Installing The Requirements

In [1]:
```
# pip install torch torchvision matplotlib
```

Loading The Requirements

In [2]:
```
import torch
import torchvision.models as models
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
import math
import os
```

Load and preprocess image

In [3]:
```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],  # VGG16 normalization
                         std=[0.229, 0.224, 0.225])
])

image_path = "/kaggle/input/art-style-transfert/a-women-portrait-beautiful-y
original_image = Image.open(image_path).convert("RGB")
input_tensor = transform(original_image).unsqueeze(0)  # [1, 3, 224, 224]
```

Load VGG16 pretrained model

In [4]:
```
vgg16 = models.vgg16(pretrained=True).features.eval()
```

```
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: Us
erWarning: The parameter 'pretrained' is deprecated since 0.13 and may be re
moved in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: Us
erWarning: Arguments other than a weight enum or `None` for 'weights' are de
precated since 0.13 and may be removed in the future. The current behavior i
s equivalent to passing `weights=VGG16_Weights.IMAGENET1K_V1`. You can also
use `weights=VGG16_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

Define conv block layer indices

In [5]:
```
block_layers = [4, 9, 16, 23, 30]
```

Extract features after each block

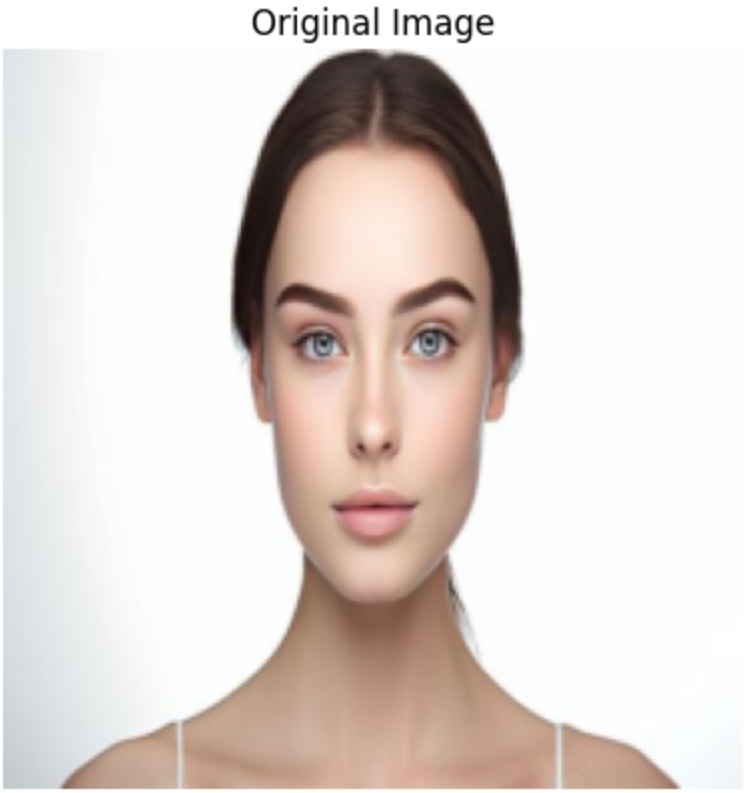
In [6]:
```
features = []
x = input_tensor.clone()
with torch.no_grad():
    for i, layer in enumerate(vgg16):
        x = layer(x)
        if i in block_layers:
            features.append(x.cpu())
```

Utility: Plot all channels in a grid

In [7]:
```
def plot_feature_maps(feature_map, title_prefix="Block"):
    num_channels = feature_map.shape[1]
    grid_cols = 8
    grid_rows = math.ceil(num_channels / grid_cols)

    fig, axs = plt.subplots(grid_rows, grid_cols, figsize=(grid_cols * 1.5,
    axs = axs.flatten()

    for i in range(num_channels):
        axs[i].imshow(feature_map[0, i], cmap='viridis')
        axs[i].axis('off')
        axs[i].set_title(f"C{i}")

    for i in range(num_channels, len(axs)):
        axs[i].axis('off')

    plt.suptitle(f"{title_prefix} - All {num_channels} Feature Maps", fontsi
