In [ ]:
```
'''
The Leung-Malik (LM) Filter Bank, implementation in python

T. Leung and J. Malik. Representing and recognizing the visual appearance o
f
materials using three-dimensional textons. International Journal of Compute
r
Vision, 43(1):29-44, June 2001.

Reference: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html
'''

import numpy as np
import cv2
import matplotlib.pyplot as plt
```

In [ ]:
```
def gaussian1d(sigma, mean, x, ord):
    x = np.array(x)
    x_ = x - mean
    var = sigma**2

    # Gaussian Function
    g1 = (1/np.sqrt(2*np.pi*var))*(np.exp((-1*x_*x_)/(2*var)))

    if ord == 0:
        g = g1
        return g
    elif ord == 1:
        g = -g1*((x_)/(var))
        return g
    else:
        g = g1*(((x_*x_) - var)/(var**2))
        return g
```

In [ ]:
```
def gaussian2d(sup, scales):
    var = scales * scales
    shape = (sup,sup)
    n,m = [(i - 1)/2 for i in shape]
    x,y = np.ogrid[-m:m+1,-n:n+1]
    g = (1/np.sqrt(2*np.pi*var))*np.exp( -(x*x + y*y) / (2*var) )
    return g
```

In [ ]:
```
def log2d(sup, scales):
    var = scales * scales
    shape = (sup,sup)
    n,m = [(i - 1)/2 for i in shape]
    x,y = np.ogrid[-m:m+1,-n:n+1]
    g = (1/np.sqrt(2*np.pi*var))*np.exp( -(x*x + y*y) / (2*var) )
    h = g*((x*x + y*y) - var)/(var**2)
    return h
```

In [ ]:
```
def makefilter(scale, phasex, phasey, pts, sup):

    gx = gaussian1d(3*scale, 0, pts[0,...], phasex)
    gy = gaussian1d(scale,   0, pts[1,...], phasey)

    image = gx*gy

    image = np.reshape(image,(sup,sup))
    return image
```

In [ ]:
```
def makeLMfilters():
    sup     = 49
    scalex  = np.sqrt(2) * np.array([1,2,3])
    norient = 6
    nrotinv = 12

    nbar  = len(scalex)*norient
    nedge = len(scalex)*norient
    nf    = nbar+nedge+nrotinv
    F     = np.zeros([sup,sup,nf])
    hsup  = (sup - 1)/2

    x = [np.arange(-hsup,hsup+1)]
    y = [np.arange(-hsup,hsup+1)]

    [x,y] = np.meshgrid(x,y)

    orgpts = [x.flatten(), y.flatten()]
    orgpts = np.array(orgpts)

    count = 0
    for scale in range(len(scalex)):
        for orient in range(norient):
            angle = (np.pi * orient)/norient
            c = np.cos(angle)
            s = np.sin(angle)
            rotpts = [[c+0,-s+0],[s+0,c+0]]
            rotpts = np.array(rotpts)
            rotpts = np.dot(rotpts,orgpts)
            F[:,:,count] = makefilter(scalex[scale], 0, 1, rotpts, sup)
            F[:,:,count+nedge] = makefilter(scalex[scale], 0, 2, rotpts, su
p)
            count = count + 1

    count = nbar+nedge
    scales = np.sqrt(2) * np.array([1,2,3,4])

    for i in range(len(scales)):
        F[:,:,count]   = gaussian2d(sup, scales[i])
        count = count + 1

    for i in range(len(scales)):
        F[:,:,count] = log2d(sup, scales[i])
        count = count + 1

    for i in range(len(scales)):
        F[:,:,count] = log2d(sup, 3*scales[i])
        count = count + 1

    return F
```

In [ ]:
```
F = makeLMfilters()
print(F.shape) #same as print(np.shape(F))
```

```
(49, 49, 48)
```

Examining Separable Filters

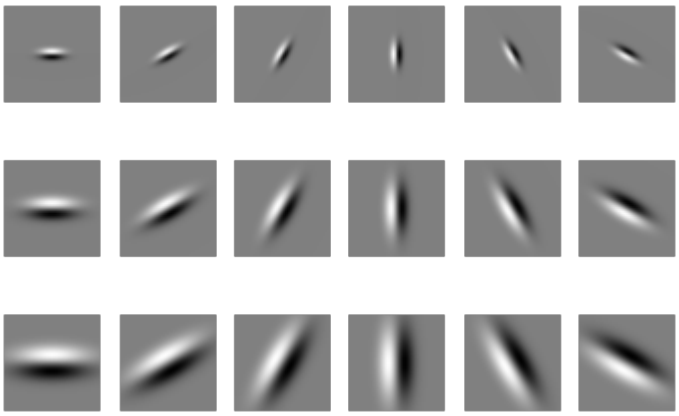
0-17 - First order Gaussian 18-35 - Second order Gaussian 36-39 - Gaussian 40-47 - Laplacian of Gaussian

In [ ]:
```
# Show singular value matrix

for i in range(0,48):
    np.set_printoptions(precision=4)
    U, E, V = np.linalg.svd(F[:,:,i])

    # If rank is 1, filter is separable.
    if(np.linalg.matrix_rank(F[:,:,i]) == 1):
        printout = "filter %d is separable, with a rank of %s" % (i, np.lin
alg.matrix_rank(F[:,:,i]))
        print(printout)
```

```
filter 0 is separable, with a rank of 1
filter 3 is separable, with a rank of 1
filter 6 is separable, with a rank of 1
filter 9 is separable, with a rank of 1
filter 12 is separable, with a rank of 1
filter 15 is separable, with a rank of 1
filter 18 is separable, with a rank of 1
filter 21 is separable, with a rank of 1
filter 24 is separable, with a rank of 1
filter 27 is separable, with a rank of 1
filter 30 is separable, with a rank of 1
filter 33 is separable, with a rank of 1
filter 36 is separable, with a rank of 1
filter 37 is separable, with a rank of 1
filter 38 is separable, with a rank of 1
filter 39 is separable, with a rank of 1
```

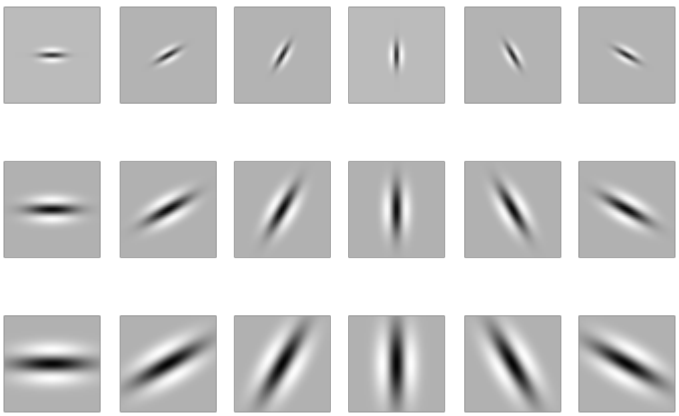
First order derivative Gaussian Filter

In [ ]:
```
for i in range(0,18):
    plt.subplot(3,6,i+1)
    plt.axis('off')
    plt.imshow(F[:,:,i], cmap = 'gray')
```



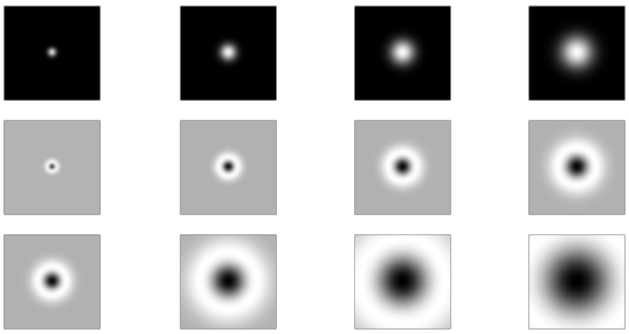
Second order derivative Gaussian Filter

In [ ]:
```
for i in range(0,18):
    plt.subplot(3,6,i+1)
    plt.axis('off')
    plt.imshow(F[:,:,i+18], cmap = 'gray')
```



Gaussian and Laplacian Filter

In [ ]:
```
for i in range(0,12):
    plt.subplot(4,4,i+1)
    plt.axis('off')
    plt.imshow(F[:,:,i+36], cmap = 'gray')
```



Load images

In [ ]:
```
img = cv2.imread('cardinal1.jpg',0)
plt.figure(figsize = [10, 10])
plt.axis('off')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB), cmap="gray")
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x1a58d17d250>
```

In [ ]:
```
# Load all Images
images = list()

# Birds
images.append(cv2.imread('cardinal1.jpg',0))
images.append(cv2.imread('cardinal2.jpg',0))
images.append(cv2.imread('woodduck1.jpeg',0))
images.append(cv2.imread('woodduck2.jpeg',0))

# Mammals
images.append(cv2.imread('leopard1.jpg',0))
images.append(cv2.imread('leopard2.jpg',0))
images.append(cv2.imread('panda1.jpg',0))
images.append(cv2.imread('panda2.jpg',0))
```

In [ ]:
```
def makeResponses(img, F):
    #insert code to compute responses for grayscale images

    w, h = img.shape
    _, _, num_filters = F.shape
    responses = np.empty([w, h, num_filters])

    for i in range (0, 48):
        responses[:,:,i] = cv2.filter2D(src = img, ddepth=-1, kernel=F[:,:,
i])

    return responses
```

In [ ]:
```
def makeResponses_sep(img, F, vectors):
    #insert code to compute responses for RBG images
    w, h = img.shape
    _, _, num_filters = F.shape
    responses = np.empty([w, h, num_filters])

    for i in range (0, 16):
        responses[:,:,i] = cv2.sepFilter2D(src = img, ddepth=-1, kernelX=ve
ctors[i], kernelY = vectors[i+16])

    return responses
```

In [ ]:
```
def makeResponses_rgb(img, F):
    #insert code to compute responses for grayscale images

    w, h, ch = img.shape
    _, _, num_filters = F.shape
    responses = np.empty([w, h, ch, num_filters])

    for i in range (0, 48):
        responses[:,:, :,i] = cv2.filter2D(src = img, ddepth=-1, kernel=F
[:,:,i])

    return responses
```

In [ ]:
```
def computeMagnitude(img):
    w, h = img.shape
    resolution = w*h

    sum = 0

    # Get the sum of the absolute value of all pixels in image
    sum = np.sum(abs(img))

    #sum = np.linalg.norm(img, 1)

    return sum/resolution #normalize by image size
```

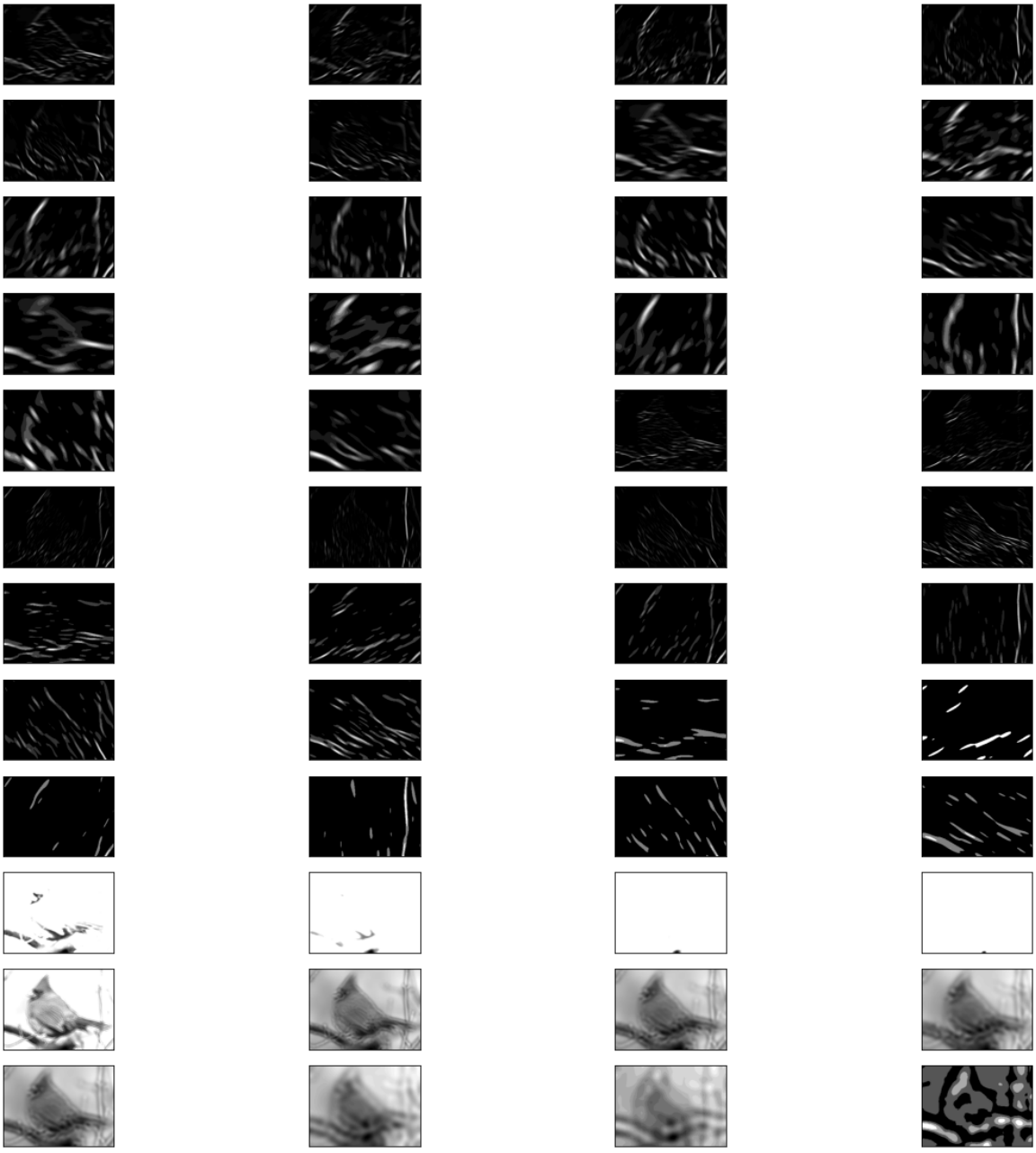
In [ ]:
```
def computeMagnitudeRGB(img):

    sumVal = 0
    w, h, ch = img.shape
    resolution = w*h

    # Get the sum of the absolute value of all pixels in image
    sum = np.sum(abs(img))

    return sum/resolution #normalize by image size
```

Loop and apply all filters to all images - Grayscale

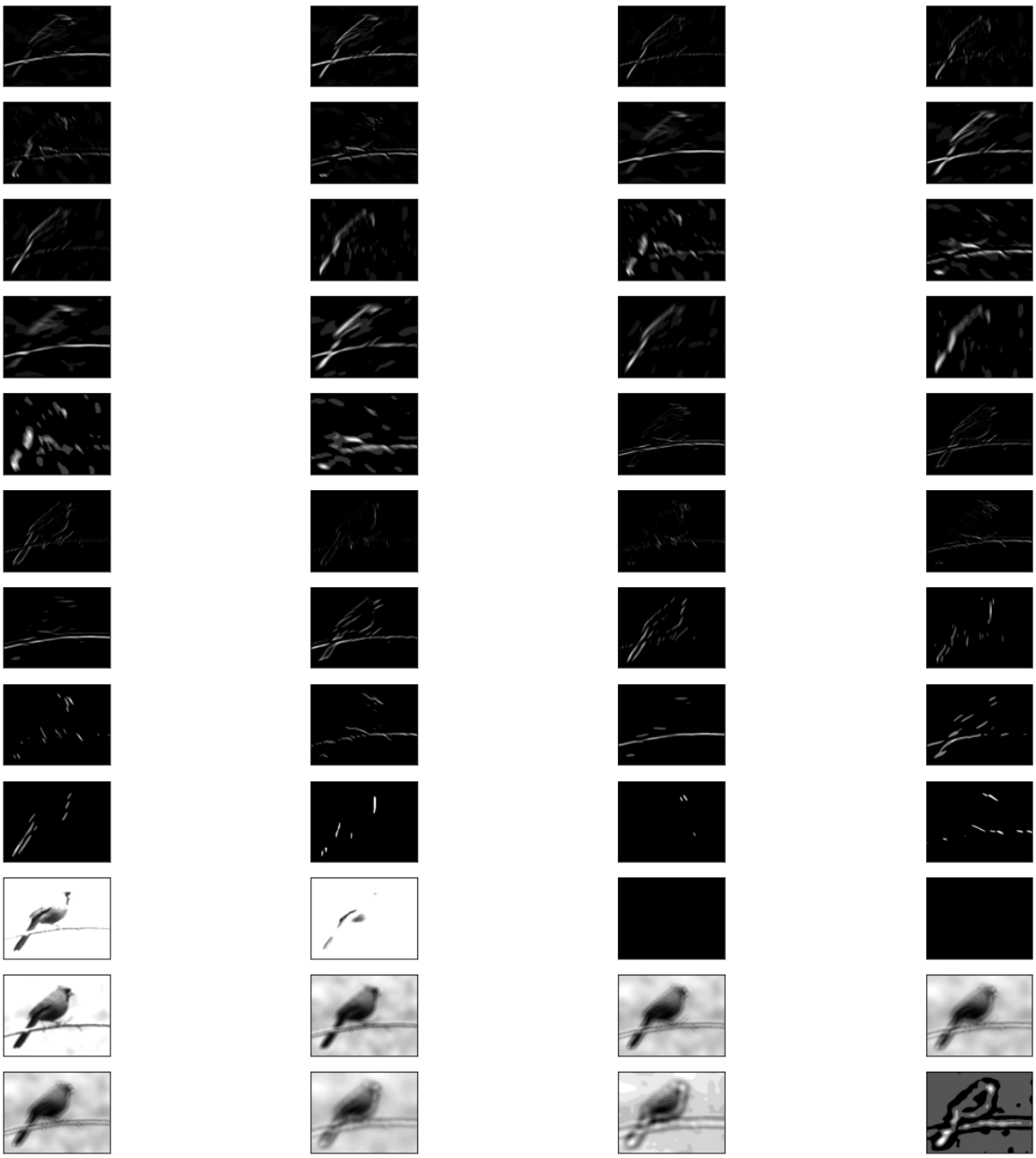
In [ ]:
```
num_images = len(images)
num_filters = 48

norms = [[0 for i in range(num_images)] for j in range(num_filters)]

image_sums = [0] * 8

sum_norm = 0
sum_norm_mine = 0

# Loop through each image
for i in range (0, num_images):

    # Filter image and store
    FB_responses = makeResponses(images[i], F)
    print(FB_responses.shape, images[i].shape)

    h, w = images[i].shape


    plt.figure(figsize=(20,20))
    plt.suptitle('LM filter responses')
    nr = 12
    nc = 4

    #For each filter...
    for j in range(num_filters):
        #Get magnitude for each filter of each image
        norms[j][i] = computeMagnitude(FB_responses[:,:,j])

        #Get sum of all filtered images
        if(j == 0):
            image_sums[i] = FB_responses[:,:,j]
        else:
            image_sums[i] = FB_responses[:,:,j] + image_sums[i]


        #Plot filtered images
        plt.subplot(nr, nc, j+1)
        fig = plt.imshow(FB_responses[:,:,j], cmap='gray')
        fig.axes.get_xaxis().set_visible(False)
        fig.axes.get_yaxis().set_visible(False)

    plt.show()

#print(sum_norm / (h*w))
#print(sum_norm_mine)
#print(np.matrix(norms))
```

```
(200, 275, 48) (200, 275)
```

