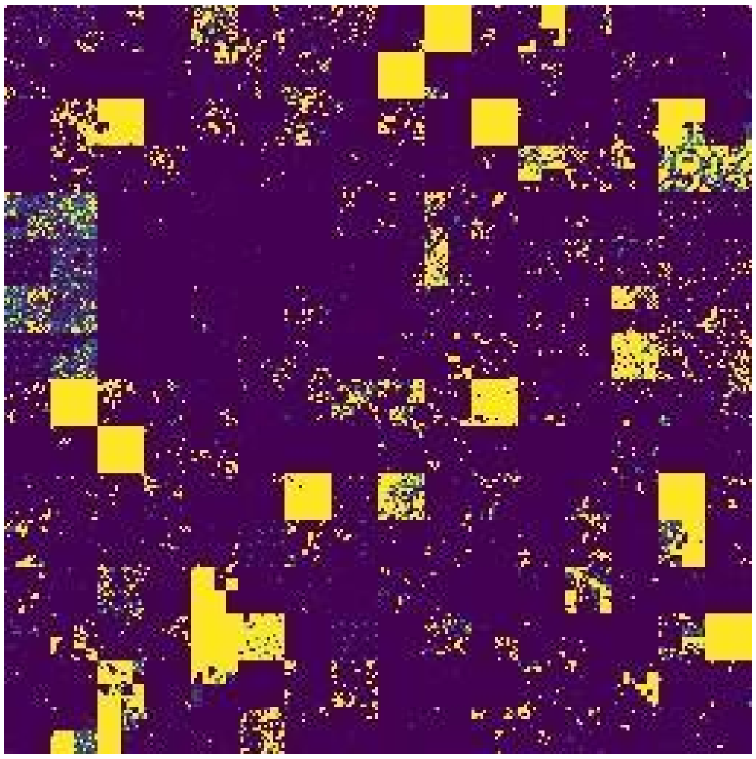
165.0.0.0/8:



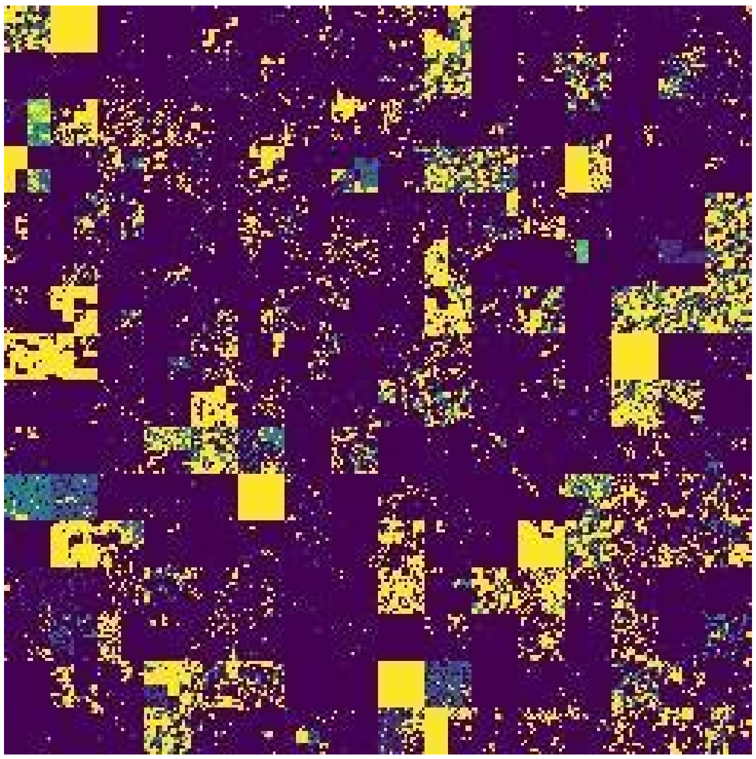
166.0.0.0/8:



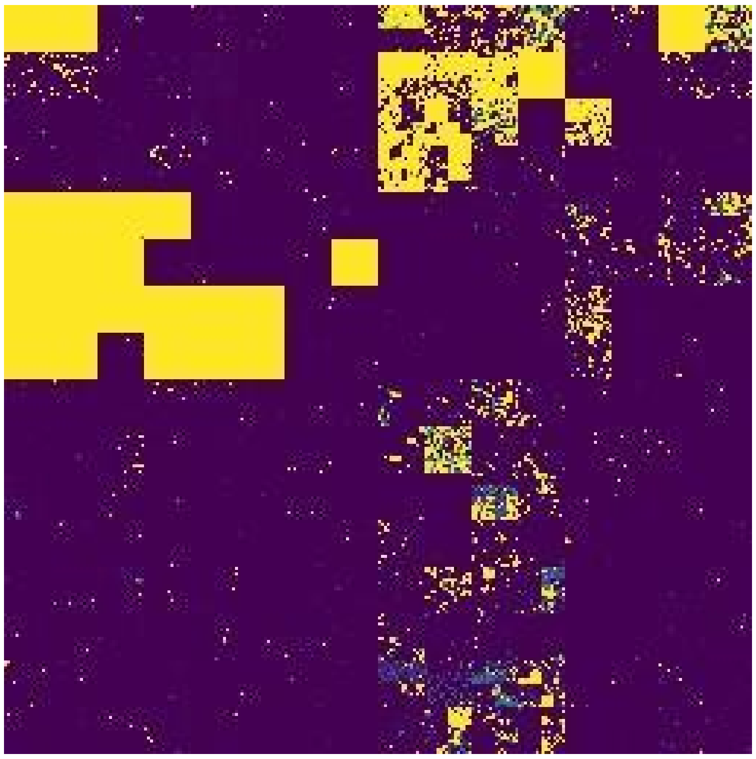
167.0.0.0/8:



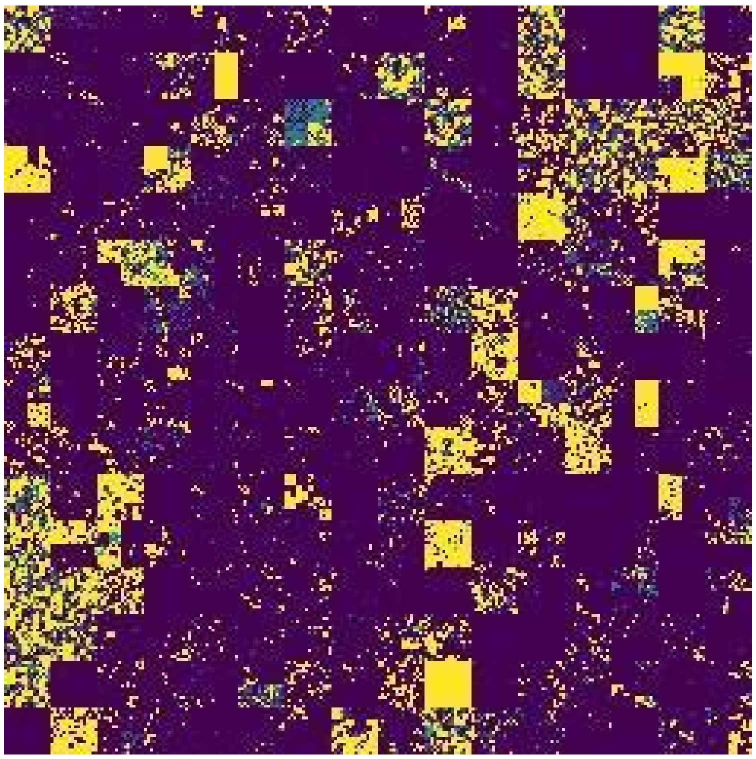
168.0.0.0/8:



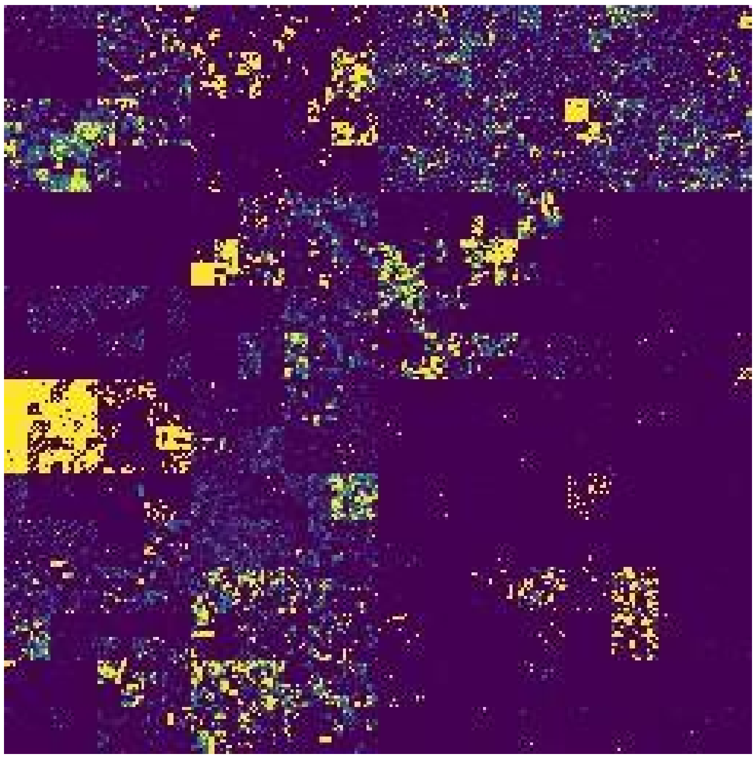
169.0.0.0/8:



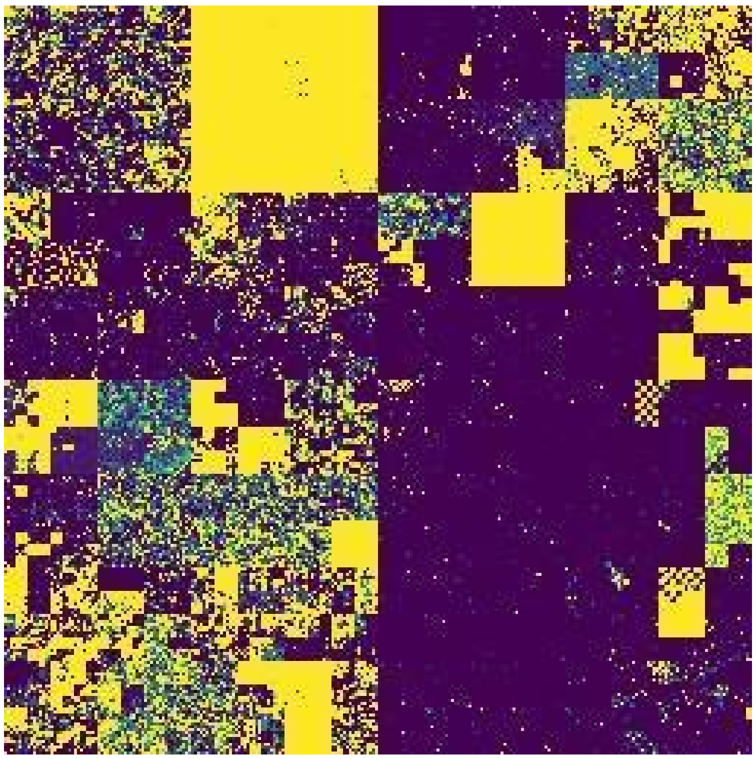
170.0.0.0/8:



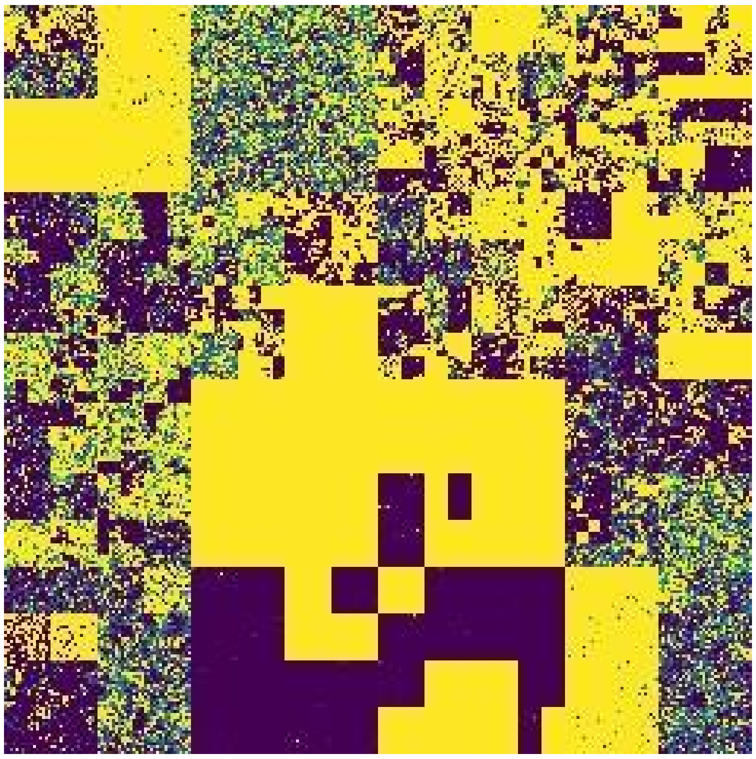
171.0.0.0/8:



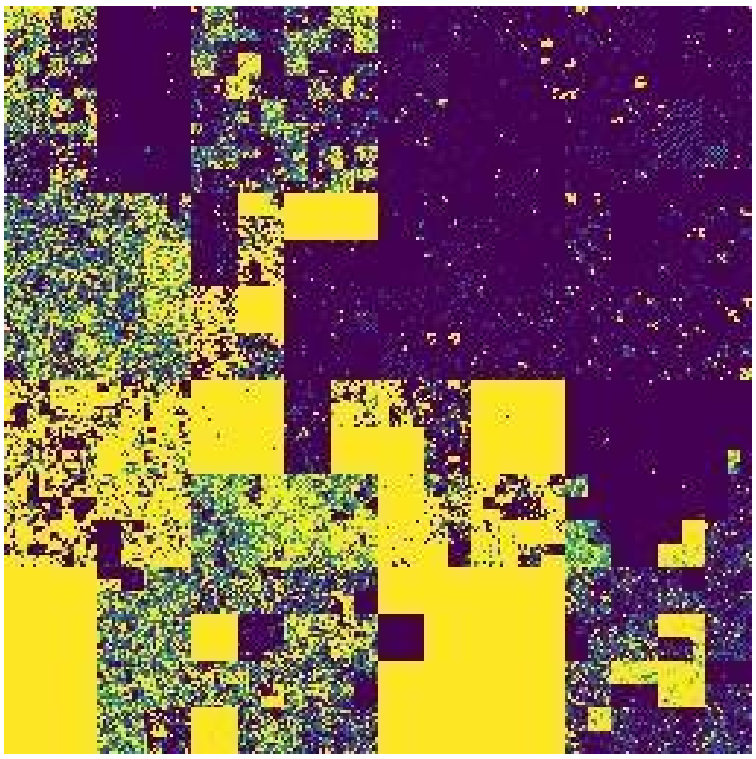
172.0.0.0/8:



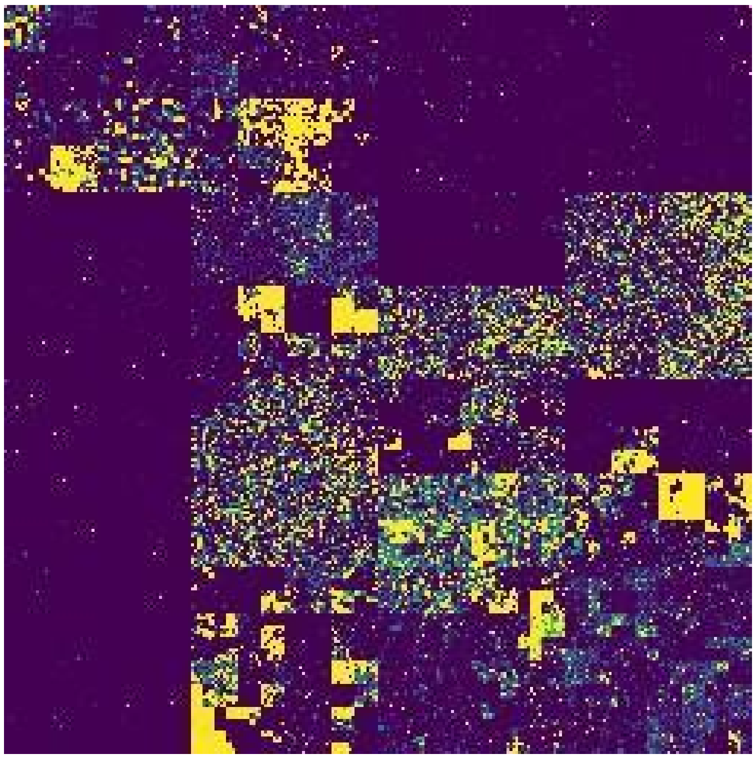
173.0.0.0/8:



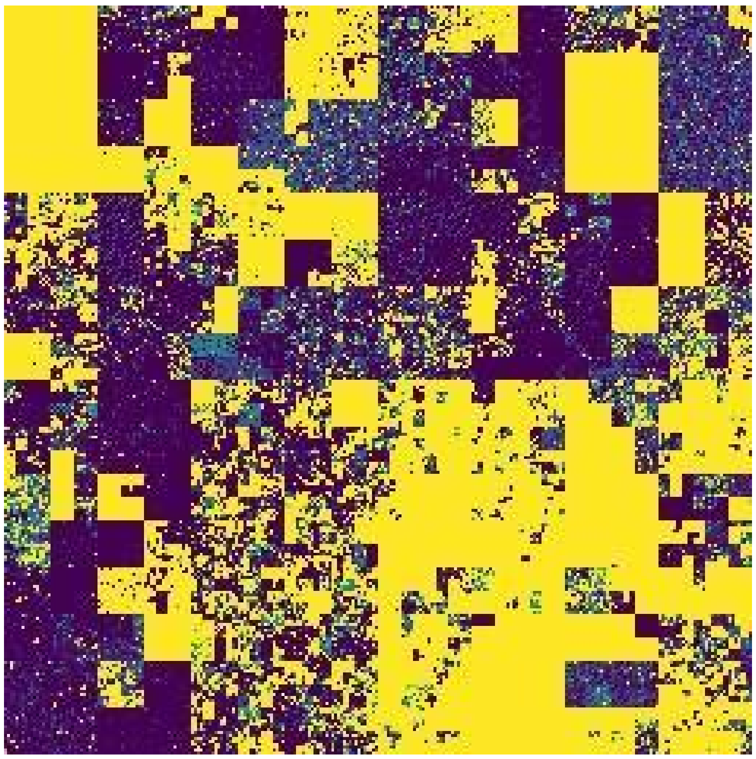
174.0.0.0/8:



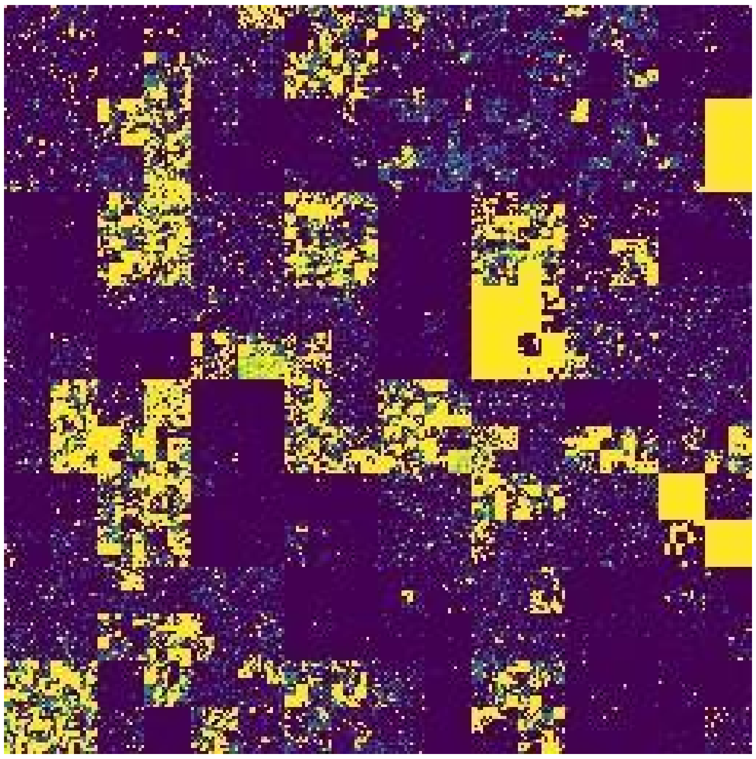
175.0.0.0/8:



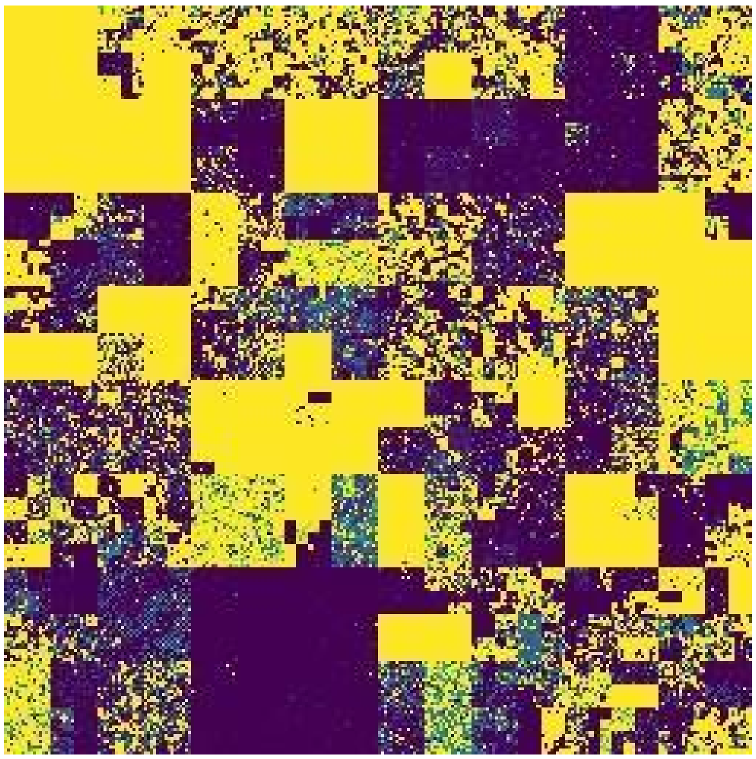
176.0.0.0/8:



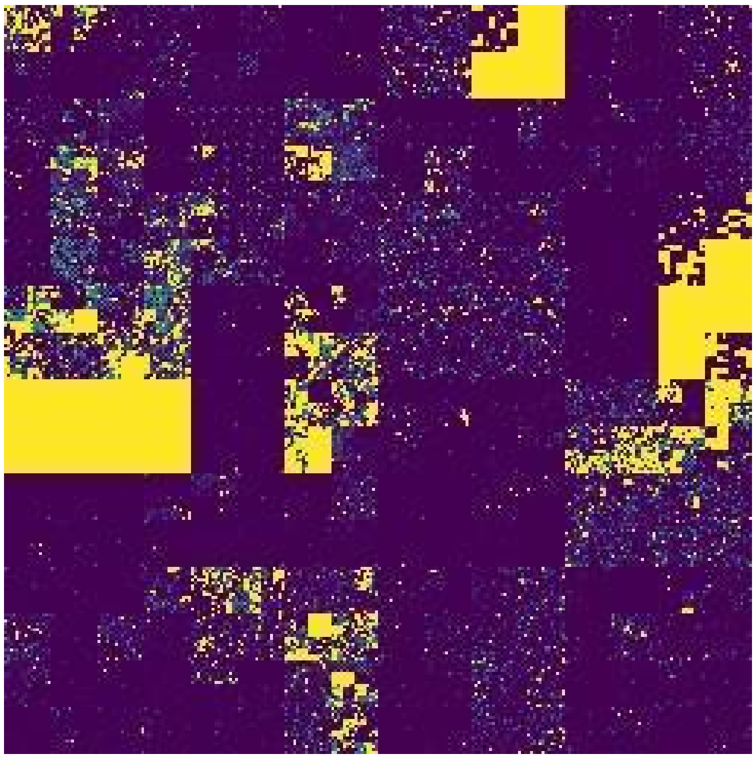
177.0.0.0/8:



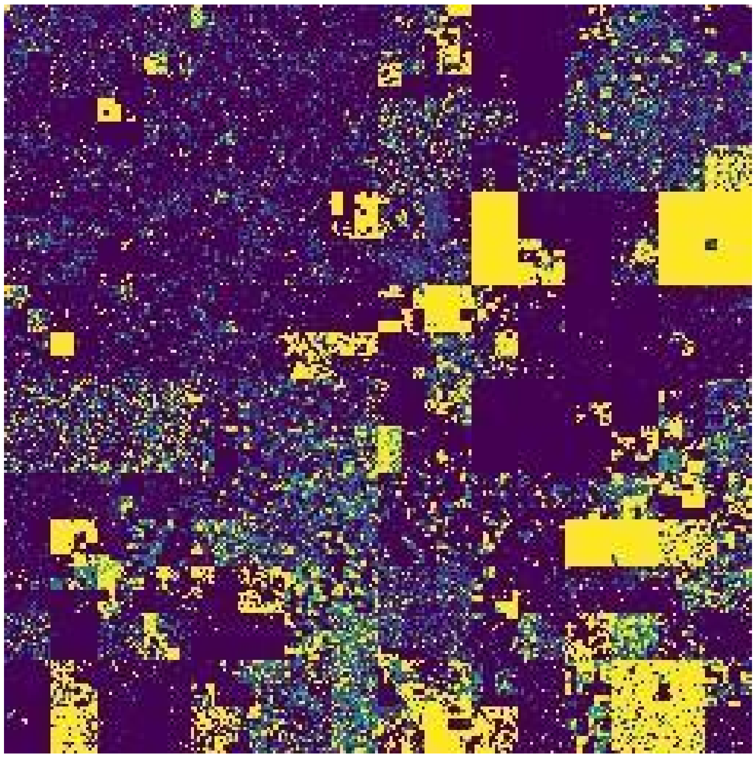
178.0.0.0/8:



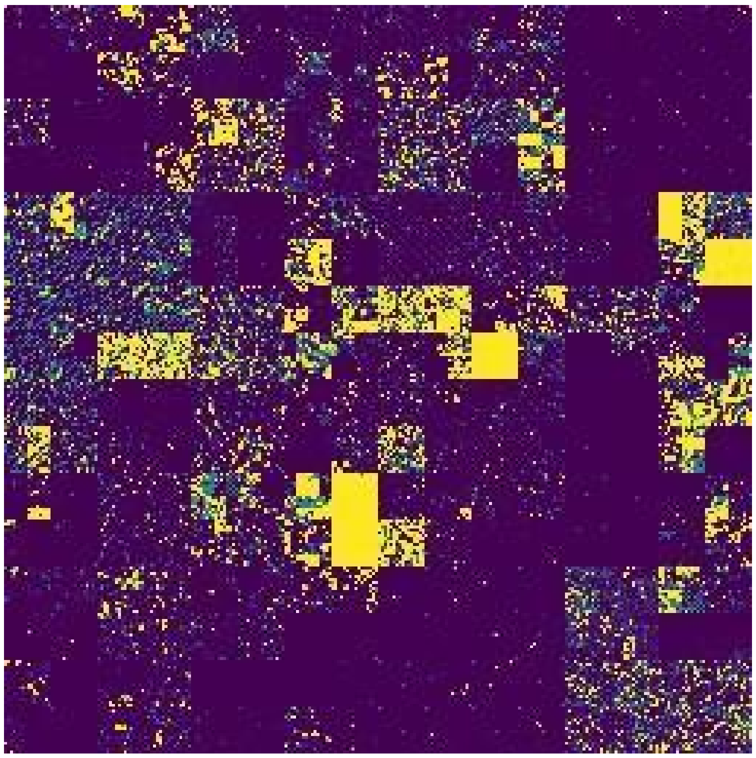
179.0.0.0/8:



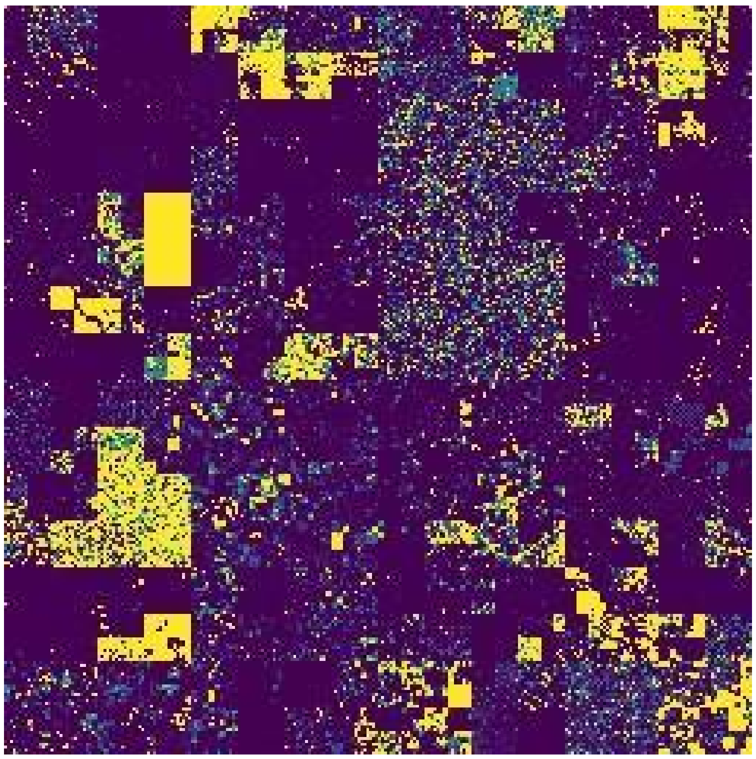
180.0.0.0/8:



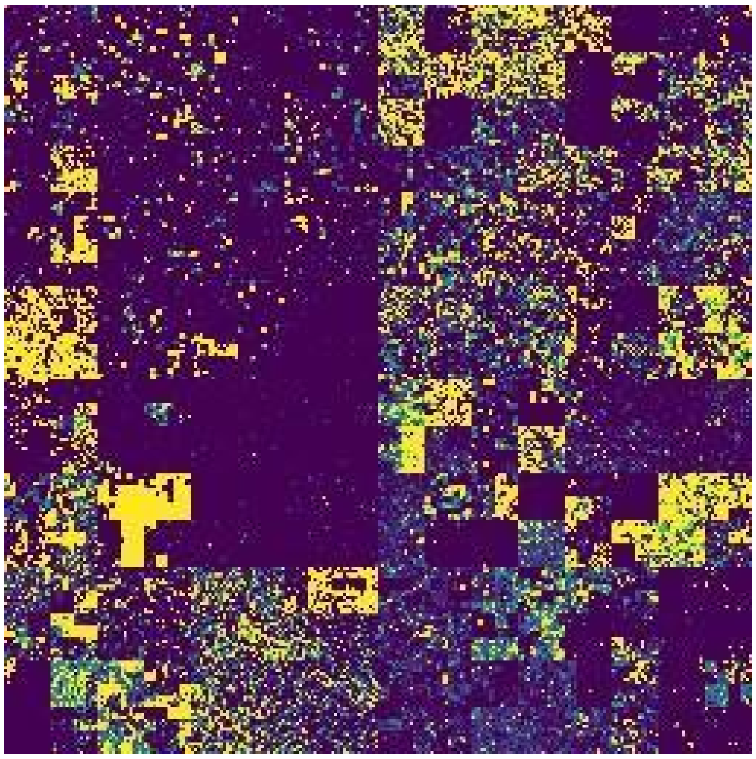
181.0.0.0/8:



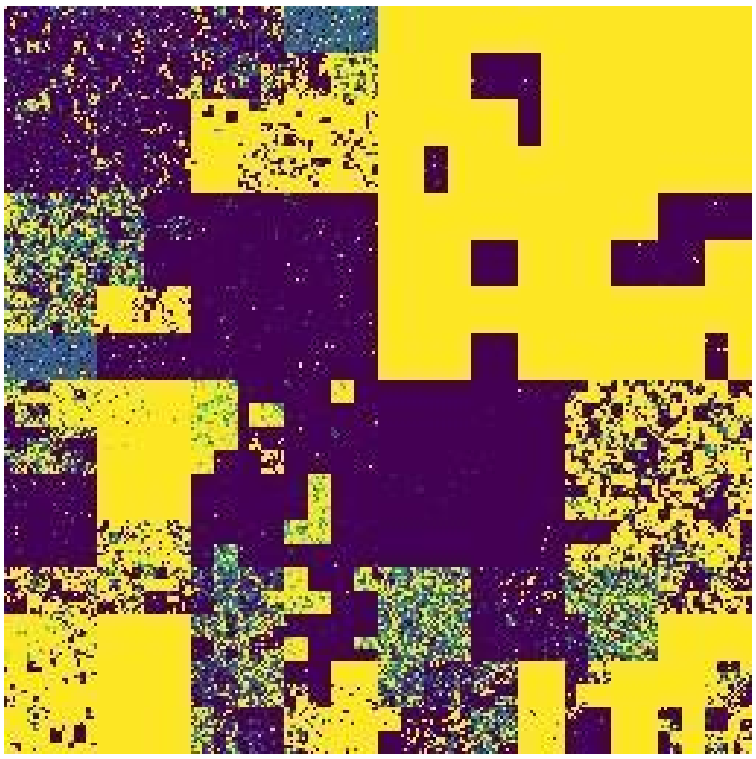
182.0.0.0/8:



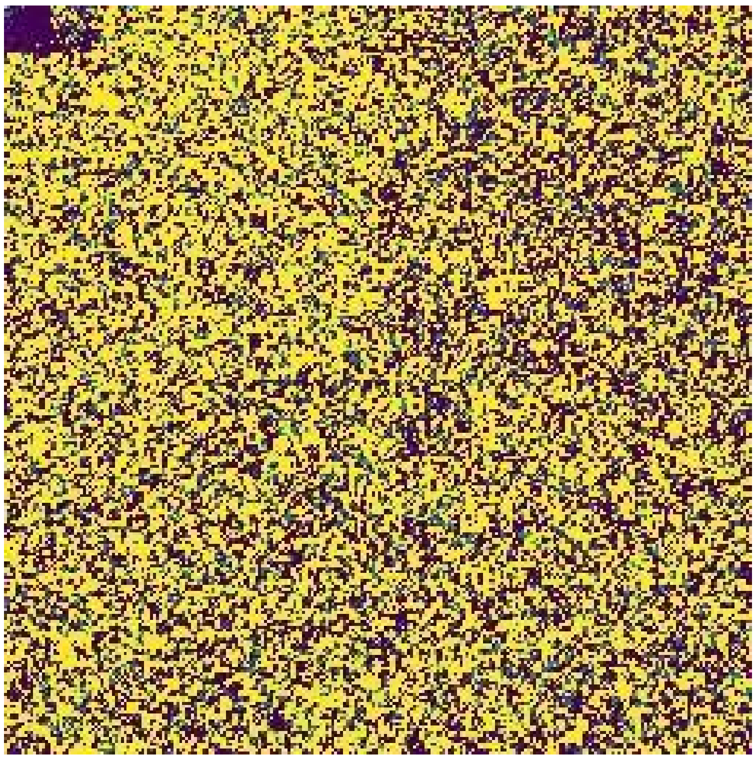
183.0.0.0/8:



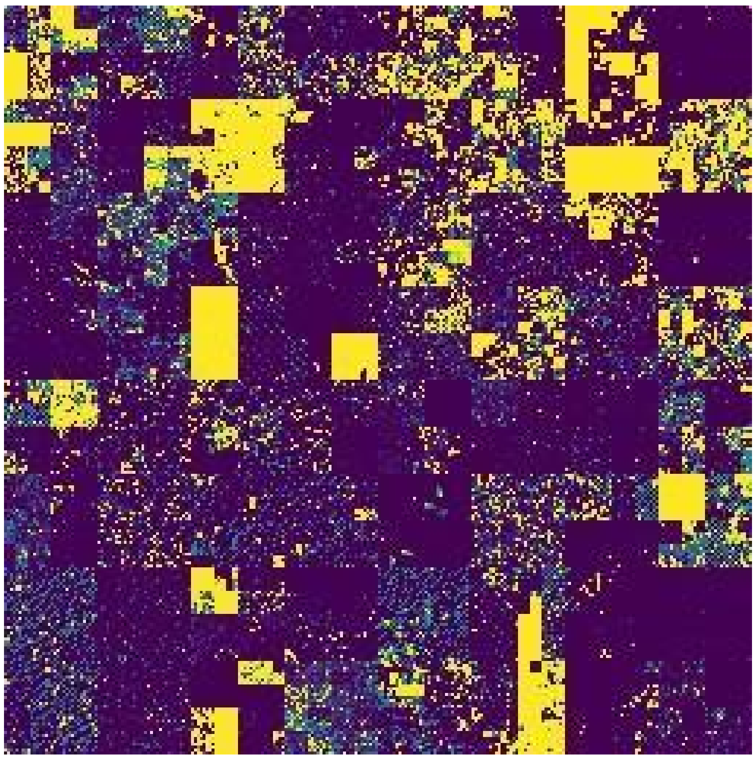
184.0.0.0/8:



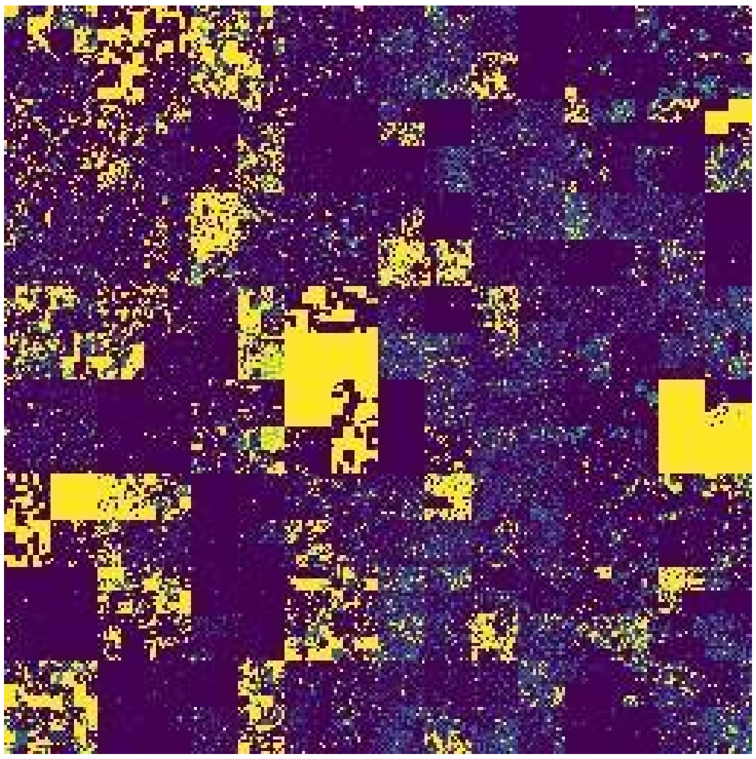
185.0.0.0/8:



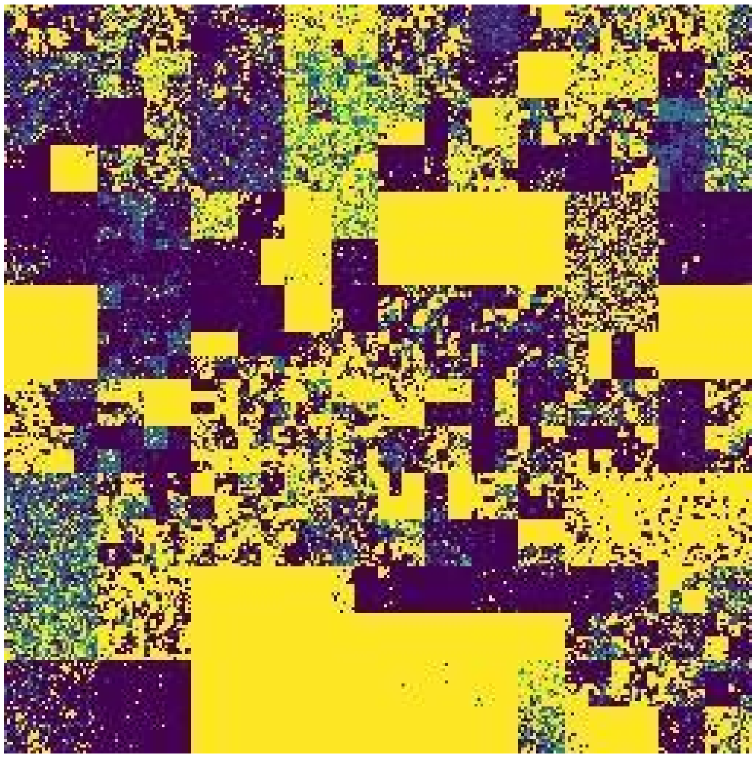
186.0.0.0/8:



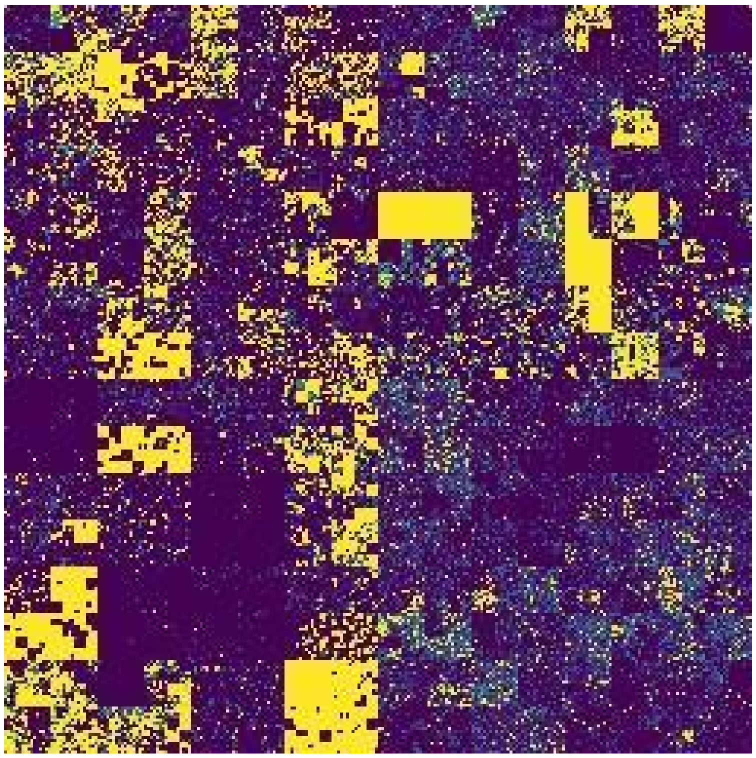
187.0.0.0/8:



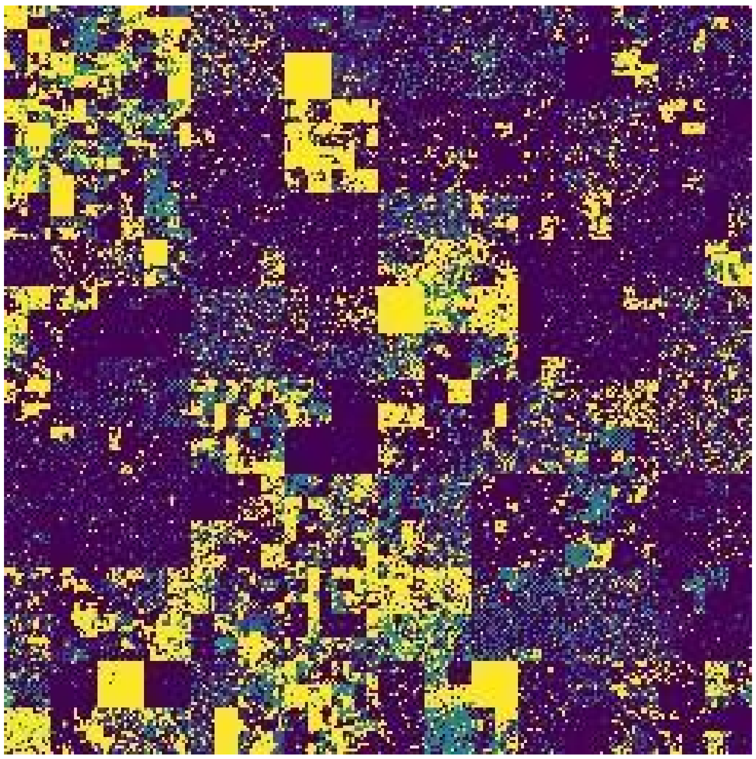
188.0.0.0/8:



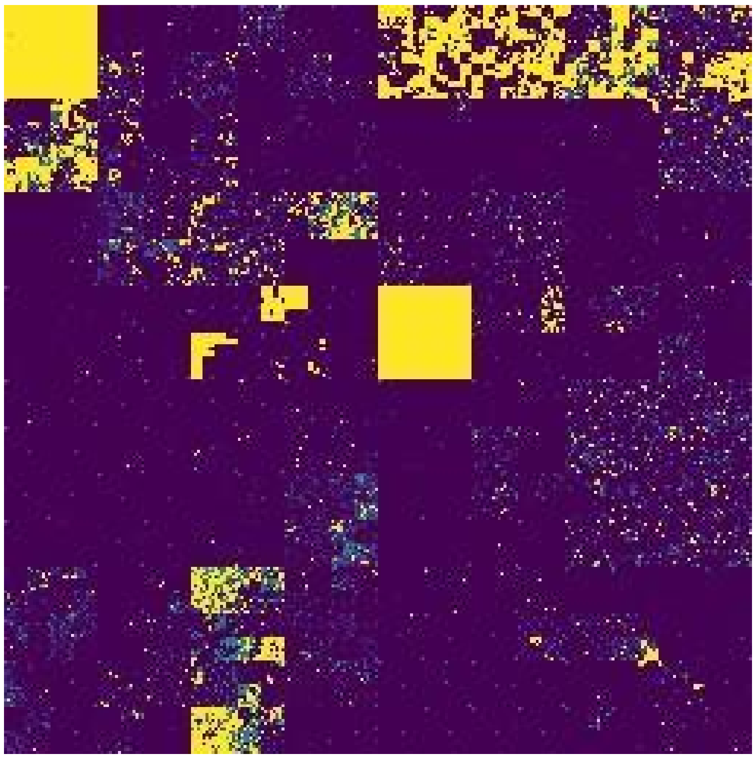
189.0.0.0/8:



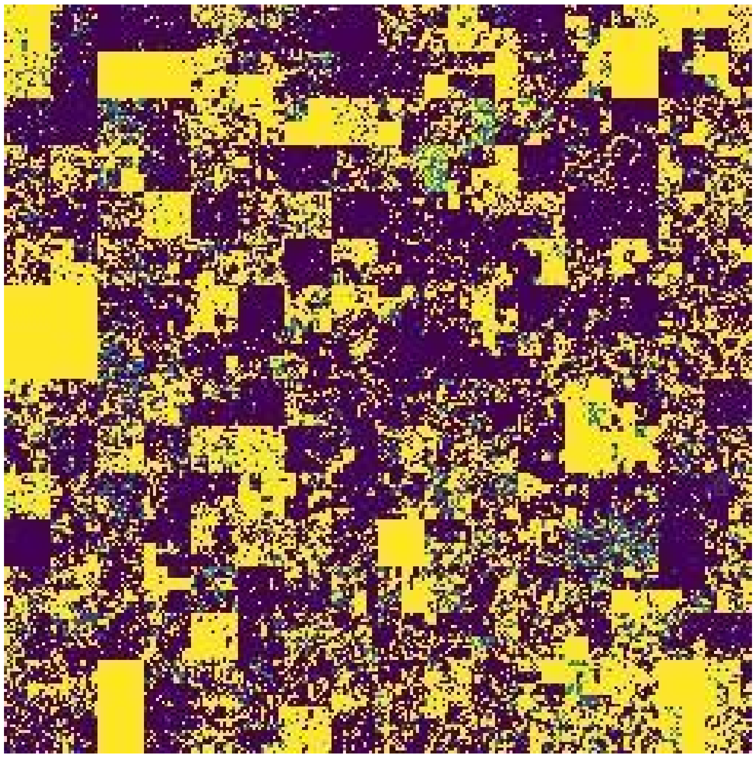
190.0.0.0/8:



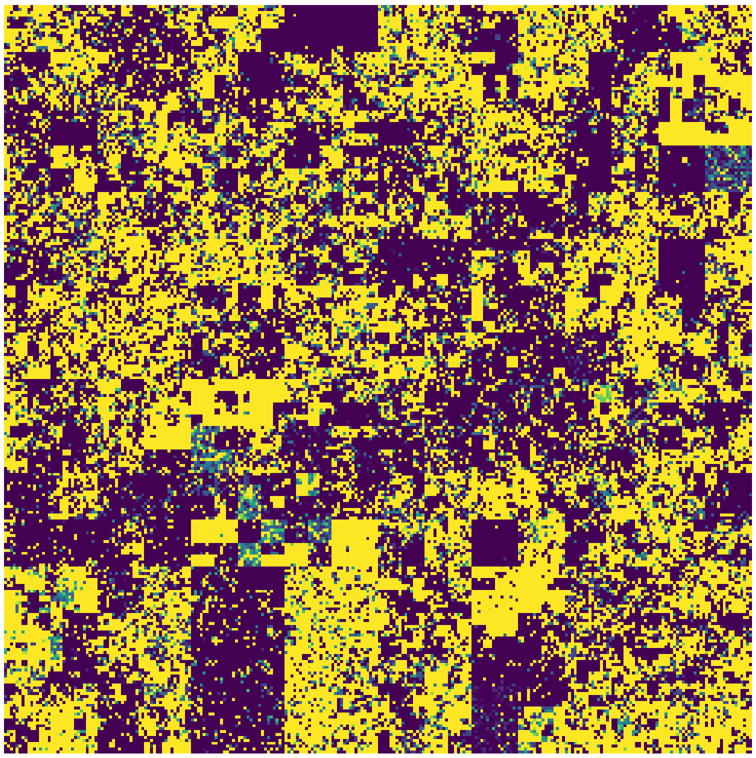
191.0.0.0/8:



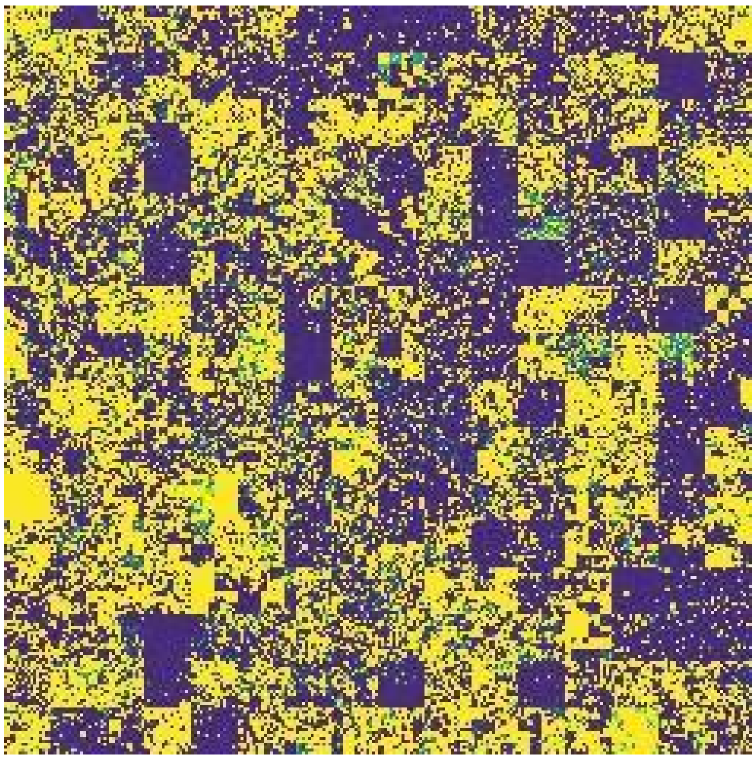
192.0.0.0/8:



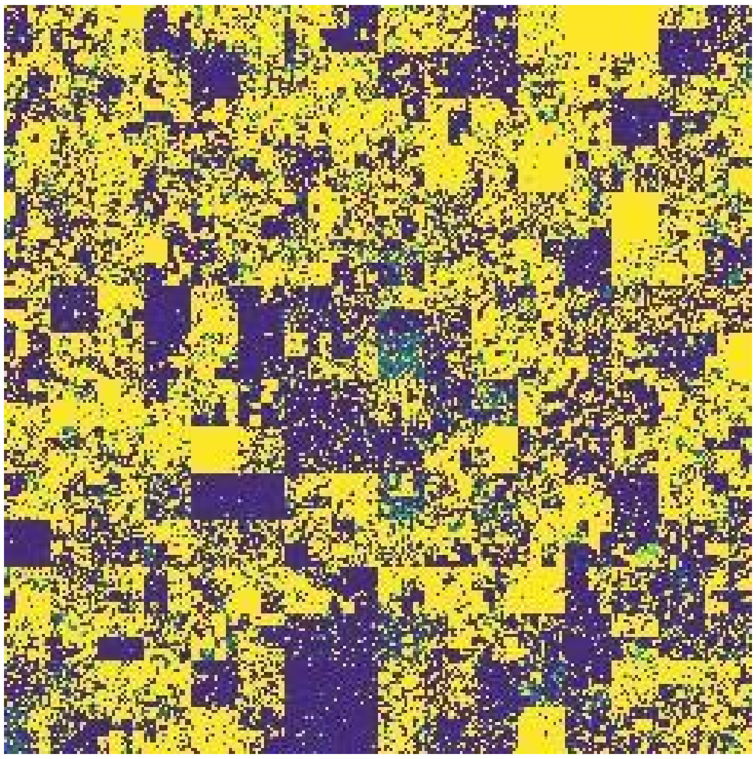
193.0.0.0/8:



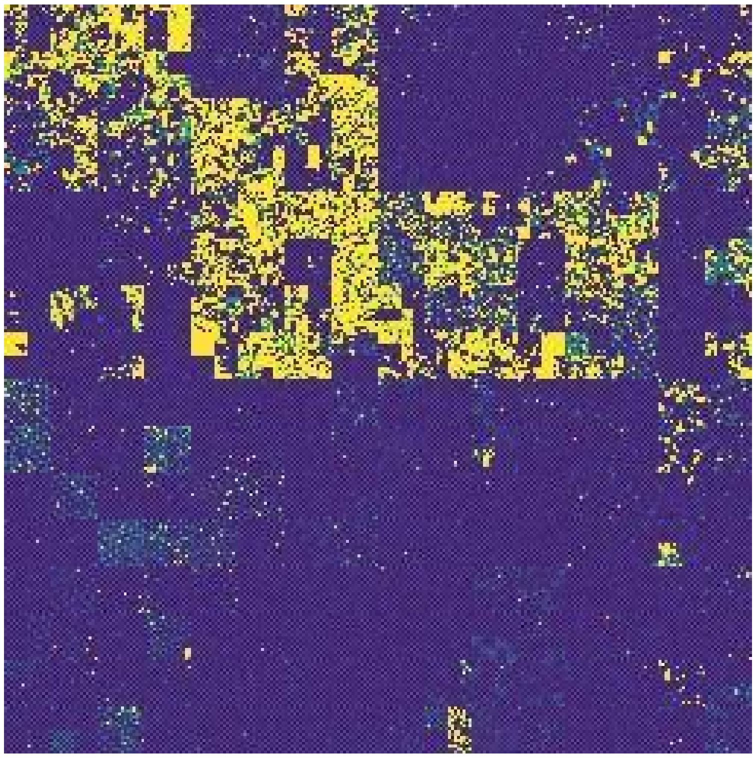
194.0.0.0/8:



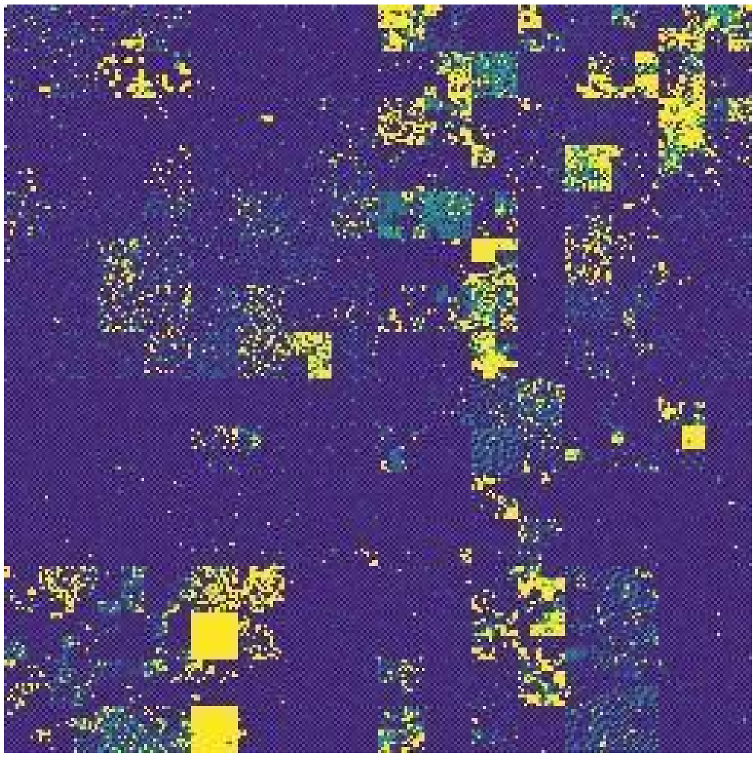
195.0.0.0/8:



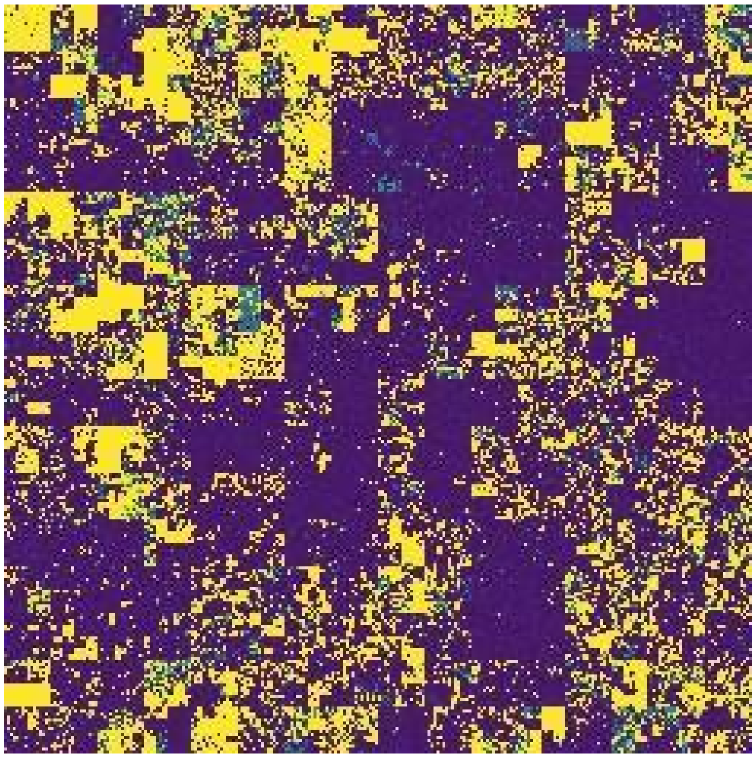
196.0.0.0/8:



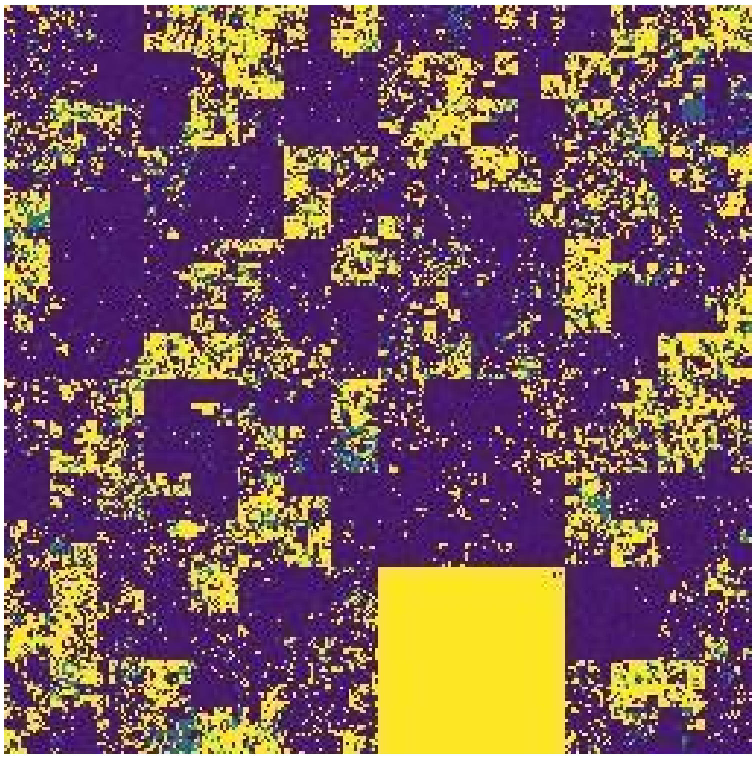
197.0.0.0/8:



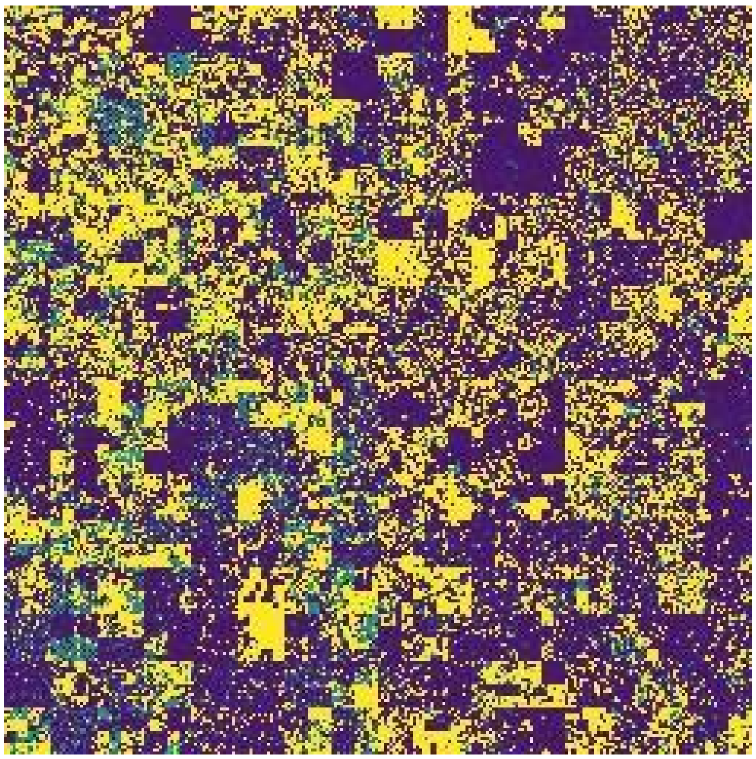
198.0.0.0/8:



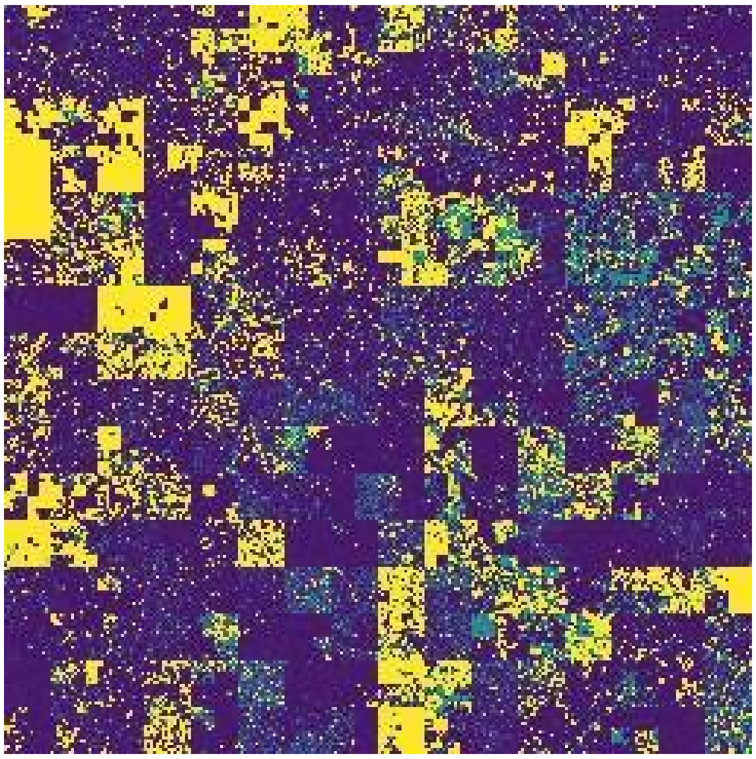
199.0.0.0/8:



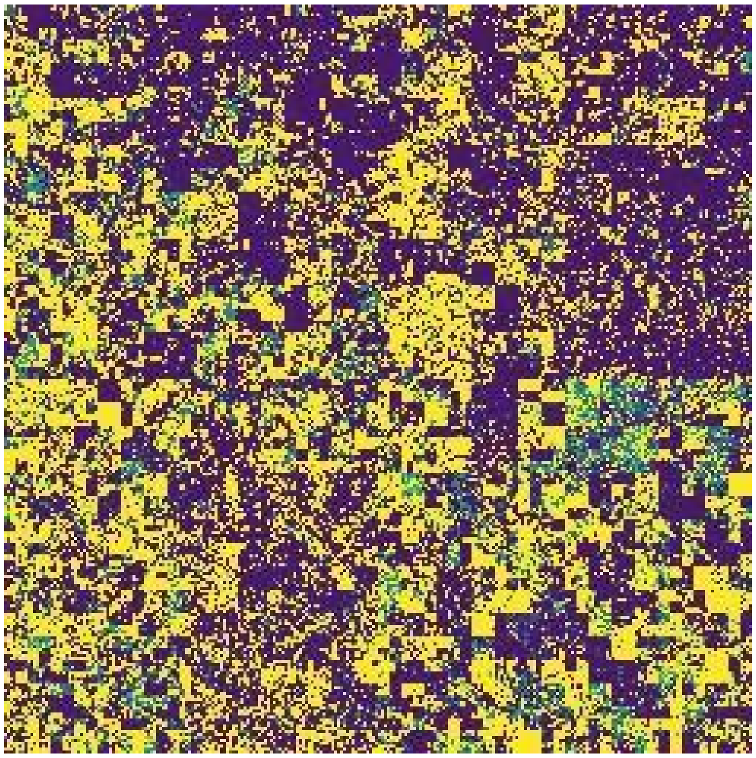
200.0.0.0/8:



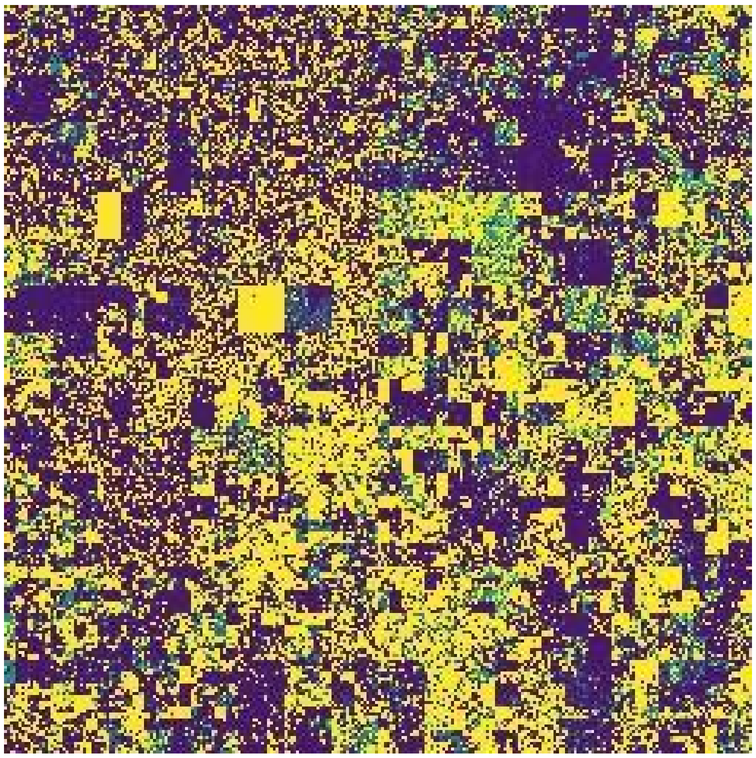
201.0.0.0/8:



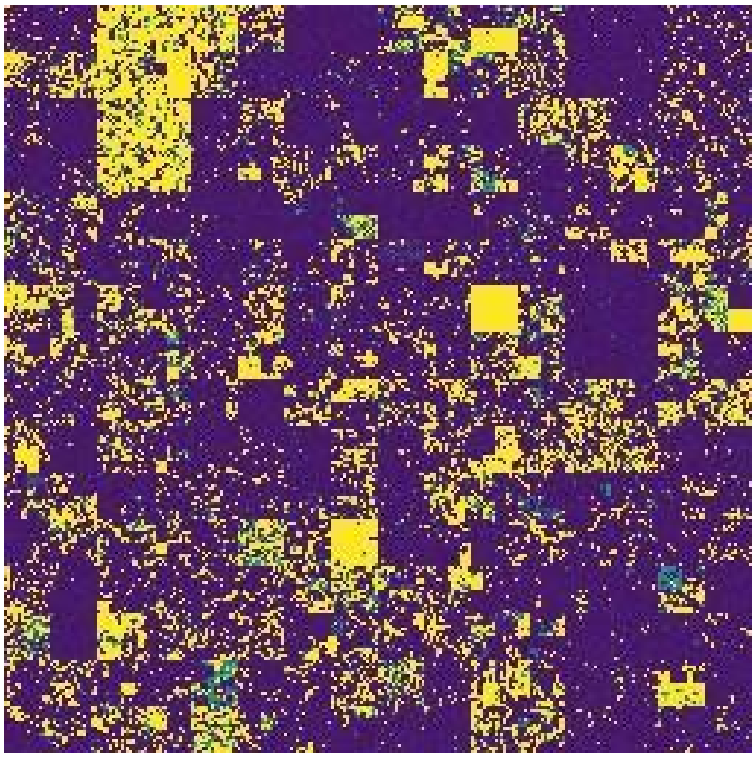
202.0.0.0/8:



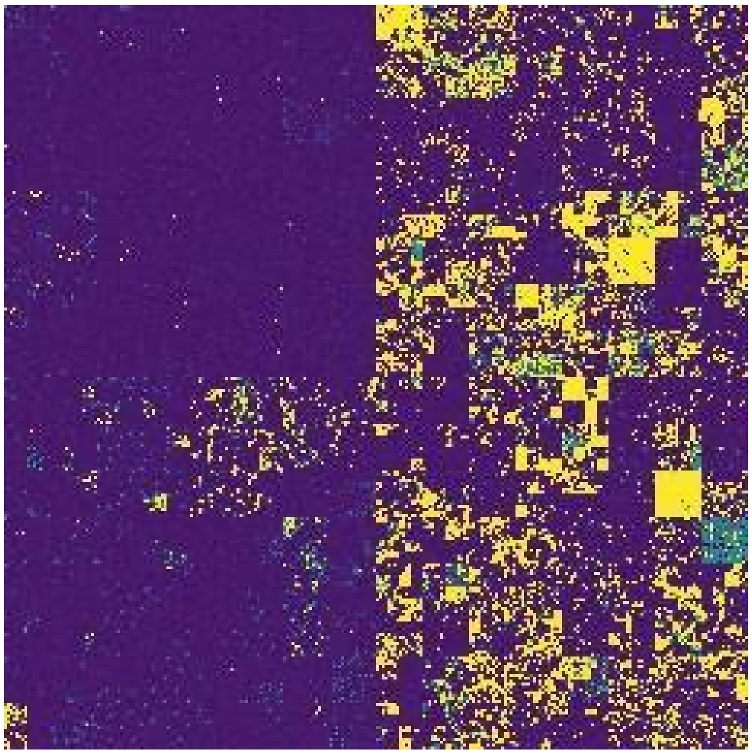
203.0.0.0/8:



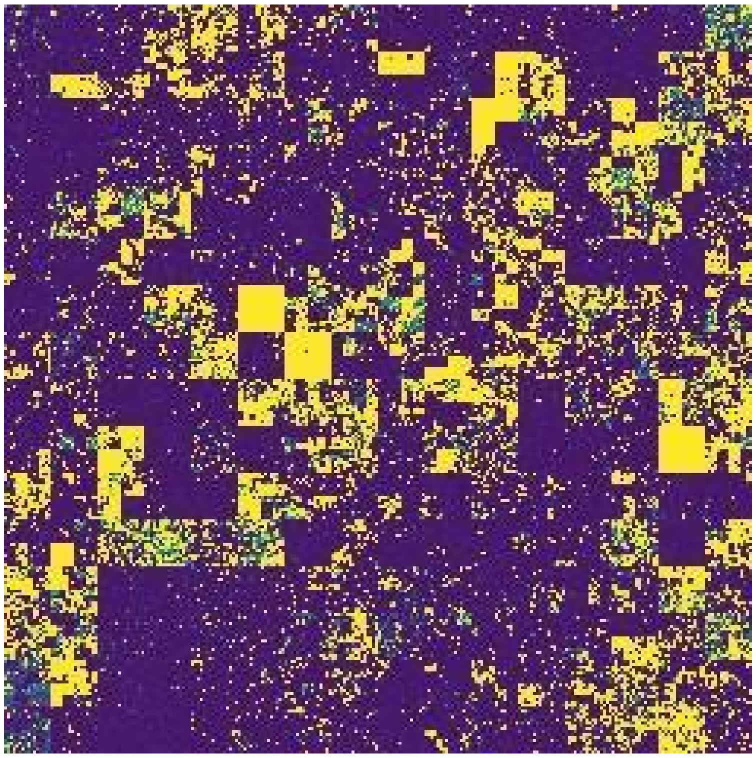
204.0.0.0/8:



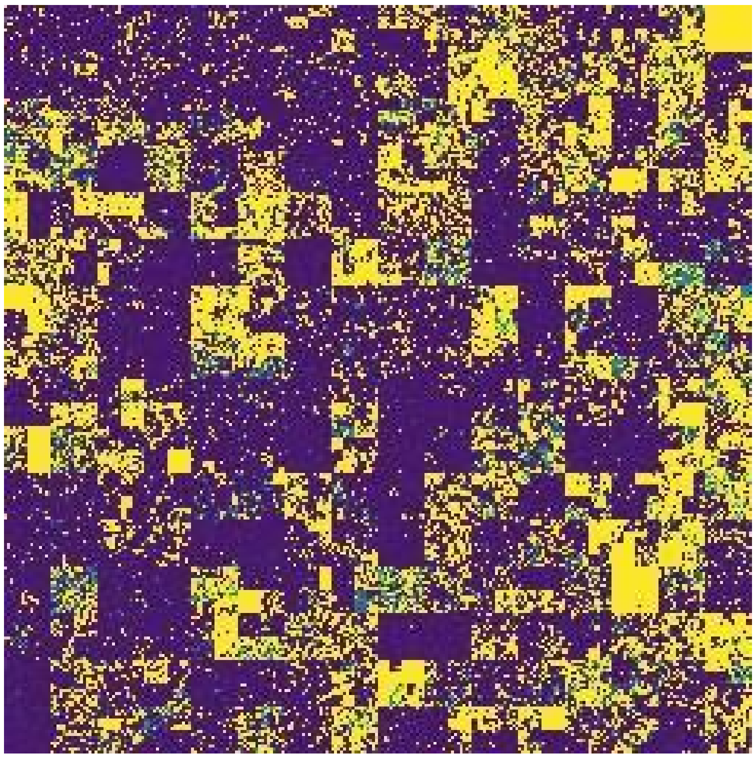
205.0.0.0/8: DOD (05/1998)



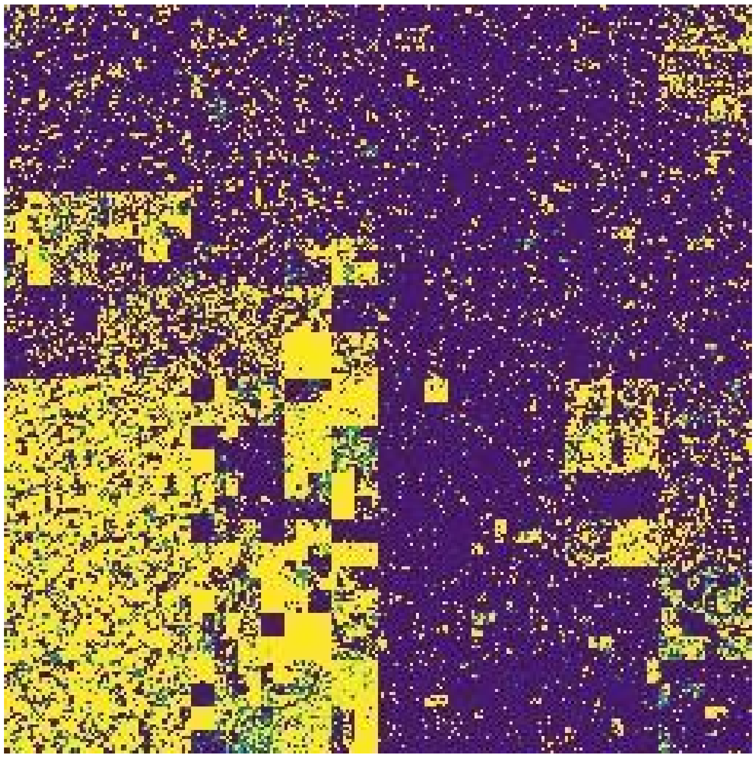
206.0.0.0/8:



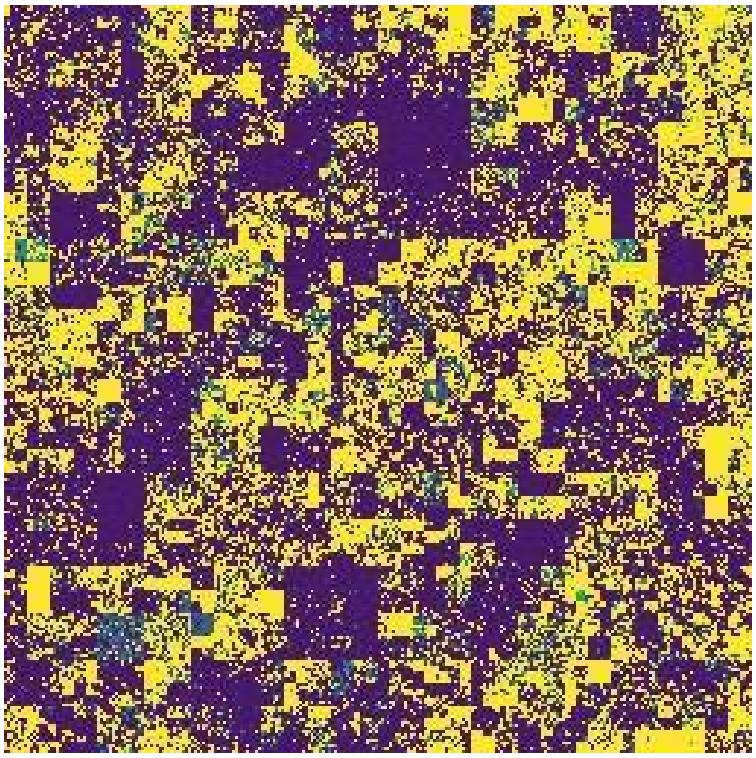
207.0.0.0/8:



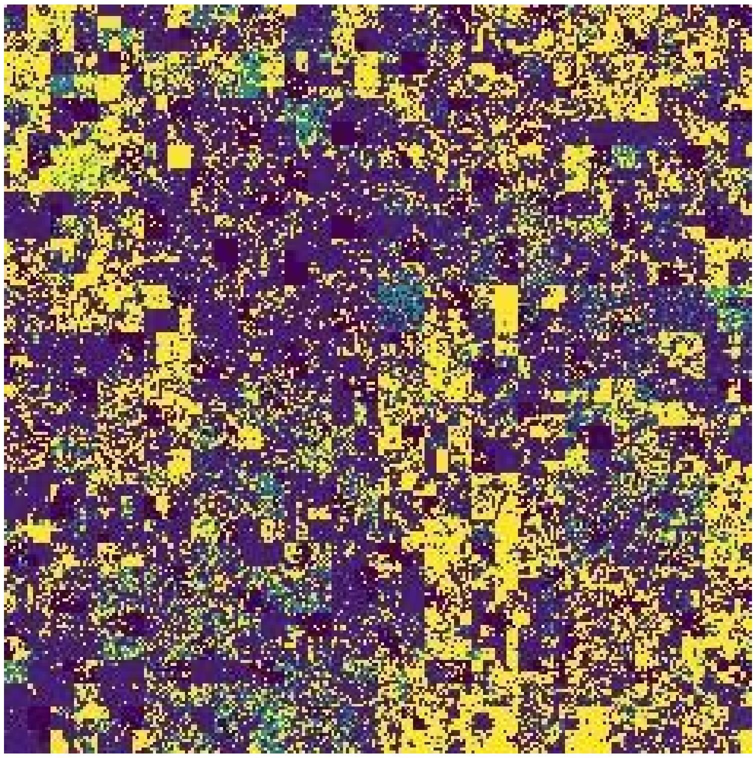
208.0.0.0/8:



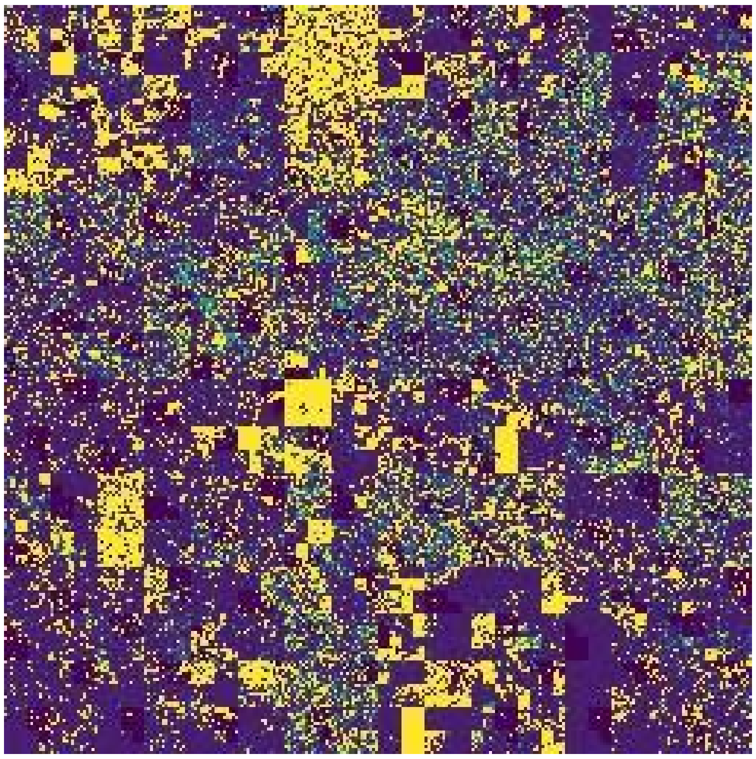
209.0.0.0/8:



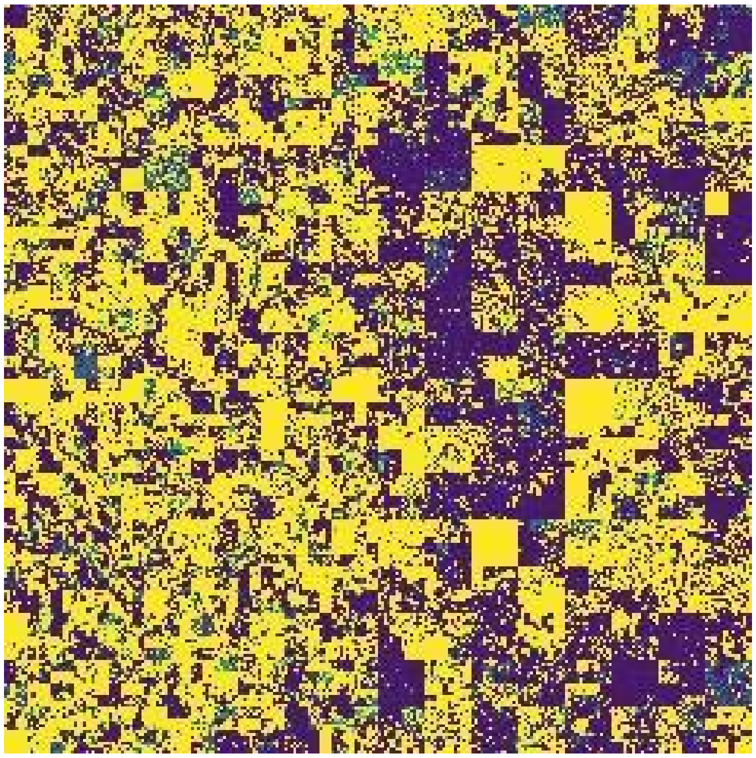
210.0.0.0/8:



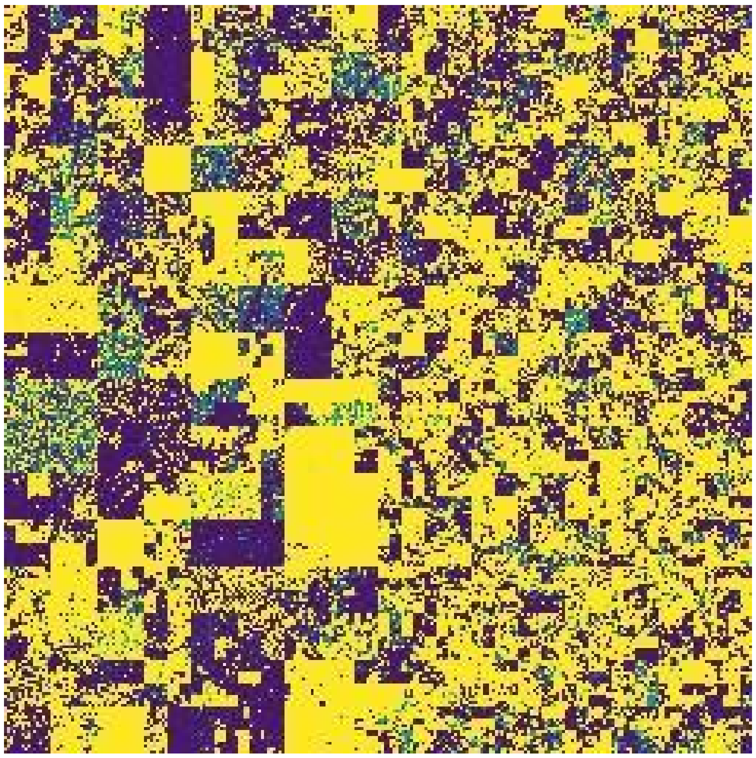
211.0.0.0/8:



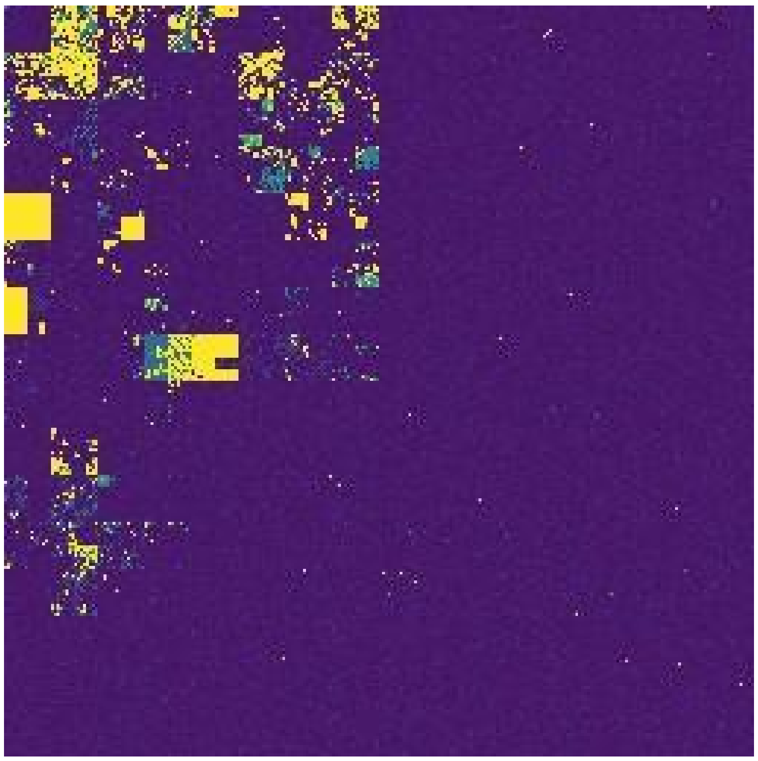
212.0.0.0/8:



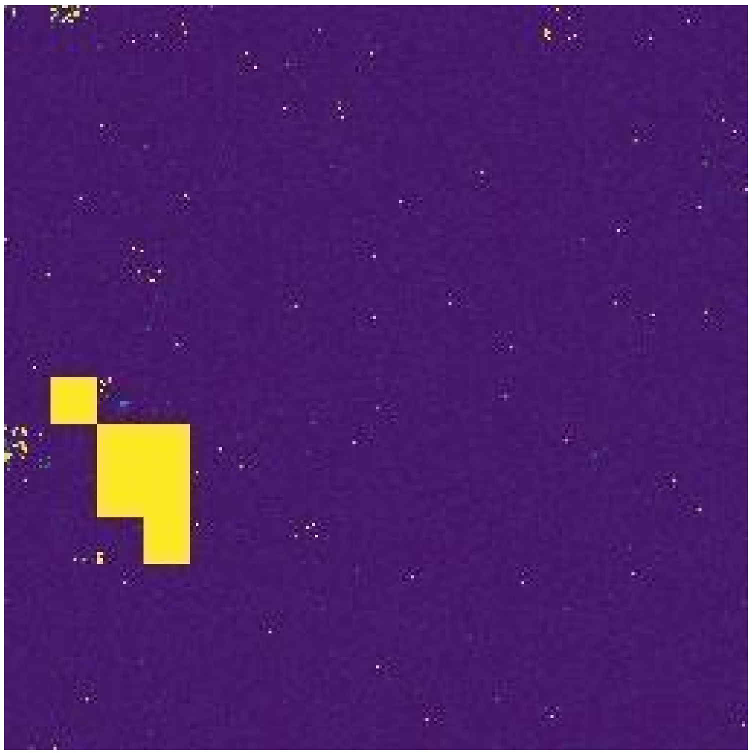
213.0.0.0/8:



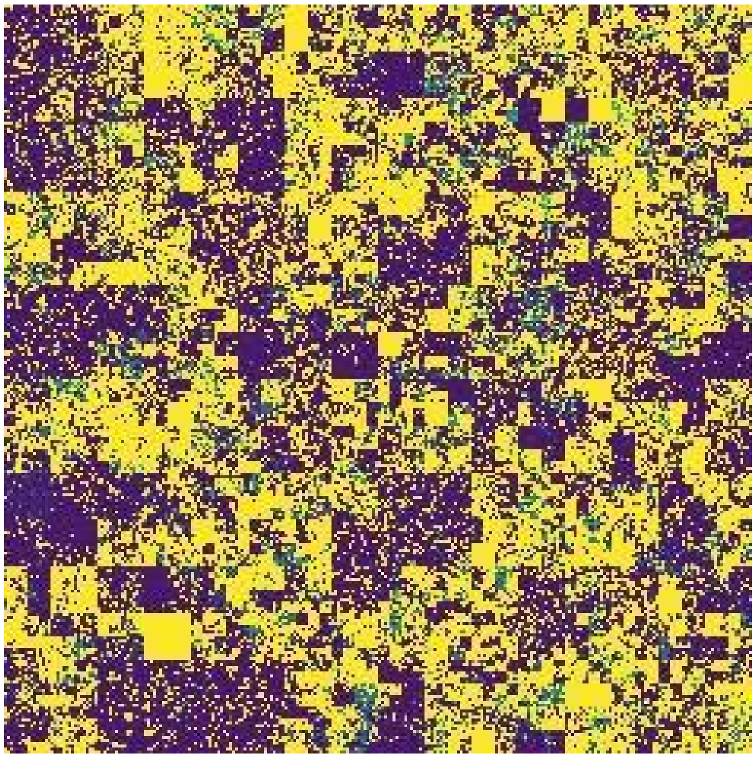
214.0.0.0/8:



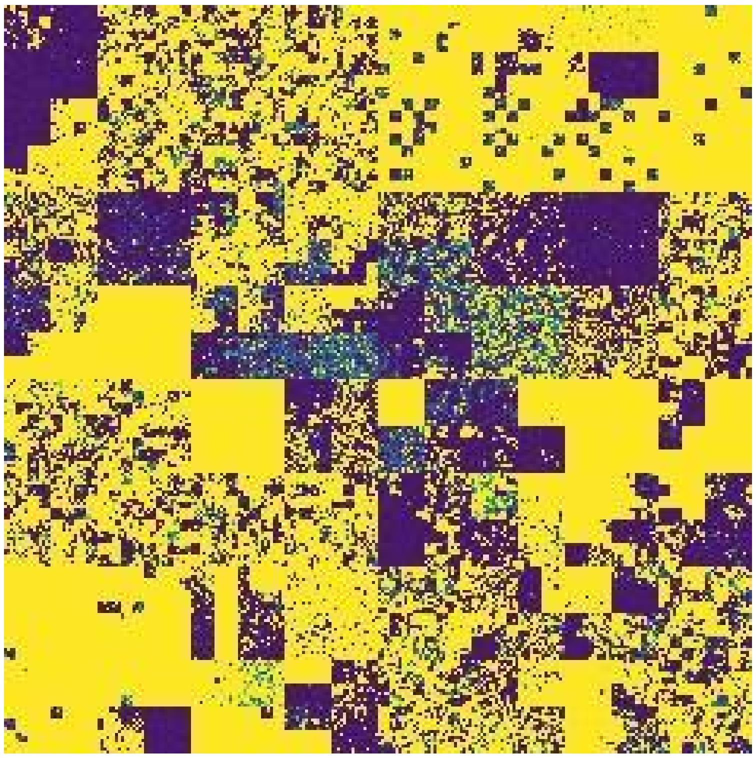
215.0.0.0/8: DOD (03/1998)



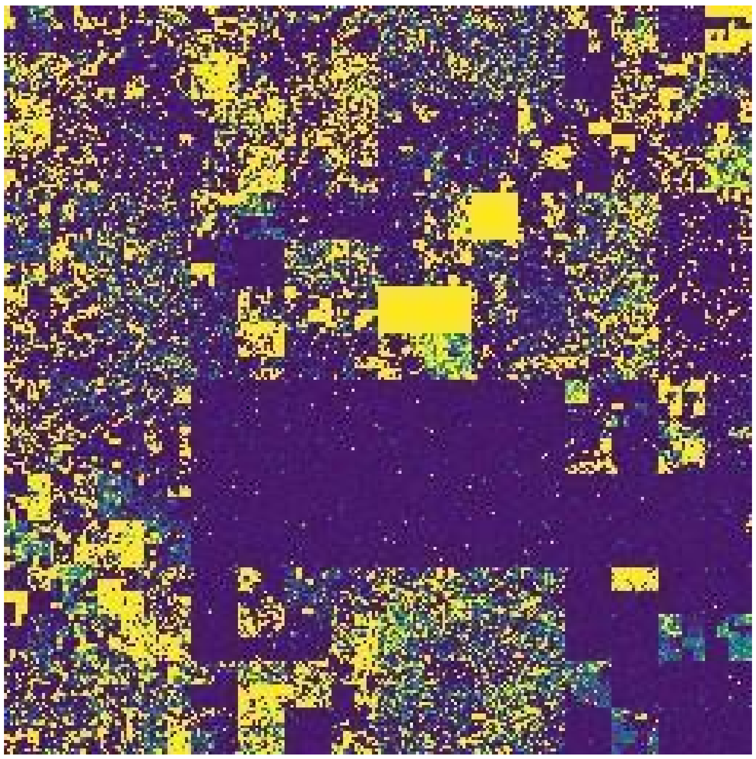
216.0.0.0/8:



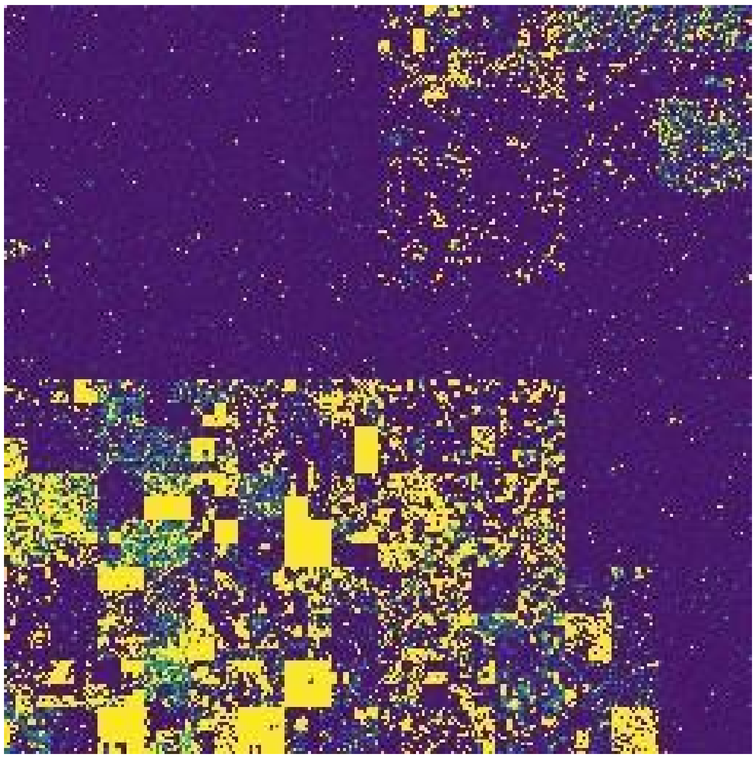
217.0.0.0/8:



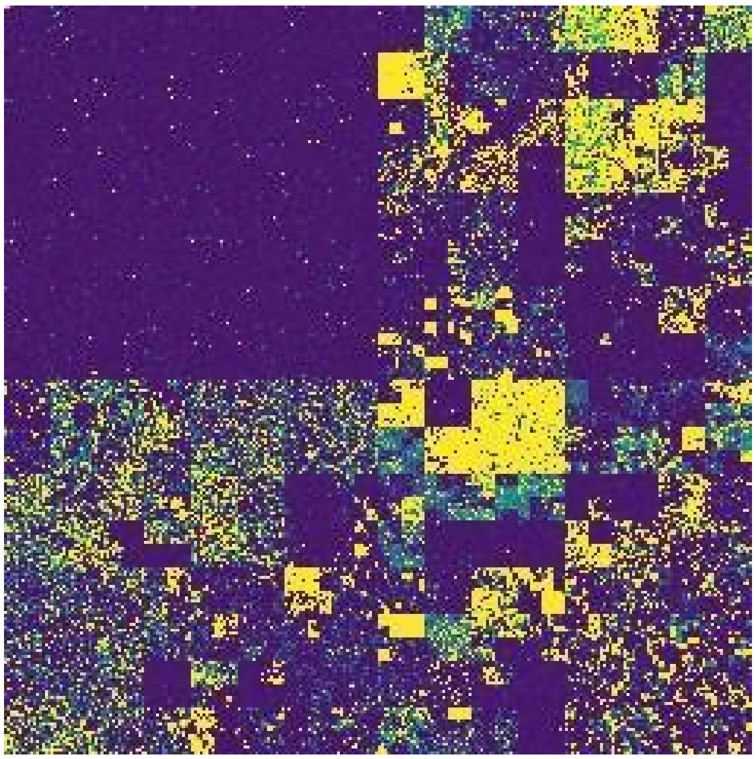
218.0.0.0/8:



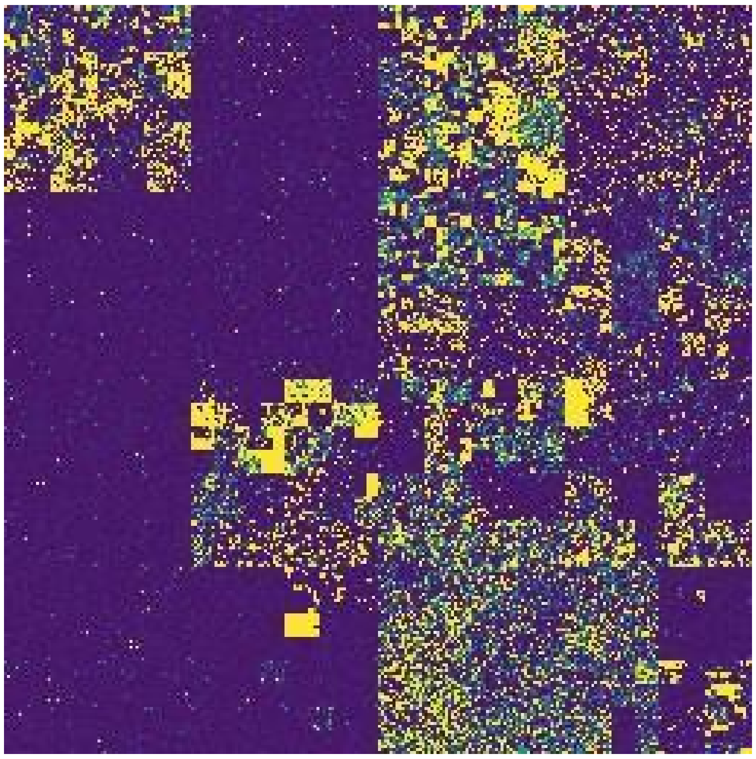
219.0.0.0/8:



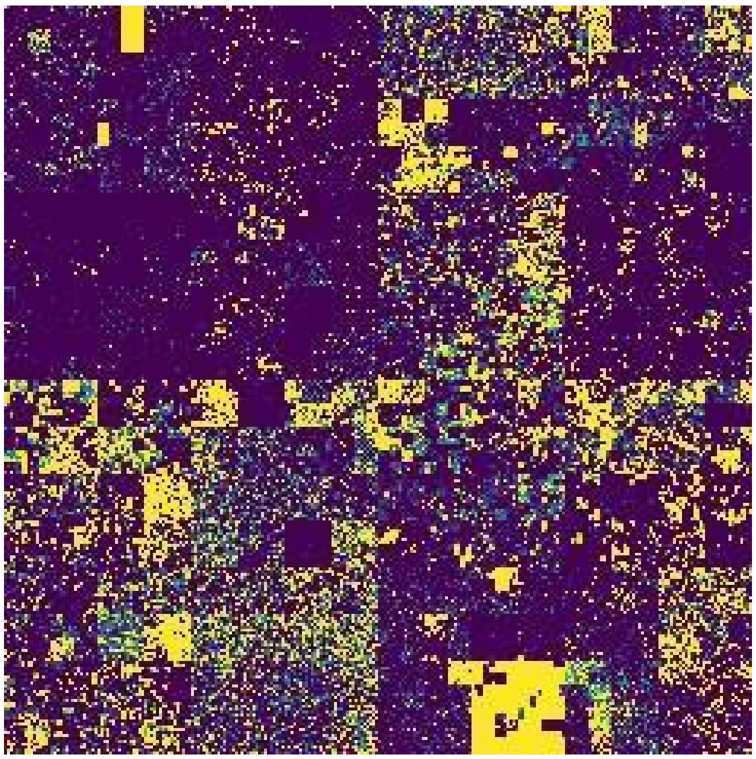
220.0.0.0/8:



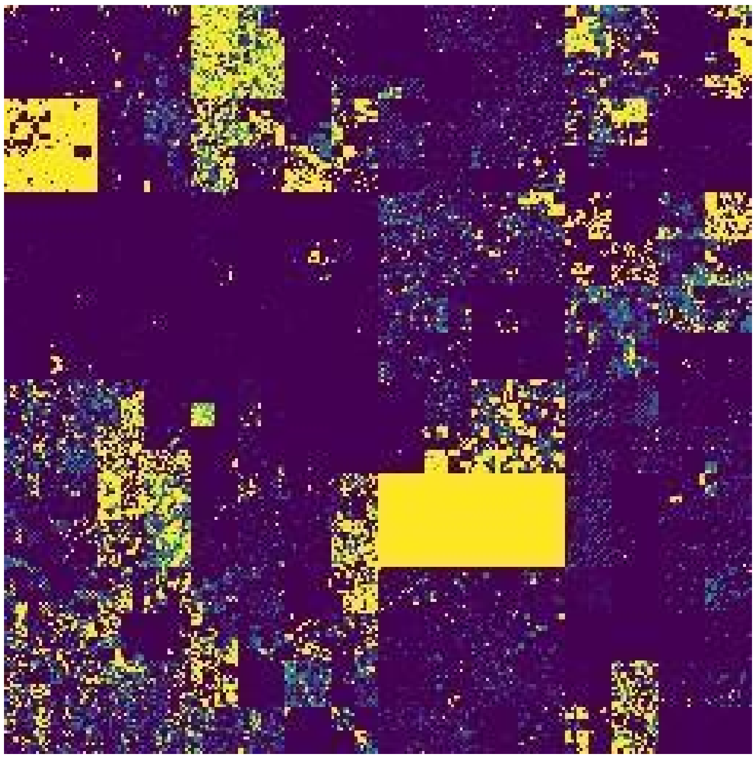
221.0.0.0/8:



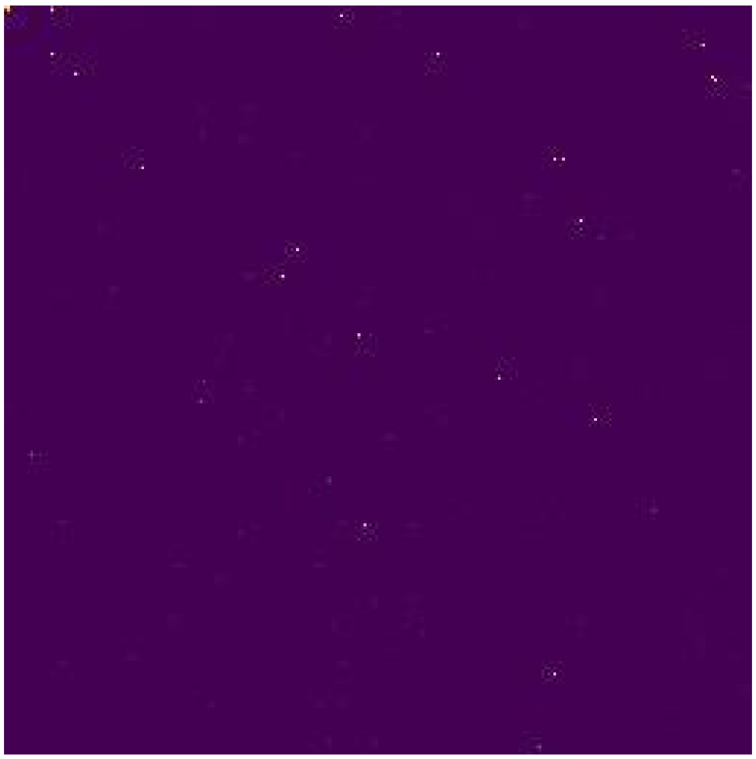
222.0.0.0/8:



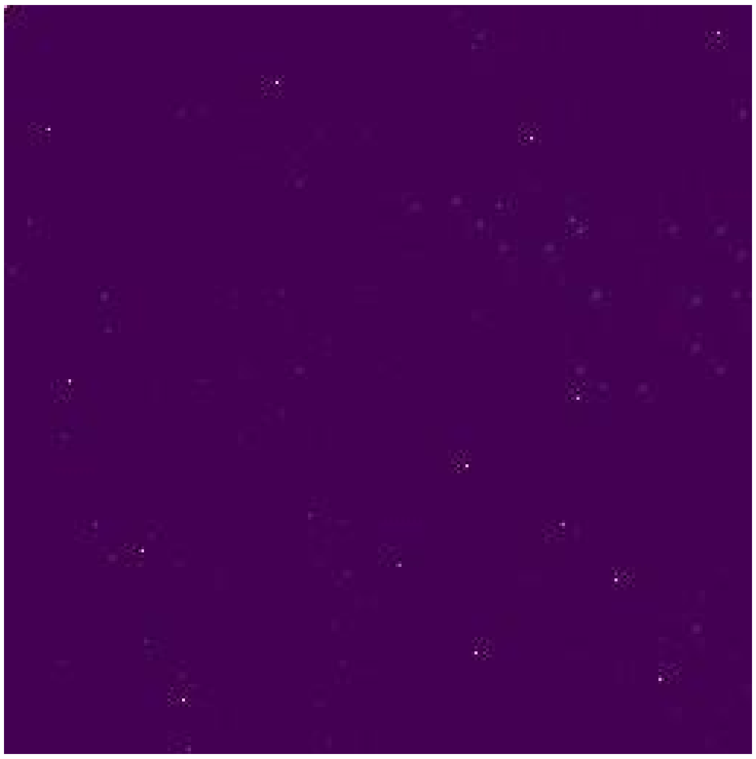
223.0.0.0/8:



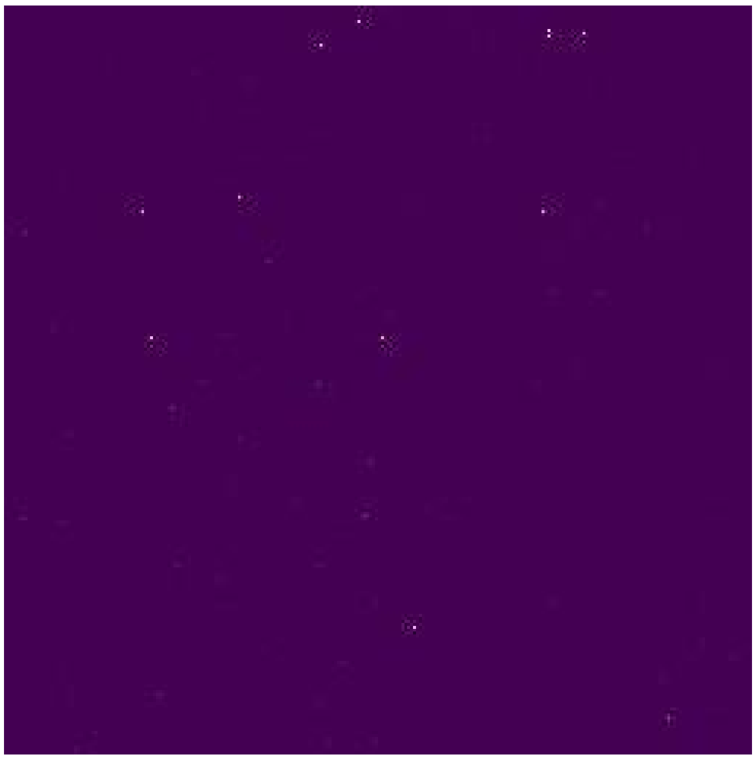
224.0.0.0/8: Multicast



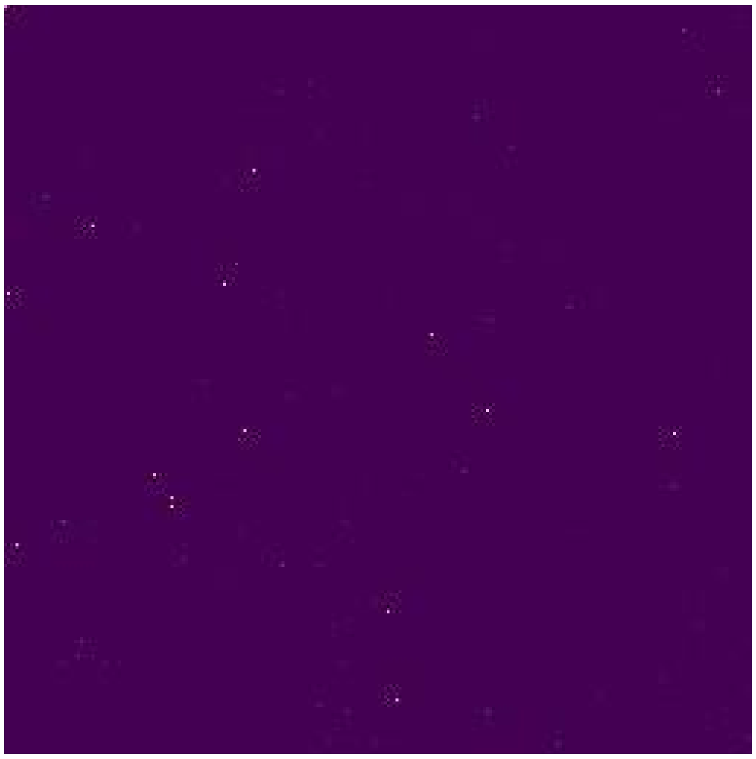
225.0.0.0/8: Multicast



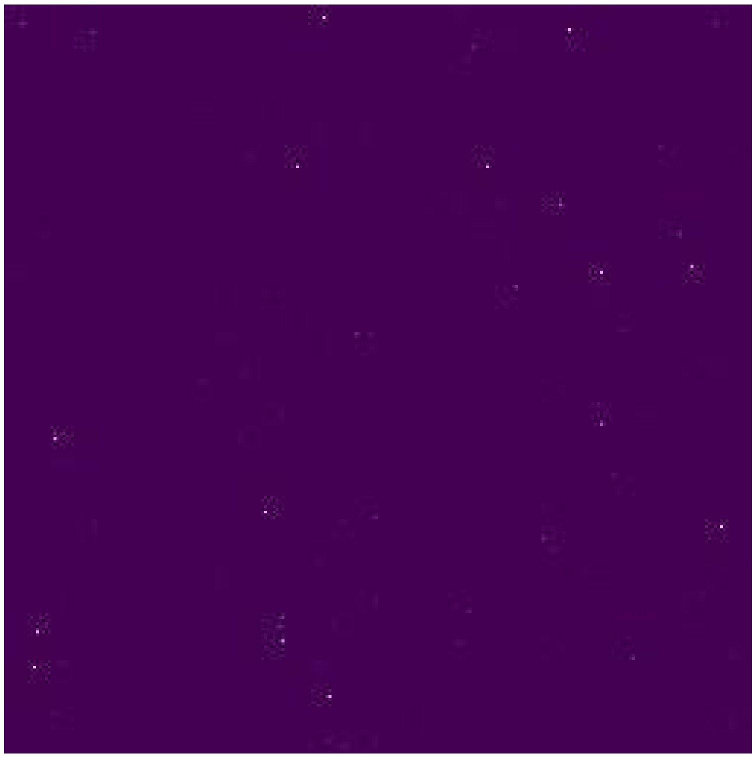
226.0.0.0/8: Multicast



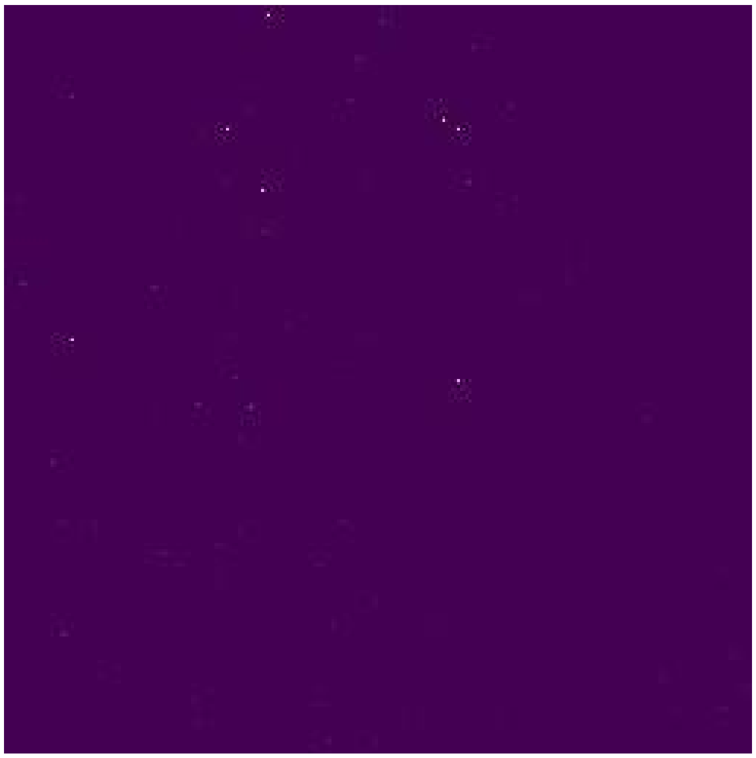
227.0.0.0/8: Multicast



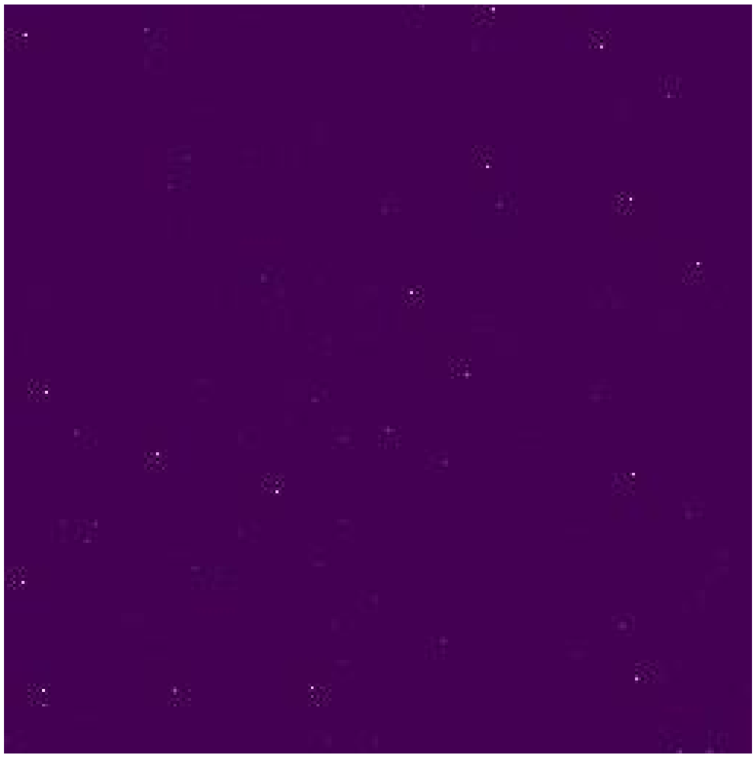
228.0.0.0/8: Multicast



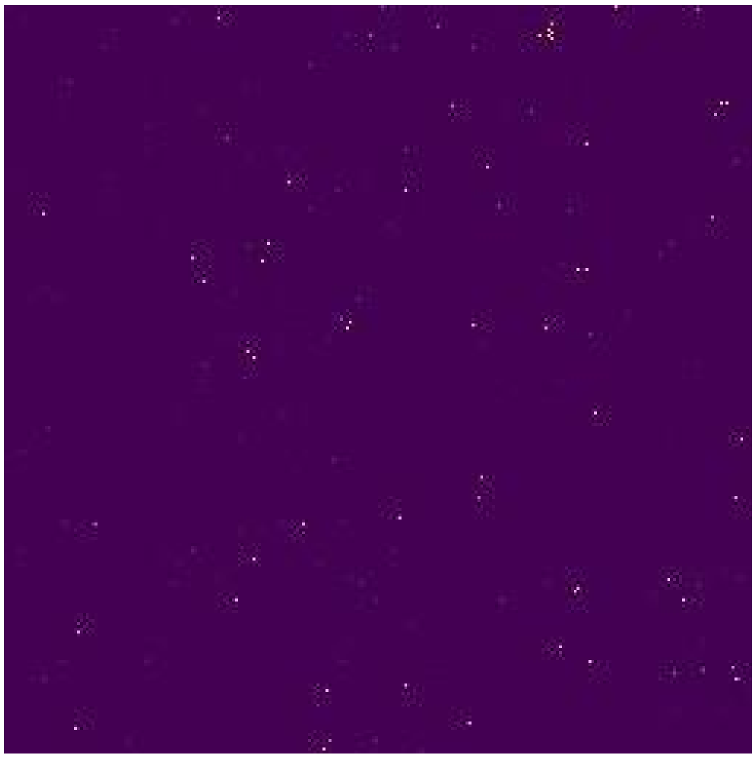
229.0.0.0/8: Multicast



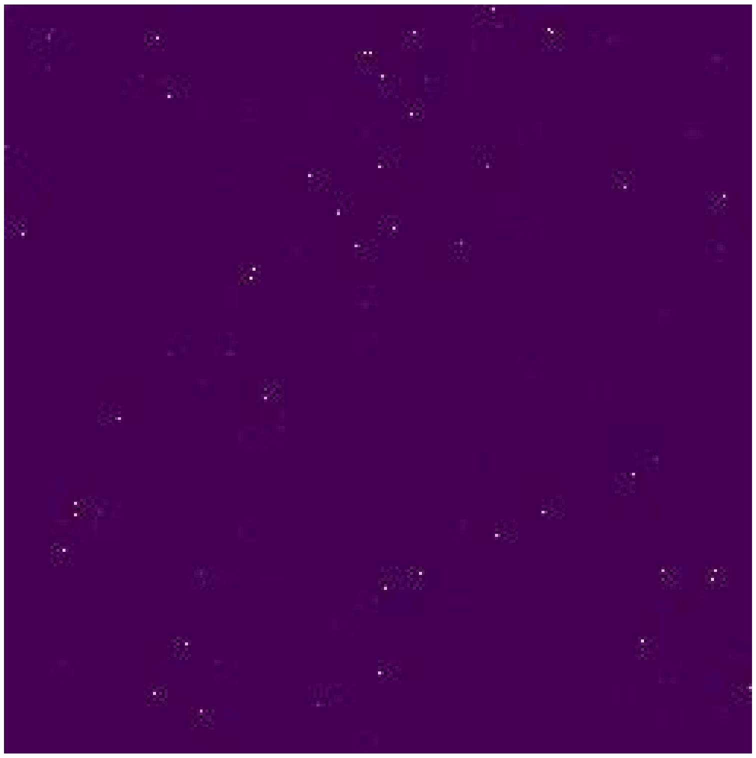
230.0.0.0/8: Multicast



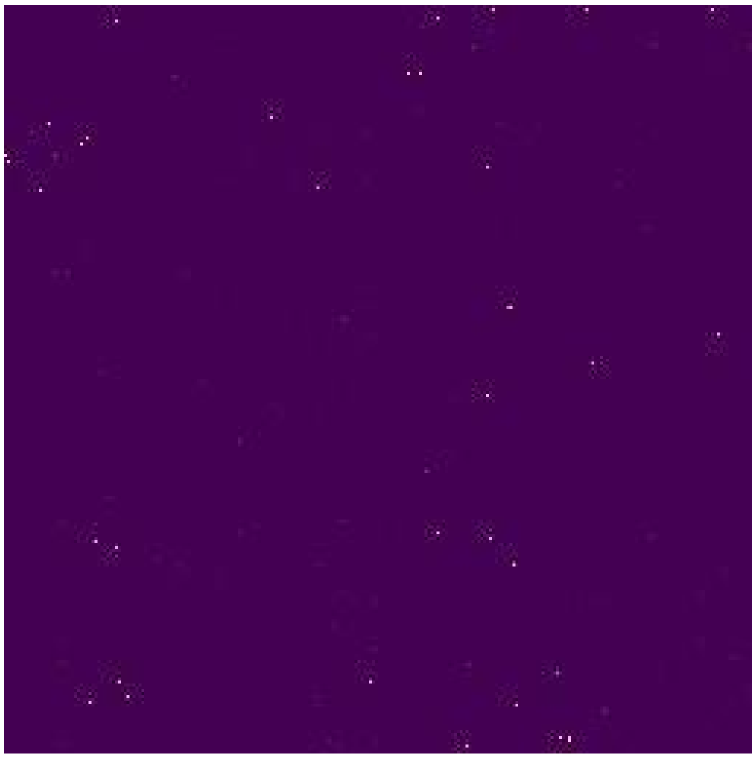
231.0.0.0/8: Multicast



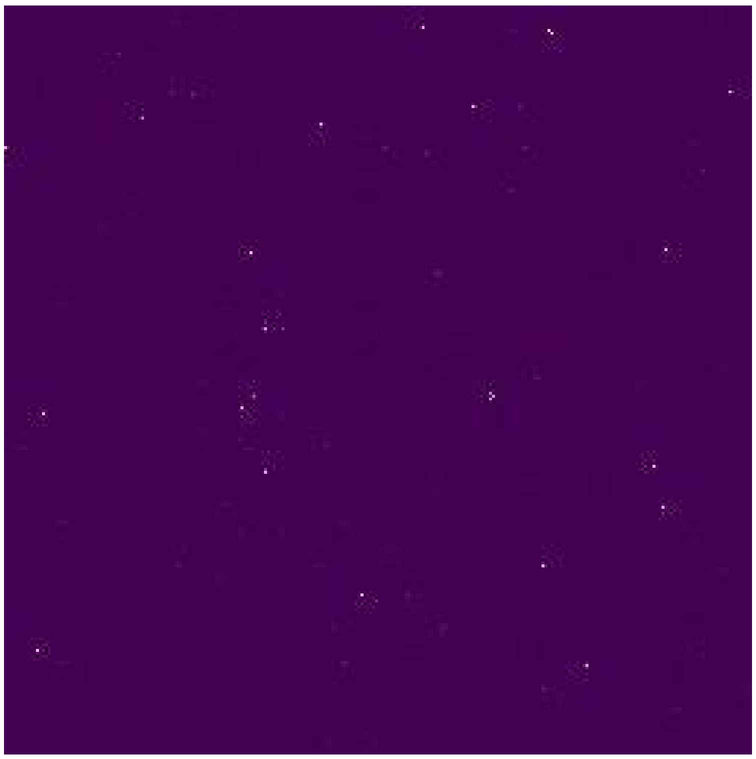
232.0.0.0/8: Multicast



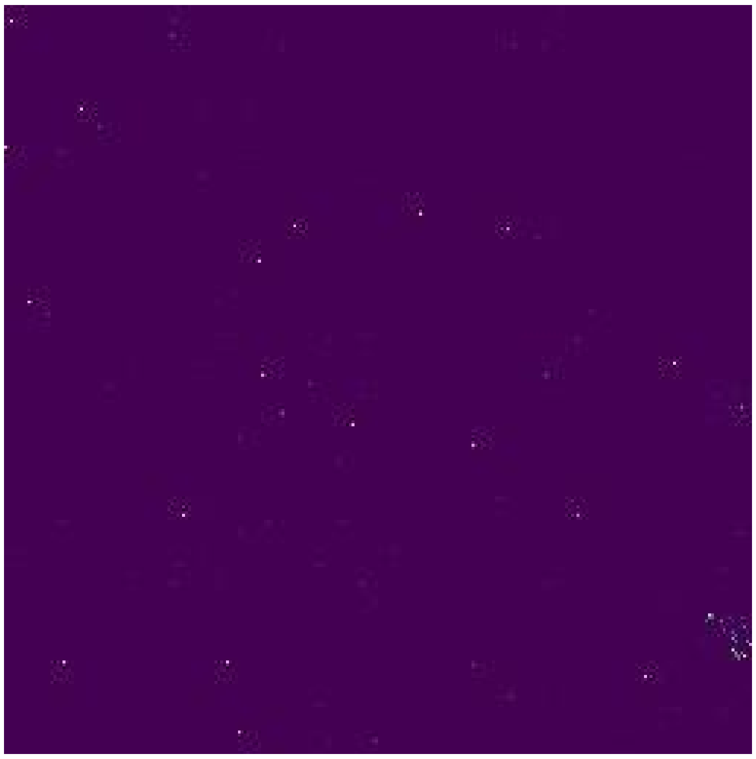
233.0.0.0/8: Multicast



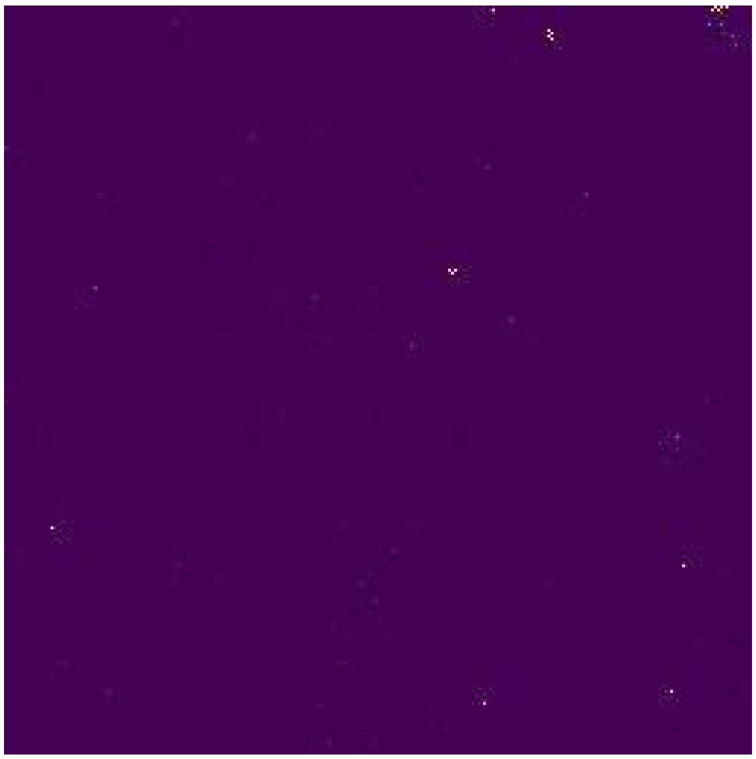
234.0.0.0/8: Multicast



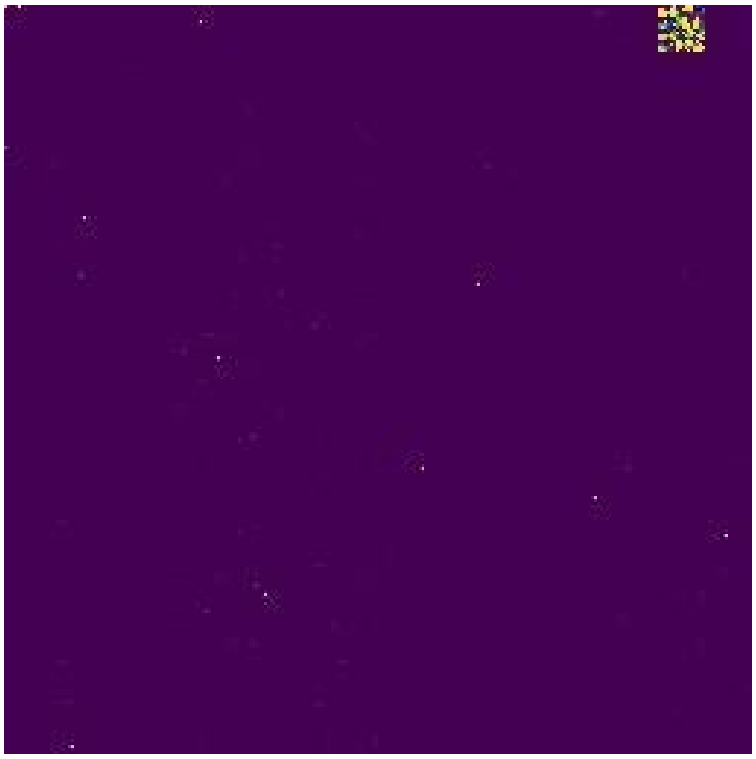
235.0.0.0/8: Multicast



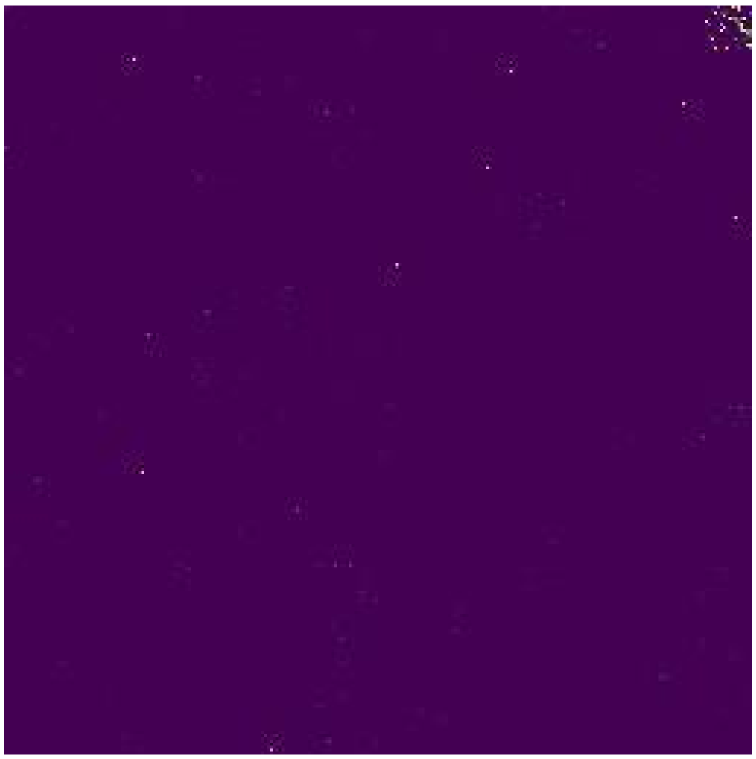
236.0.0.0/8: Multicast



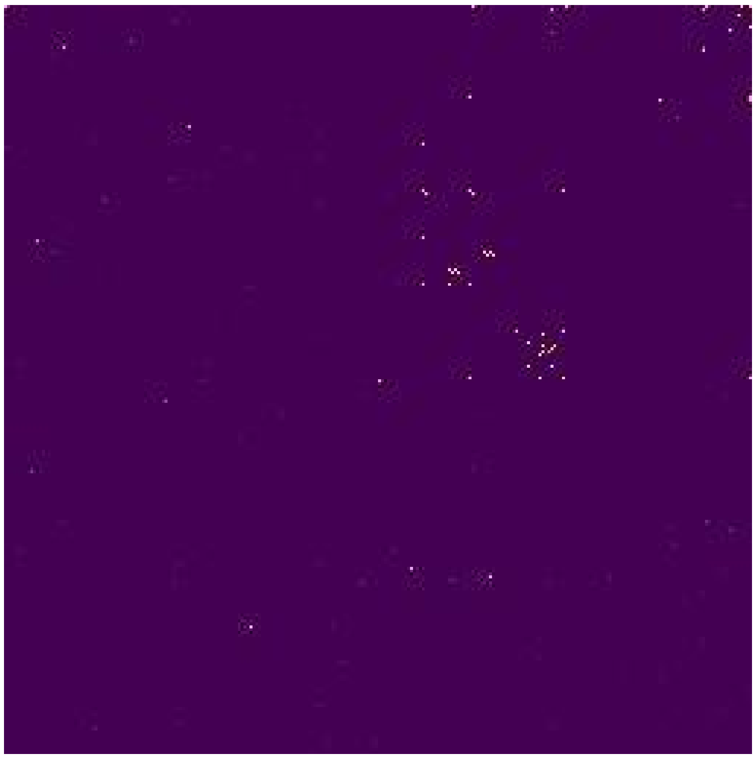
237.0.0.0/8: Multicast



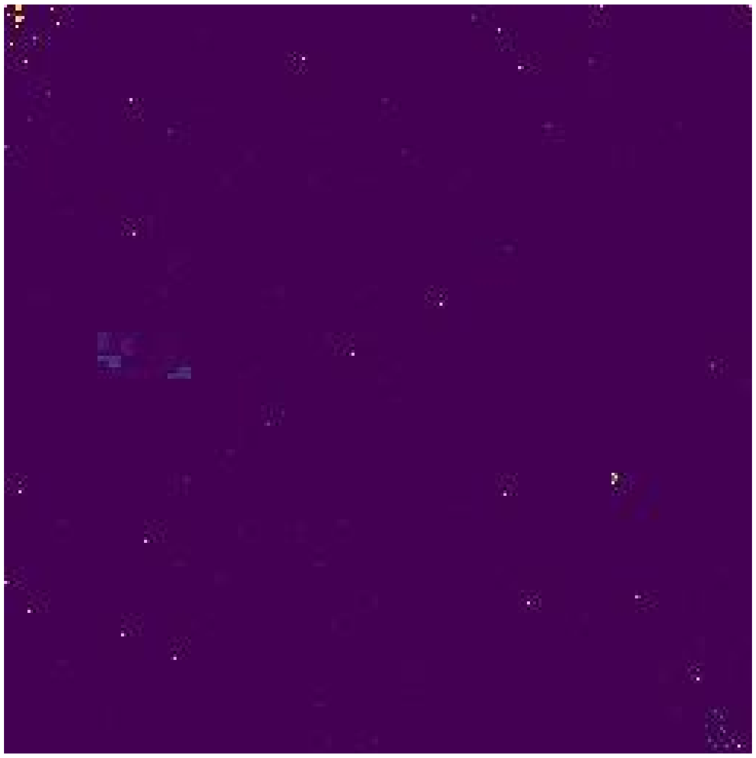
238.0.0.0/8: Multicast



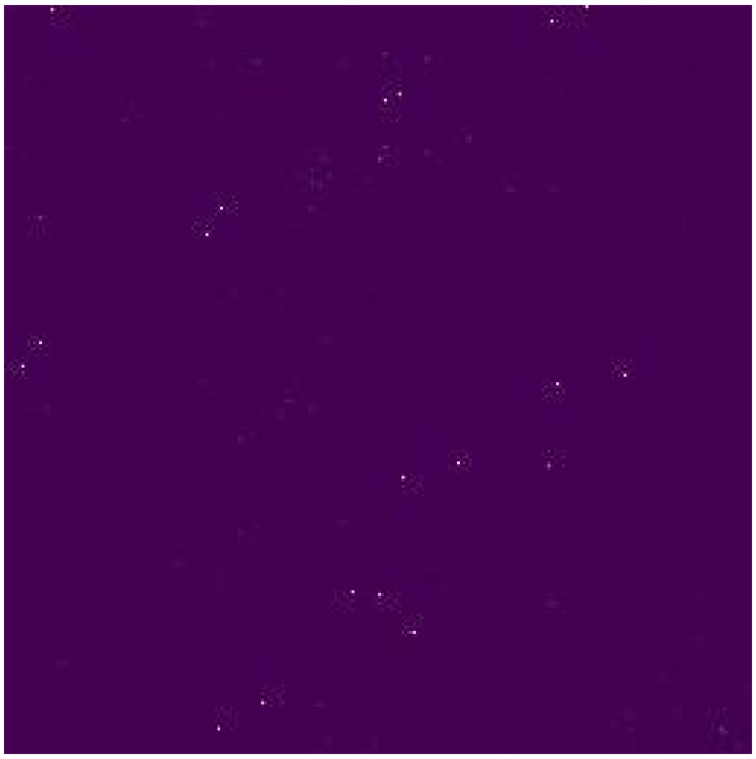
239.0.0.0/8: Multicast



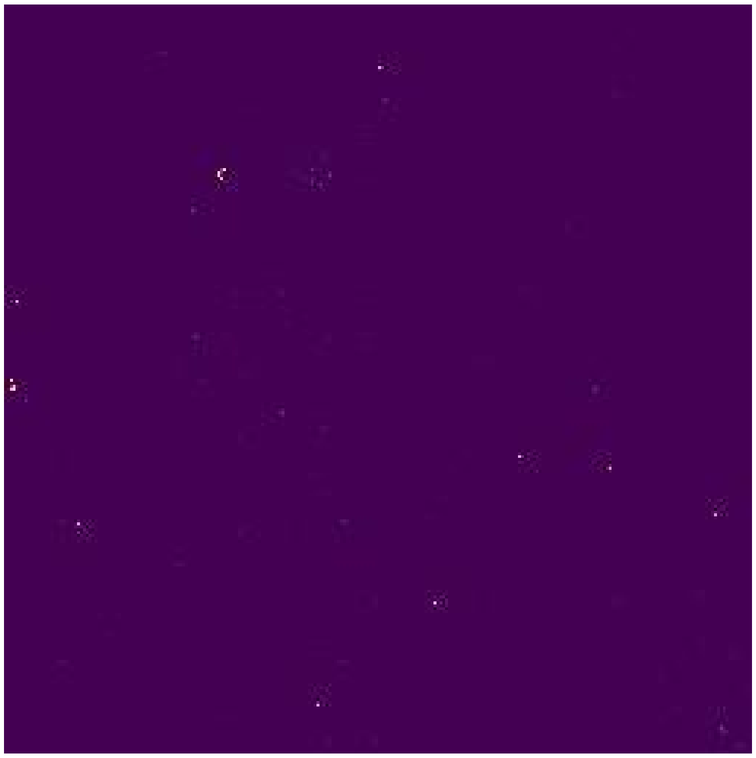
240.0.0.0/8: Class “E”



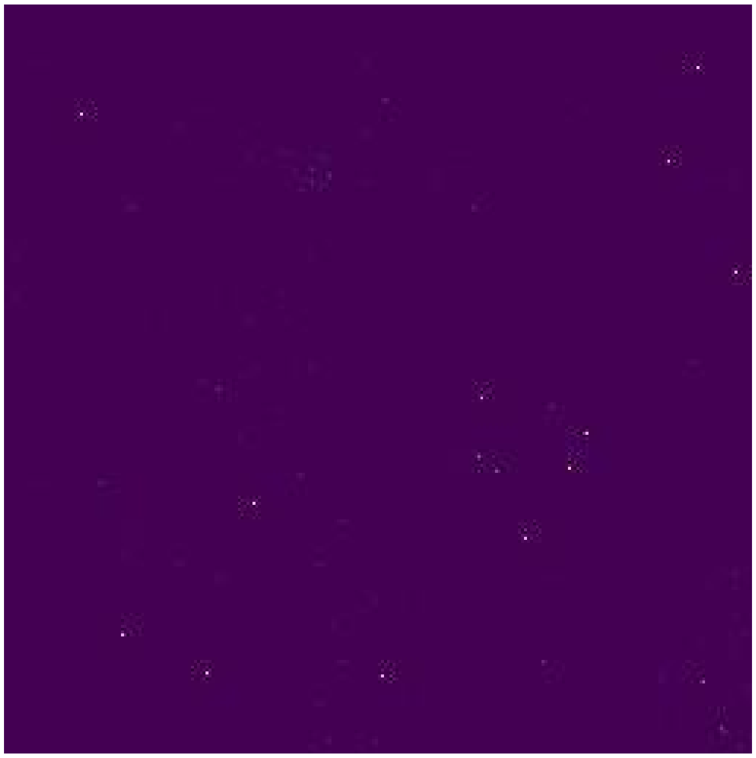
241.0.0.0/8: Class “E”



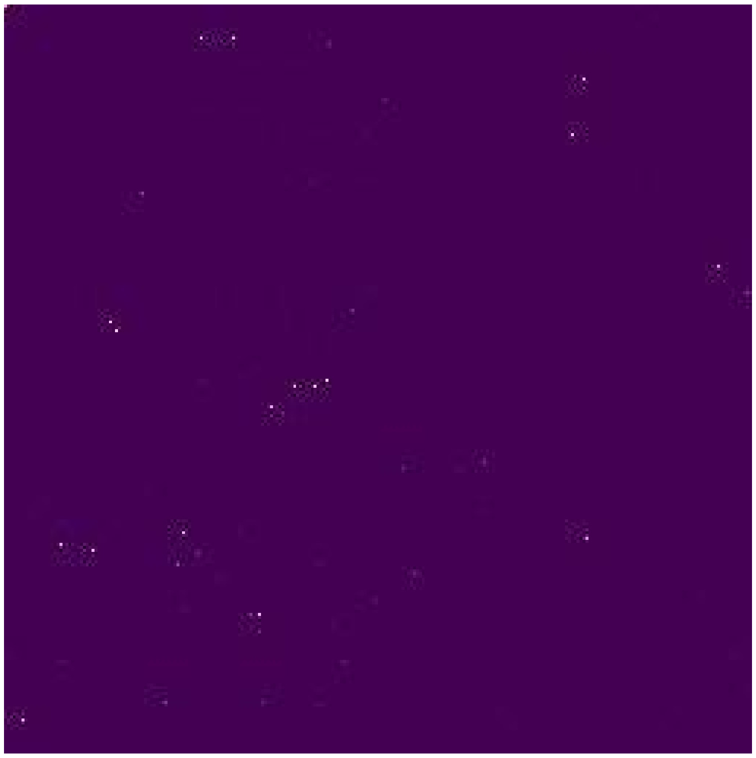
242.0.0.0/8: Class “E”



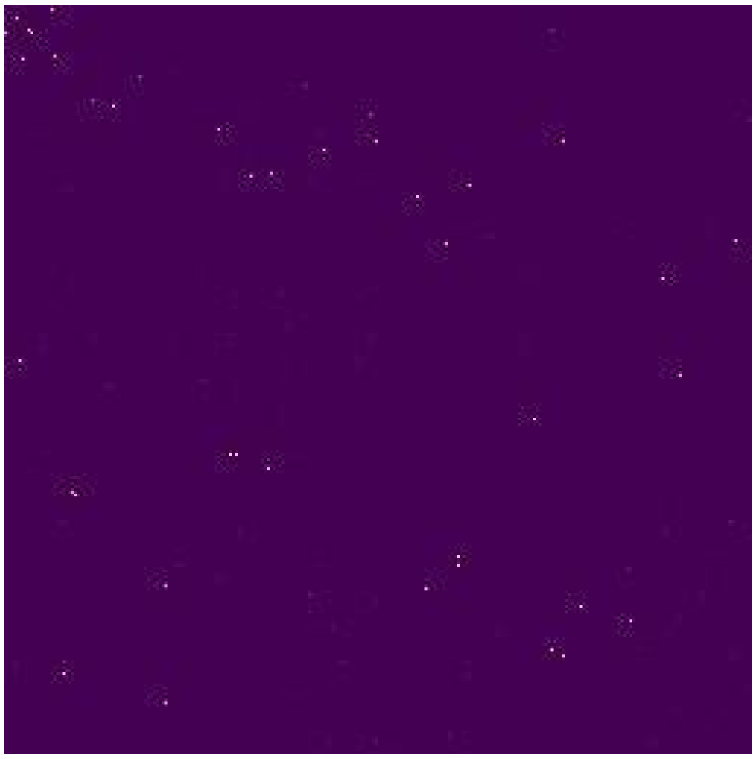
243.0.0.0/8: Class “E”