    plt.tight_layout()
    plt.subplots_adjust(top=0.92)
    plt.show()
```

Visualize original image

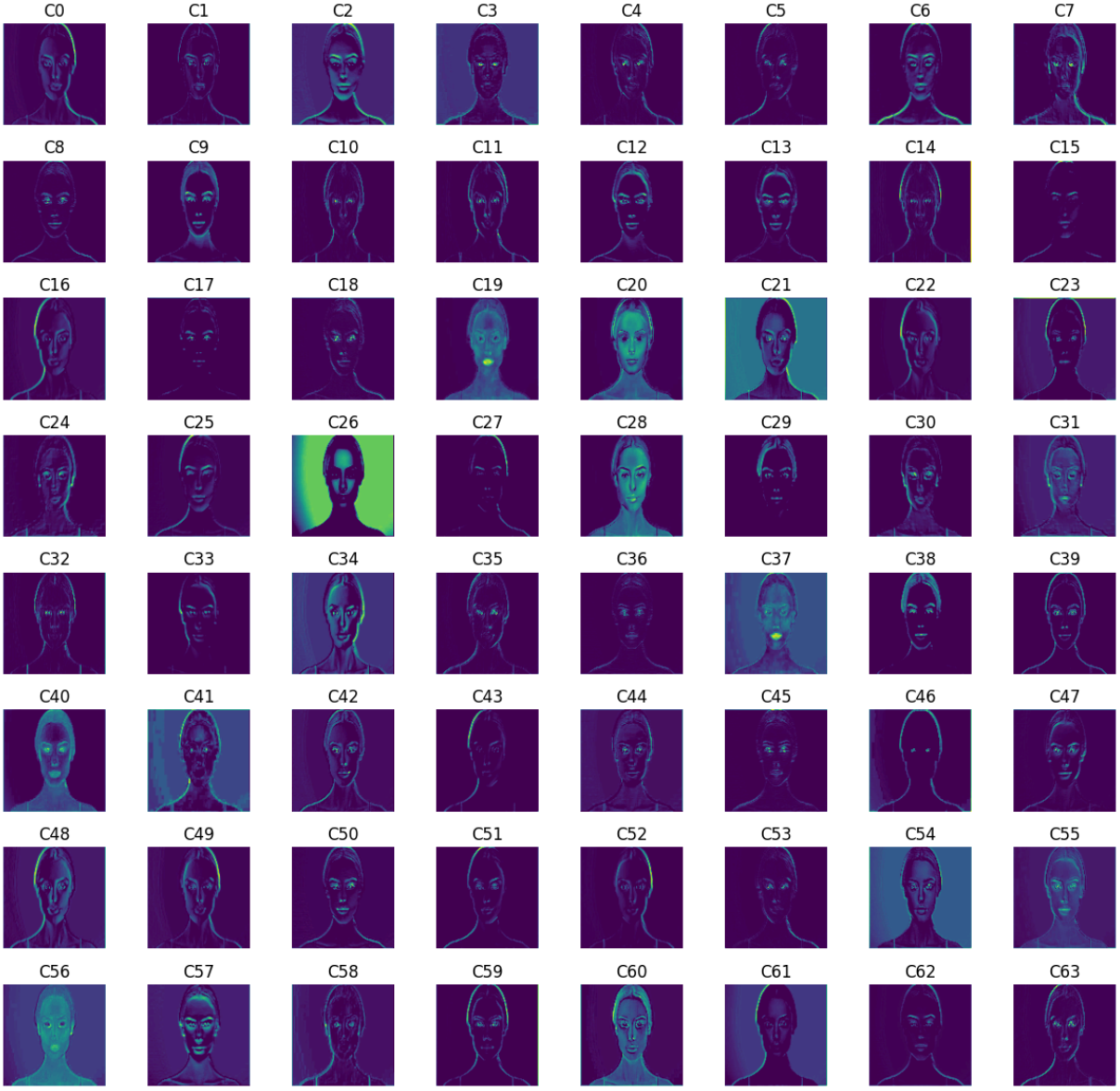
In [8]:
```
def denormalize(tensor):
    mean = torch.tensor([0.485, 0.456, 0.406]).reshape(3, 1, 1)
    std = torch.tensor([0.229, 0.224, 0.225]).reshape(3, 1, 1)
    return tensor * std + mean

plt.imshow(np.transpose(denormalize(input_tensor[0]).numpy(), (1, 2, 0)))
plt.title("Original Image")
plt.axis("off")
plt.show()
```



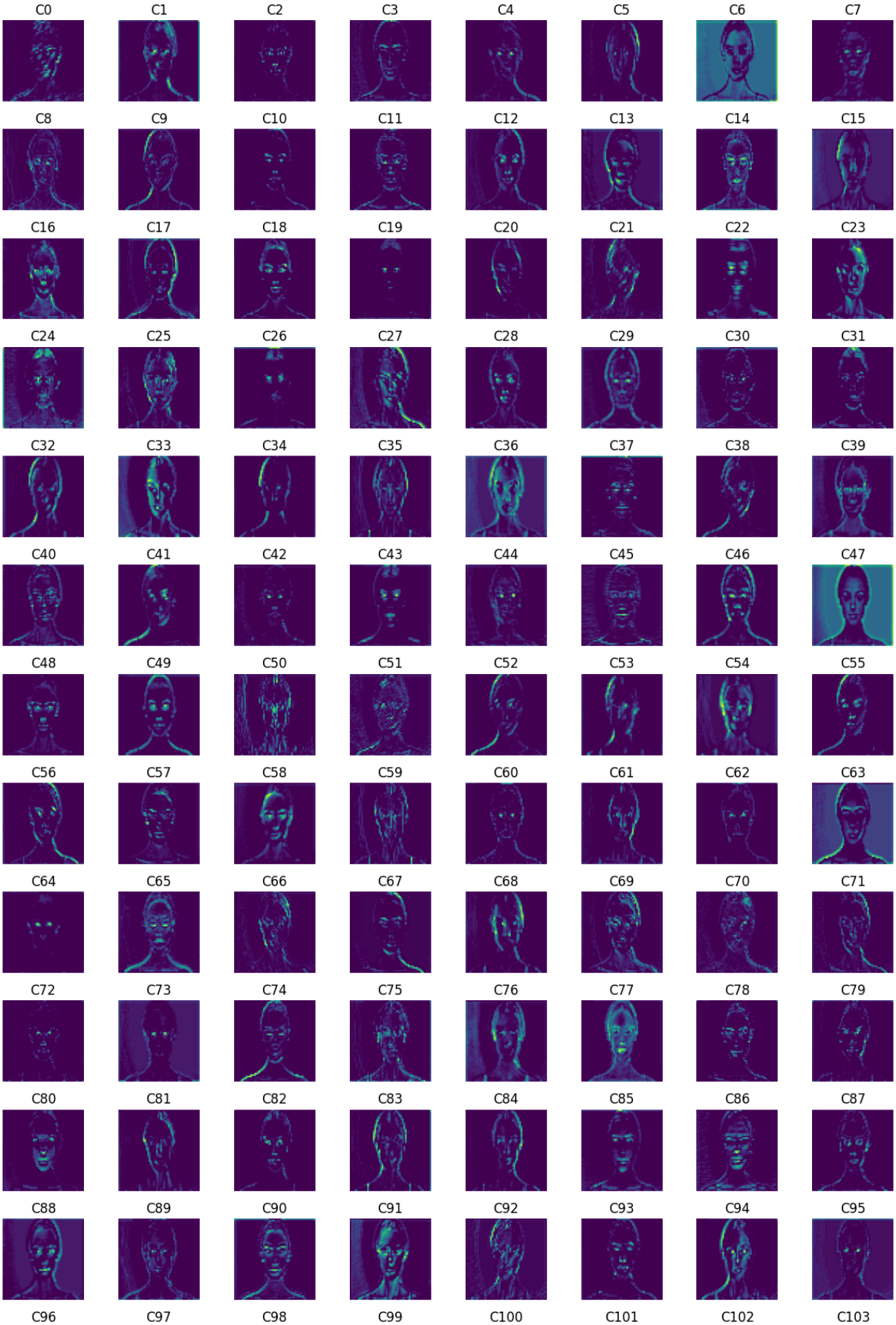
Visualize all channels for each block

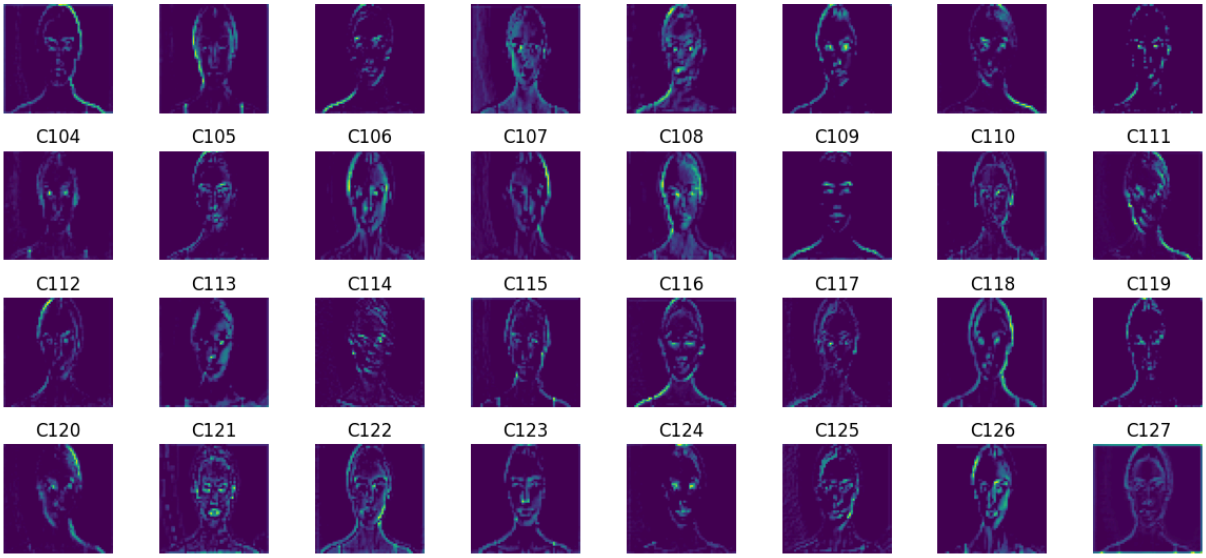
In [9]:
```
for i, feat in enumerate(features):
    plot_feature_maps(feat, title_prefix=f"Block {i+1}")
```

Block 1 - All 64 Feature Maps

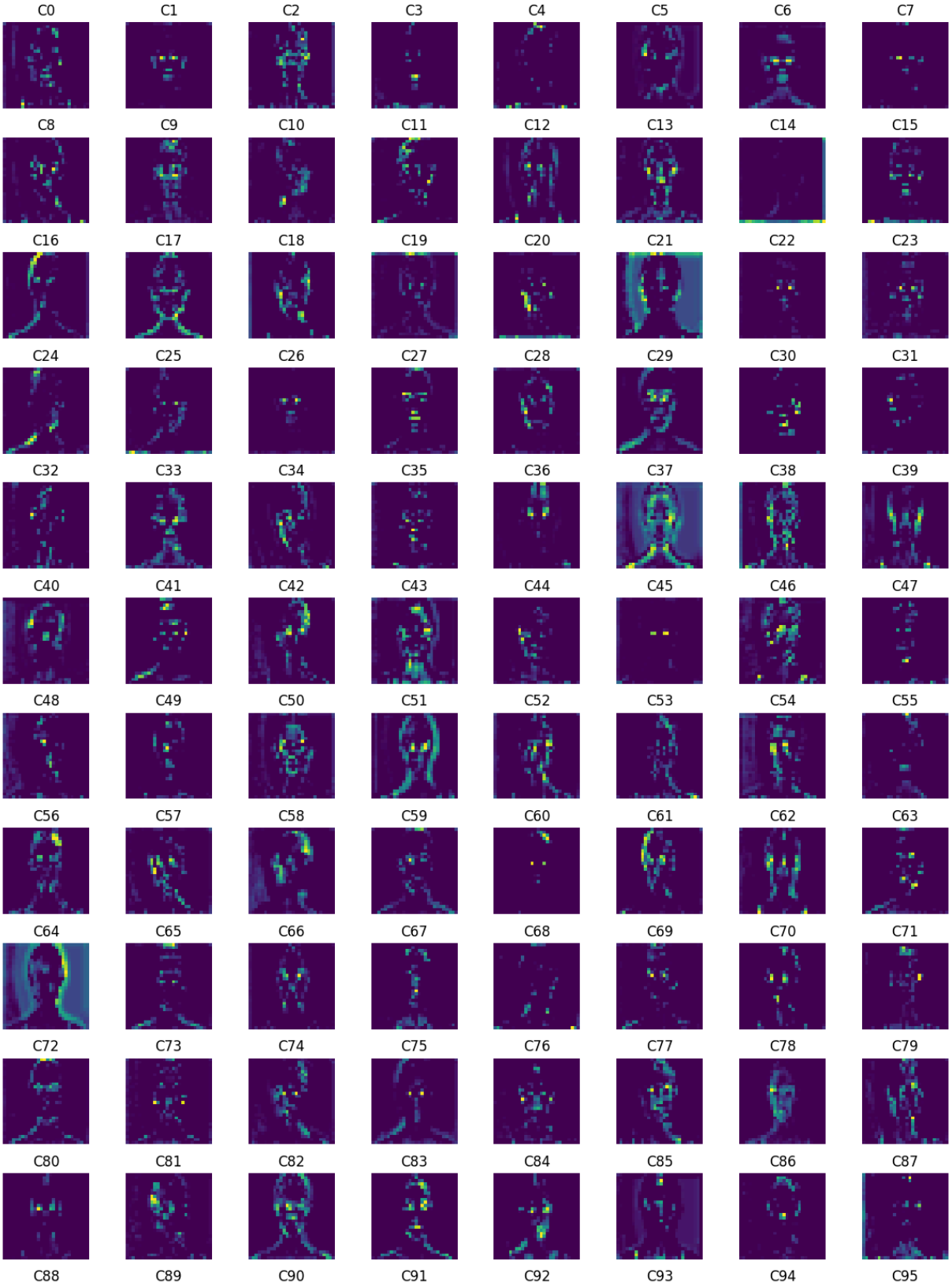


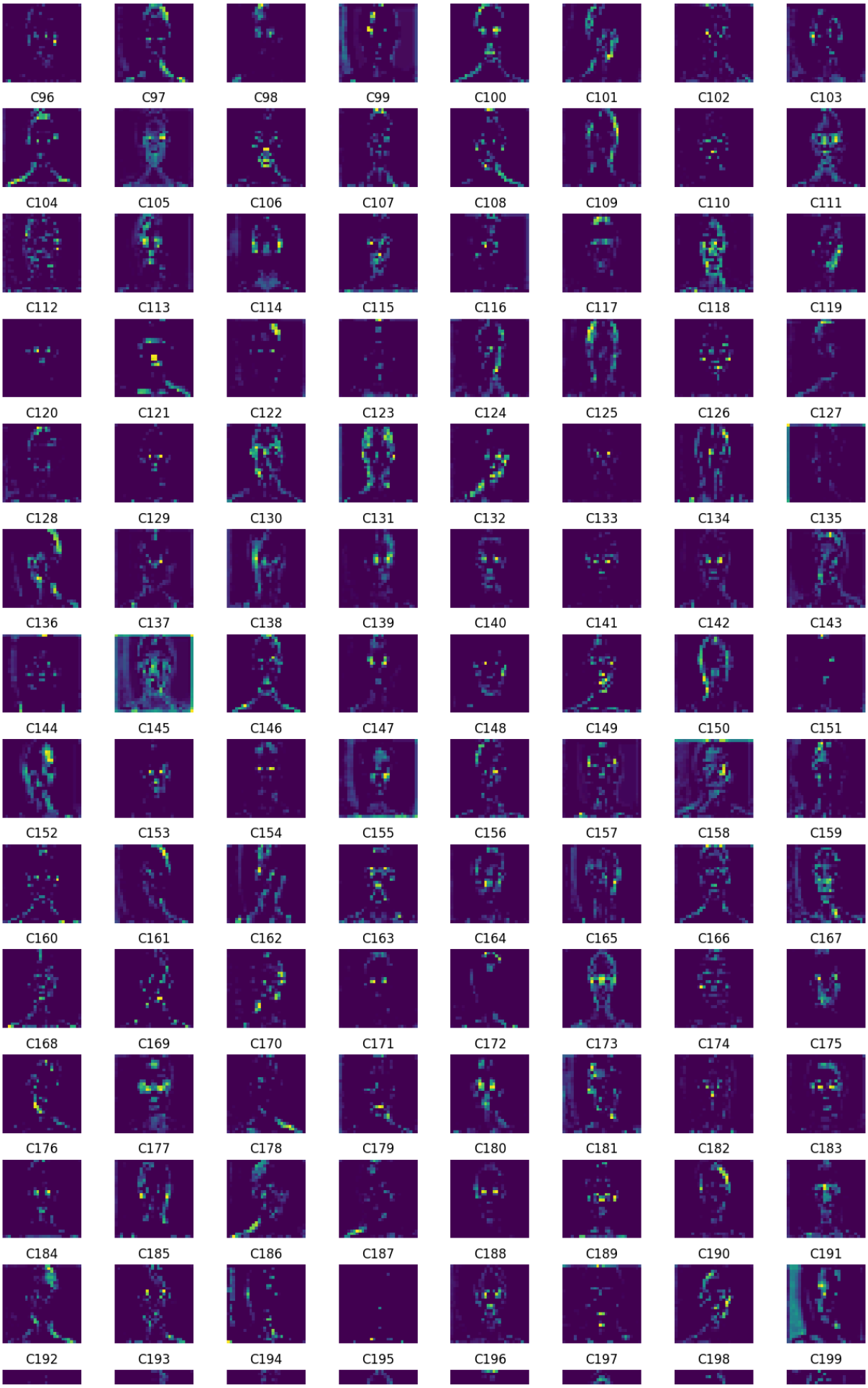
Block 2 - All 128 Feature Maps

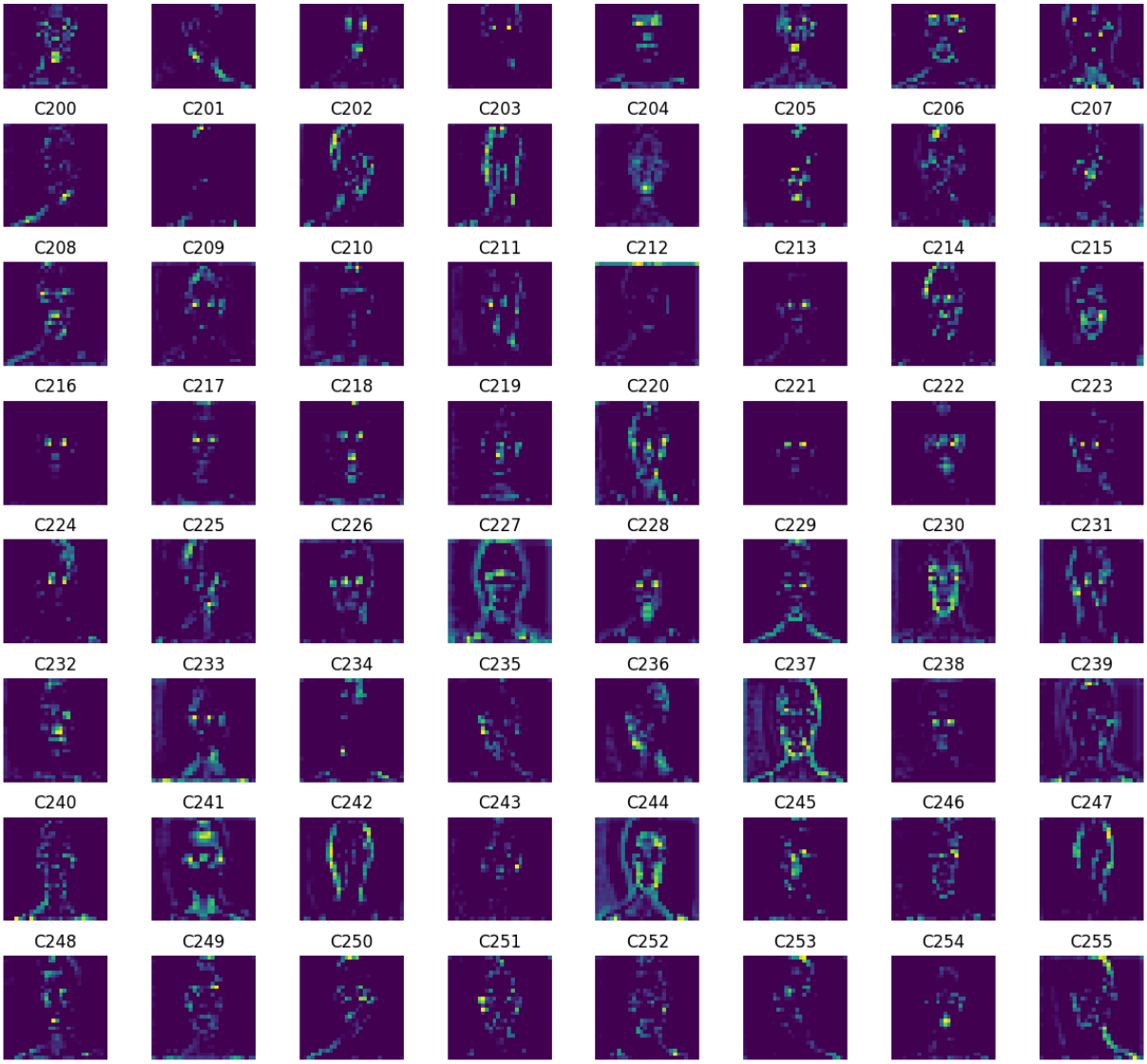



C104
C105
C106
C107
C108
C109
C110
C111
C112
C113
C114
C115
C116
C117
C118
C119
C120
C121
C122
C123
C124
C125
C126
C127

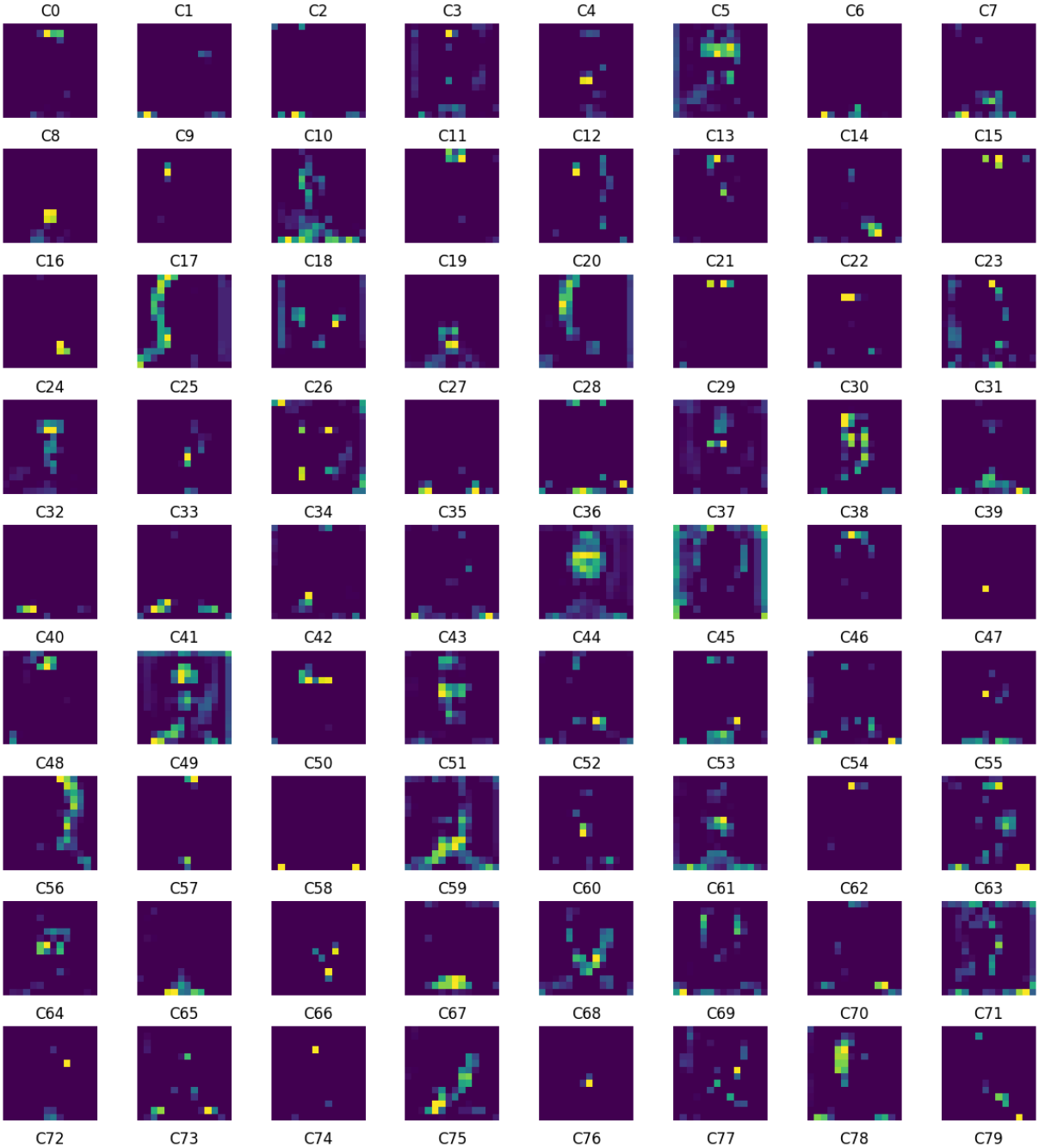