LM filter responses

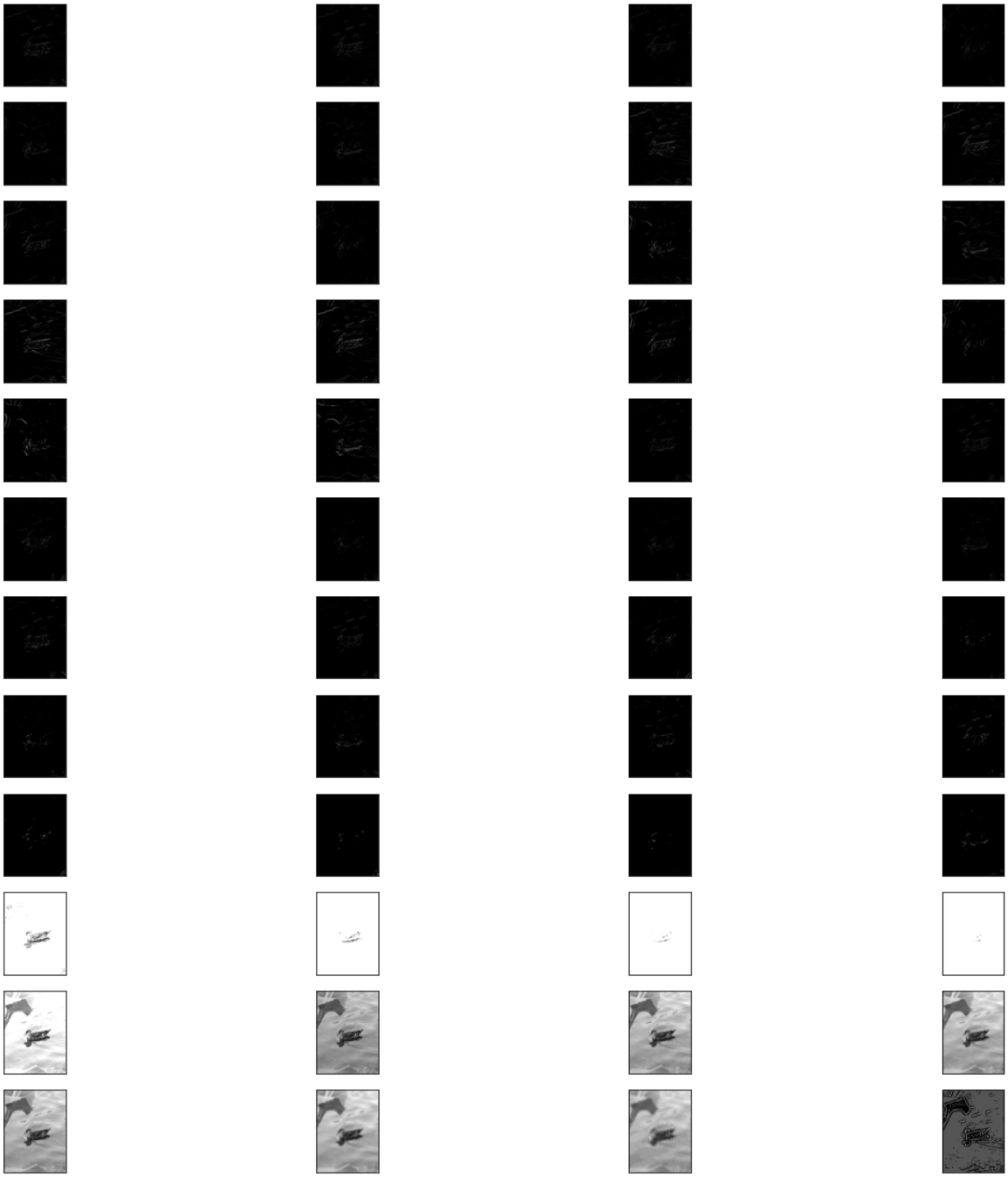


```
(302, 398, 48) (302, 398)
```



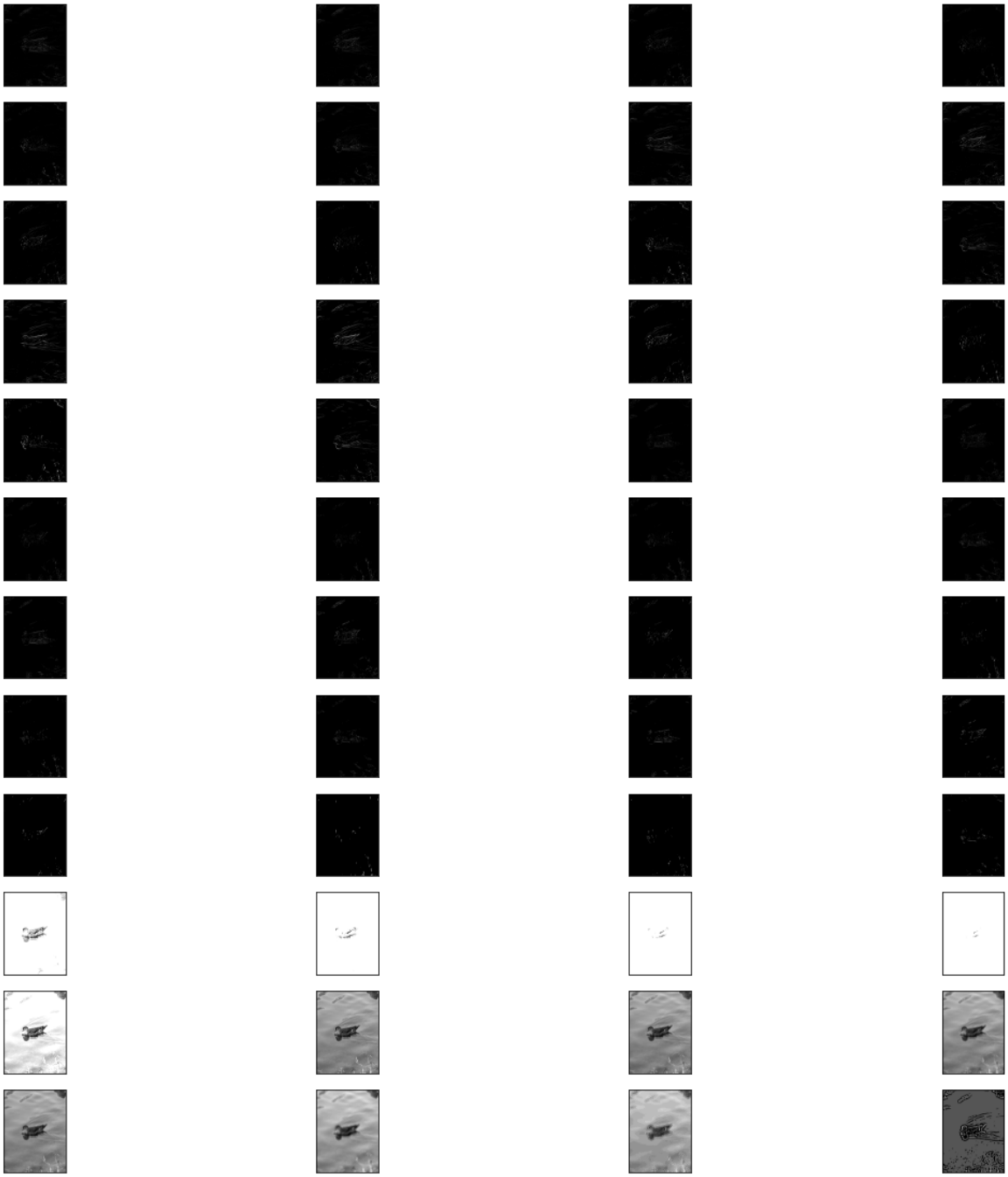

```
(2048, 1536, 48) (2048, 1536)
```

LM filter responses

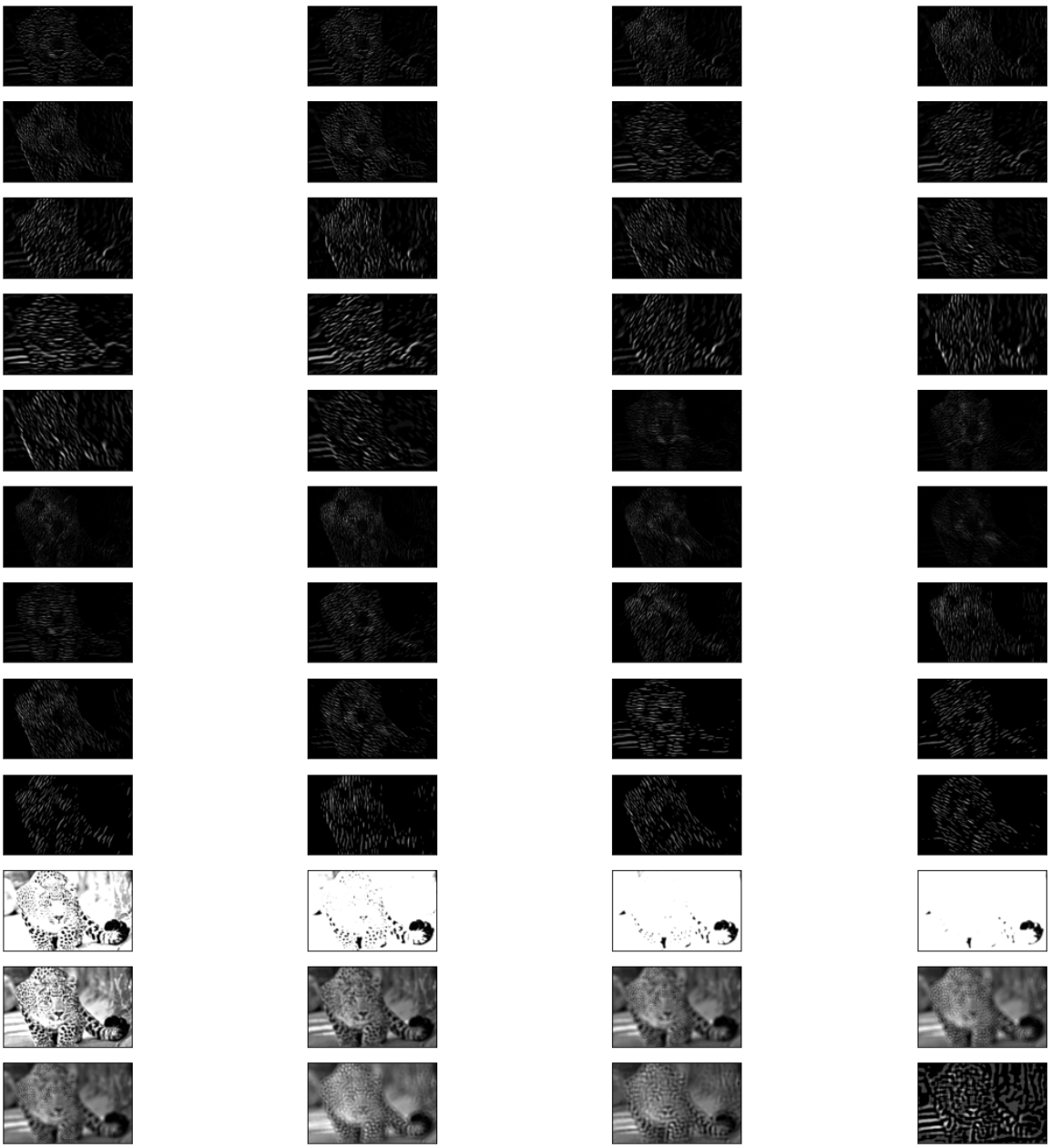


```
(2048, 1536, 48) (2048, 1536)
```

LM filter responses

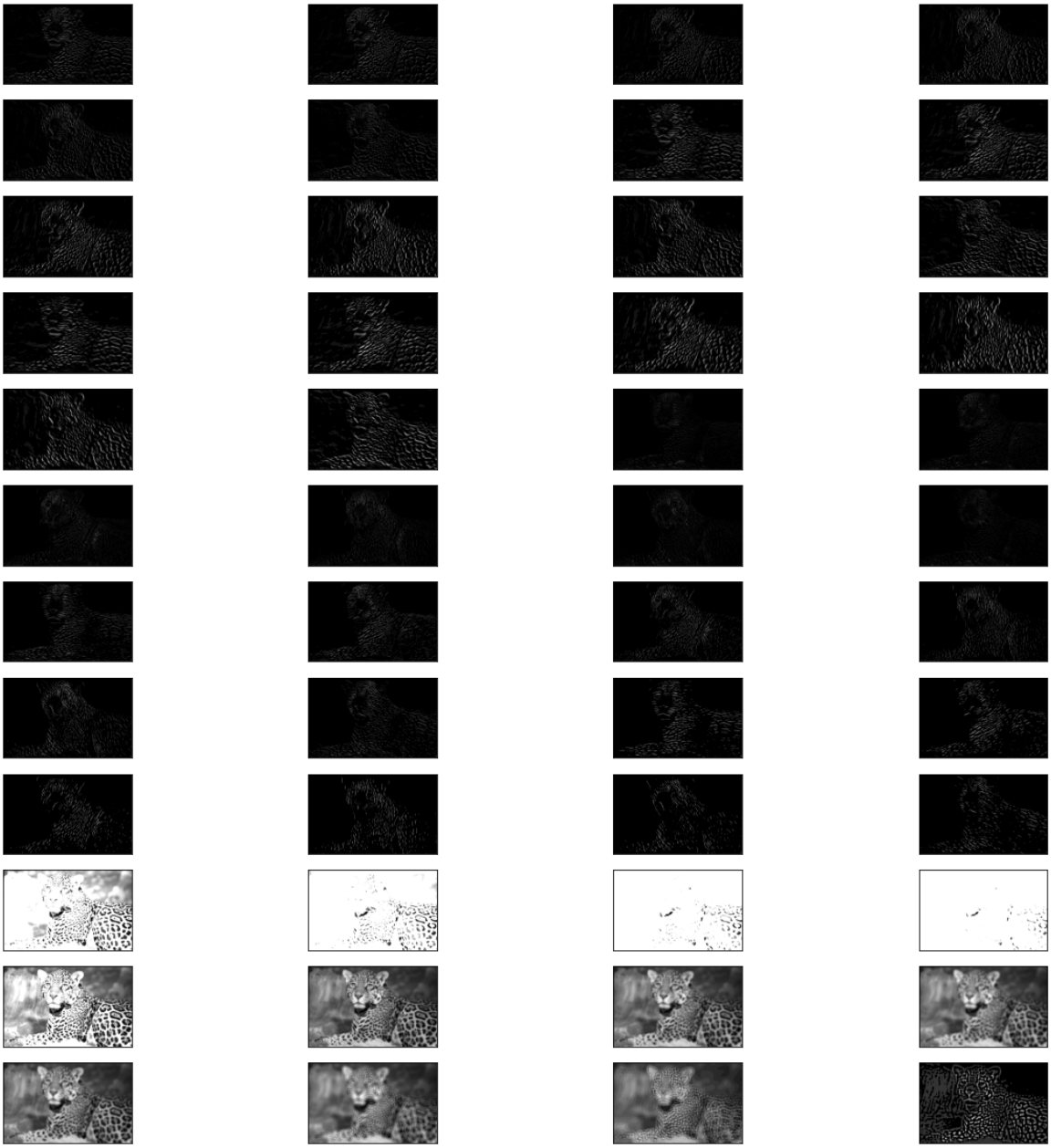


(585, 940, 48) (585, 940)



