Name: ____

EID: ____

CS5495 - Tutorial 5

CNNs - Feature Visualization

In this tutorial, you use feature visualization to examine the learned features in a CNN.

First we need to initialize Python. Run the below cell.

In [1]:
```
# setup
%matplotlib inline
import matplotlib_inline   # setup output image format
matplotlib_inline.backend_inline.set_matplotlib_formats('svg')
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100  # display larger images
import matplotlib
from numpy import *
from sklearn import *
from scipy import stats
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
pd.set_option('display.precision', 5)
import statsmodels.api as sm
import lime
import shap
from sklearn.model_selection import train_test_split
import os
from PIL import Image
```

In [2]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import Dataset, DataLoader

import torchvision

print(f"Using pytorch version: {torch.__version__}")

if (torch.cuda.is_available()):
    device = torch.device("cuda:0")
    print(f"Using: {torch.cuda.get_device_name(device)}")
elif (torch.backends.mps.is_available()):
    device = torch.device("mps")
    print(f"Using: Apple MPS")
else:
    raise("no GPU available")
```

```
Using pytorch version: 2.8.0+cu128
Using: Tesla T4
```

Helper functions

- These are helper functions from the lecture

In [3]:
```
def show_imgs(W_list, nc=10, highlight_green=None, highlight_red=None, titles=Nc
    nfilter = len(W_list)
    nr = (nfilter - 1) // nc + 1
    for i in range(nr):
        for j in range(nc):
            idx = i * nc + j
            if idx == nfilter:
                break
            plt.subplot(nr, nc, idx + 1)
            cur_W = W_list[idx]
            plt.imshow(cur_W,cmap='gray', interpolation='nearest')
            if titles is not None:
                if isinstance(titles, str):
                    plt.title(titles % idx)
                else:
                    plt.title(titles[idx])

            if ((highlight_green is not None) and highlight_green[idx]) or \
               ((highlight_red is not None) and highlight_red[idx]):
                ax = plt.gca()
                if highlight_green[idx]:
                    mycol = '#00FF00'
                else:
                    mycol = 'r'
                for S in ['bottom', 'top', 'right', 'left']:
                    ax.spines[S].set_color(mycol)
                    ax.spines[S].set_lw(2.0)
                ax.xaxis.set_ticks_position('none')
                ax.yaxis.set_ticks_position('none')
                ax.set_xticks([])
                ax.set_yticks([])
            else:
                plt.gca().set_axis_off()
```

Dog vs Cat Dataset

The task is to classify an image as having a cat or a dog. This dataset is from Kaggle.

In our dataset Class `0` is cat, and class `1` is dog.

First let's load the validation images. Make sure you have unzipped the validation and model files into your directory.

In [4]:
```
# the transform of the data for input into the network
test_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
```

```
    transforms.Normalize((0.5,), (0.5,))
])
```

In [5]:
```
# class to store the dataset
class CatDogDataset(Dataset):
    def __init__(self, image_paths, transform):
        super().__init__()
        self.paths = image_paths
        self.len = len(self.paths)
        self.transform = transform

    def __len__(self): return self.len

    def __getitem__(self, index):
        path = self.paths[index]
        image = Image.open(path).convert('RGB')
        image = self.transform(image)
        label = 0 if 'cat' in path else 1
        return (image, label)
```

In [6]:
```
# collect validation data file names
img_files = os.listdir('valid/')
len(img_files)
img_files = list(filter(lambda x: x != 'valid', img_files))
def valid_path(p): return f"valid/{p}"
img_files = list(map(valid_path, img_files))[::10] # subsample to make it faster
```

In [7]:
```
# load the dataset
valid_ds = CatDogDataset(img_files, test_transform)
valid_dl = DataLoader(valid_ds, batch_size=100)
classnames = ['cat', 'dog']
len(valid_ds), len(valid_dl)
```

Out[7]:
```
(500, 5)
```

Now let's view a few samples

In [8]:
```
eg_imgs  = []
eg_names = []
for X,Y in valid_dl:
    for i in range(15):
        # transform the image from [-1,1] to [0,1]
        eg_imgs.append( (1+transpose(X[i], (1,2,0)))/2 )
        eg_names.append( classnames[Y[i]] )
    break
plt.figure(figsize=(10,7))
show_imgs(eg_imgs, titles=eg_names, nc=5)
plt.show()
plt.close()
```

Deep CNNs

We have trained four Deep CNNs on the training set:

1. Model 1: 5 layers of convolutions, then global-average pooling and a linear classifier layer.
2. Model 2: 7 layers of convolutions, then global average pooling and a linear classifier layer.
3. Model 3: ResNet-18 (17 convolution layers), then global average pooling and a linear classifier layer. Trained from scratch.
4. Model 4: ResNet-18 (same as Model 3), but the network is initialized with pre-trained weights based on ImageNet.

Note that the global-average pooling takes the last convolution feature map (say C x H x W), and then averages over all the spatial nodes to obtain a (C x 1 x 1) feature vector. This is then used by the linear classifier layer. Thus, the weights in the linear classifier layer will indicate which of the C feature channels was useful for the classification task.

Now we will load each model. If your GPU has limited memory, probably you should only load one model at a time.

Model 1 (5 conv layers)

In [9]:
```
class CatAndDogNet4(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 3, out_channels = 16, kernel_size=(
        self.conv2 = nn.Conv2d(in_channels = 16, out_channels = 32, kernel_size=
        self.conv3 = nn.Conv2d(in_channels = 32, out_channels = 64, kernel_size=
        self.conv4 = nn.Conv2d(in_channels = 64, out_channels = 128, kernel_size
        self.conv5 = nn.Conv2d(in_channels = 128, out_channels = 128, kernel_siz
        self.gap   = nn.AdaptiveAvgPool2d((1, 1))