Block 3 - All 256 Feature Maps

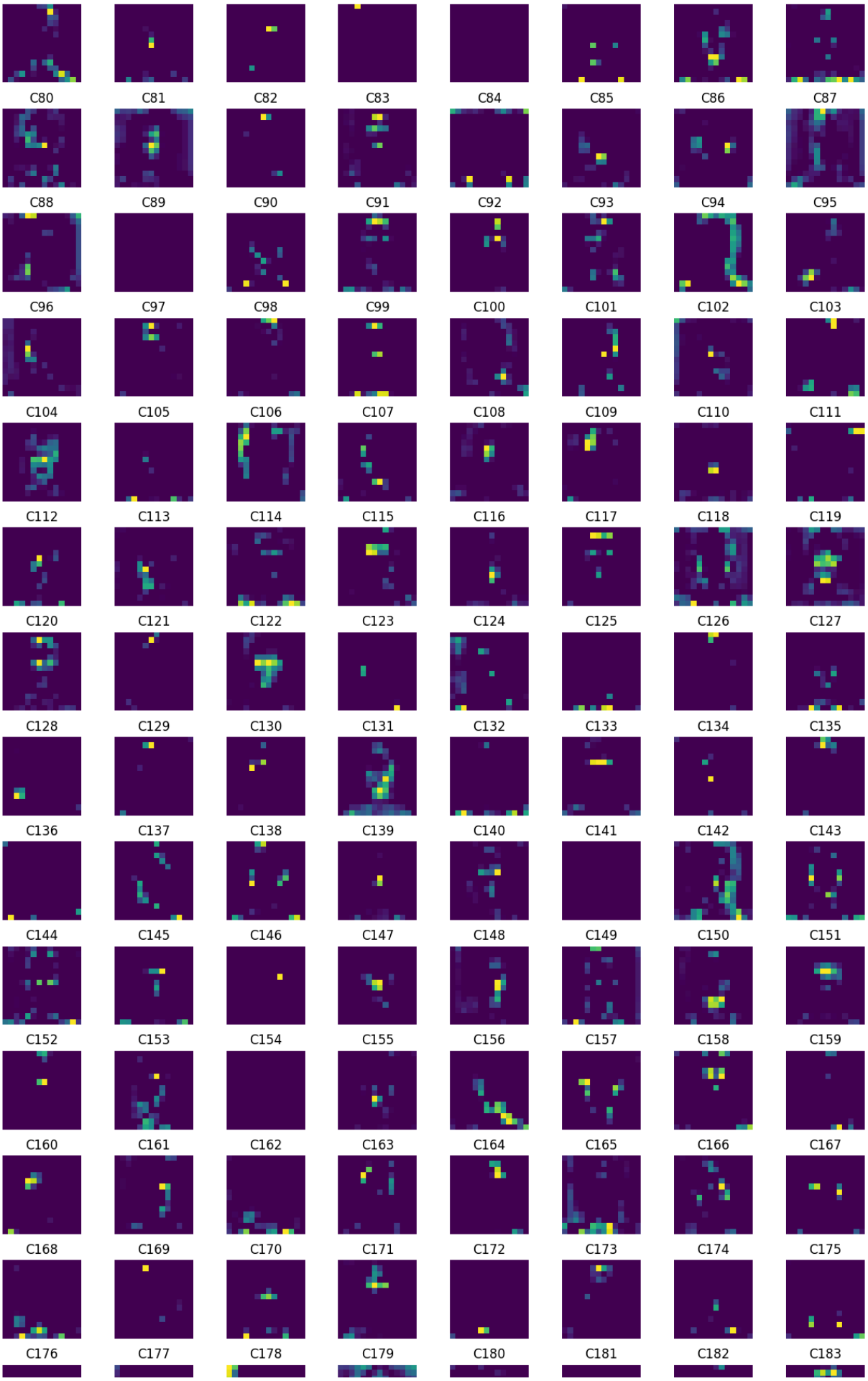


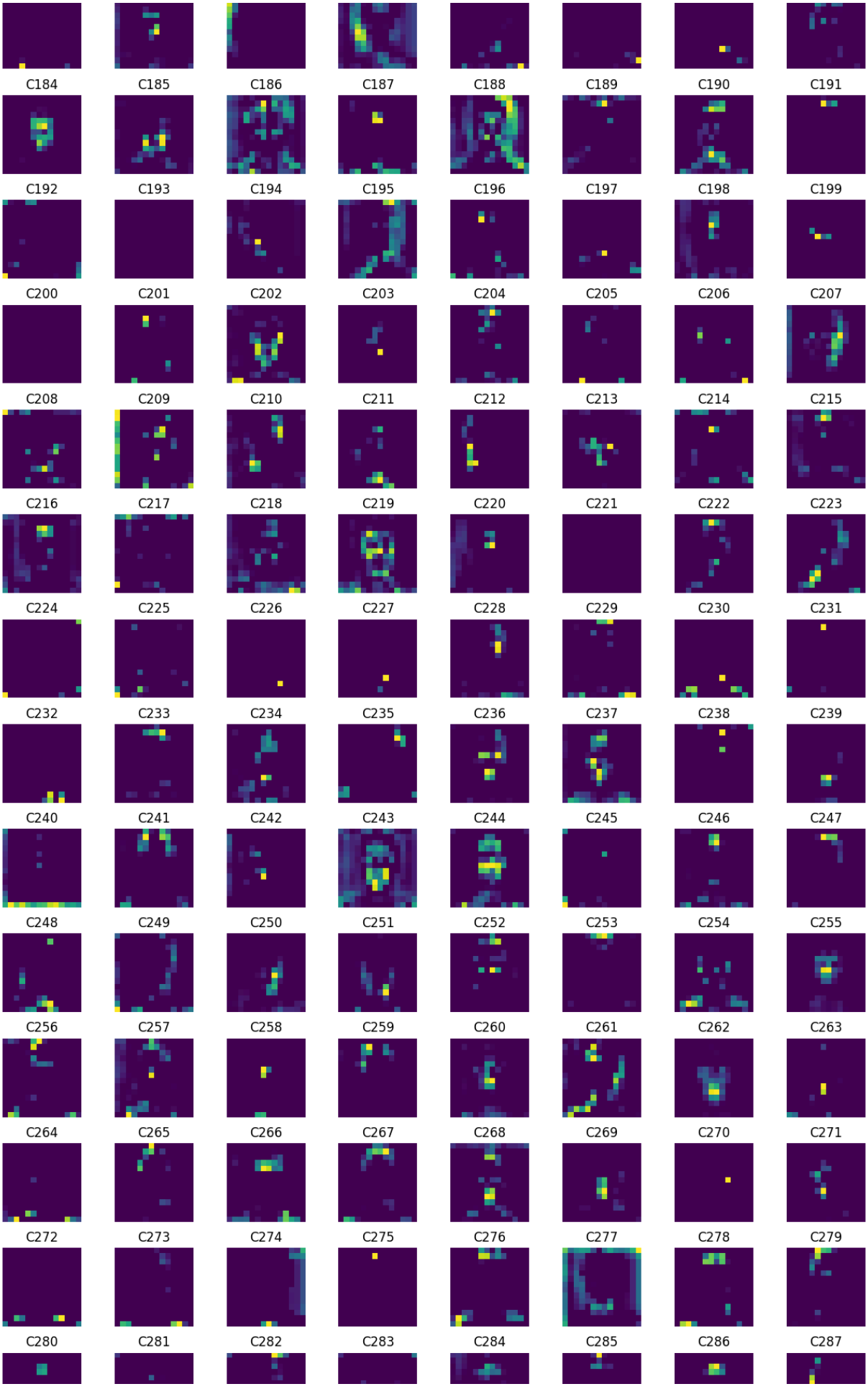

C96 C97 C98 C99 C100 C101 C102 C103
C104 C105 C106 C107 C108 C109 C110 C111
C112 C113 C114 C115 C116 C117 C118 C119
C120 C121 C122 C123 C124 C125 C126 C127
C128 C129 C130 C131 C132 C133 C134 C135
C136 C137 C138 C139 C140 C141 C142 C143
C144 C145 C146 C147 C148 C149 C150 C151
C152 C153 C154 C155 C156 C157 C158 C159
C160 C161 C162 C163 C164 C165 C166 C167
C168 C169 C170 C171 C172 C173 C174 C175
C176 C177 C178 C179 C180 C181 C182 C183
C184 C185 C186 C187 C188 C189 C190 C191
C192 C193 C194 C195 C196 C197 C198 C199

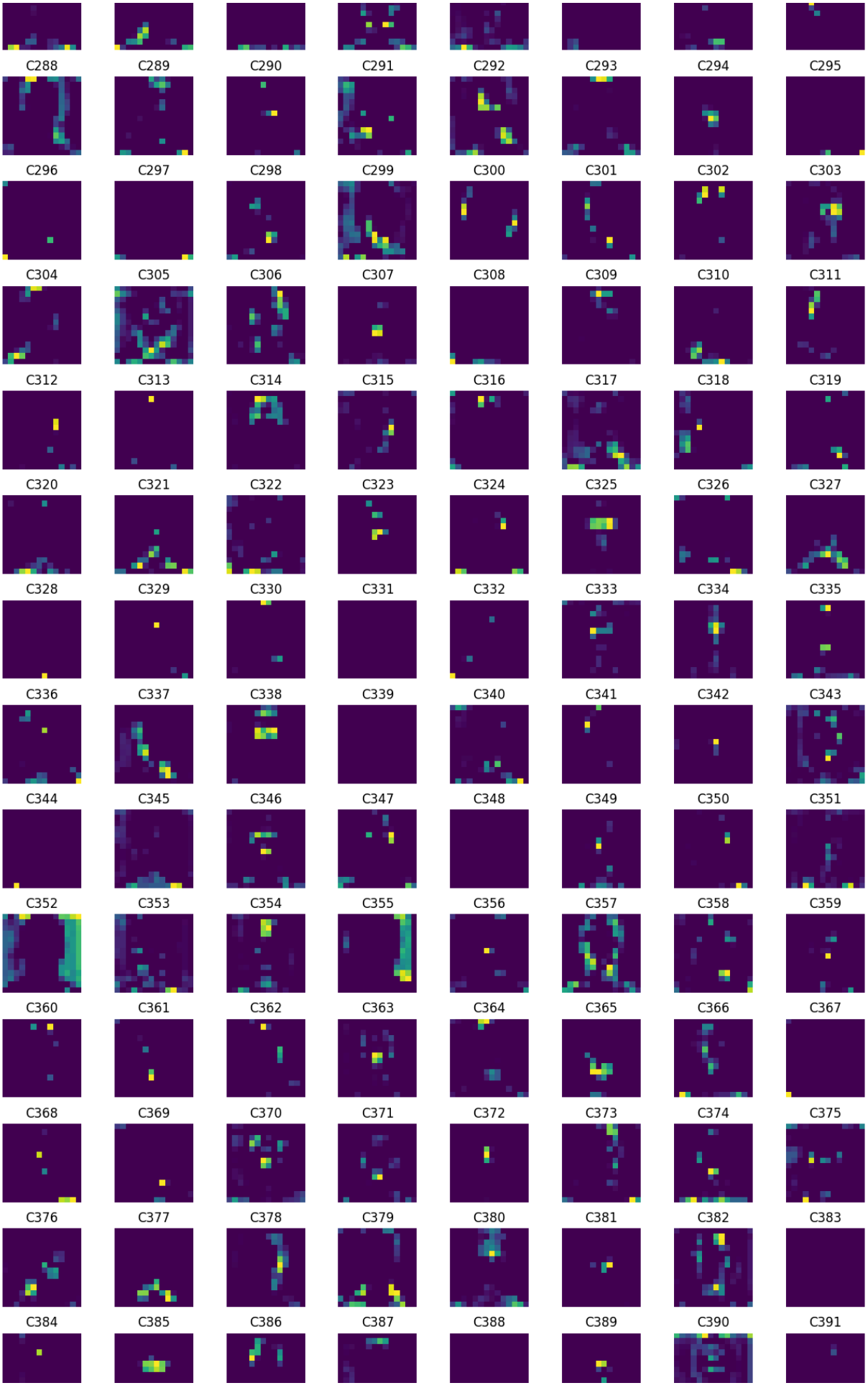

C200
C201
C202
C203
C204
C205
C206
C207
C208
C209
C210
C211
C212
C213
C214
C215
C216
C217
C218
C219
C220
C221
C222
C223
C224
C225
C226
C227
C228
C229
C230
C231
C232
C233
C234
C235
C236
C237
C238
C239
C240
C241
C242
C243
C244
C245
C246
C247
C248
C249
C250
C251
C252
C253
C254
C255

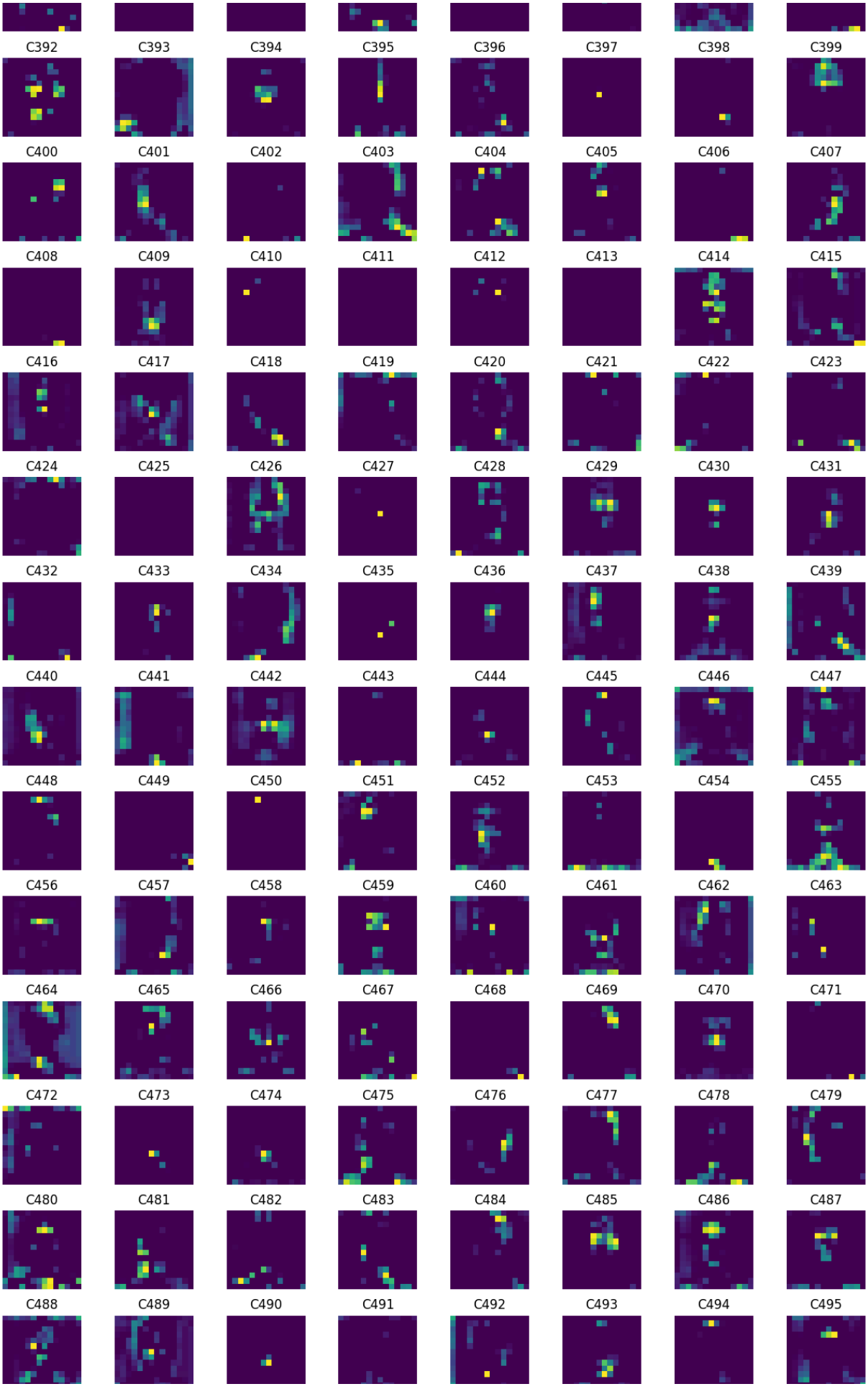
Block 4 - All 512 Feature Maps

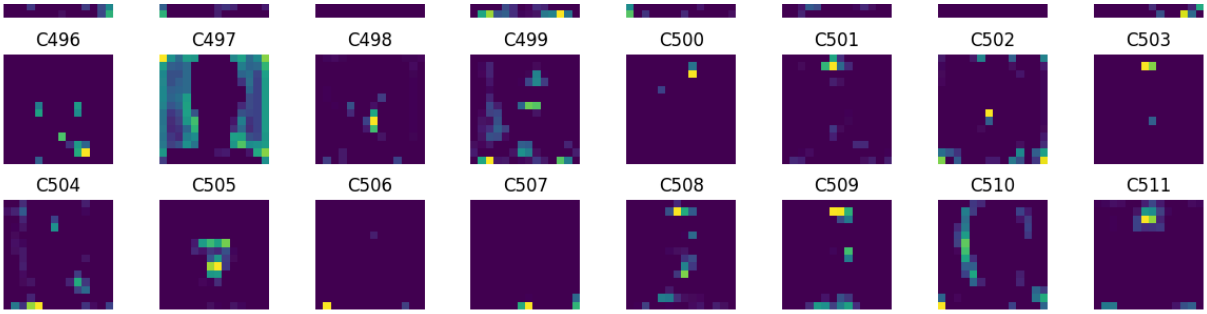



C80 C81 C82 C83 C84 C85 C86 C87