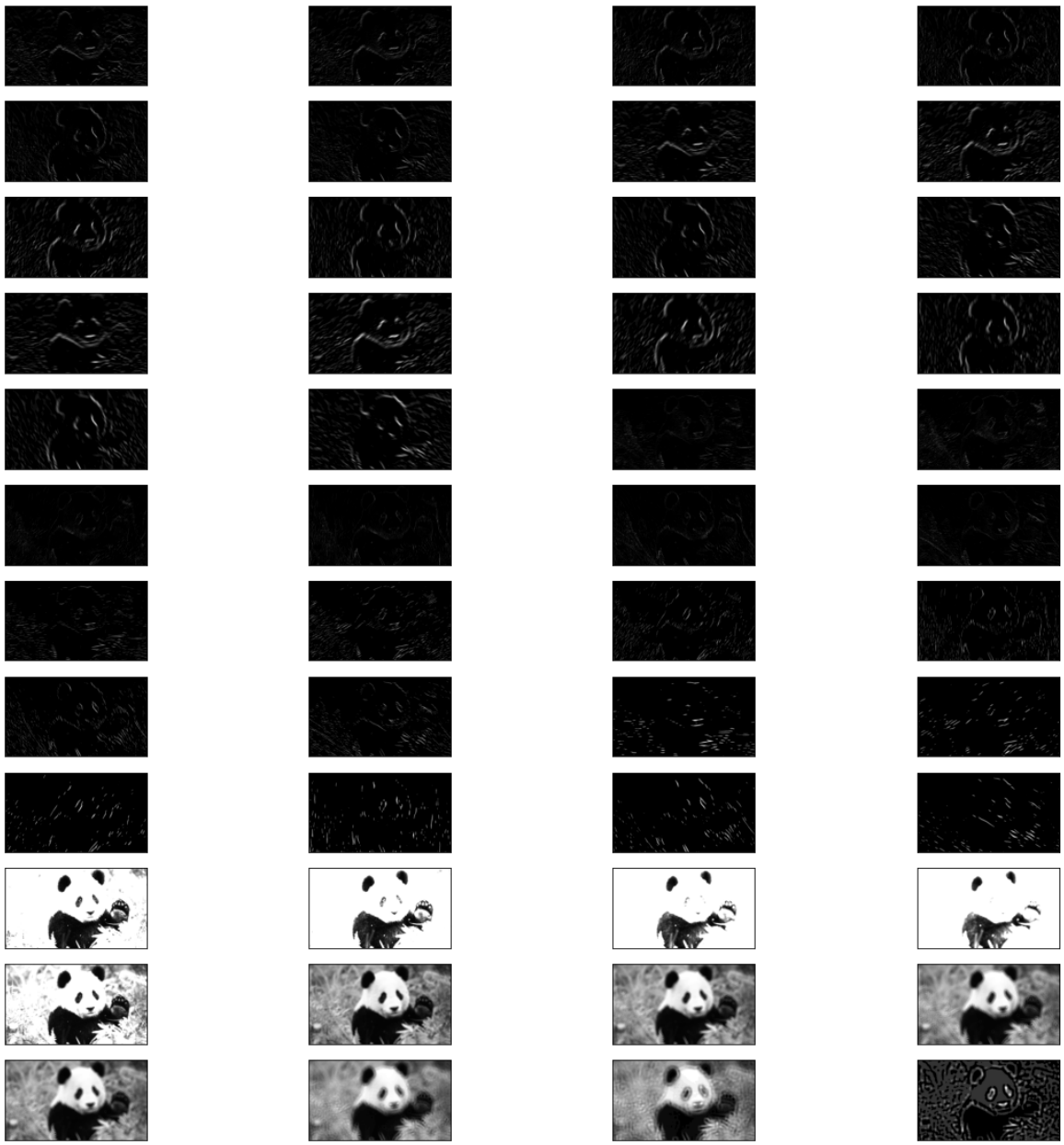

```
(1200, 1920, 48) (1200, 1920)
```

LM filter responses



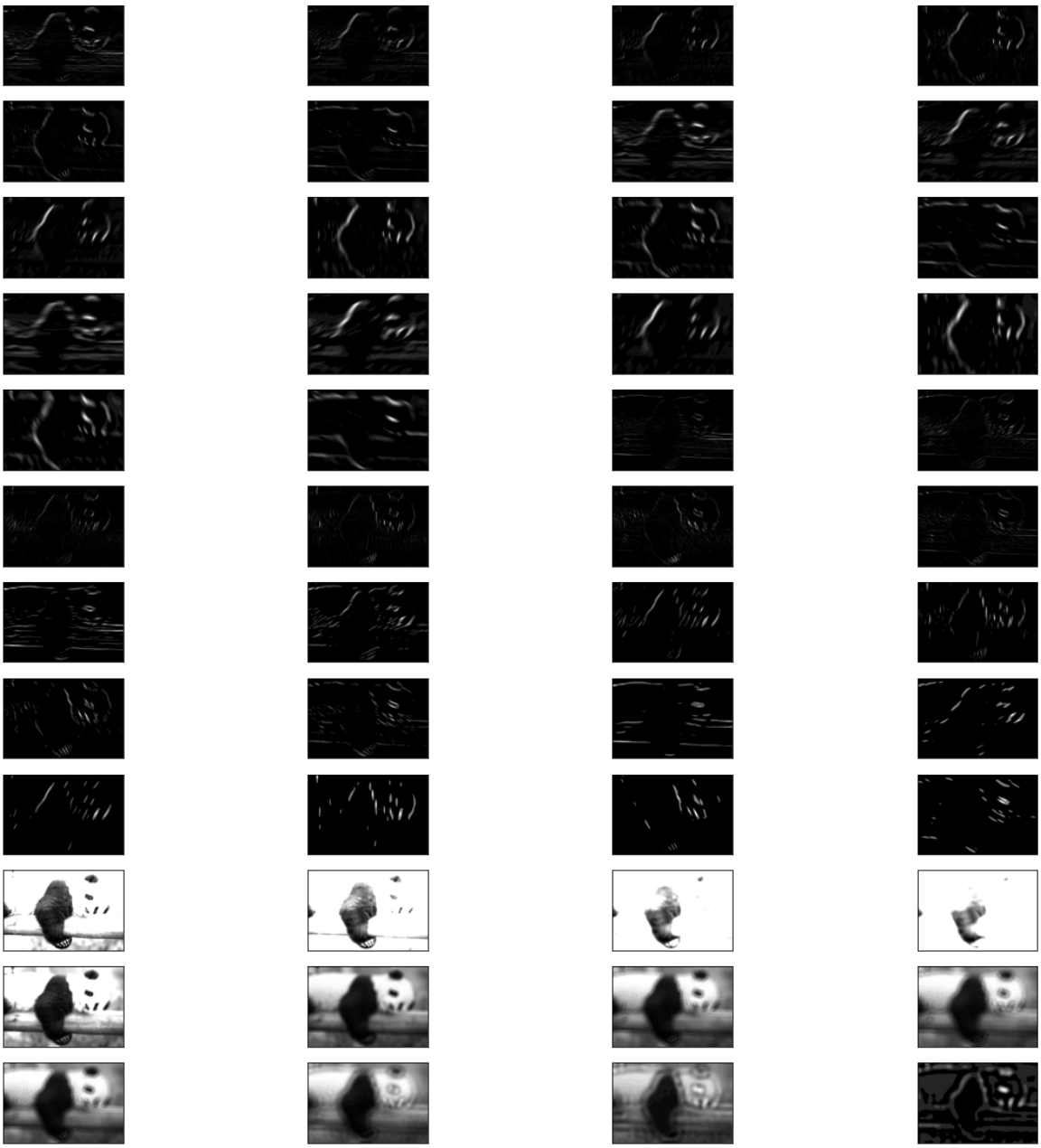
```
(788, 1400, 48) (788, 1400)
```

LM filter responses



```
(367, 550, 48) (367, 550)
```

LM filter responses



In [ ]:
```
print(np.matrix(norms))
```

```
[[1.5794e+00 8.1464e-01 4.3845e-01 5.0041e-01 1.8699e+00 1.1327e+00
  1.0086e+00 1.2455e+00]
 [1.3468e+00 7.7757e-01 4.0026e-01 4.5296e-01 1.8752e+00 1.1546e+00
  1.0094e+00 1.1461e+00]
 [1.2048e+00 5.6296e-01 3.0309e-01 3.4176e-01 1.9061e+00 1.1334e+00
  9.9642e-01 9.7741e-01]
 [1.2467e+00 3.9228e-01 2.2559e-01 2.7366e-01 1.9078e+00 1.1007e+00
  1.0067e+00 8.5987e-01]
 [1.2751e+00 3.8611e-01 2.4781e-01 3.0590e-01 1.9508e+00 1.1000e+00
  9.9216e-01 8.8021e-01]
 [1.3697e+00 5.3409e-01 3.4372e-01 4.0899e-01 1.9425e+00 1.1515e+00
  9.8318e-01 9.4917e-01]
 [1.0489e+00 5.9543e-01 2.6867e-01 2.9340e-01 1.1504e+00 8.2011e-01
  7.0283e-01 9.8269e-01]
 [8.6151e-01 5.5305e-01 2.4265e-01 2.5906e-01 1.1202e+00 8.4103e-01
  7.1098e-01 8.8825e-01]
 [7.2705e-01 3.9144e-01 1.6475e-01 1.7726e-01 1.1176e+00 8.2538e-01
  7.0206e-01 7.4064e-01]
 [7.3635e-01 2.8307e-01 1.1071e-01 1.3397e-01 1.1395e+00 8.0579e-01
  7.1495e-01 6.4438e-01]
 [7.2765e-01 2.5827e-01 1.1887e-01 1.4931e-01 1.1316e+00 8.1028e-01
  7.1327e-01 6.5498e-01]
 [7.9740e-01 3.4398e-01 1.9075e-01 2.2203e-01 1.1207e+00 8.5245e-01
  6.8504e-01 7.0978e-01]
 [8.0727e-01 4.4273e-01 1.9299e-01 2.0051e-01 7.6097e-01 6.1656e-01
  5.3962e-01 8.3132e-01]
 [6.6118e-01 4.2336e-01 1.7759e-01 1.8237e-01 7.4088e-01 6.3048e-01
  5.5439e-01 7.6256e-01]
 [5.1145e-01 2.9942e-01 1.1317e-01 1.1430e-01 7.2803e-01 6.2014e-01
  5.5202e-01 6.1173e-01]
 [4.9060e-01 2.2012e-01 7.7952e-02 8.8351e-02 7.4707e-01 6.0633e-01
  5.5604e-01 5.3282e-01]
 [4.8038e-01 1.9313e-01 7.7143e-02 9.5470e-02 7.2910e-01 6.2248e-01
  5.6301e-01 5.3795e-01]
 [5.3611e-01 2.6141e-01 1.3556e-01 1.5310e-01 7.2230e-01 6.6098e-01
  5.3253e-01 5.9721e-01]
 [6.0949e-01 2.6218e-01 1.3984e-01 1.8208e-01 6.8722e-01 3.4031e-01
  3.2165e-01 4.2008e-01]
 [4.8333e-01 2.4474e-01 1.1195e-01 1.4769e-01 6.7695e-01 3.3706e-01
  3.1454e-01 3.1471e-01]
 [4.6193e-01 1.6639e-01 7.5769e-02 1.0241e-01 6.9837e-01 3.1614e-01
  2.9519e-01 2.6649e-01]
 [5.5076e-01 1.0089e-01 5.8771e-02 9.2055e-02 7.2218e-01 2.9250e-01
  3.0416e-01 2.7084e-01]
 [5.8275e-01 8.2873e-02 6.3040e-02 9.4268e-02 7.4299e-01 2.9037e-01
  2.9577e-01 2.6616e-01]
 [7.0671e-01 1.3720e-01 1.0160e-01 1.3473e-01 7.2922e-01 3.1089e-01
  3.0382e-01 3.1114e-01]
 [1.7815e-01 1.1021e-01 3.6229e-02 5.1320e-02 2.6254e-01 1.4034e-01
  9.4024e-02 1.3843e-01]
 [1.2947e-01 9.1958e-02 2.8384e-02 3.9597e-02 2.4049e-01 1.4260e-01
  9.0748e-02 8.7961e-02]
 [1.2209e-01 5.9095e-02 1.7324e-02 2.4855e-02 2.3827e-01 1.3412e-01
  8.4395e-02 7.5972e-02]
```

```
[1.6638e-01 2.6399e-02 1.0587e-02 2.2435e-02 2.5597e-01 1.2401e-01
 9.2738e-02 7.5452e-02]
[1.7862e-01 1.4942e-02 1.0552e-02 2.2050e-02 2.5400e-01 1.1650e-01
 8.7587e-02 6.7852e-02]
[2.1889e-01 3.7647e-02 2.2403e-02 3.3605e-02 2.4378e-01 1.2515e-01
 8.6783e-02 8.4781e-02]
[7.5509e-02 5.0676e-02 1.1813e-02 1.6308e-02 1.1934e-01 6.7680e-02
 2.9465e-02 6.2393e-02]
[3.9582e-02 3.5584e-02 8.2124e-03 1.0626e-02 8.9818e-02 6.1640e-02
 2.4826e-02 2.9983e-02]
[2.7873e-02 2.1723e-02 4.3310e-03 5.5993e-03 8.6643e-02 5.5872e-02
 2.2681e-02 3.4674e-02]
[5.5855e-02 7.7457e-03 2.3982e-03 6.4535e-03 1.0608e-01 5.6328e-02
 2.8942e-02 3.4189e-02]
[5.7545e-02 1.7887e-03 1.8727e-03 4.7509e-03 9.2688e-02 4.4620e-02
 2.5757e-02 2.3250e-02]
[8.7782e-02 9.1184e-03 4.8958e-03 7.0047e-03 8.4746e-02 4.9807e-02
 2.3445e-02 2.7753e-02]
[2.4422e+02 2.4501e+02 2.5059e+02 2.5092e+02 2.0955e+02 2.1806e+02
 2.0887e+02 2.0105e+02]
[2.5325e+02 2.5397e+02 2.5403e+02 2.5403e+02 2.3845e+02 2.4467e+02
 2.2117e+02 2.2546e+02]
[2.5471e+02 2.5500e+02 2.5471e+02 2.5466e+02 2.4576e+02 2.5072e+02
 2.2864e+02 2.3691e+02]
[2.5495e+02 2.5500e+02 2.5492e+02 2.5488e+02 2.4897e+02 2.5326e+02
 2.3460e+02 2.4310e+02]
[2.1506e+02 2.2886e+02 2.2663e+02 2.3280e+02 1.4880e+02 1.6371e+02
 1.8225e+02 1.6074e+02]
[1.2143e+02 1.2189e+02 1.1954e+02 1.2462e+02 7.7581e+01 8.7530e+01
 1.0679e+02 9.8497e+01]
[8.0956e+01 8.1264e+01 7.9690e+01 8.3079e+01 5.1724e+01 5.8353e+01
 7.1193e+01 6.5662e+01]
[6.0676e+01 6.0909e+01 5.9731e+01 6.2272e+01 3.8769e+01 4.3738e+01
 5.3363e+01 4.9222e+01]
[8.0956e+01 8.1264e+01 7.9690e+01 8.3079e+01 5.1724e+01 5.8353e+01
 7.1193e+01 6.5662e+01]
[3.7297e+01 3.7442e+01 3.6715e+01 3.8277e+01 2.3831e+01 2.6885e+01
 3.2799e+01 3.0252e+01]
[1.1850e+01 1.1894e+01 1.1664e+01 1.2164e+01 7.5692e+00 8.5432e+00
 1.0429e+01 9.6193e+00]
[7.5011e-01 8.0014e-01 8.3638e-01 8.8042e-01 5.0926e-01 5.5946e-01
 6.2606e-01 5.9779e-01]]
```

Grayscale Thresholds

In [ ]:
```
# Get norms of summed responses
image_norms = [0] * num_images

for i in range (num_images):

    h, w = image_sums[i].shape
    image_norms[i] = np.linalg.norm(image_sums[i], 1) / (h*w)
```

In [ ]:
```
filter_thresholds = [0] * 49
sum_threshold = 0

bird_norms = [0] * 4
mammal_norms = [0] * 4

bird_norms_sum = [0] * 4
mammal_norms_sum = [0] * 4

for i in range (num_filters):

    #Split norms into birds and mammals
    bird_norms[0] = norms[i][0]
    bird_norms[1] = norms[i][1]
    bird_norms[2] = norms[i][2]
    bird_norms[3] = norms[i][3]

    mammal_norms[0] = norms[i][4]
    mammal_norms[1] = norms[i][5]
    mammal_norms[2] = norms[i][6]
    mammal_norms[3] = norms[i][7]

    # Calculate ideal threshold value
    if(max(bird_norms) < min(mammal_norms)):
        filter_thresholds[i] = (max(bird_norms) + min(mammal_norms)) / 2
    else:
        filter_thresholds[i] = min(mammal_norms)

    printout = "Filter #%d has the following norms:\n birds %s \n mammals %
s" % (i, bird_norms, mammal_norms)
    print(printout)

#Summed threshold
bird_norms_sum[0] = image_norms[0]
bird_norms_sum[1] = image_norms[1]
bird_norms_sum[2] = image_norms[2]
bird_norms_sum[3] = image_norms[3]

mammal_norms_sum[0] = image_norms[4]
mammal_norms_sum[1] = image_norms[5]
mammal_norms_sum[2] = image_norms[6]
mammal_norms_sum[3] = image_norms[7]

if(max(bird_norms_sum) < min(mammal_norms_sum)):
    sum_threshold = (max(bird_norms_sum) + min(mammal_norms_sum)) / 2
else:
    sum_threshold = min(mammal_norms_sum)

print(filter_thresholds)
print(image_norms)
print(sum_threshold)
```

```
Filter #0 has the following norms:
 birds [1.5793636363636363, 0.8146360943791807, 0.4384454091389974, 0.50041
42125447592]
 mammals [1.8698927077650482, 1.1326948784722222, 1.0086085931834663, 1.245
5040871934604]
Filter #1 has the following norms:
 birds [1.3468363636363636, 0.7775716329994342, 0.40025901794433594, 0.4529
581069946289]
 mammals [1.8752409529005274, 1.15461328125, 1.0093817984046412, 1.14612335
89298985]
Filter #2 has the following norms:
 birds [1.2047818181818182, 0.5629555060068555, 0.3030923207600911, 0.34175
745646158856]
 mammals [1.9061211129296236, 1.1334214409722223, 0.9964204133430021, 0.977
4139212286351]
Filter #3 has the following norms:
 birds [1.246690909090909, 0.3922842690272555, 0.22559134165445963, 0.27365
52556355794]
 mammals [1.907766866703037, 1.1007430555555555, 1.0066579042784627, 0.8598
662373049294]
Filter #4 has the following norms:
 birds [1.275090909090909, 0.3861110186695065, 0.24781004587809244, 0.30589
834849039715]
 mammals [1.9508110565557375, 1.1000334201388888, 0.9921646120377084, 0.880
2130294773347]
Filter #5 has the following norms:
 birds [1.369709090909091, 0.5340859928782988, 0.3437156677246094, 0.408991
17787679035]
 mammals [1.9425150027277687, 1.1515494791666667, 0.9831816533720087, 0.949
1701758731731]
Filter #6 has the following norms:
 birds [1.0489272727272727, 0.5954274684681686, 0.26867421468098956, 0.2934
0489705403644]
 mammals [1.1503600654664485, 0.8201067708333334, 0.7028308556925308, 0.982
685162249195]
Filter #7 has the following norms:
 birds [0.8615090909090909, 0.5530466904056708, 0.242645263671875, 0.259056
7270914714]
 mammals [1.1201563920712856, 0.8410282118055555, 0.7109835025380711, 0.888
2486995293535]
Filter #8 has the following norms:
 birds [0.7270545454545455, 0.3914439748410929, 0.16475296020507812, 0.1772
6357777913412]
 mammals [1.1176395708310602, 0.8253832465277777, 0.7020621827411168, 0.740
6440426058954]
Filter #9 has the following norms:
 birds [0.7363454545454545, 0.2830709840593697, 0.11071268717447917, 0.1339
6644592285156]
 mammals [1.1395344608110565, 0.8057916666666667, 0.7149456127628716, 0.644
3844438939806]
Filter #10 has the following norms:
 birds [0.7276545454545454, 0.2582698259509468, 0.11887041727701823, 0.1493
1233723958334]
 mammals [1.1315839243498818, 0.81027734375, 0.7132704858593183, 0.65497646
```

```
76740154]
Filter #11 has the following norms:
 birds [0.7974, 0.3439798329395321, 0.1907498041788737, 0.2220274607340494
7]
 mammals [1.1206546644844517, 0.8524474826388889, 0.6850371646120377, 0.709
7795392618281]
Filter #12 has the following norms:
 birds [0.8072727272727273, 0.4427268794302639, 0.19298871358235678, 0.2005
1352183024088]
 mammals [0.7609656301145663, 0.6165559895833334, 0.5396174764321973, 0.831
3153331681942]
Filter #13 has the following norms:
 birds [0.6611818181818182, 0.4233585144264368, 0.17758909861246744, 0.1823
6700693766275]
 mammals [0.7408837970540099, 0.6304761284722222, 0.5543863306744018, 0.762
5613079019073]
Filter #14 has the following norms:
 birds [0.5114545454545455, 0.2994192818396619, 0.11316935221354167, 0.1143
023173014323]
 mammals [0.7280305510092744, 0.6201384548611111, 0.5520195794053662, 0.611
7314837750805]
Filter #15 has the following norms:
 birds [0.4906, 0.22012379779693167, 0.07795174916585286, 0.088350931803385
42]
 mammals [0.7470685579196218, 0.6063298611111111, 0.556035170413343, 0.5328
214020312113]
Filter #16 has the following norms:
 birds [0.4803818181818182, 0.19312622716230157, 0.07714271545410156, 0.095
46979268391927]
 mammals [0.7290961993089653, 0.6224809027777778, 0.5630067077592459, 0.537
953926182809]
Filter #17 has the following norms:
 birds [0.5361090909090909, 0.261406369596326, 0.1355555852254232, 0.153102
55686442056]
 mammals [0.7223022367703219, 0.6609796006944444, 0.532532632342277, 0.5972
05845925192]
Filter #18 has the following norms:
 birds [0.6094909090909091, 0.262180105827149, 0.13984298706054688, 0.18208
47193400065]
 mammals [0.687219494453537, 0.3403090277777778, 0.32164612037708484, 0.420
08422095615555]
Filter #19 has the following norms:
 birds [0.4833272727272727, 0.24474192152817065, 0.11194674173990886, 0.147
69299825032553]
 mammals [0.676950354609929, 0.3370564236111111, 0.314537708484409, 0.31470
894228387414]
Filter #20 has the following norms:
 birds [0.46192727272727274, 0.16638656860461246, 0.07576878865559895, 0.10
240618387858073]
 mammals [0.6983651573013275, 0.3161410590277778, 0.29518582306018853, 0.26
64899677978697]
Filter #21 has the following norms:
 birds [0.5507636363636363, 0.10088522080601683, 0.05877113342285156, 0.092
0550028483073]
```

```
 mammals [0.7221822149481724, 0.29249913194444443, 0.30416243654822334, 0.2
7084468664850136]
Filter #22 has the following norms:
 birds [0.5827454545454546, 0.08287297414223435, 0.06304009755452473, 0.094
26816304524739]
 mammals [0.7429878159665394, 0.2903689236111111, 0.2957695794053662, 0.266
16299232103047]
Filter #23 has the following norms:
 birds [0.706709090909091, 0.1372009051881926, 0.10159683227539062, 0.13473
288218180338]
 mammals [0.7292198581560284, 0.31088802083333333, 0.3038179840464104, 0.31
114193708199156]
Filter #24 has the following norms:
 birds [0.17814545454545455, 0.11021165429797997, 0.03622881571451823, 0.05
1320393880208336]
 mammals [0.26254228041462085, 0.1403420138888889, 0.09402374909354605, 0.1
3842952687639337]
Filter #25 has the following norms:
 birds [0.12947272727272727, 0.09195813504609139, 0.028383572896321613, 0.0
395971934000651]
 mammals [0.24048736133842516, 0.14260026041666668, 0.09074782451051487, 0.
08796135744364628]
Filter #26 has the following norms:
 birds [0.1220909090909091, 0.05909514459715798, 0.017324129740397137, 0.02
4855295817057293]
 mammals [0.23827059465357336, 0.13411762152777779, 0.08439539521392313, 0.
07597225662620757]
Filter #27 has the following norms:
 birds [0.1663818181818182, 0.026398549036573595, 0.010587056477864584, 0.0
22434552510579426]
 mammals [0.25596653937079467, 0.12400737847222222, 0.09273839738941261, 0.
0754520683675997]
Filter #28 has the following norms:
 birds [0.17861818181818181, 0.014942260973742887, 0.010551770528157553, 0.
022049903869628906]
 mammals [0.2539970903800691, 0.11650086805555555, 0.08758701957940537, 0.0
6785236561803319]
Filter #29 has the following norms:
 birds [0.21889090909090908, 0.03764684348896802, 0.02240276336669922, 0.03
360525767008463]
 mammals [0.24378068739770867, 0.1251484375, 0.08678299492385787, 0.0847807
7780530097]
Filter #30 has the following norms:
 birds [0.07550909090909091, 0.05067556324669706, 0.011813163757324219, 0.0
16307830810546875]
 mammals [0.11933806146572104, 0.06768012152777778, 0.029465192168237852,
0.06239286598959624]
Filter #31 has the following norms:
 birds [0.03958181818181818, 0.03558354687344005, 0.008212407430013021, 0.0
10626475016276041]
 mammals [0.08981814875431897, 0.06163975694444444, 0.02482596084118927, 0.
02998266039137974]
Filter #32 has the following norms:
 birds [0.02787272727272727, 0.021722852673965856, 0.004330952962239583, 0.
```

```
0055993398030598955]
 mammals [0.08664302600472813, 0.05587152777777778, 0.022681290790427848,
0.03467426306663364]
Filter #33 has the following norms:
 birds [0.055854545454545455, 0.0077456820526473425, 0.0023981730143229165,
0.006453514099121094]
 mammals [0.10608110565557374, 0.05632769097222222, 0.028942168237853515,
0.034188754025266284]
Filter #34 has the following norms:
 birds [0.057545454545454545, 0.0017887450497520716, 0.001872698465983073,
0.004750887552897136]
 mammals [0.09268776141116566, 0.044619791666666665, 0.025756889050036258,
0.023249938072826357]
Filter #35 has the following norms:
 birds [0.08778181818181818, 0.00911843988152684, 0.004895846048990886, 0.0
07004737854003906]
 mammals [0.08474631751227496, 0.04980729166666667, 0.023445431472081217,
0.02775328214020312]
Filter #36 has the following norms:
 birds [244.2154181818182, 245.01279576691405, 250.59400749206543, 250.9242
4297332764]
 mammals [209.5456973995272, 218.05806076388888, 208.8650525743292, 201.051
2558830815]
Filter #37 has the following norms:
 birds [253.2549272727273, 253.96708709108455, 254.0306453704834, 254.03466
701507568]
 mammals [238.4463447899618, 244.66769618055557, 221.17237218999276, 225.45
759722566262]
Filter #38 has the following norms:
 birds [254.71058181818182, 255.0, 254.70826148986816, 254.659836769104]
 mammals [245.76242225859247, 250.71989713541666, 228.64426667875273, 236.9
1217735942533]
Filter #39 has the following norms:
 birds [254.9497090909091, 255.0, 254.92179012298584, 254.87793827056885]
 mammals [248.97472995090016, 253.2586306423611, 234.59520123277738, 243.10
317562546444]
Filter #40 has the following norms:
 birds [215.06436363636362, 228.85784052713902, 226.63290532430014, 232.802
4056752523]
 mammals [148.798399709038, 163.70957508680556, 182.2474011965192, 160.7432
8957146395]
Filter #41 has the following norms:
 birds [121.43258181818182, 121.89353223068987, 119.53542709350586, 124.618
14594268799]
 mammals [77.58124931805783, 87.52983767361111, 106.78985768672952, 98.4966
4602427545]
Filter #42 has the following norms:
 birds [80.9556909090909, 81.26397717062132, 79.68952814737956, 83.07924747
467041]
 mammals [51.72440443717039, 58.35252951388889, 71.19299764321973, 65.66185
781520932]
Filter #43 has the following norms:
 birds [60.675563636363634, 60.90869080501847, 59.730752309163414, 62.27194
118499756]
```

```
 mammals [38.76909256228405, 43.73771137152778, 53.36260152284264, 49.22171
909834035]
Filter #44 has the following norms:
 birds [80.9556909090909, 81.26397717062132, 79.68952814737956, 83.07924747
467041]
 mammals [51.72440443717039, 58.35252951388889, 71.19299764321973, 65.66185
781520932]
Filter #45 has the following norms:
 birds [37.297436363636365, 37.44221105527638, 36.71491940816244, 38.277393
02317301]
 mammals [23.831234769958176, 26.884582465277777, 32.79862128353879, 30.252
370572207084]
Filter #46 has the following norms:
 birds [11.84989090909091, 11.894081333821426, 11.663843154907227, 12.16374
1111755371]
 mammals [7.569214402618658, 8.543243489583332, 10.428798948513416, 9.61934
6049046321]
Filter #47 has the following norms:
 birds [0.750109090909091, 0.8001430996039801, 0.8363768259684244, 0.880417
8237915039]
 mammals [0.5092562284051646, 0.5594578993055556, 0.6260623640319072, 0.597
7854842704979]
[1.0086085931834663, 1.0093817984046412, 0.9774139212286351, 0.859866237304
9294, 0.8802130294773347, 0.9491701758731731, 0.7028308556925308, 0.7109835
025380711, 0.7020621827411168, 0.6443844438939806, 0.6549764676740154, 0.68
50371646120377, 0.5396174764321973, 0.5543863306744018, 0.5317370624299558,
0.5117107010156057, 0.5091678721823136, 0.532532632342277, 0.32164612037708
484, 0.314537708484409, 0.2664899677978697, 0.27084468664850136, 0.26616299
232103047, 0.3038179840464104, 0.09402374909354605, 0.08796135744364628, 0.
07597225662620757, 0.0754520683675997, 0.06785236561803319, 0.0847807778053
0097, 0.029465192168237852, 0.02482596084118927, 0.022681290790427848, 0.02
8942168237853515, 0.023249938072826357, 0.023445431472081217, 201.051255883
0815, 221.17237218999276, 228.64426667875273, 234.59520123277738, 148.79839
9709038, 77.58124931805783, 51.72440443717039, 38.76909256228405, 51.724404
43717039, 23.831234769958176, 7.569214402618658, 0.5092562284051646, 0]
[6.525418181818182, 4.328946054777197, 1.1044931411743164, 1.10993989308675
```