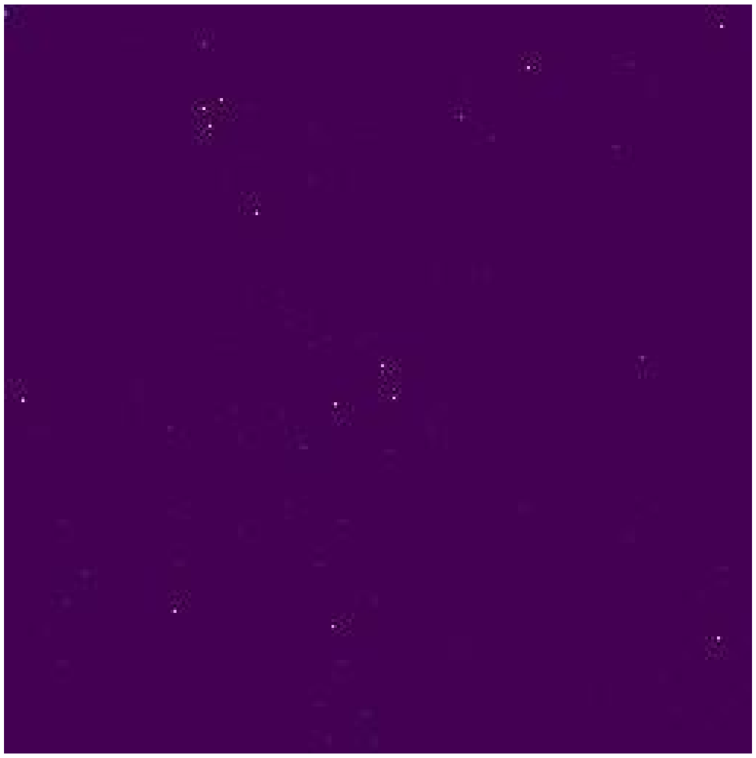
244.0.0.0/8: Class “E”



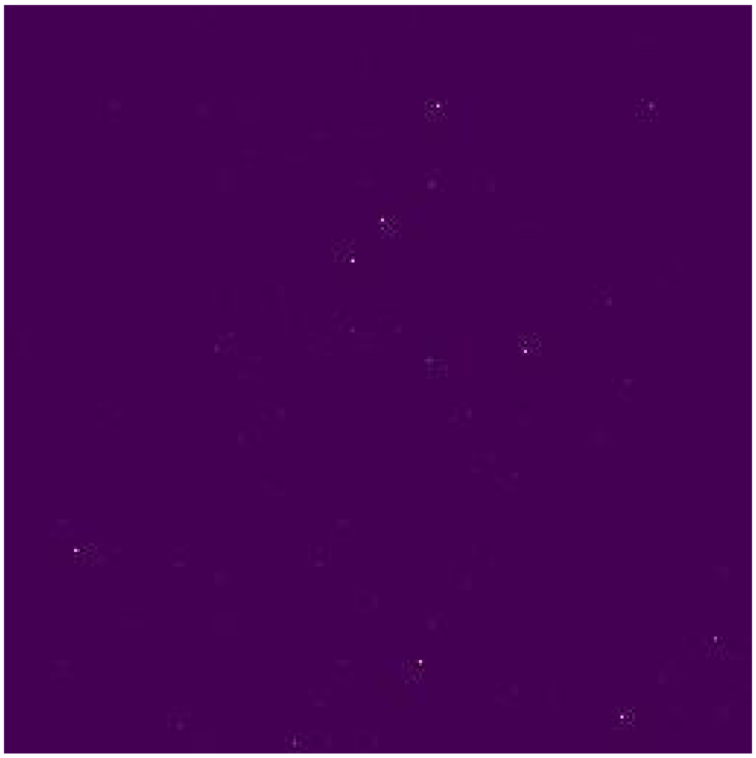
245.0.0.0/8: Class “E”



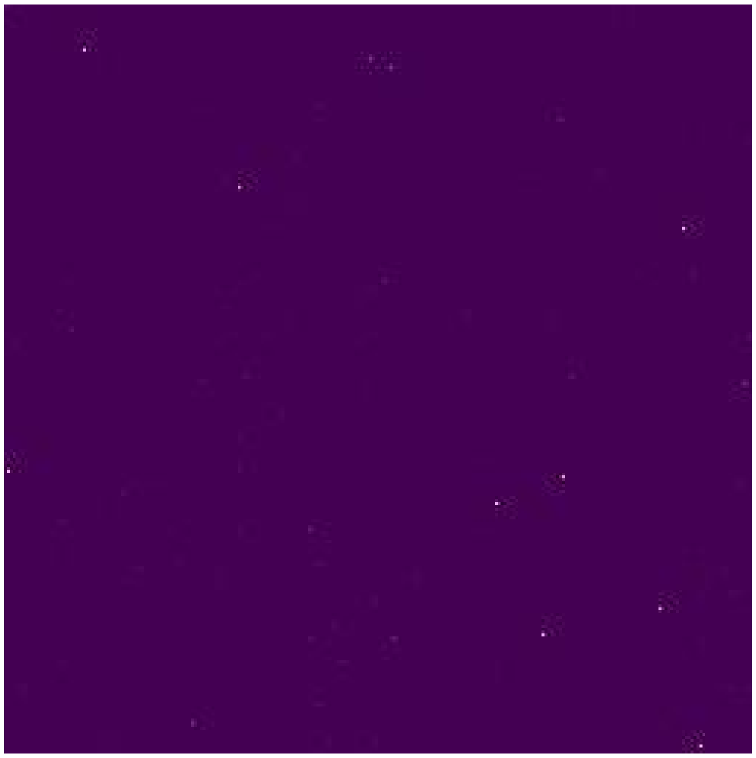
246.0.0.0/8: Class “E”



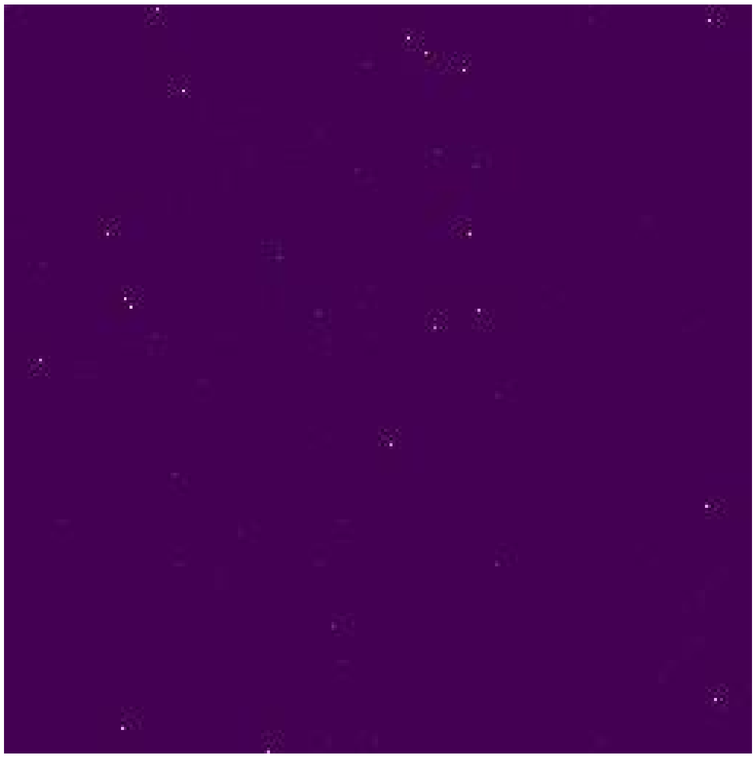
247.0.0.0/8: Class “E”



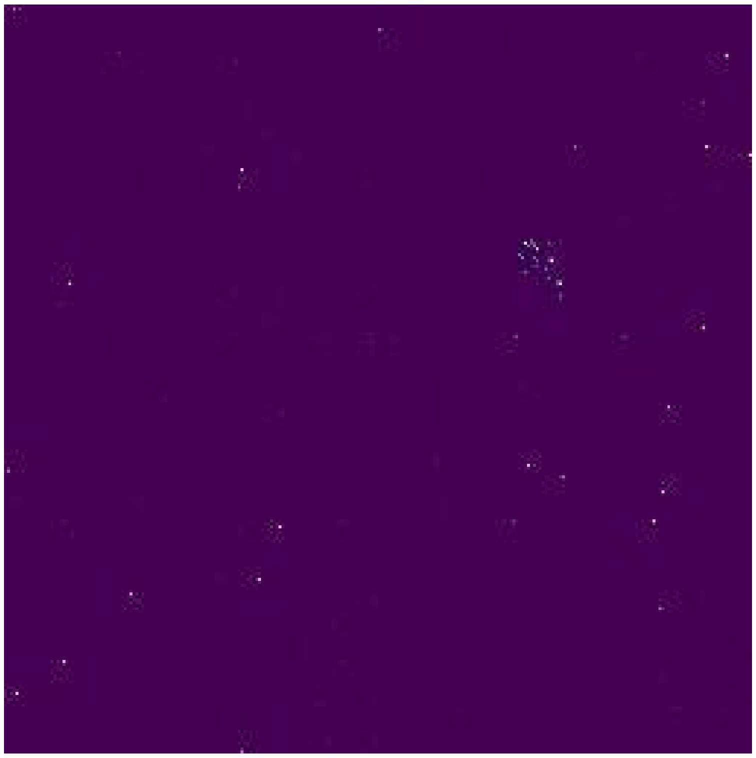
248.0.0.0/8: Class “E”



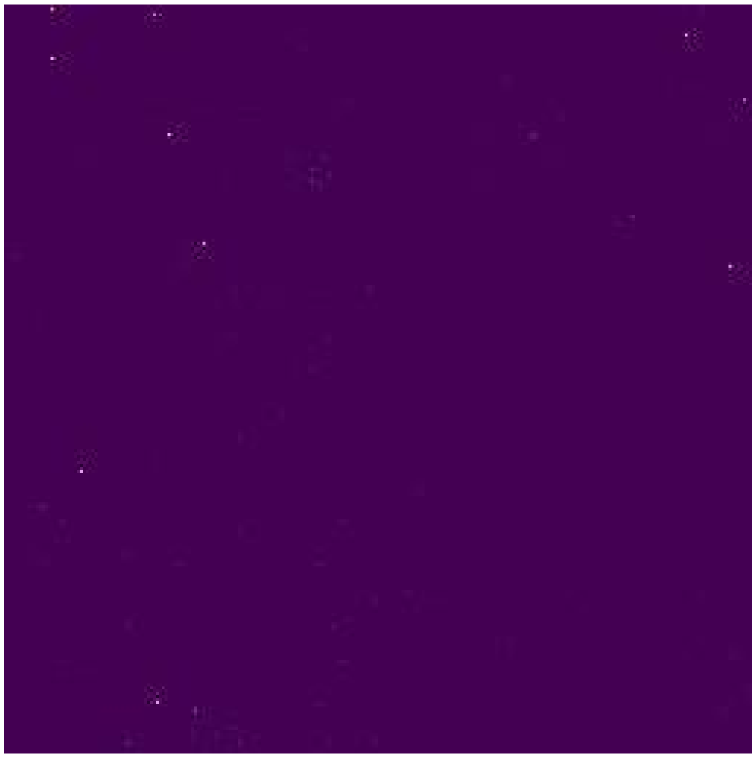
249.0.0.0/8: Class “E”



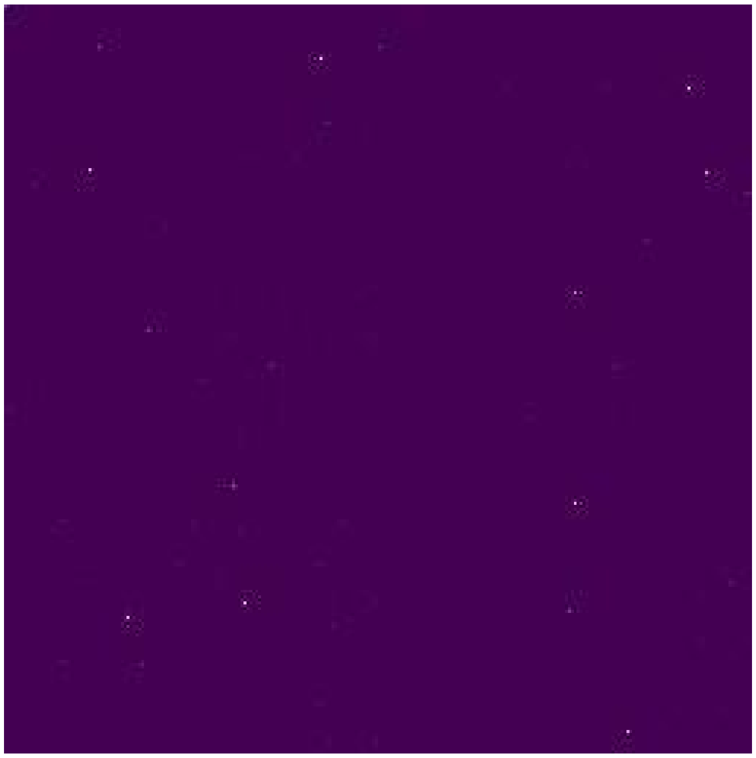
250.0.0.0/8: Class “E”



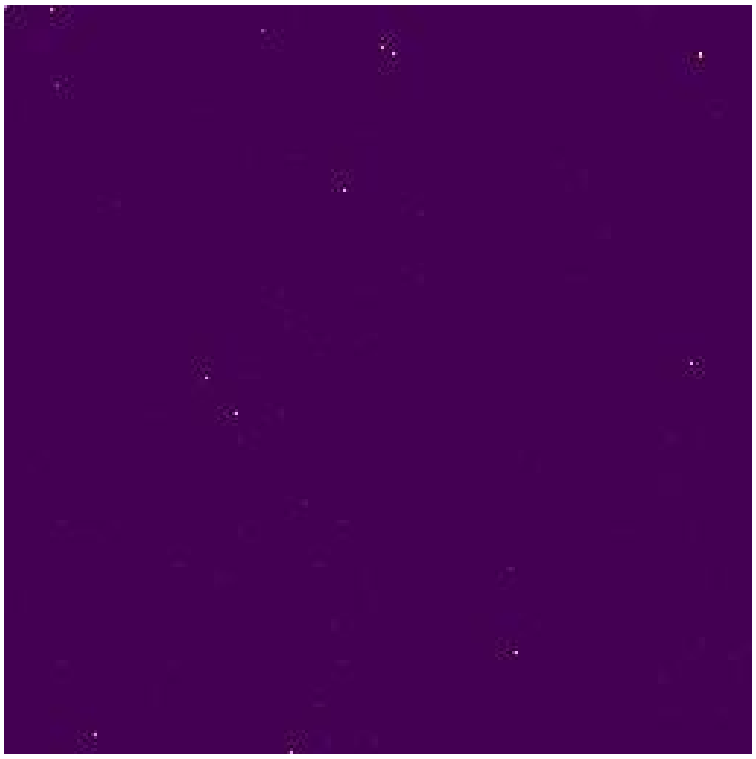
251.0.0.0/8: Class “E”



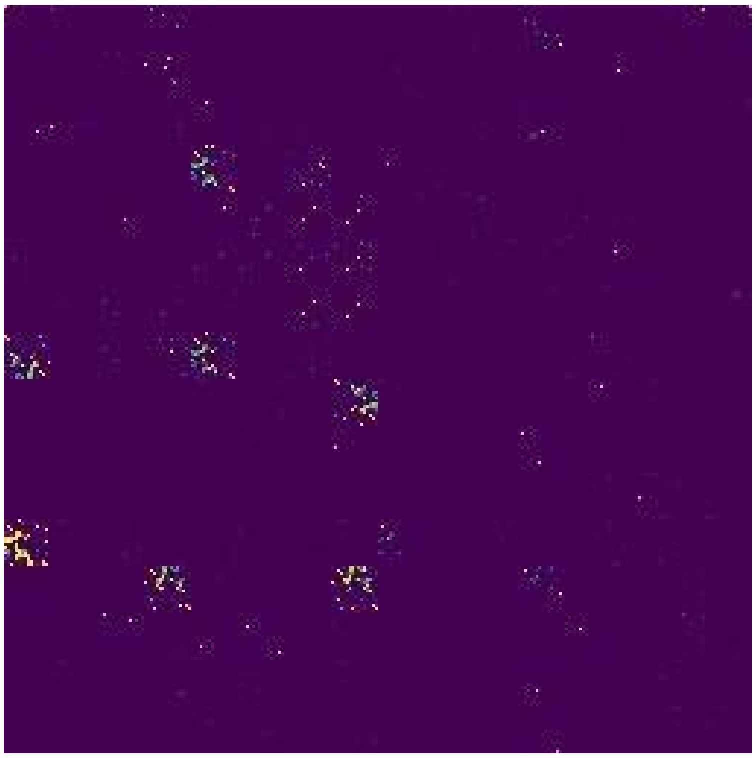
252.0.0.0/8: Class “E”



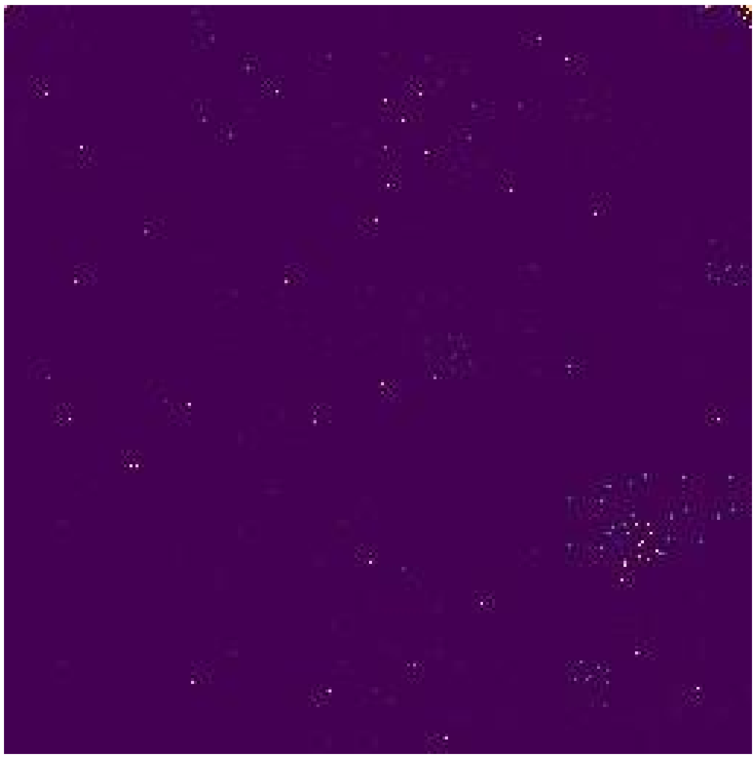
253.0.0.0/8: Class “E”



254.0.0.0/8: Class “E”



255.0.0.0/8: Class “E”



Conclusion

The allocation and usage of the IPv4 address space has changed dramatically in the last 6 years. Nearly 7.4 times more addresses are now routinely seen in DNSDB over the course of a month since 2017.

Not content, we decided to look deeper in his report. We looked at aggregated cache miss counts associated with each /24 netblock for the entire Internet IPv4 address space. We collected data for that review by performing over 16.7 million timefenced DNSDB API `-V summarize` queries. Queries were time-fenced to the preceding 90 days to ensure timeliness and relevance.

In reviewing those results, we identify some /24 netblocks with unusually large aggregate counts – aggregate cache miss counts in excess of **175 billion**. We dug in on some of those "heavy hitters," reporting on the domains that appear to be most heavily contributing to those huge cache miss counts. At the other end of the cache miss "aggregate count" spectrum, we found that 75% of all /24's have aggregate cache miss counts of 100 or fewer. Most of the Internet is not popular.

To drive this point home, violin plots and individual Hilbert curves were produced and provided for each /8. These visualizations show the uneven usage of the IPv4 space across /8s. If we assume that nearly all IPv4 address space has now been allocated, these visualizations highlight which parts of the Internet are associated with DNS and hosting domains and which parts are not.

In concluding this report, we hope you now have a somewhat better sense of IPv4 address utilization in Farsight DNSDB, and how you can pull data for individual `/24`'s – or even the whole Internet's address space. Finally, this report introduced and demonstrated the value of improvements to query metadata reporting DNSNB's command line client, `dnsdbq`, when used for bulk queries.

“Four Straight Dance Songs”
McClellan, Robedeaux, and Stoner

https://www.youtube.com/watch?v=6BHy2YFGLtE