C88 C89 C90 C91 C92 C93 C94 C95
C96 C97 C98 C99 C100 C101 C102 C103
C104 C105 C106 C107 C108 C109 C110 C111
C112 C113 C114 C115 C116 C117 C118 C119
C120 C121 C122 C123 C124 C125 C126 C127
C128 C129 C130 C131 C132 C133 C134 C135
C136 C137 C138 C139 C140 C141 C142 C143
C144 C145 C146 C147 C148 C149 C150 C151
C152 C153 C154 C155 C156 C157 C158 C159
C160 C161 C162 C163 C164 C165 C166 C167
C168 C169 C170 C171 C172 C173 C174 C175
C176 C177 C178 C179 C180 C181 C182 C183

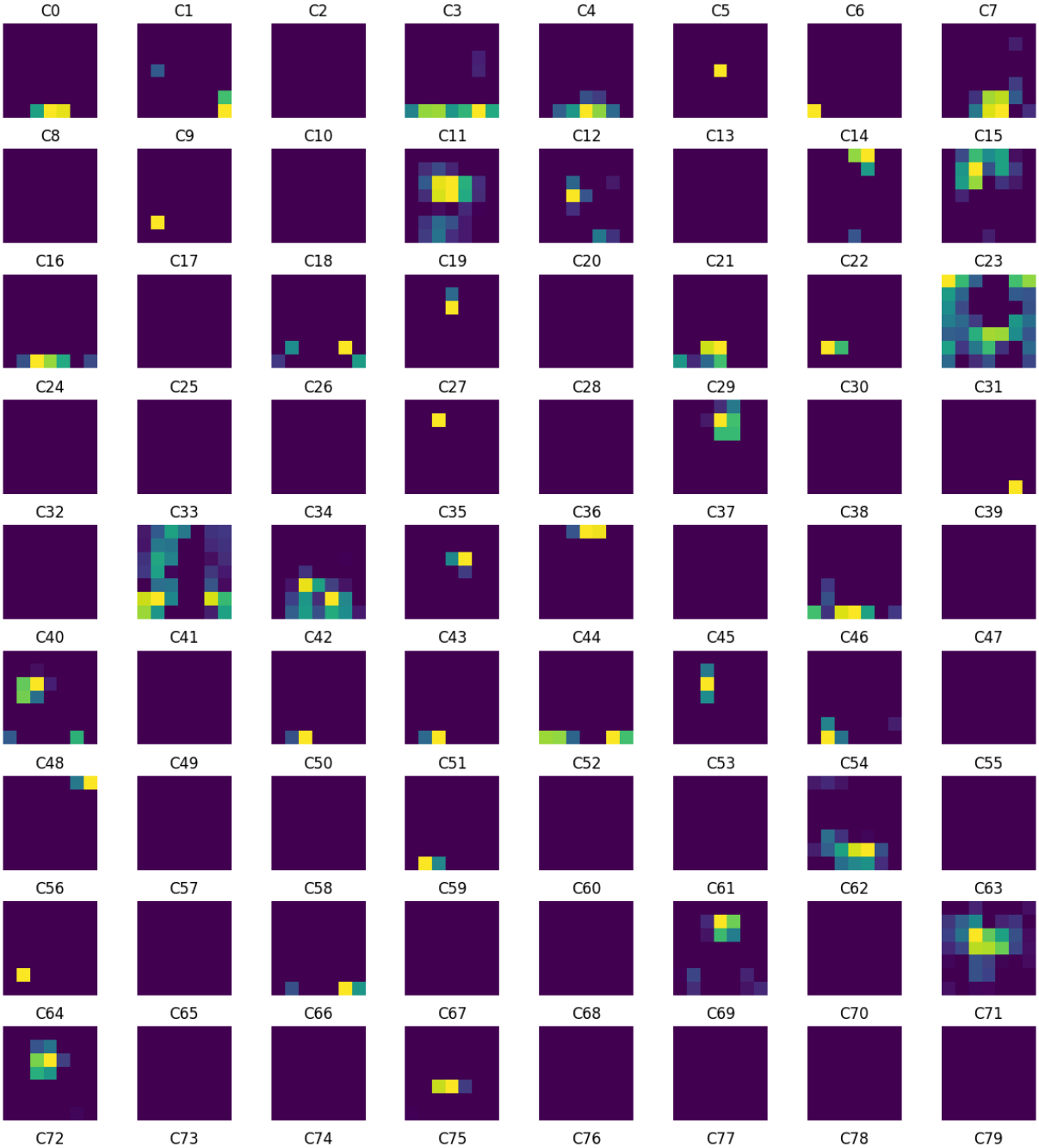

C184 C185 C186 C187 C188 C189 C190 C191
C192 C193 C194 C195 C196 C197 C198 C199
C200 C201 C202 C203 C204 C205 C206 C207
C208 C209 C210 C211 C212 C213 C214 C215
C216 C217 C218 C219 C220 C221 C222 C223
C224 C225 C226 C227 C228 C229 C230 C231
C232 C233 C234 C235 C236 C237 C238 C239
C240 C241 C242 C243 C244 C245 C246 C247
C248 C249 C250 C251 C252 C253 C254 C255
C256 C257 C258 C259 C260 C261 C262 C263
C264 C265 C266 C267 C268 C269 C270 C271
C272 C273 C274 C275 C276 C277 C278 C279
C280 C281 C282 C283 C284 C285 C286 C287

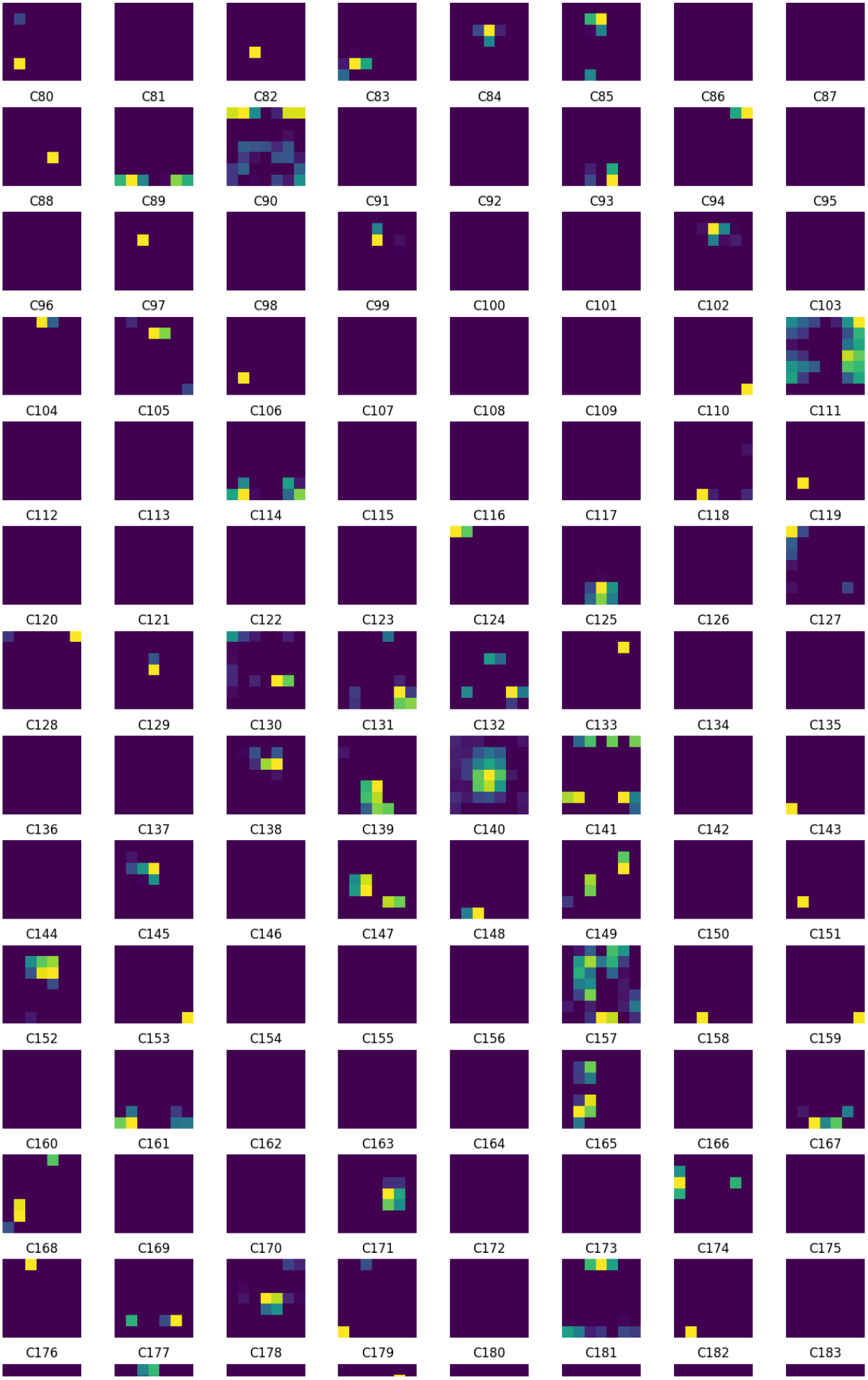

C288 C289 C290 C291 C292 C293 C294 C295