Testing Threshold classifier where bird < threshold, mammal > threshold

In [ ]:
```
#store the number of correct classifications for each filter
correct_counts = [0] * 48

for i in range(num_filters):
    for j in range(num_images):
        if(norms[i][j] >= filter_thresholds[i]):
            if(j >= 4):
                correct_counts[i] = correct_counts[i] + 1
            printout = "Theshold states image #%d is a MAMMAL" % j
        else:
            if(j < 4):
                correct_counts[i] = correct_counts[i] + 1
            printout = "Theshold states image #%d is a BIRD" % j
        #print(printout)

average_counts = np.average(correct_counts)
max_counts = max(correct_counts)
min_counts = min(correct_counts)

printout = "Average correct: %d, highest correct: %d, least correct: %d" %
(average_counts, max_counts, min_counts)
print(printout)

summed_correct = 0

for i in range(num_images):
    if(image_norms[i] >= sum_threshold):
        if(i >= 4):
            summed_correct = summed_correct + 1
        printout = "Theshold states image #%d is a MAMMAL" % j
    else:
        if(j < 4):
            summed_correct = summed_correct + 1
        printout = "Theshold states image #%d is a BIRD" % j

printout = "Summed threshold correct: %s" % summed_correct
print(printout)
```

```
Average correct: 6, highest correct: 8, least correct: 4
Summed threshold correct: 4
```

In [ ]:
```
for i in range(num_filters):
    if(correct_counts[i] == 8):
        printout = "Filter #%d classified all correctly" % i
        print(printout)
```

```
Filter #14 classified all correctly
Filter #15 classified all correctly
Filter #16 classified all correctly
```

Try in RGB (Color) Format

In [ ]:
```
img = cv2.imread('cardinal1.jpg')
plt.figure(figsize = [10, 10])
plt.axis('off')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x1a58cff6e50>
```

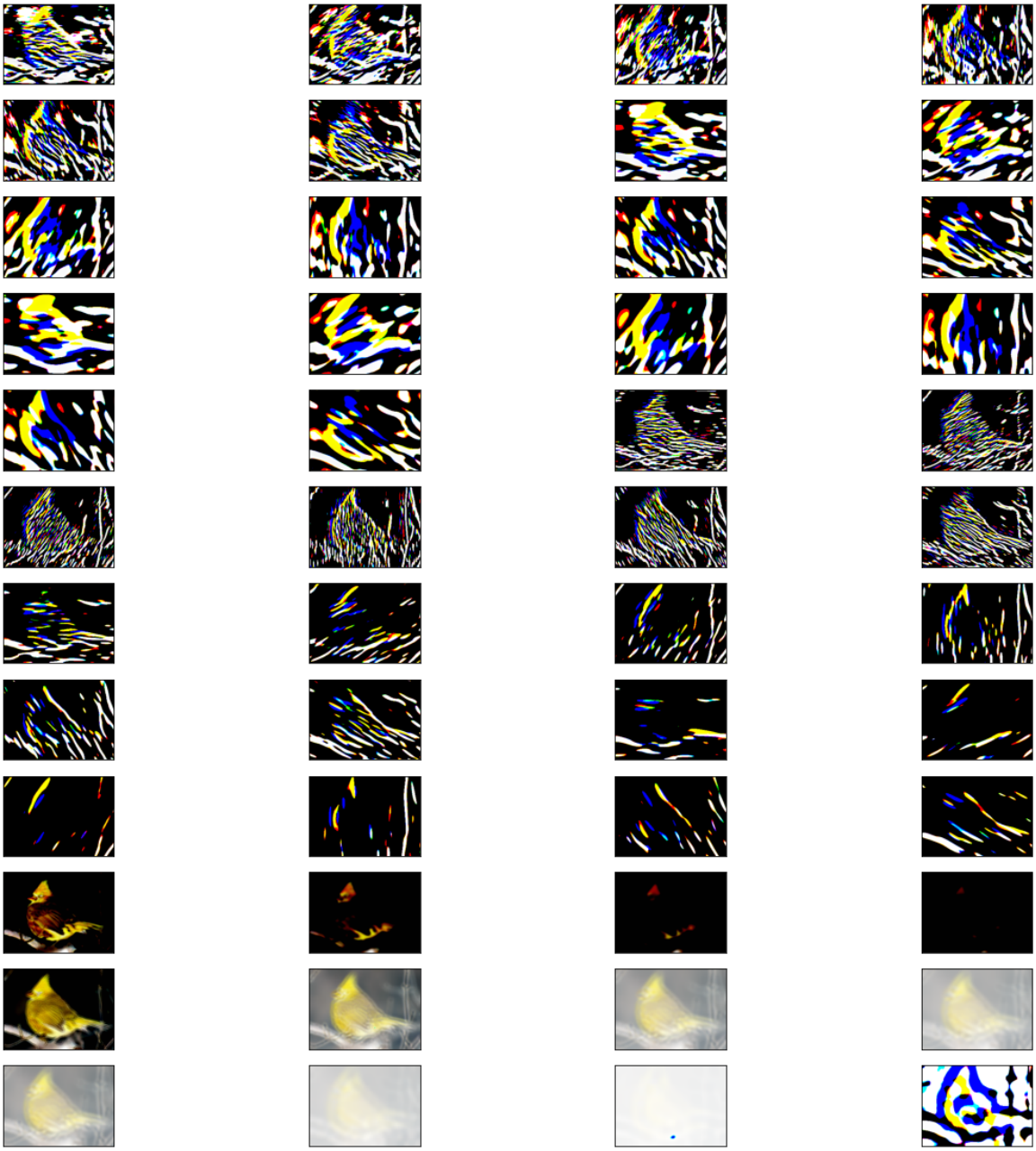
In [ ]:
```
# Load all Images
images_rgb = list()

# Birds
images_rgb.append(cv2.imread('cardinal1.jpg',1))
images_rgb.append(cv2.imread('cardinal2.jpg',1))
images_rgb.append(cv2.imread('woodduck1.jpeg',1))
images_rgb.append(cv2.imread('woodduck2.jpeg',1))

# Mammals
images_rgb.append(cv2.imread('leopard1.jpg',1))
images_rgb.append(cv2.imread('leopard2.jpg',1))
images_rgb.append(cv2.imread('panda1.jpg',1))
images_rgb.append(cv2.imread('panda2.jpg',1))
```

Loop and apply all filters to all images - RGB

In [ ]:
```
norms = [[0 for i in range(8)] for j in range(48)]

image_sums = [0] * 8

# Loop through each image
for i in range (num_images):

    # Filter image and store
    w, h, ch = images_rgb[i].shape
    _, _, num_filters = F.shape
    FB_responses = makeResponses_rgb(images_rgb[i], F)

    print(FB_responses.shape, images_rgb[i].shape)

    plt.figure(figsize=(20,20))
    plt.suptitle('LM filter responses')
    nr = 12
    nc = 4

    #For each filter...
    for j in range(num_filters):
        #Get magnitude for each filter of each image
        norms[j][i] = (computeMagnitudeRGB(FB_responses[:,:,:,j]))

        #Get sum of all filtered images
        if(j == 0):
            image_sums[i] = FB_responses[:,:,:,j]
        else:
            image_sums[i] = FB_responses[:,:,:,j] + image_sums[i]

        #Plot filtered images
        plt.subplot(nr, nc, j+1)
        fig = plt.imshow((FB_responses[:,:,:,j] * 255).astype(np.uint8))
        fig.axes.get_xaxis().set_visible(False)
        fig.axes.get_yaxis().set_visible(False)

    plt.show()

#print(np.matrix(norms))
```

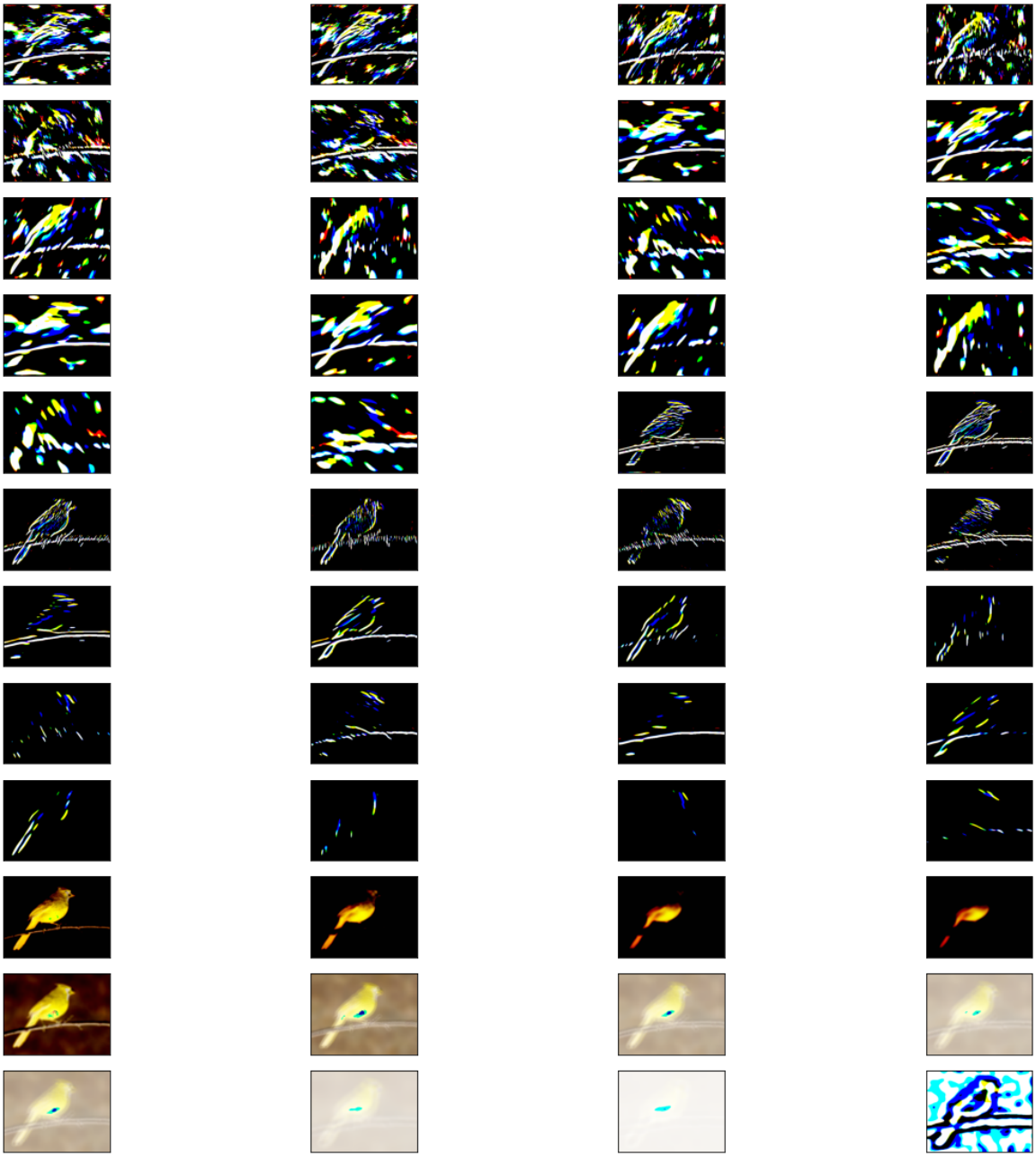
```
(200, 275, 3, 48) (200, 275, 3)
```

LM filter responses



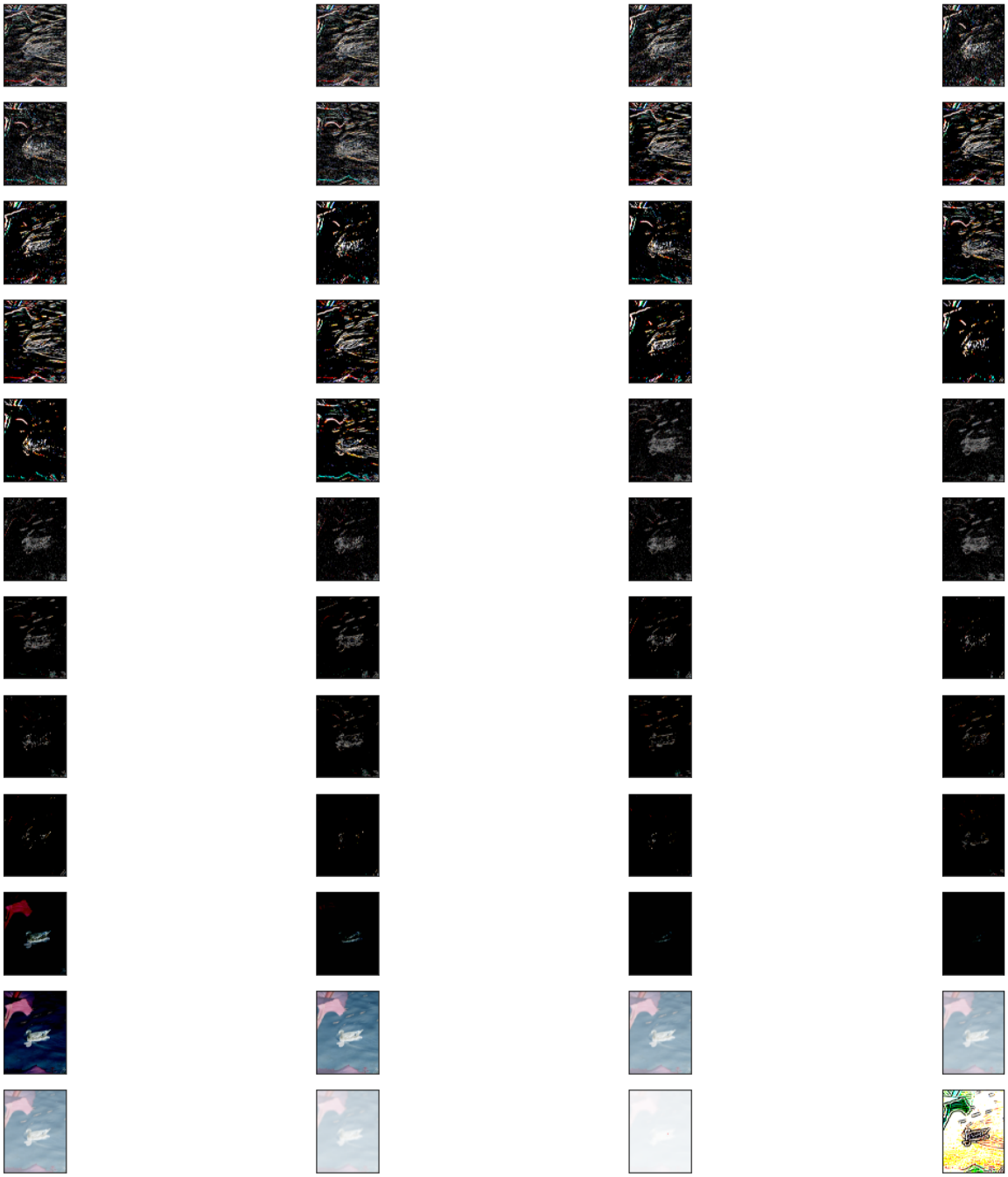
```
(302, 398, 3, 48) (302, 398, 3)
```

LM filter responses



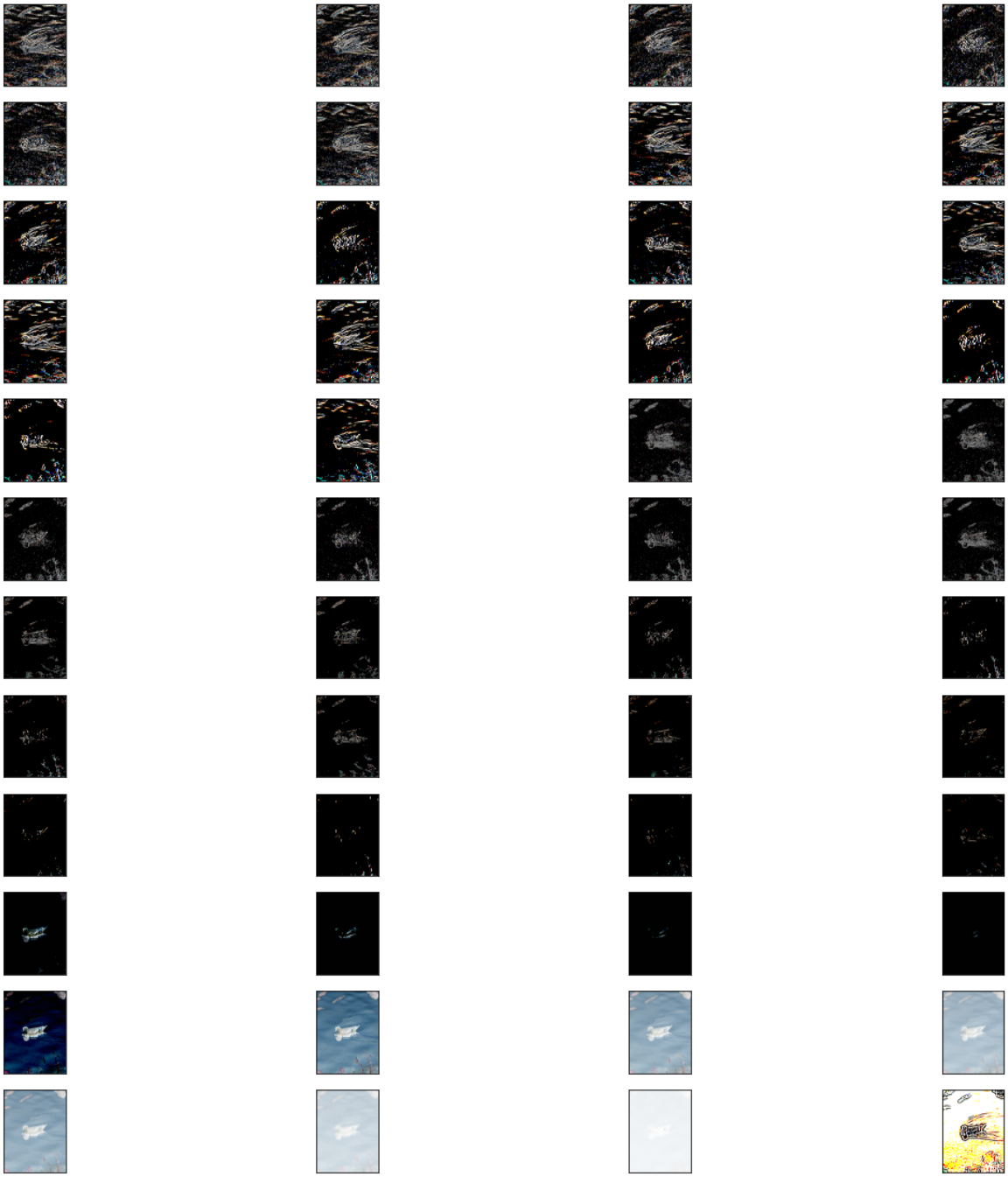
```
(2048, 1536, 3, 48) (2048, 1536, 3)
```

LM filter responses



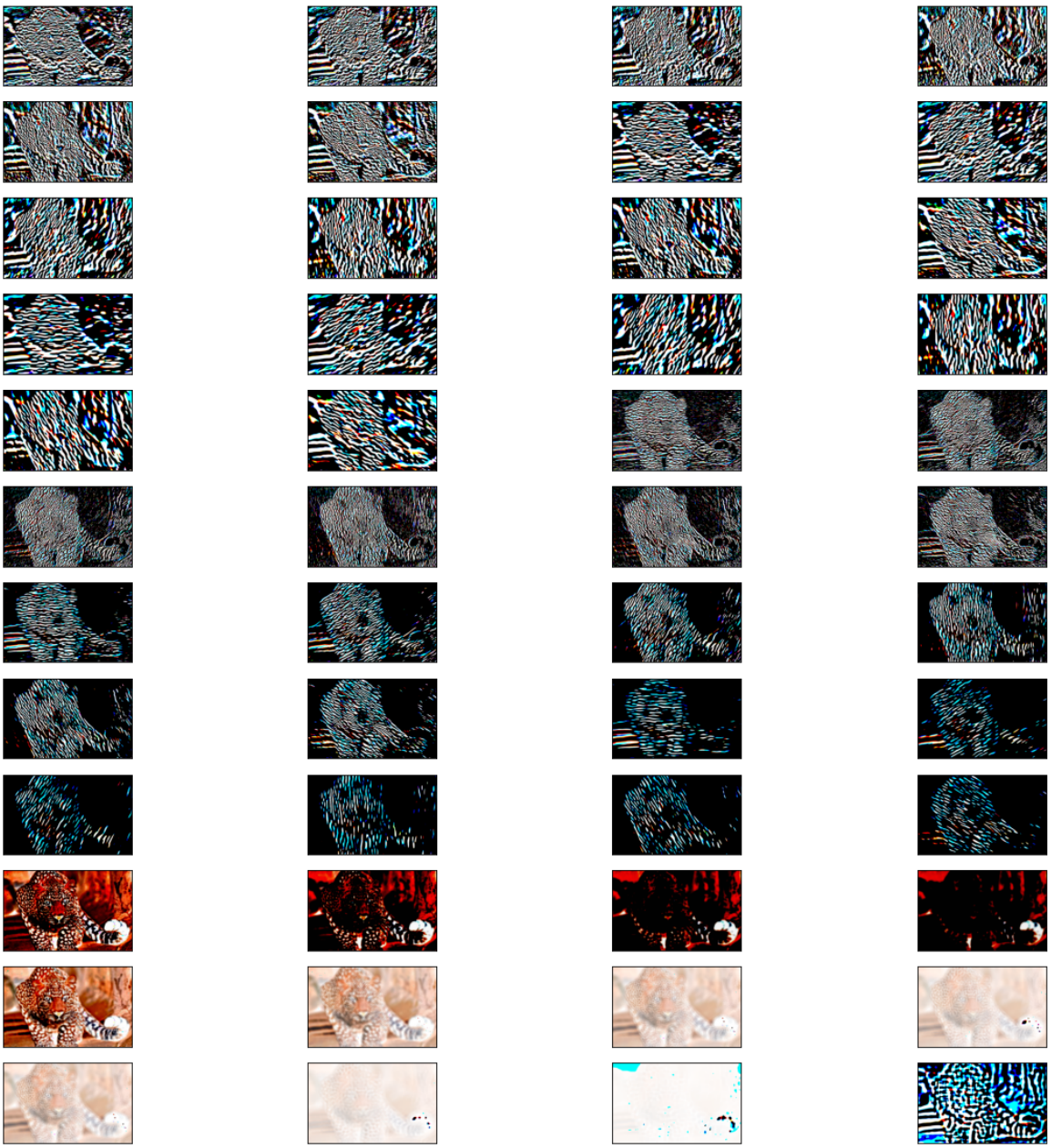
(2048, 1536, 3, 48) (2048, 1536, 3)

LM filter responses



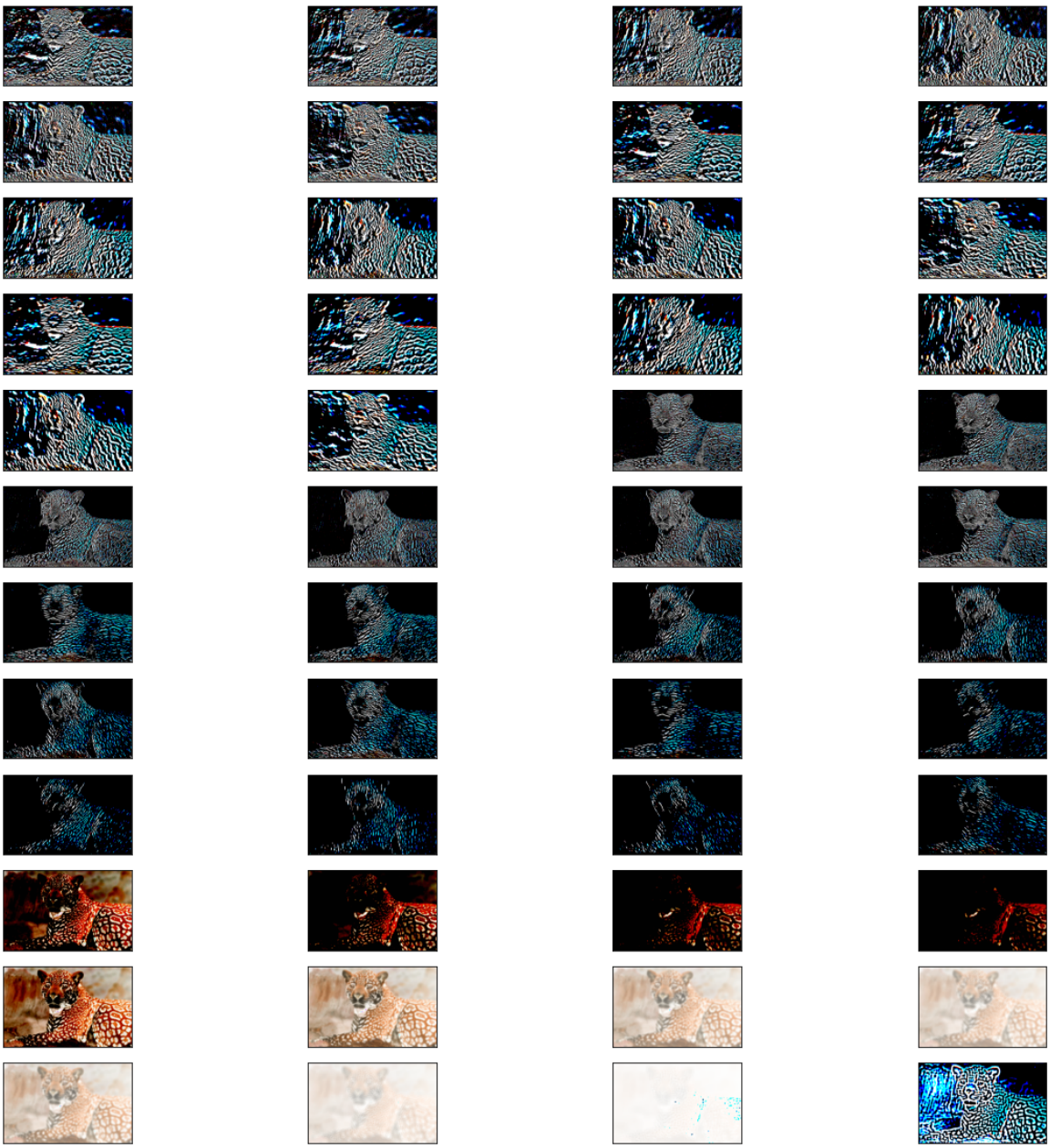
```
(585, 940, 3, 48) (585, 940, 3)
```

LM filter responses



```
(1200, 1920, 3, 48) (1200, 1920, 3)
```

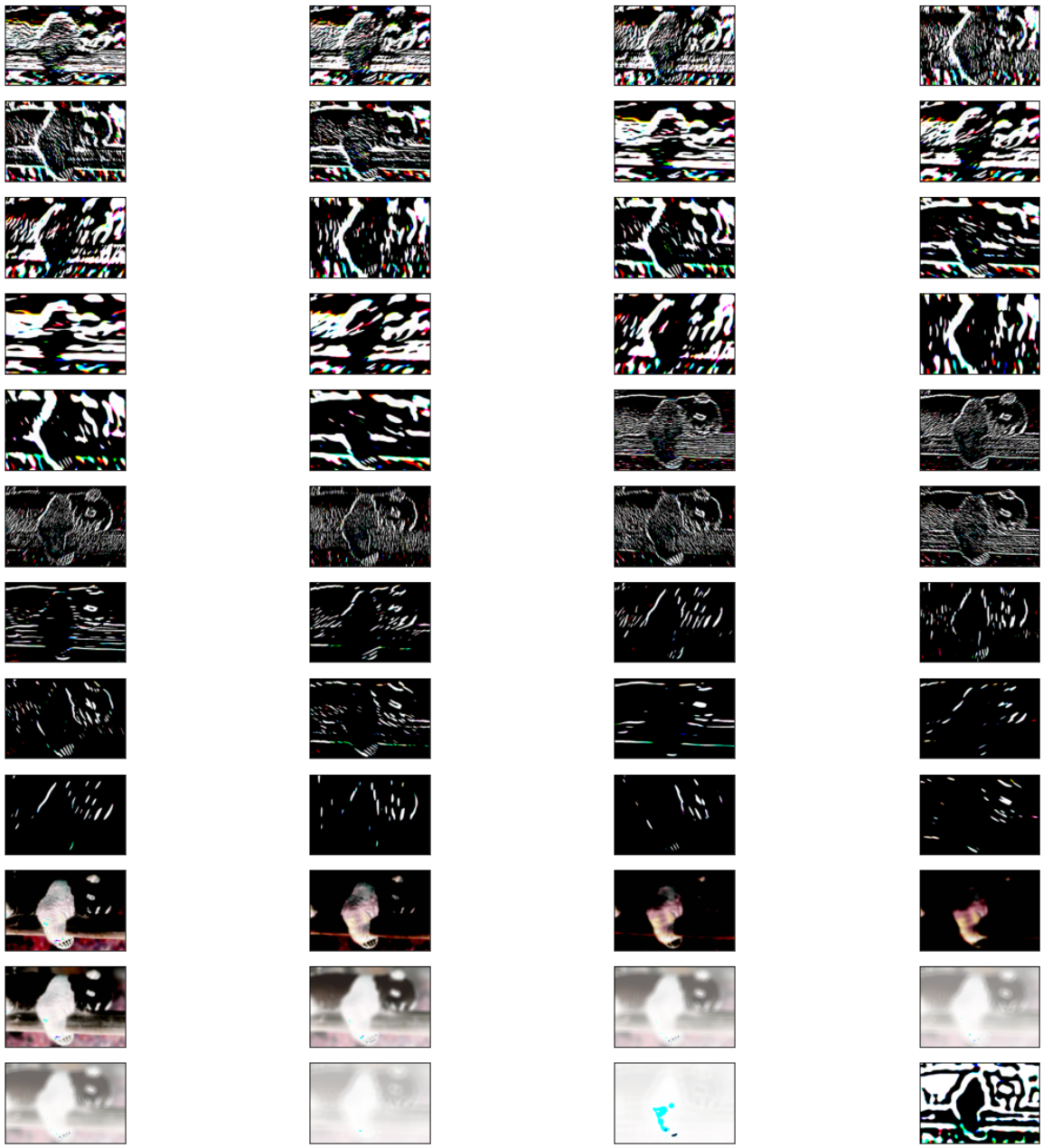
LM filter responses



(788, 1400, 3, 48) (788, 1400, 3)

LM filter responses

```
(367, 550, 3, 48) (367, 550, 3)
```



Thresholds for RGB images

In [ ]:
```
# Get norms of summed responses
image_norms = [0] * num_images


for i in range (num_images):
    w, h, ch = images_rgb[i].shape
    resolution = w*h
    image_norms[i] = np.sum(abs(image_sums[i]))/resolution

print(image_norms)
```

```
[4871.357090909091, 4752.825851109854, 4922.892136891683, 5025.65284824371
3, 3832.031067466812, 4112.056933159723, 4081.0276078680204, 4163.959033936
091]
```

In [ ]:
```
filter_thresholds = [0] * 49
sum_threshold = 0

bird_norms = [0] * 4
mammal_norms = [0] * 4

bird_norms_sum = [0] * 4
mammal_norms_sum = [0] * 4

for i in range (num_filters):

    #Split norms into birds and mammals
    bird_norms[0] = norms[i][0]
    bird_norms[1] = norms[i][1]
    bird_norms[2] = norms[i][2]
    bird_norms[3] = norms[i][3]

    mammal_norms[0] = norms[i][4]
    mammal_norms[1] = norms[i][5]
    mammal_norms[2] = norms[i][6]
    mammal_norms[3] = norms[i][7]

    # Calculate ideal threshold value
    if(max(bird_norms) < min(mammal_norms)):
        filter_thresholds[i] = (max(bird_norms) + min(mammal_norms)) / 2
    else:
        filter_thresholds[i] = min(mammal_norms)

    printout = "Filter #%d has the following norms:\n birds %s \n mammals %
s" % (i, bird_norms, mammal_norms)
    print(printout)

#Summed threshold
bird_norms_sum[0] = image_norms[0]
bird_norms_sum[1] = image_norms[1]
bird_norms_sum[2] = image_norms[2]
bird_norms_sum[3] = image_norms[3]

mammal_norms_sum[0] = image_norms[4]
mammal_norms_sum[1] = image_norms[5]
mammal_norms_sum[2] = image_norms[6]
mammal_norms_sum[3] = image_norms[7]

if(max(bird_norms_sum) < min(mammal_norms_sum)):
    sum_threshold = (max(bird_norms_sum) + min(mammal_norms_sum)) / 2
else:
    sum_threshold = min(mammal_norms_sum)

print(filter_thresholds)
print(sum_threshold)
```

```
Filter #0 has the following norms:
 birds [5.077036363636363, 2.4731937834869715, 1.3591200510660808, 1.536248
8428751628]
 mammals [5.371705764684488, 3.2363980034722224, 3.1209191443074693, 3.7715
9772108001]
Filter #1 has the following norms:
 birds [4.3978, 2.368706113348198, 1.239296277364095, 1.3882598876953125]
 mammals [5.374839061647572, 3.2905876736111113, 3.1234109862219, 3.4653108
74411692]
Filter #2 has the following norms:
 birds [3.994327272727273, 1.717661153449366, 0.9413544336954752, 1.0508257
548014324]
 mammals [5.453296963084197, 3.2327421875, 3.0786430384336474, 2.9566955660
143672]
Filter #3 has the following norms:
 birds [4.125381818181818, 1.2003976837831543, 0.7070344289143881, 0.843354
860941569]
 mammals [5.459408983451537, 3.144774739583333, 3.107391225525743, 2.600094
1293039386]
Filter #4 has the following norms:
 birds [4.158472727272727, 1.2020449931778097, 0.768975575764974, 0.9361321
131388346]
 mammals [5.596917621385707, 3.1429305555555556, 3.072102066715011, 2.64118
90017339608]
Filter #5 has the following norms:
 birds [4.424581818181818, 1.6439315784219108, 1.058672587076823, 1.2487312
95267741]
 mammals [5.582853246044736, 3.291665798611111, 3.0562844452501814, 2.84289
3237552638]
Filter #6 has the following norms:
 birds [3.3940727272727274, 1.7856917035508668, 0.8441912333170573, 0.91274
45220947266]
 mammals [3.291382069467176, 2.3068802083333333, 2.158105511240029, 2.97627
4461233589]
Filter #7 has the following norms:
 birds [2.813290909090909, 1.659006955306333, 0.7651576995849609, 0.8065137
86315918]
 mammals [3.205282778687034, 2.3662938368055557, 2.176333393763597, 2.68865
98959623482]
Filter #8 has the following norms:
 birds [2.429, 1.1802555825485042, 0.5233866373697916, 0.553307851155599]
 mammals [3.1902891434806326, 2.3278880208333335, 2.1488769035532993, 2.246
3958384939313]
Filter #9 has the following norms:
 birds [2.497218181818182, 0.858664181836334, 0.35155709584554035, 0.414901
4155069987]
 mammals [3.2459210765593745, 2.2753962673611112, 2.1831680565627267, 1.943
1904879861284]
Filter #10 has the following norms:
 birds [2.437018181818182, 0.7957003560850611, 0.37615203857421875, 0.45965
735117594403]
 mammals [3.24092925986543, 2.2860264756944444, 2.187891588107324, 1.958563
289571464]
Filter #11 has the following norms:
```

```
 birds [2.611781818181818, 1.0542780125794535, 0.596660296122233, 0.6848344
802856445]
 mammals [3.219234406255683, 2.402087673611111, 2.116092277012328, 2.118835
769135497]
Filter #12 has the following norms:
 birds [2.6074545454545452, 1.313612765815834, 0.6141157150268555, 0.630739
5299275717]
 mammals [2.185473722494999, 1.7282135416666666, 1.6447951414068165, 2.5201
535793906364]
Filter #13 has the following norms:
 birds [2.1522363636363635, 1.2602748843555527, 0.5690148671468099, 0.57445
49433390299]
 mammals [2.126759410801964, 1.7709513888888888, 1.6819760696156636, 2.3107
70374040129]
Filter #14 has the following norms:
 birds [1.7218, 0.8970265233452028, 0.36248048146565753, 0.360110600789388]
 mammals [2.0859556282960536, 1.7452799479166667, 1.680084300217549, 1.8531
483775080506]
Filter #15 has the following norms:
 birds [1.7160181818181819, 0.6676428500116477, 0.2503407796223958, 0.27413
050333658856]
 mammals [2.13409347154028, 1.710763454861111, 1.6894307469180565, 1.603586
8218974485]
Filter #16 has the following norms:
 birds [1.674, 0.6014093647043163, 0.24918047587076822, 0.2939138412475586]
 mammals [2.0962975086379343, 1.751884982638889, 1.7223404641044235, 1.6069
804310131286]
Filter #17 has the following norms:
 birds [1.7979272727272728, 0.7990199341076242, 0.42957719167073566, 0.4762
795766194661]
 mammals [2.0883942535006366, 1.8570889756944444, 1.6409662799129805, 1.779
2073321773594]
Filter #18 has the following norms:
 birds [1.9756363636363636, 0.8155429465206829, 0.43026065826416016, 0.5544
583002726237]
 mammals [2.010412802327696, 1.0083272569444444, 1.0191642494561277, 1.2698
687143918752]
Filter #19 has the following norms:
 birds [1.6216727272727274, 0.7499584012779127, 0.34057776133219403, 0.4462
728500366211]
 mammals [1.9699927259501728, 0.9833676215277778, 0.9918255982596084, 0.945
4446371067624]
Filter #20 has the following norms:
 birds [1.5406181818181819, 0.515183533561849, 0.23132832845052084, 0.31005
57327270508]
 mammals [2.020574649936352, 0.9195993923611111, 0.9302338651196519, 0.8015
011146891255]
Filter #21 has the following norms:
 birds [1.8057818181818182, 0.32024360211654296, 0.18440723419189453, 0.280
8418273925781]
 mammals [2.096973995271868, 0.85805859375, 0.967141950688905, 0.8227198414
664355]
Filter #22 has the following norms:
 birds [1.8723272727272726, 0.26891077906086724, 0.19286346435546875, 0.285
```

```
04498799641925]
 mammals [2.151258410620113, 0.8516488715277778, 0.9313043872371284, 0.8020
708446866485]
Filter #23 has the following norms:
 birds [2.2628545454545455, 0.4367200239608639, 0.3086945215861003, 0.40691
407521565753]
 mammals [2.1198508819785418, 0.9168255208333334, 0.9526686004350979, 0.933
5595739410454]
Filter #24 has the following norms:
 birds [0.582709090909091, 0.3422326866118673, 0.11121495564778645, 0.15610
91740926107]
 mammals [0.742522276777596, 0.3880251736111111, 0.29846718636693254, 0.411
82561307901905]
Filter #25 has the following norms:
 birds [0.43994545454545453, 0.27894439082831374, 0.08780956268310547, 0.12
097422281901042]
 mammals [0.6780160029096199, 0.39387022569444446, 0.2874818709209572, 0.26
310131285608124]
Filter #26 has the following norms:
 birds [0.41858181818181817, 0.17412393091284237, 0.05420239766438802, 0.07
589499155680339]
 mammals [0.6708619749045281, 0.3719513888888889, 0.26574691805656275, 0.22
790686153083972]
Filter #27 has the following norms:
 birds [0.5550727272727273, 0.08234883024393491, 0.033032735188802086, 0.06
791400909423828]
 mammals [0.7216166575741044, 0.34442838541666665, 0.2880293691080493, 0.22
77235570968541]
Filter #28 has the following norms:
 birds [0.5723090909090909, 0.049419281839661884, 0.0327447255452474, 0.066
81187947591145]
 mammals [0.7187761411165666, 0.3235190972222222, 0.2745839376359681, 0.203
33911320287343]
Filter #29 has the following norms:
 birds [0.6974545454545454, 0.12155978568338381, 0.06879234313964844, 0.102
17698415120442]
 mammals [0.6917857792325878, 0.34581119791666665, 0.2725235678027556, 0.25
22764429031459]
Filter #30 has the following norms:
 birds [0.24543636363636365, 0.1563279976039136, 0.03716564178466797, 0.050
71576436360677]
 mammals [0.3288288779778141, 0.18456553819444443, 0.09607777374909354, 0.1
8387416398315581]
Filter #31 has the following norms:
 birds [0.1418, 0.10696695397517389, 0.026563644409179688, 0.03354708353678
385]
 mammals [0.24389707219494453, 0.16883723958333333, 0.08144670050761421, 0.
08966559326232351]
Filter #32 has the following norms:
 birds [0.10394545454545455, 0.06581749808645879, 0.014115651448567709, 0.0
17748196919759113]
 mammals [0.23089288961629387, 0.15228342013888888, 0.07302121102248006, 0.
102764429031459]
Filter #33 has the following norms:
```

```
 birds [0.20703636363636363, 0.02323704615794203, 0.007548967997233073, 0.0
19765218098958332]
 mammals [0.2857501363884343, 0.15419401041666667, 0.09179024655547498, 0.1
0168937329700273]
Filter #34 has the following norms:
 birds [0.19283636363636364, 0.009409630936137642, 0.0060304005940755205,
0.014407475789388021]
 mammals [0.2526204764502637, 0.12650434027777777, 0.07874728063814358, 0.0
6983403517463463]
Filter #35 has the following norms:
 birds [0.2776727272727273, 0.029218942394089654, 0.015852610270182293, 0.0
22176742553710938]
 mammals [0.23090380069103472, 0.13743402777777777, 0.07443437273386512, 0.
08335397572454793]
Filter #36 has the following norms:
 birds [708.0578363636364, 713.8875004159872, 746.0532073974609, 753.107298
8510132]
 mammals [561.8210420076377, 615.4265407986111, 593.8919987309645, 592.8734
505821154]
Filter #37 has the following norms:
 birds [749.4895636363636, 734.6568355020133, 762.115241686503, 762.2295668
919882]
 mammals [669.5216312056738, 711.6100451388888, 656.3985360768673, 670.4188
803567005]
Filter #38 has the following norms:
 birds [761.1848363636364, 743.532862990449, 764.2174612681071, 764.0578155
517578]
 mammals [708.3588797963266, 737.3992521701389, 685.4718772661349, 706.5736
091156799]
Filter #39 has the following norms:
 birds [764.5038727272728, 749.4549569037239, 764.7910324732462, 764.663864
4536337]
 mammals [727.2327295871977, 749.2702074652777, 705.4531236403191, 725.9242
655437206]
Filter #40 has the following norms:
 birds [633.6964545454546, 636.038861526174, 664.0179961522421, 687.1330267
588297]
 mammals [397.66315148208764, 449.3359075520833, 472.6640609137056, 474.252
66782264055]
Filter #41 has the following norms:
 birds [365.2576363636364, 353.6495390861593, 372.2029571533203, 393.922877
62959796]
 mammals [210.9112965993817, 245.54362760416666, 282.8641252719362, 290.466
01436710426]
Filter #42 has the following norms:
 birds [243.4988, 235.76322839362373, 248.13358052571616, 262.6150865554809
6]
 mammals [140.61666848517913, 163.69437022569446, 188.57527556200145, 193.6
4033192965073]
Filter #43 has the following norms:
 birds [182.50905454545455, 176.71302705580885, 185.98713811238608, 196.841
35945638022]
 mammals [105.39749408983451, 122.69727517361112, 141.3477782813633, 145.14
761456527125]
```

```
Filter #44 has the following norms:
 birds [243.4988, 235.76322839362373, 248.13358052571616, 262.6150865554809
6]
 mammals [140.61666848517913, 163.69437022569446, 188.57527556200145, 193.6
4033192965073]
Filter #45 has the following norms:
 birds [112.18527272727273, 108.62122699590668, 114.32243537902832, 120.998
14351399739]
 mammals [64.78721767594108, 75.41623697916667, 86.88685279187817, 89.21870
696061431]
Filter #46 has the following norms:
 birds [35.63038181818182, 34.51934340577058, 36.31736183166504, 38.4297733
3068848]
 mammals [20.57806146572104, 23.957166666666666, 27.634176033357505, 28.353
371315333167]
Filter #47 has the following norms:
 birds [2.3014727272727273, 2.2005557589270857, 2.4106709162394204, 2.56197
8022257487]
 mammals [1.411656664848154, 1.514828125, 1.7010270123277738, 1.77372306167
9465]
[3.1209191443074693, 3.1234109862219, 2.9566955660143672, 2.600094129303938
6, 2.6411890017339608, 2.842893237552638, 2.158105511240029, 2.176333393763
597, 2.1488769035532993, 1.9431904879861284, 1.958563289571464, 2.116092277
012328, 1.6447951414068165, 1.6819760696156636, 1.680084300217549, 1.603586
8218974485, 1.6069804310131286, 1.6409662799129805, 1.0083272569444444, 0.9
454446371067624, 0.8015011146891255, 0.8227198414664355, 0.802070844686648
5, 0.9168255208333334, 0.29846718636693254, 0.26310131285608124, 0.22790686
153083972, 0.2277235570968541, 0.20333911320287343, 0.2522764429031459, 0.0
9607777374909354, 0.08144670050761421, 0.07302121102248006, 0.0917902465554
7498, 0.06983403517463463, 0.07443437273386512, 561.8210420076377, 656.3985
360768673, 685.4718772661349, 705.4531236403191, 397.66315148208764, 210.91
```