C296 C297 C298 C299 C300 C301 C302 C303
C304 C305 C306 C307 C308 C309 C310 C311
C312 C313 C314 C315 C316 C317 C318 C319
C320 C321 C322 C323 C324 C325 C326 C327
C328 C329 C330 C331 C332 C333 C334 C335
C336 C337 C338 C339 C340 C341 C342 C343
C344 C345 C346 C347 C348 C349 C350 C351
C352 C353 C354 C355 C356 C357 C358 C359
C360 C361 C362 C363 C364 C365 C366 C367
C368 C369 C370 C371 C372 C373 C374 C375
C376 C377 C378 C379 C380 C381 C382 C383
C384 C385 C386 C387 C388 C389 C390 C391

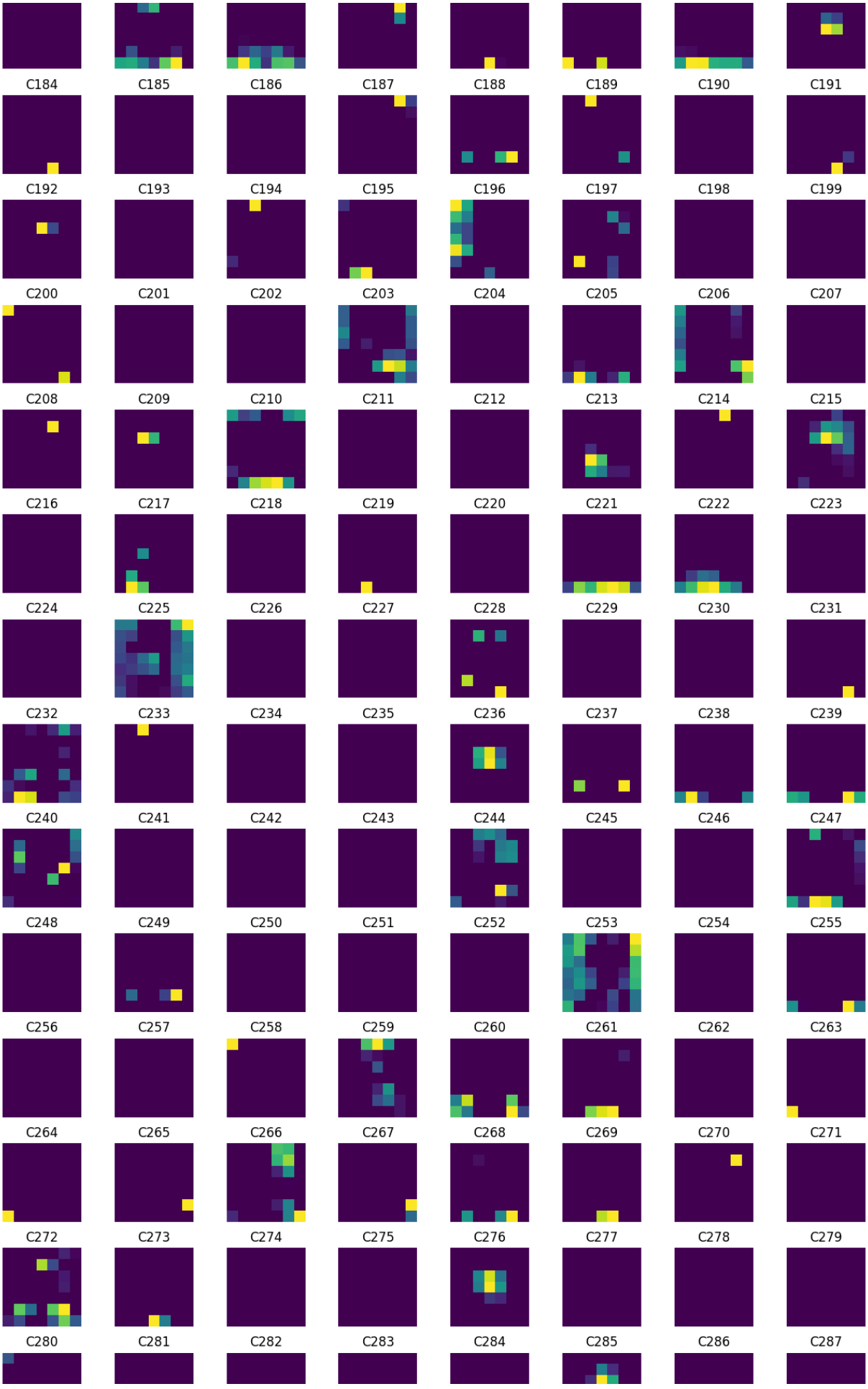

C392 C393 C394 C395 C396 C397 C398 C399
C400 C401 C402 C403 C404 C405 C406 C407
C408 C409 C410 C411 C412 C413 C414 C415
C416 C417 C418 C419 C420 C421 C422 C423
C424 C425 C426 C427 C428 C429 C430 C431
C432 C433 C434 C435 C436 C437 C438 C439
C440 C441 C442 C443 C444 C445 C446 C447
C448 C449 C450 C451 C452 C453 C454 C455
C456 C457 C458 C459 C460 C461 C462 C463
C464 C465 C466 C467 C468 C469 C470 C471
C472 C473 C474 C475 C476 C477 C478 C479
C480 C481 C482 C483 C484 C485 C486 C487
C488 C489 C490 C491 C492 C493 C494 C495

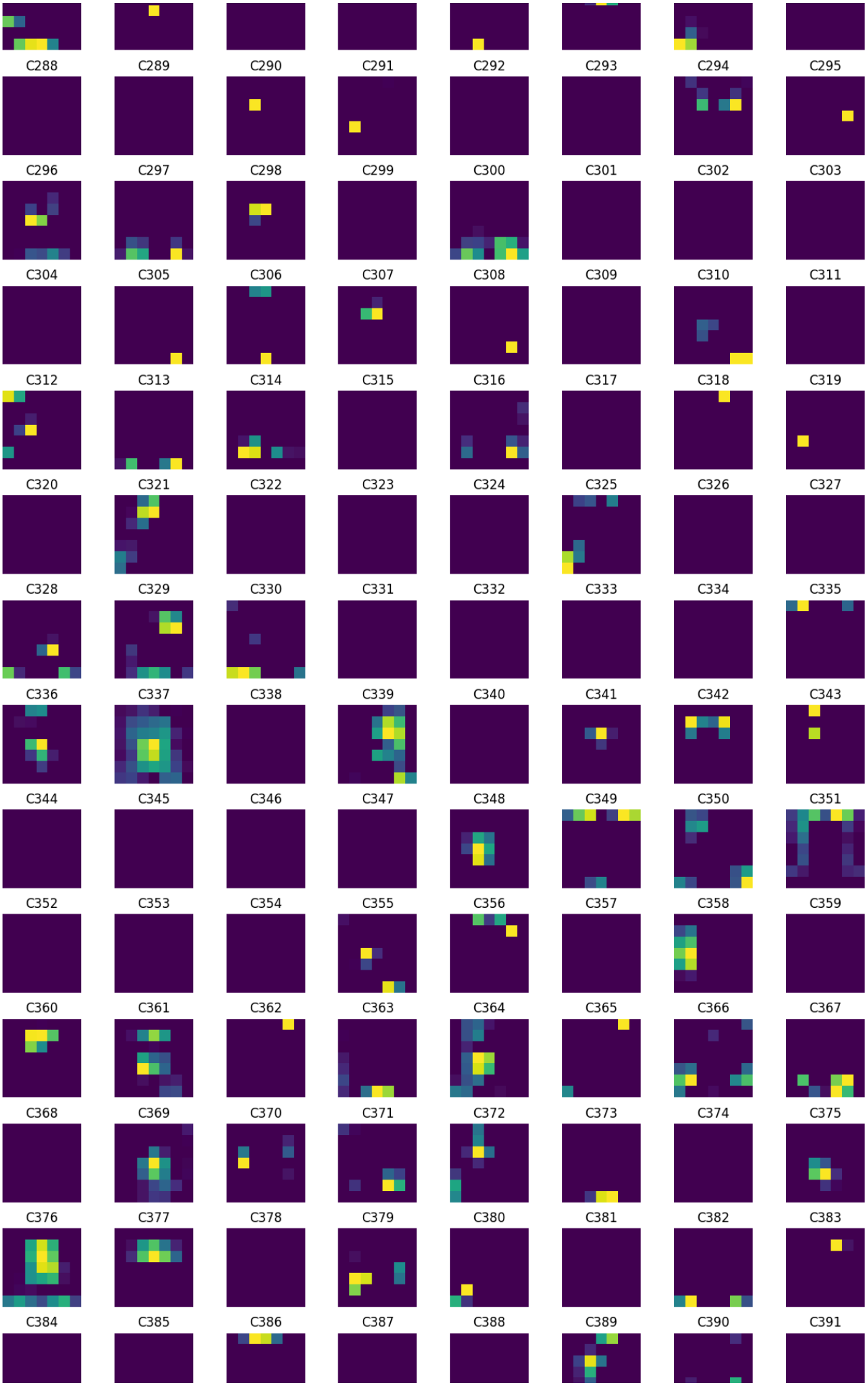

C496
C497
C498