In [ ]:
```
#store the number of correct classifications for each filter
correct_counts = [0] * 48

for i in range(num_filters):
    for j in range(num_images):
        if(norms[i][j] >= filter_thresholds[i]):
            if(j >= 4):
                correct_counts[i] = correct_counts[i] + 1
            printout = "Theshold states image #%d is a MAMMAL" % j
        else:
            if(j < 4):
                correct_counts[i] = correct_counts[i] + 1
            printout = "Theshold states image #%d is a BIRD" % j
        print(printout)

average_counts = np.average(correct_counts)
max_counts = max(correct_counts)
min_counts = min(correct_counts)

printout = "Average correct: %d, highest correct: %d, least correct: %d" %
(average_counts, max_counts, min_counts)
print(printout)
```

```
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
```

```
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
```

```
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
```

```
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
```

```
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
```

```
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a BIRD
Theshold states image #2 is a BIRD
Theshold states image #3 is a BIRD
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
```

```
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
Theshold states image #4 is a MAMMAL
Theshold states image #5 is a MAMMAL
Theshold states image #6 is a MAMMAL
Theshold states image #7 is a MAMMAL
Theshold states image #0 is a MAMMAL
Theshold states image #1 is a MAMMAL
```

```
Theshold states image #2 is a MAMMAL
Theshold states image #3 is a MAMMAL
```

In [ ]:
```
for i in range(num_filters):
    if(correct_counts[i] == 8):
        printout = "Filter #%d classified all correctly" % i
        print(printout)
```

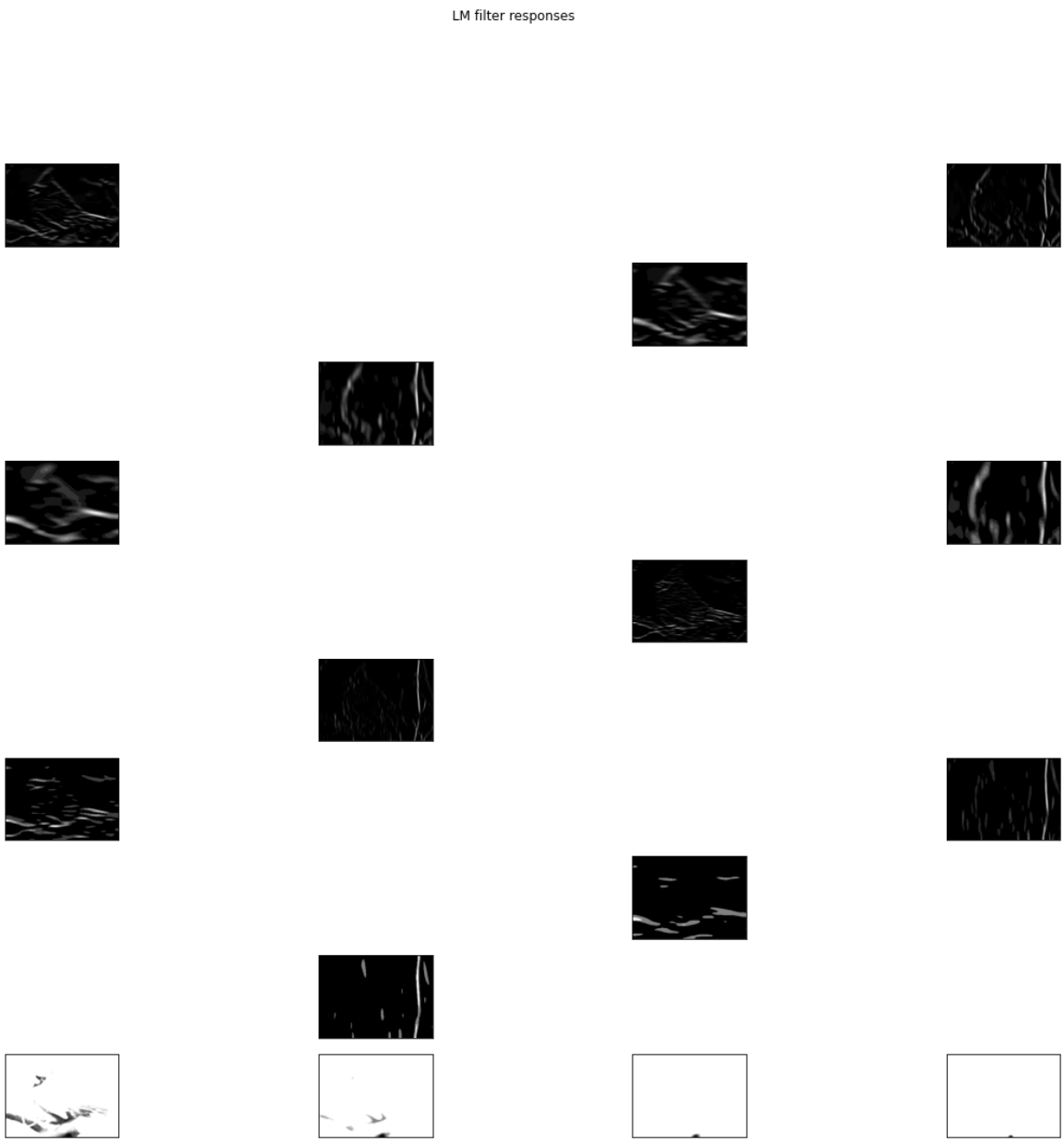
Comparing Separable vs. Non-Separable compute times

In [ ]:
```
# Get separable filter arrays for filters 0, 3, 6, 9, 12, 15, 18, 21, 24, 2
7, 30, 33, 36, 37, 38, 39

sep_list = [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 37, 38, 39]

# i = kernelX, i + len(sep_list) = kernelY
filter_vectors = [0] * len(sep_list)*2


index = 0

for i in (sep_list):
    U, E, V = np.linalg.svd(F[:,:,i])

    VT = np.transpose(V)

    kernelX = np.sqrt(E[0]) * U[0]
    kernelY = np.sqrt(E[0]) * VT[0]

    filter_vectors[index] = kernelX
    filter_vectors[index + len(sep_list)] = kernelY

    index = index + 1
```

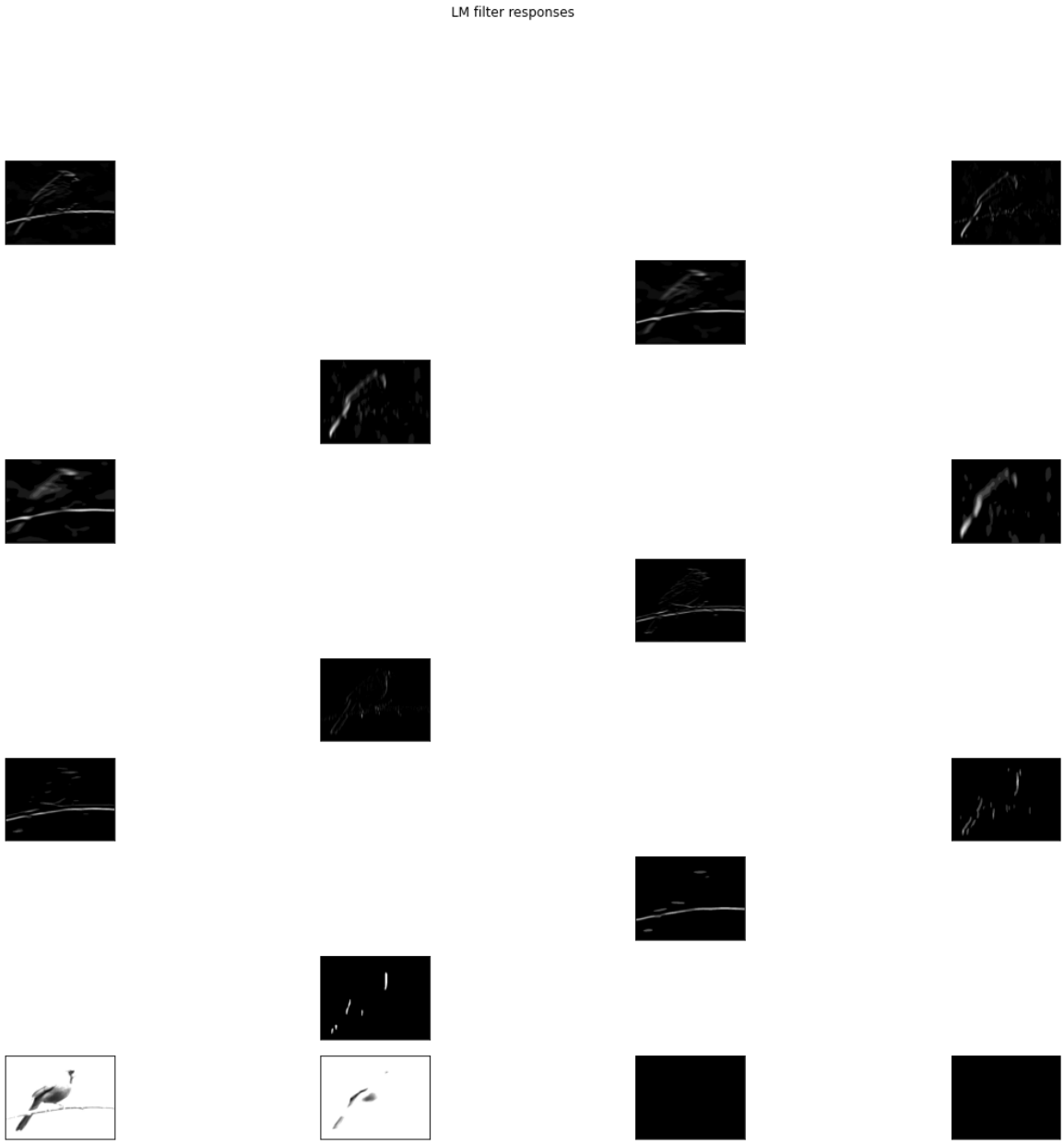
Filter grayscale images with non-separated separable filters only

In [ ]:
```
num_images = len(images)
num_filters = 48


# Loop through each image
for i in range (0, num_images):

    # Filter image and store
    FB_responses = makeResponses(images[i], F)
    print(FB_responses.shape, images[i].shape)

    plt.figure(figsize=(20,20))
    plt.suptitle('LM filter responses')
    nr = 12
    nc = 4

    #For each filter...
    for j in sep_list:

        #Plot filtered images
        plt.subplot(nr, nc, j+1)
        fig = plt.imshow(FB_responses[:,:,j], cmap='gray')
        fig.axes.get_xaxis().set_visible(False)
        fig.axes.get_yaxis().set_visible(False)

    plt.show()

#print(np.matrix(norms))
```