C499
C500
C501
C502
C503
C504
C505
C506
C507
C508
C509
C510
C511

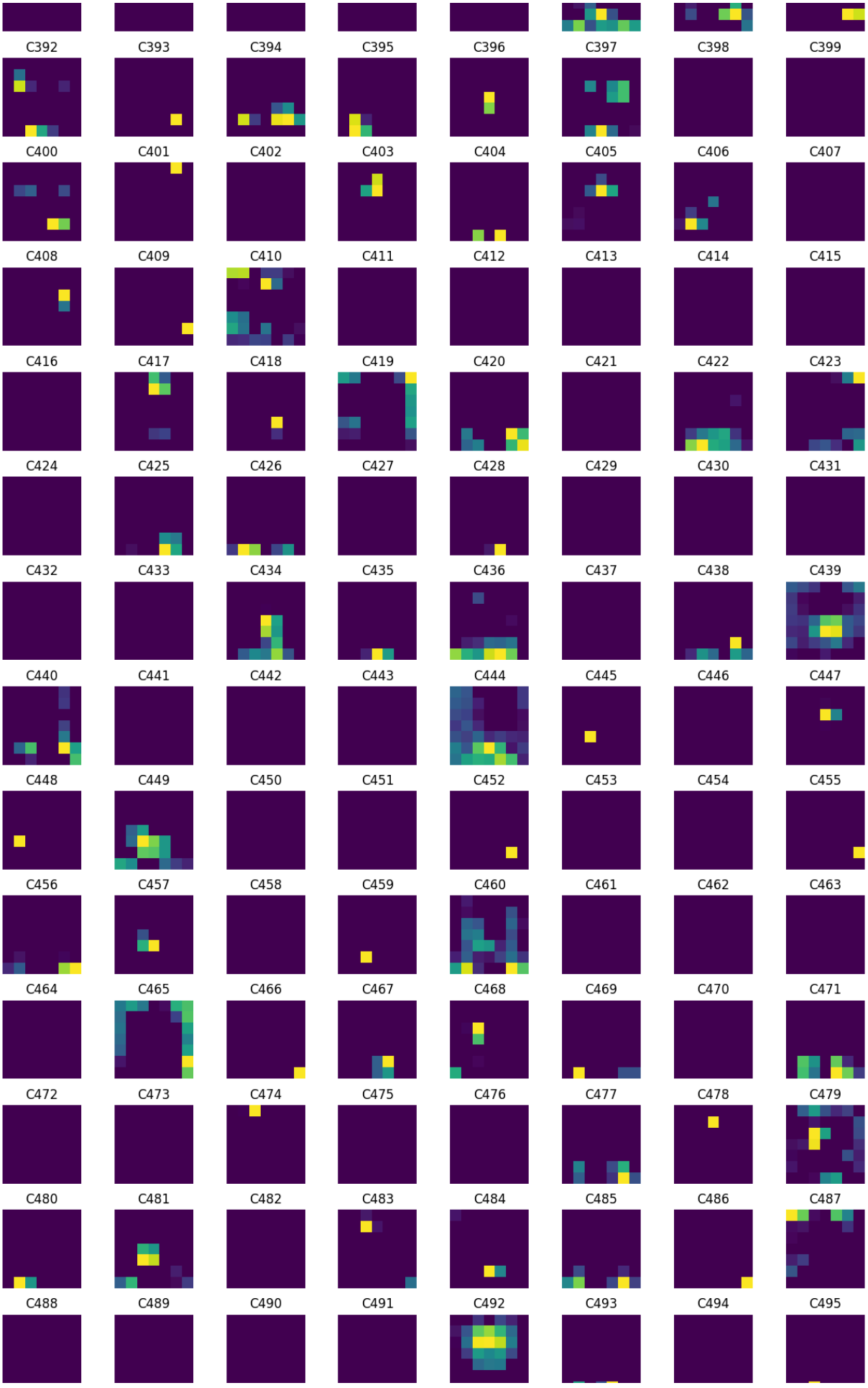
Block 5 - All 512 Feature Maps




C80
C81
C82
C83
C84
C85
C86
C87
C88
C89
C90
C91
C92
C93
C94
C95
C96
C97
C98
C99
C100
C101
C102
C103
C104
C105
C106
C107
C108
C109
C110
C111
C112
C113
C114
C115
C116
C117
C118
C119
C120
C121
C122
C123
C124
C125
C126
C127
C128
C129
C130
C131
C132
C133
C134
C135
C136
C137
C138
C139
C140
C141
C142
C143
C144
C145
C146
C147
C148
C149
C150
C151
C152
C153
C154
C155
C156
C157
C158
C159
C160
C161
C162
C163
C164
C165
C166
C167
C168
C169
C170
C171
C172
C173
C174
C175
C176
C177
C178
C179
C180
C181
C182
C183


C184
C185
C186
C187
C188
C189
C190
C191
C192
C193
C194
C195
C196
C197
C198
C199
C200
C201
C202
C203
C204
C205
C206
C207
C208
C209
C210
C211
C212
C213
C214
C215
C216
C217
C218
C219
C220
C221
C222
C223
C224
C225
C226
C227
C228
C229
C230
C231
C232
C233
C234
C235
C236
C237
C238
C239
C240
C241
C242
C243
C244
C245
C246
C247
C248
C249
C250
C251
C252
C253
C254
C255
C256
C257
C258
C259
C260
C261
C262
C263
C264
C265
C266
C267
C268
C269
C270
C271
C272
C273
C274
C275
C276
C277
C278
C279
C280
C281
C282
C283
C284
C285
C286
C287


C288
C289
C290
C291
C292
C293
C294
C295
C296
C297
C298
C299
C300
C301
C302
C303
C304
C305
C306
C307
C308
C309
C310
C311
C312
C313
C314
C315
C316
C317
C318
C319
C320
C321
C322
C323
C324
C325
C326
C327
C328
C329
C330
C331
C332
C333
C334
C335
C336
C337
C338
C339
C340
C341
C342
C343
C344
C345
C346
C347
C348
C349
C350
C351
C352
C353
C354
C355
C356
C357
C358
C359
C360
C361
C362
C363
C364
C365
C366
C367
C368
C369
C370
C371
C372
C373
C374
C375
C376
C377
C378
C379
C380
C381
C382
C383
C384
C385
C386
C387
C388
C389
C390
C391


C392
C393
C394
C395
C396
C397
C398
C399
C400
C401
C402
C403
C404
C405
C406
C407
C408
C409
C410
C411
C412
C413
C414
C415
C416
C417
C418
C419
C420
C421
C422
C423
C424
C425
C426
C427
C428
C429
C430
C431
C432
C433
C434
C435
C436
C437
C438
C439
C440
C441
C442
C443
C444
C445
C446
C447
C448
C449
C450
C451
C452
C453
C454
C455
C456
C457
C458
C459
C460
C461
C462
C463
C464
C465
C466
C467
C468
C469
C470
C471
C472
C473
C474
C475
C476
C477
C478
C479
C480
C481
C482
C483
C484
C485
C486
C487
C488
C489
C490
C491
C492
C493
C494
C495