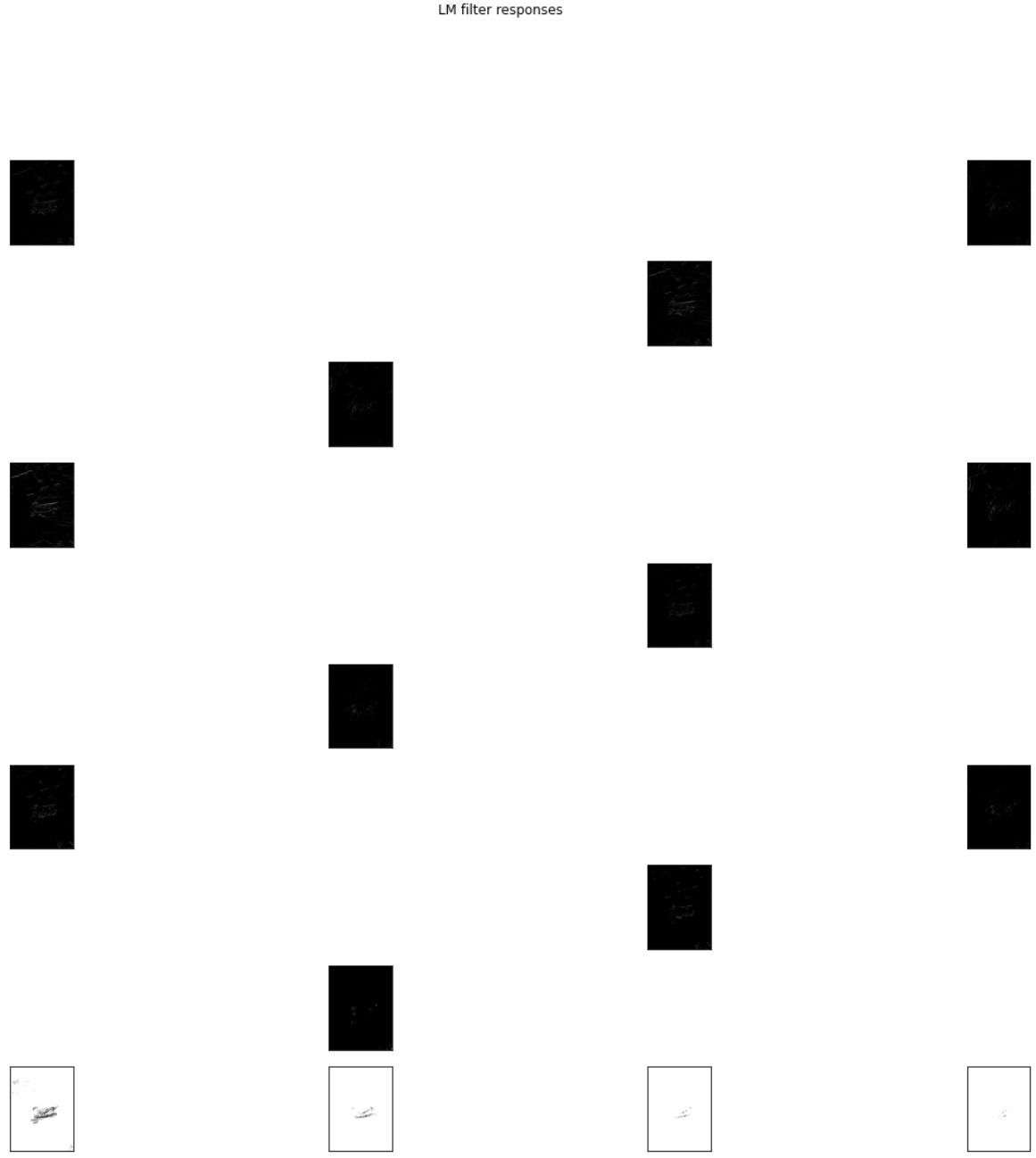
```
(200, 275, 48) (200, 275)
```



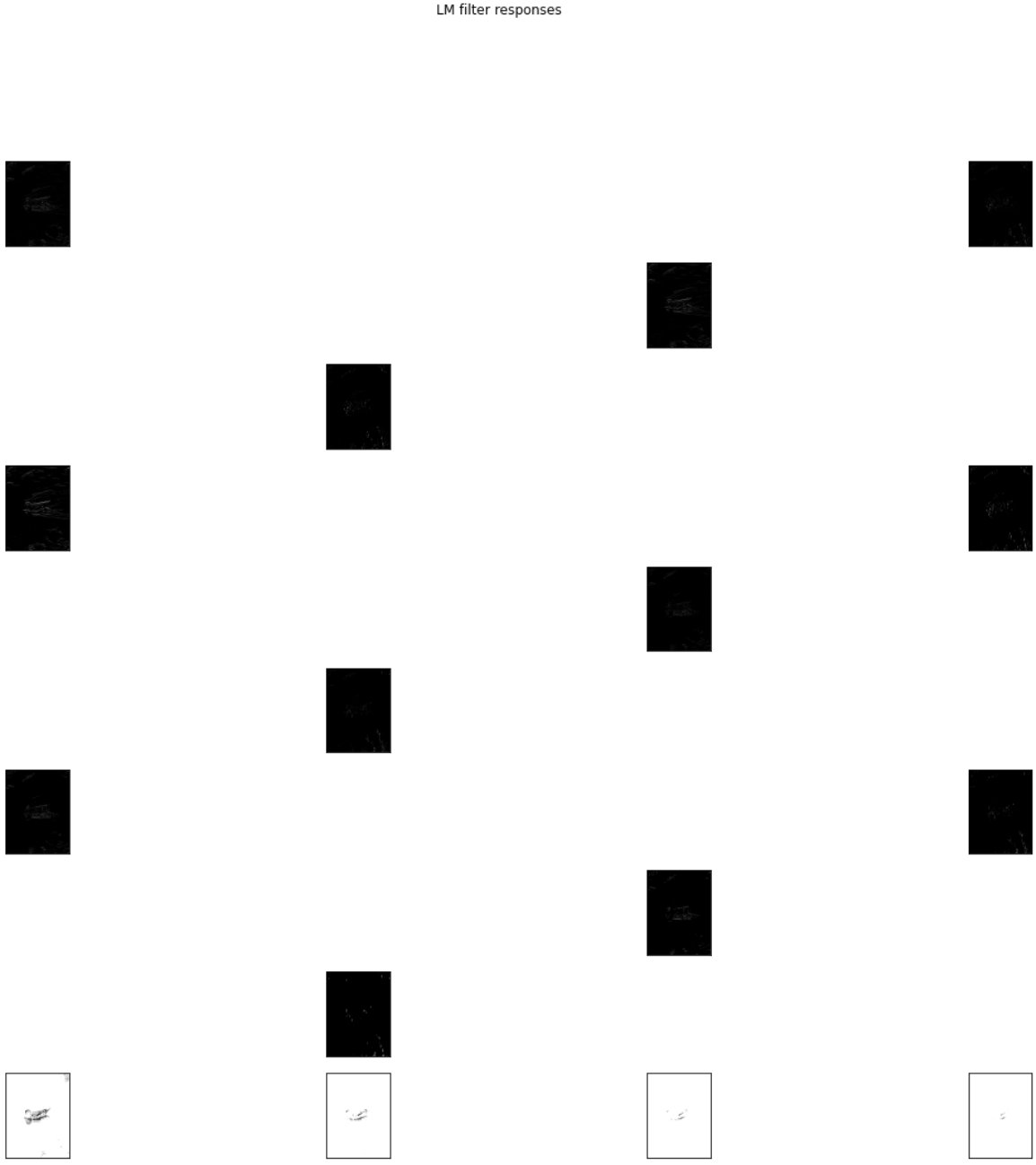

```
(302, 398, 48) (302, 398)
```



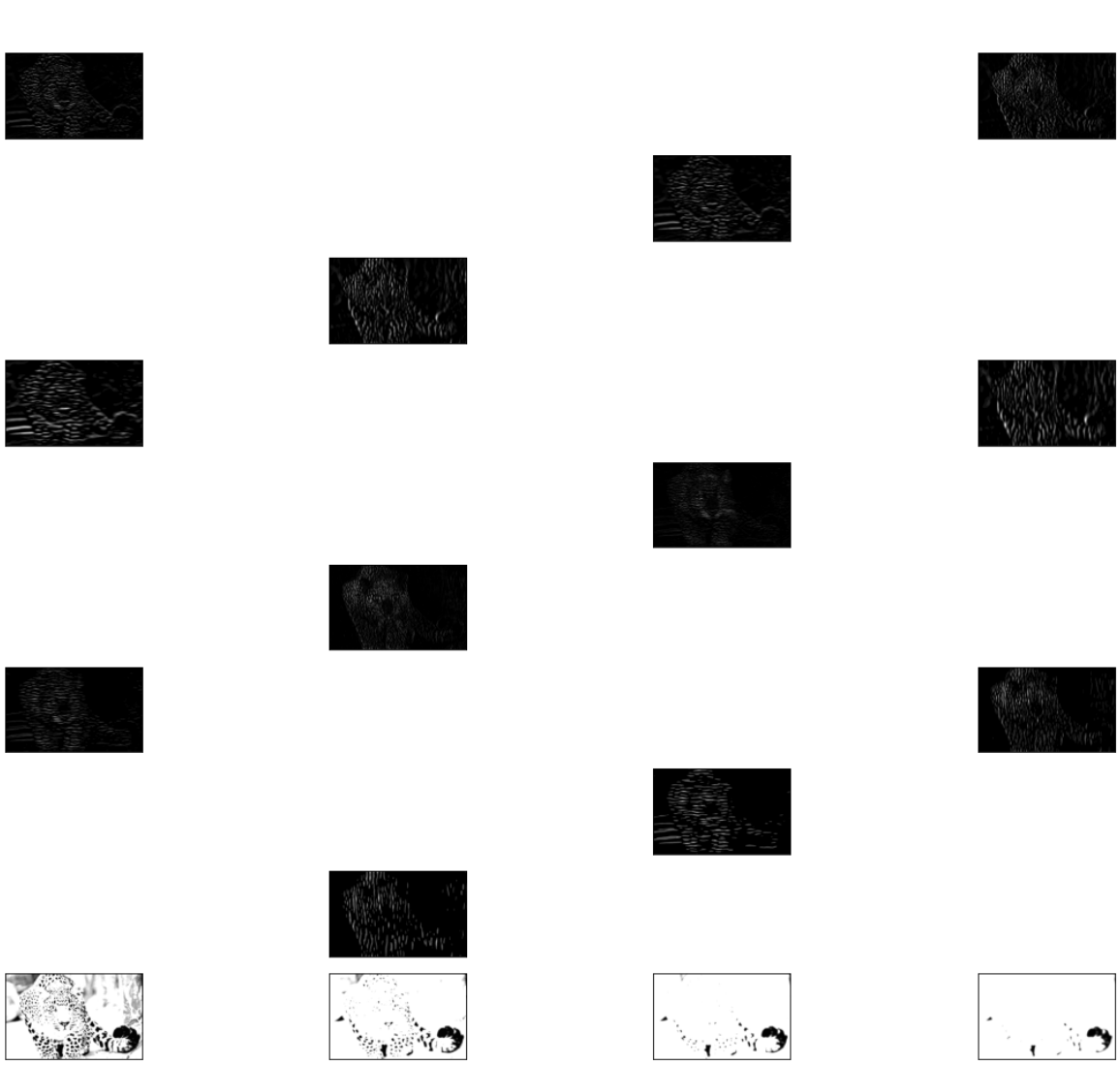

```
(2048, 1536, 48) (2048, 1536)
```



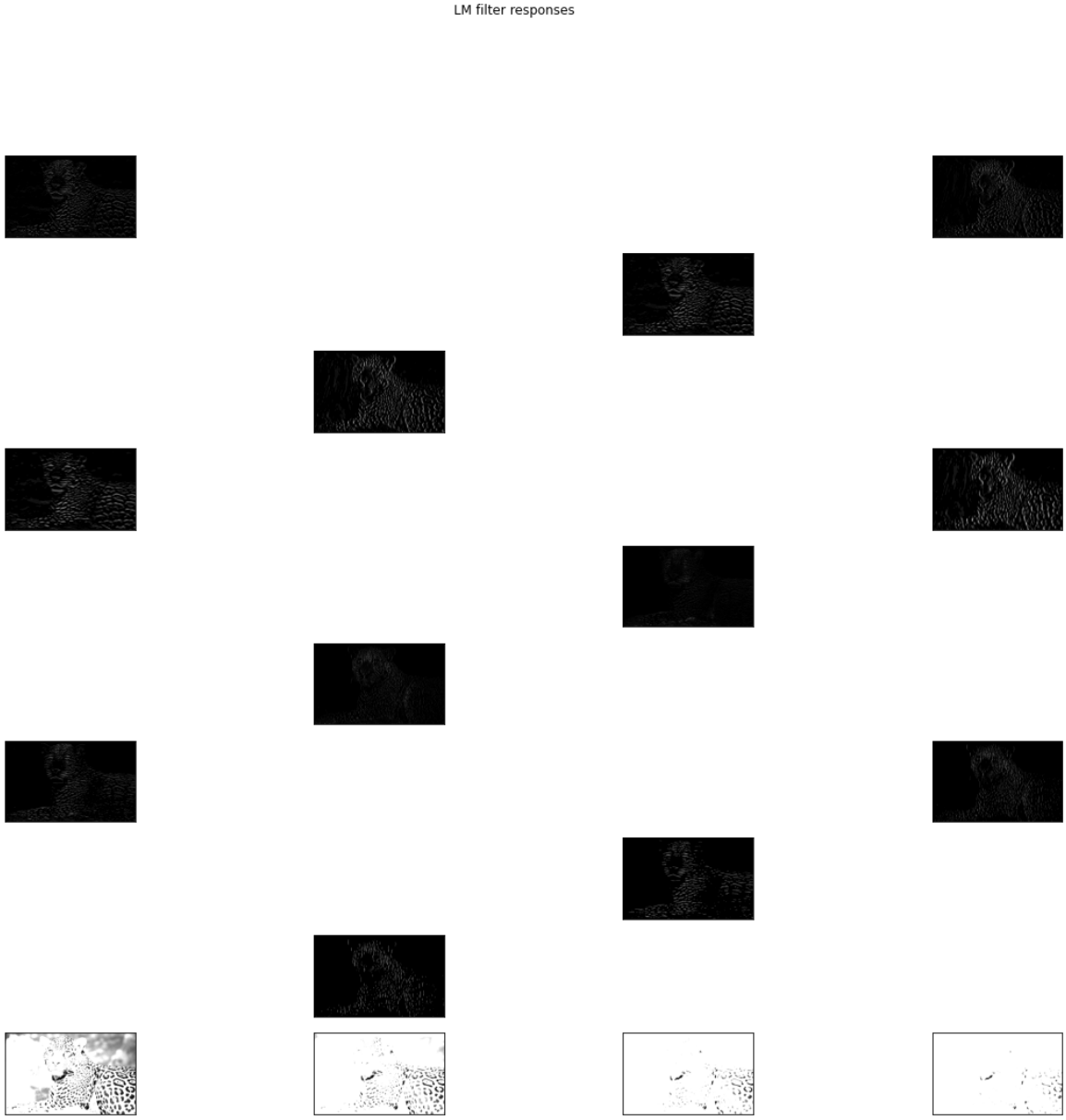
(2048, 1536, 48) (2048, 1536)



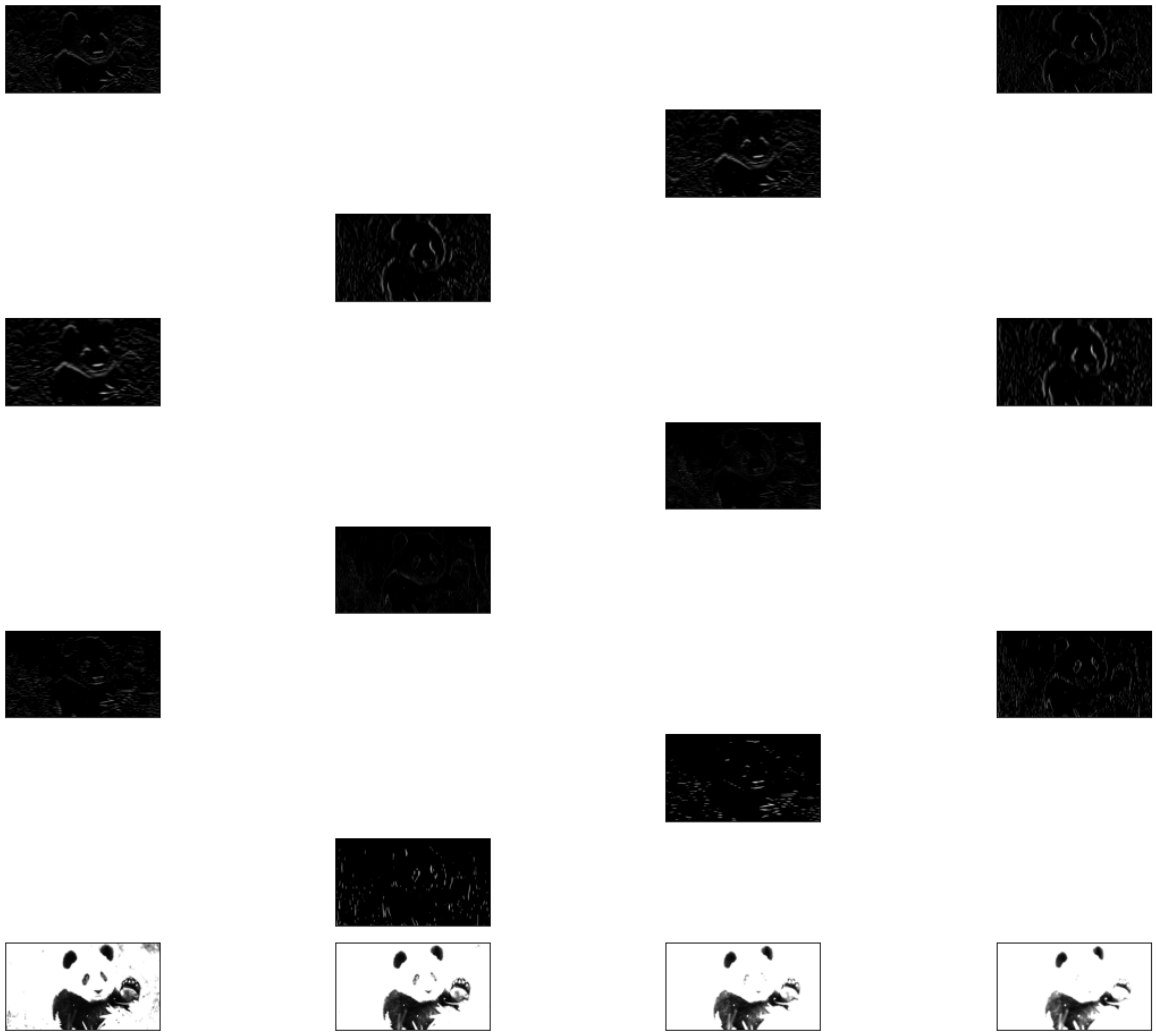

```
(585, 940, 48) (585, 940)
```



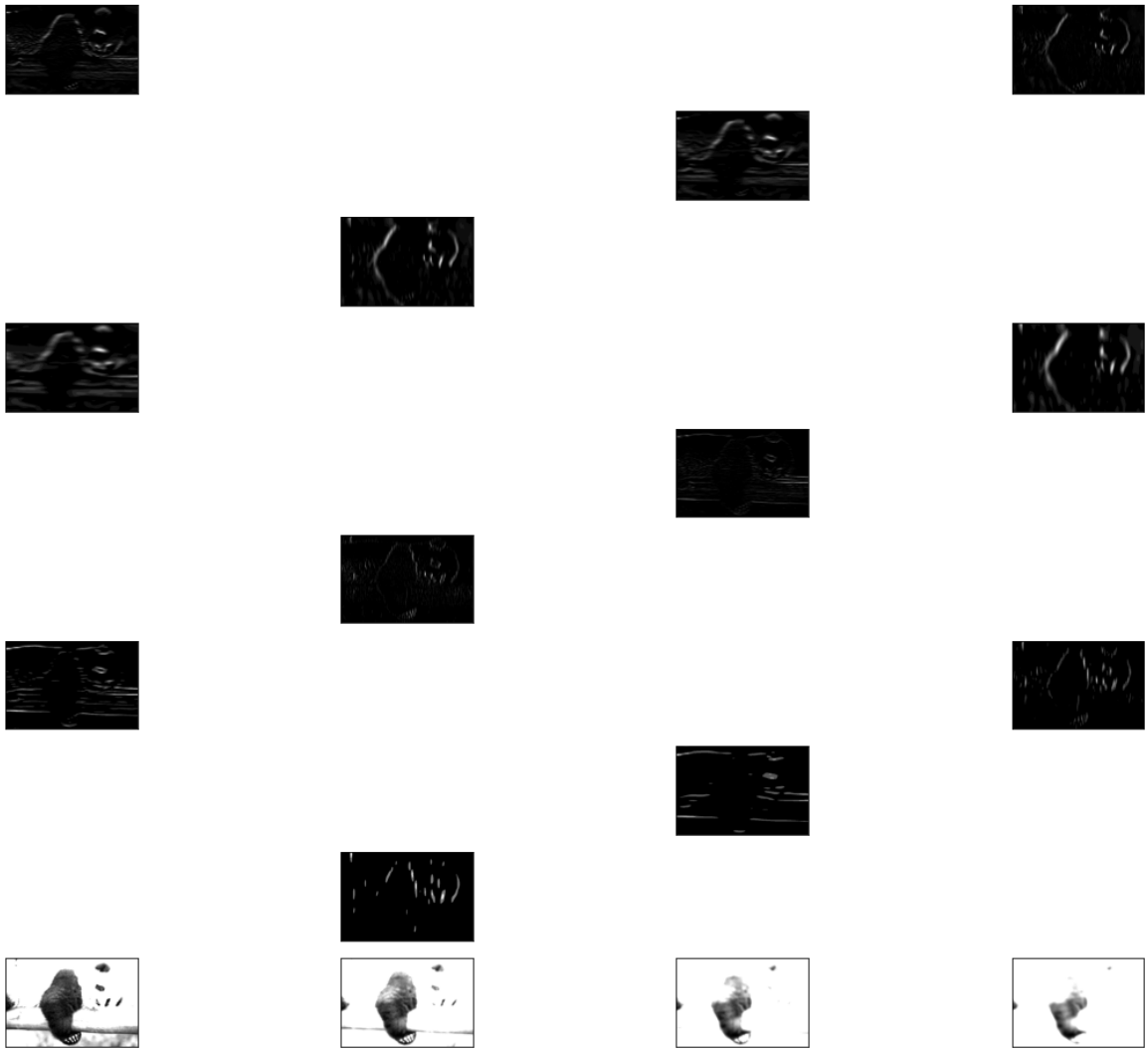

```
(1200, 1920, 48) (1200, 1920)
```




```
(788, 1400, 48) (788, 1400)
```



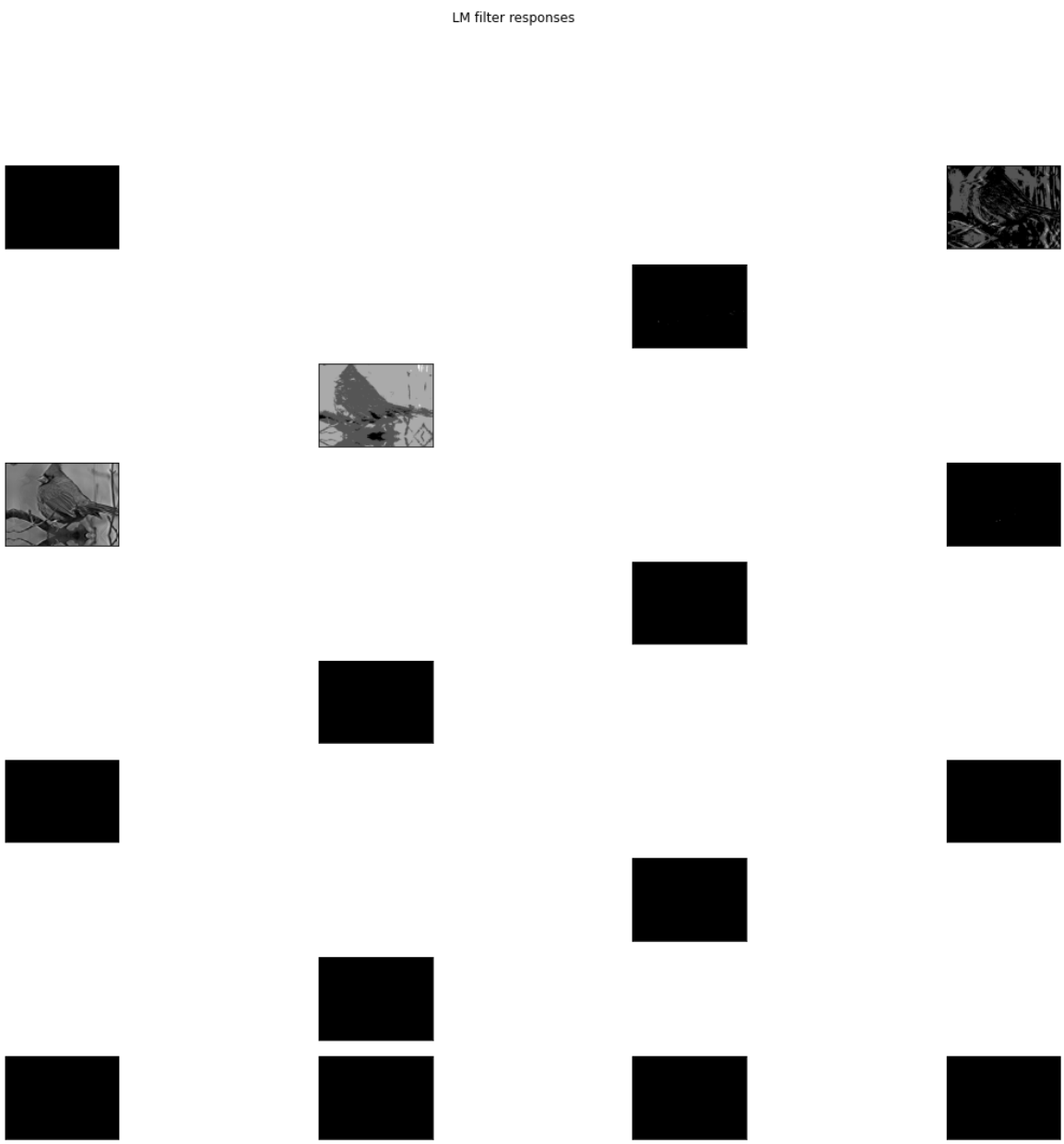

```
(367, 550, 48) (367, 550)
```



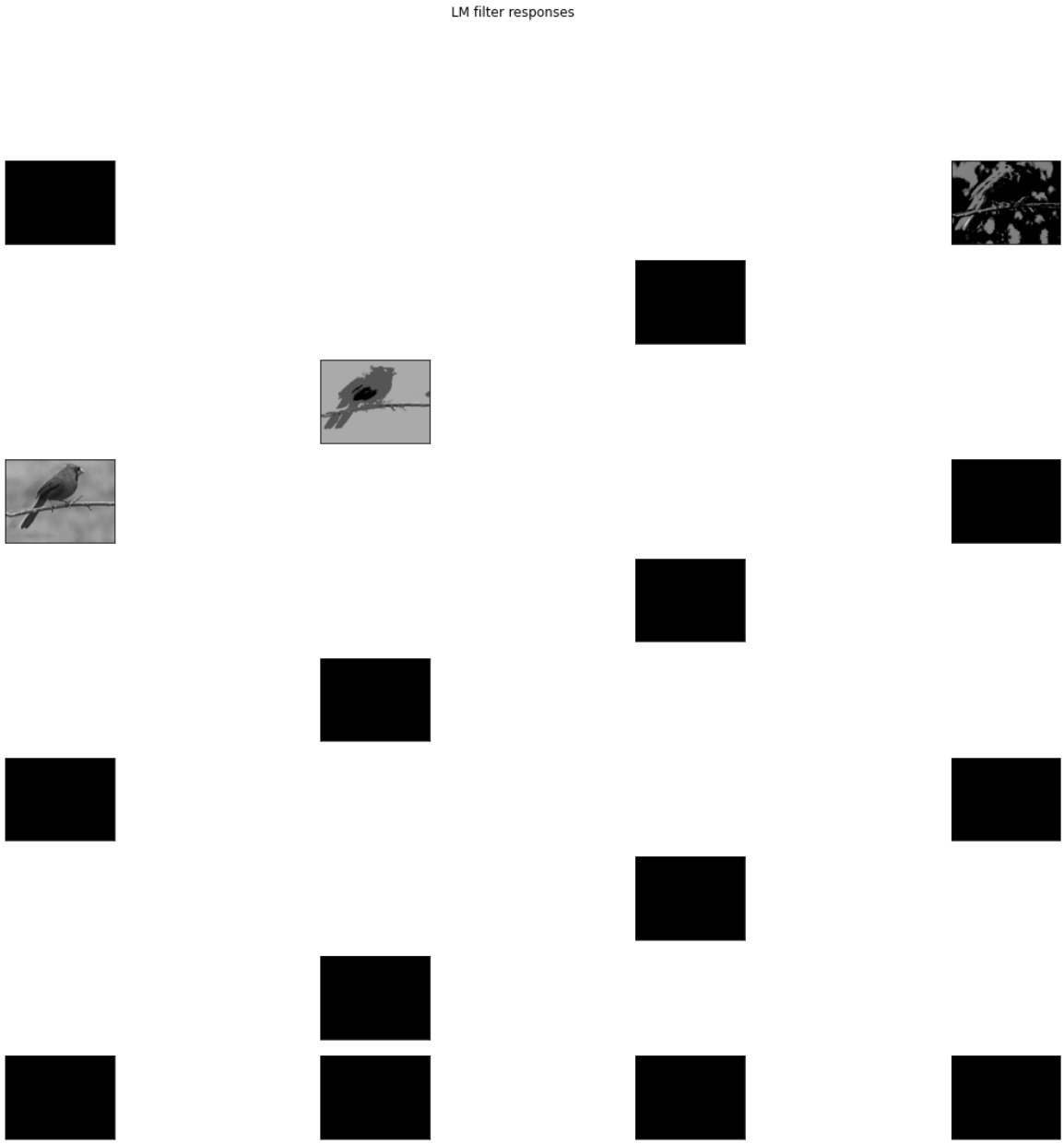
Separated separable filters

In [ ]:
```
# Loop through each image
for i in range (0, num_images):

    # Filter image and store
    FB_responses = makeResponses_sep(images[i], F, filter_vectors)
    print(FB_responses.shape, images[i].shape)

    plt.figure(figsize=(20,20))
    plt.suptitle('LM filter responses')
    nr = 12
    nc = 4

    #For each filter...
    for j in sep_list:

        #Plot filtered images
        plt.subplot(nr, nc, j+1)
        fig = plt.imshow(FB_responses[:,:,j], cmap='gray')
        fig.axes.get_xaxis().set_visible(False)
        fig.axes.get_yaxis().set_visible(False)

    plt.show()
```

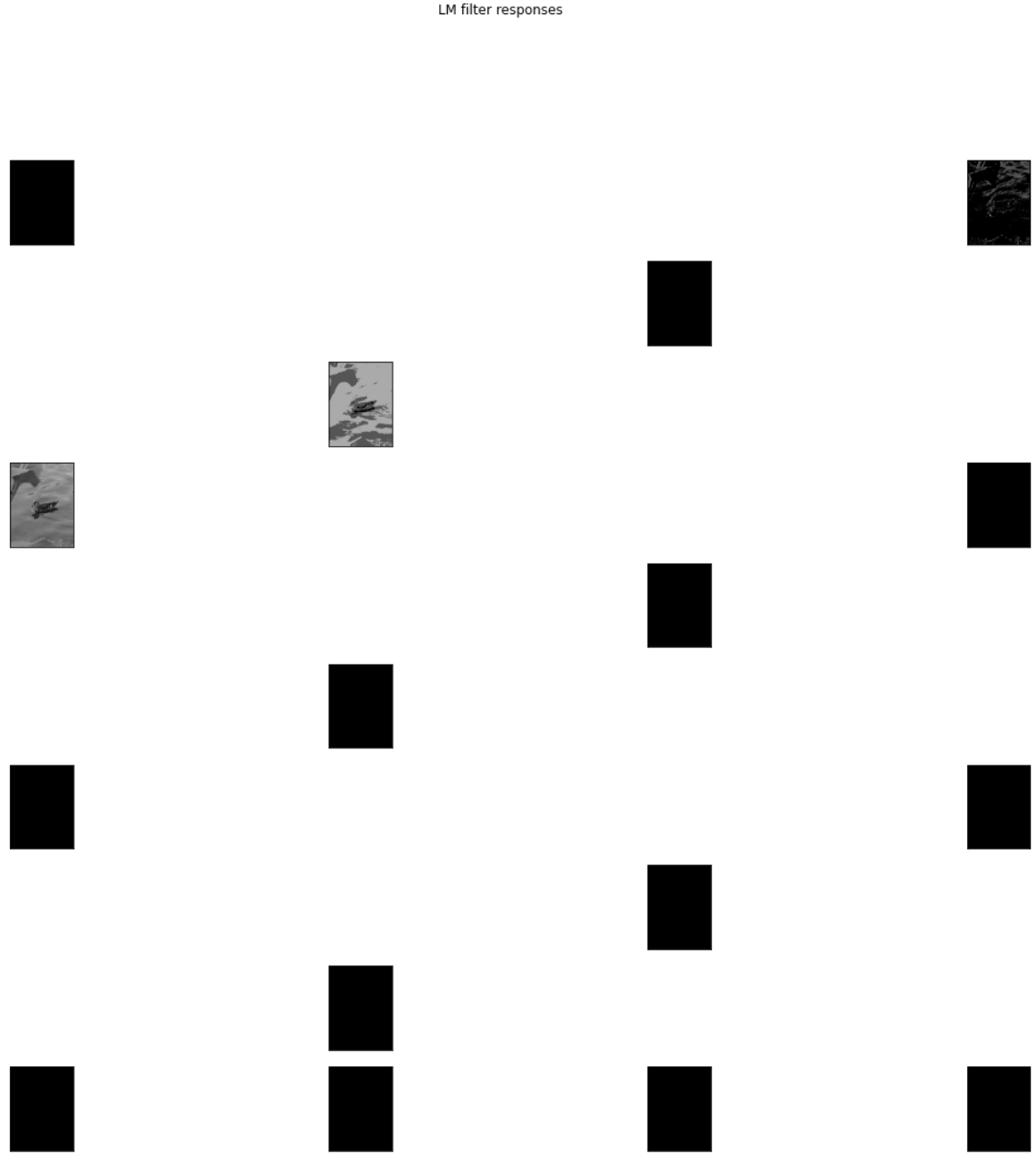
```
(200, 275, 48) (200, 275)
```



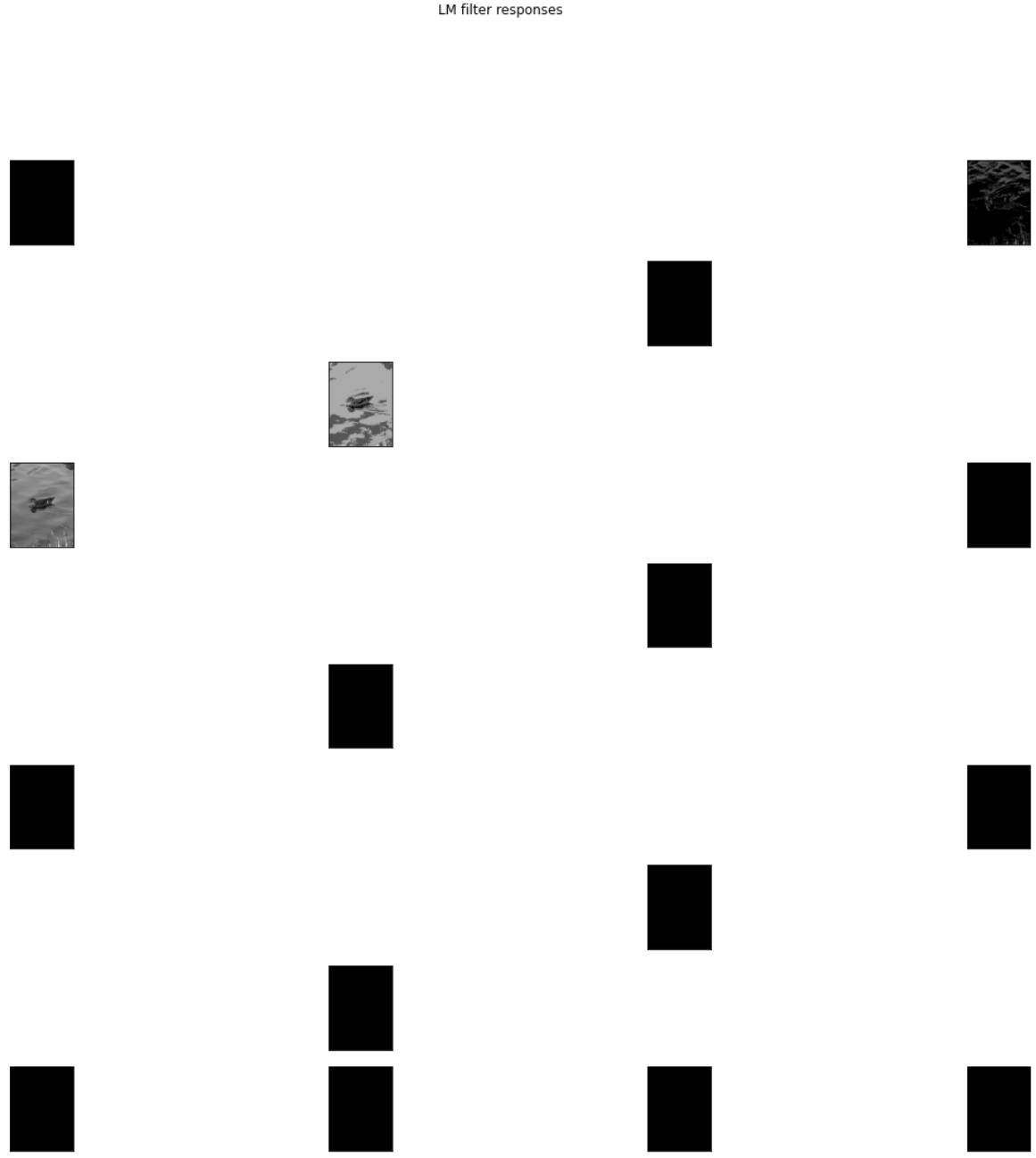

```
(302, 398, 48) (302, 398)
```



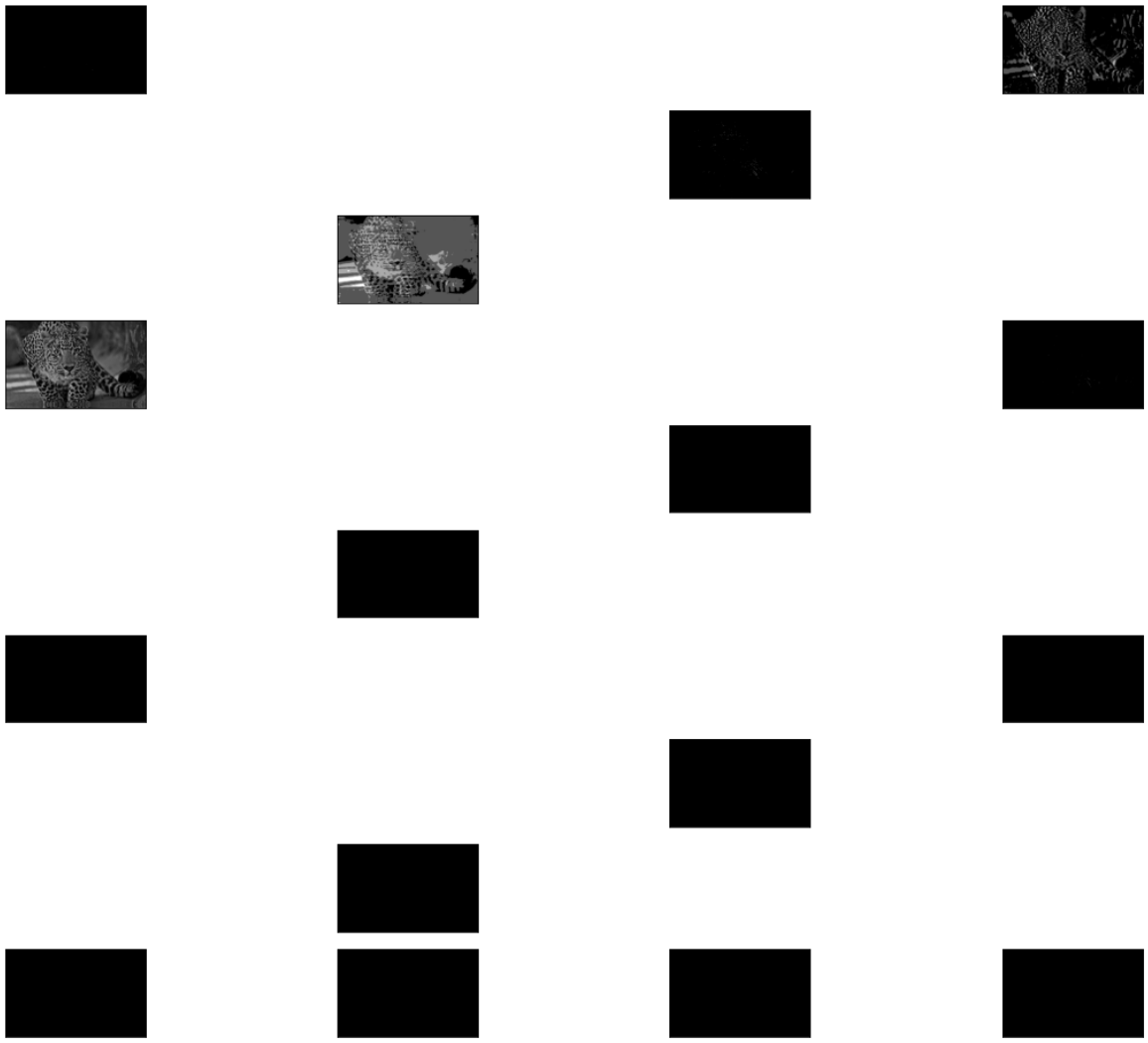

```
(2048, 1536, 48) (2048, 1536)
```



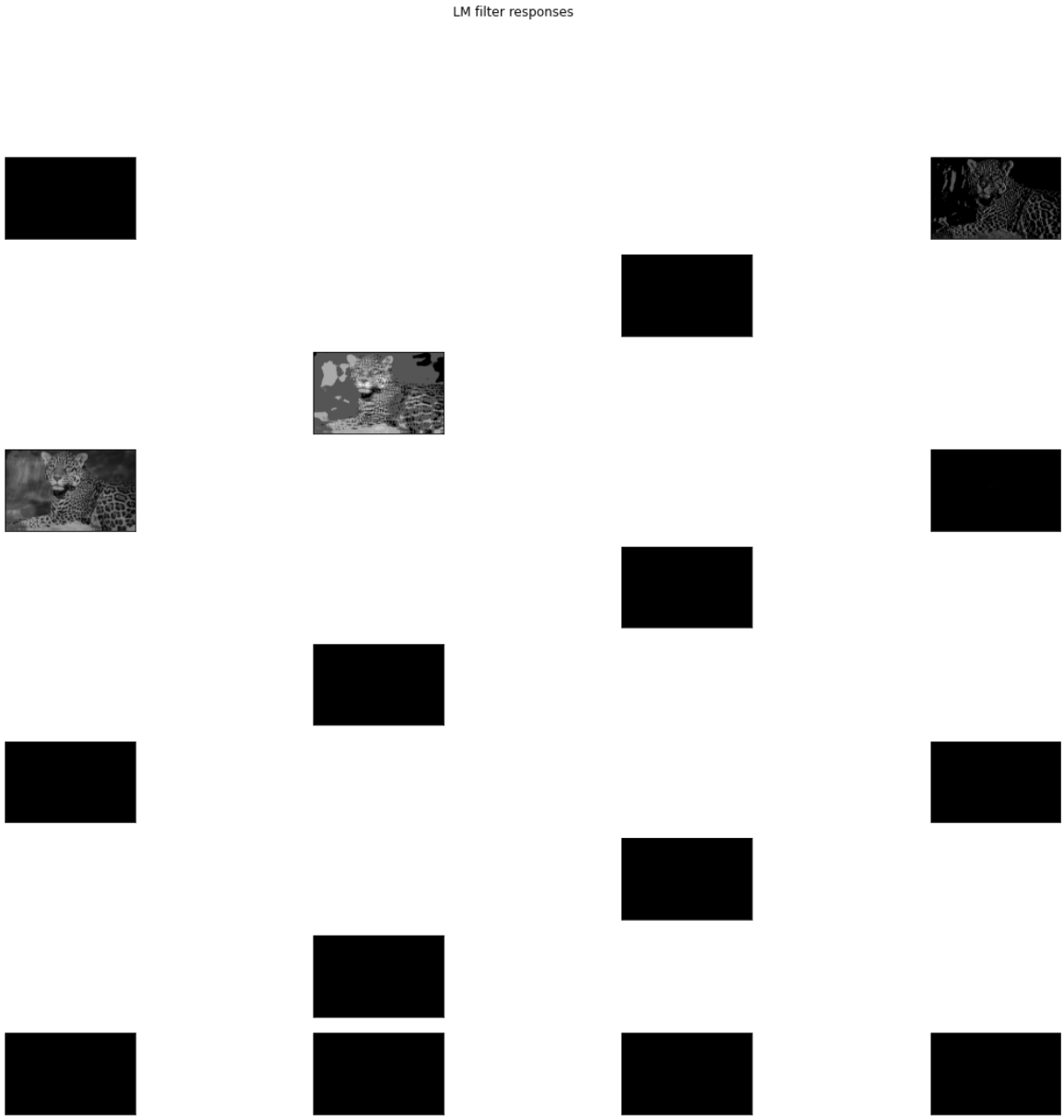

```
(2048, 1536, 48) (2048, 1536)
```



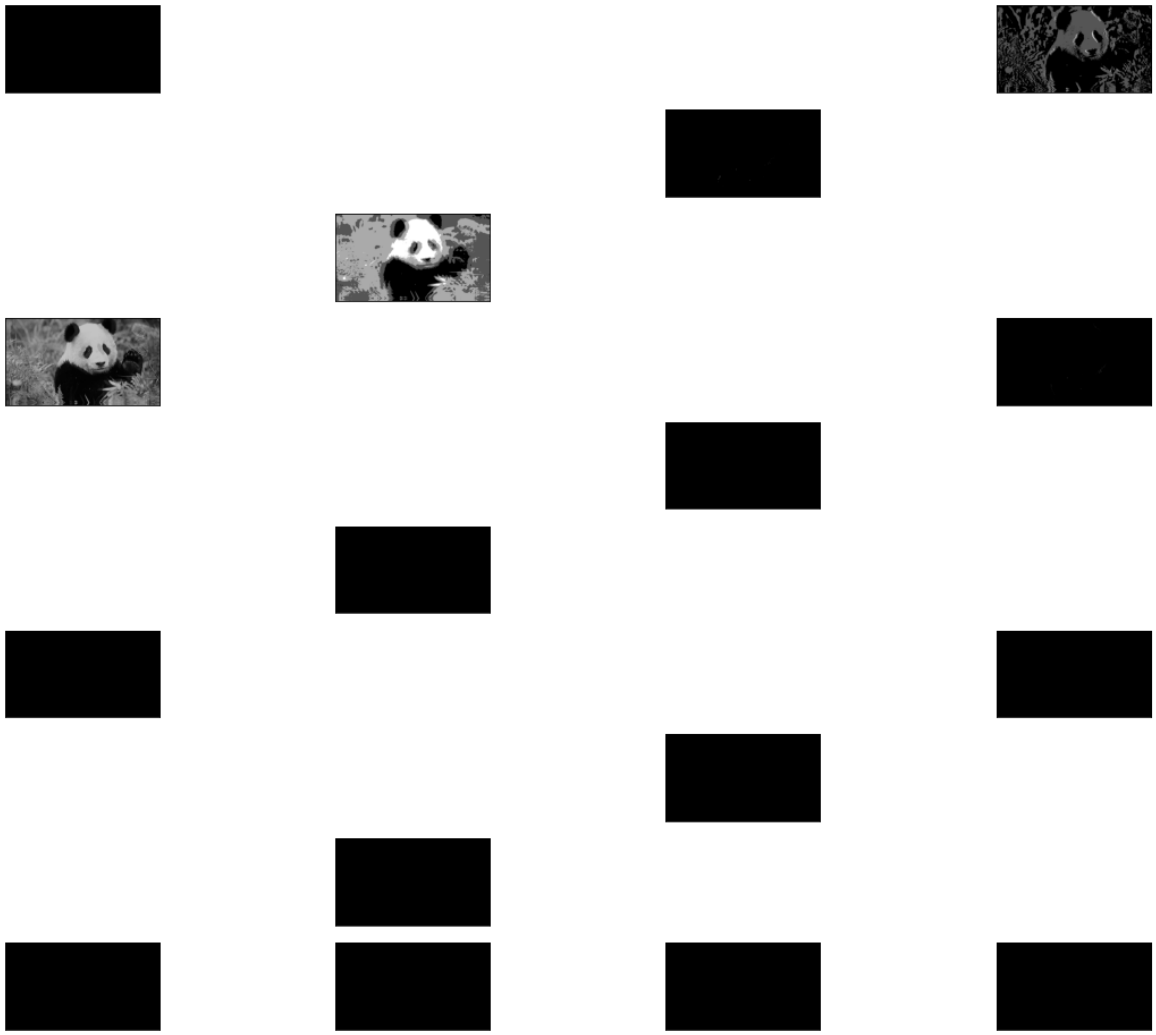
(585, 940, 48) (585, 940)




```
(1200, 1920, 48) (1200, 1920)
```




```
(788, 1400, 48) (788, 1400)
```




```
(367, 550, 48) (367, 550)
```

LM filter responses