        self.fc1 = nn.Linear(in_features= 128, out_features=2)


    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv3(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv4(X))

        X = F.relu(self.conv5(X))

        X = self.gap(X)

        X = X.view(X.shape[0], -1)
        X = self.fc1(X)

        return X
```

In [10]:
```
# load model
model1 = CatAndDogNet4().to(device)
model1.load_state_dict(torch.load('models/model4-19.pth', map_location="cpu"))
model1.to(device).eval()
print(model1)
```

```
CatAndDogNet4(
  (conv1): Conv2d(3, 16, kernel_size=(5, 5), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(5, 5), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (gap): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc1): Linear(in_features=128, out_features=2, bias=True)
)
```

Model 2 (7 conv layers)

In [11]:
```
class CatAndDogNet6(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 3, out_channels = 16, kernel_size=(
        self.conv2 = nn.Conv2d(in_channels = 16, out_channels = 32, kernel_size=
        self.conv3 = nn.Conv2d(in_channels = 32, out_channels = 64, kernel_size=
        self.conv4 = nn.Conv2d(in_channels = 64, out_channels = 128, kernel_size
        self.conv5 = nn.Conv2d(in_channels = 128, out_channels = 128, kernel_siz
        self.conv6 = nn.Conv2d(in_channels = 128, out_channels = 128, kernel_siz
        self.conv7 = nn.Conv2d(in_channels = 128, out_channels = 128, kernel_siz
        self.gap   = nn.AdaptiveAvgPool2d((1, 1))
        self.fc1 = nn.Linear(in_features= 128, out_features=2)


    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv3(X))
        X = F.max_pool2d(X, 2)

        X = F.relu(self.conv4(X))

        X = F.relu(self.conv5(X))

        X = F.relu(self.conv6(X))

        X = F.relu(self.conv7(X))

        X = self.gap(X)

        X = X.view(X.shape[0], -1)
        X = self.fc1(X)
        return X
```

In [12]:
```
# Load model
model2 = CatAndDogNet6().to(device)
model2.load_state_dict(torch.load('models/model6-19.pth', map_location="cpu"))
model2.to(device).eval()
print(model2)
```

```
CatAndDogNet6(
  (conv1): Conv2d(3, 16, kernel_size=(5, 5), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(5, 5), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv6): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (gap): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc1): Linear(in_features=128, out_features=2, bias=True)
)
```

Model 3 (ResNet-18, from scratch)

In [13]:
```
model3 = models.resnet18().to(device)
in_feats = model3.fc.in_features
model3.fc = nn.Linear(in_feats, 2)
model3.load_state_dict(torch.load('models/model8-17.pth', map_location="cpu"))
model3.to(device).eval()
```

Out[13]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bia
s=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mod
e=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
```

```
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
```

```
  (fc): Linear(in_features=512, out_features=2, bias=True)
)
```

Model 4 (ResNet-18, pretrained on ImageNet)

In [14]:
```
model4 = models.resnet18().to(device)
in_feats = model4.fc.in_features
model4.fc = nn.Linear(in_feats, 2)
model4.load_state_dict(torch.load('models/model7-18.pth', map_location="cpu"))
model4.to(device).eval()
```

Out[14]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bia
s=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mod
e=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
```

```
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
```

```
  (fc): Linear(in_features=512, out_features=2, bias=True)
)
```

Evaluation

Now we will evaluate the models. First consolidate them.

In [15]:
```
all_models = {
    'CNN5': model1,
    'CNN7': model2,
    'ResNet18': model3,
    'ResNet18p': model4
}
modelnames = all_models.keys()
```

Evaluate each model on the validation data.

In [16]:
```
# Evaluation
correct = {}
total = {}
for myname,mymodel in all_models.items():
    mymodel.to(device).eval()
    correct[myname] = 0
    total[myname] = 0

with torch.no_grad():
    # for each validation batch
    for data, targets in valid_dl:
        print('.', end='', flush=True)
        data, targets = data.to(device), targets.to(device)

        # test each model
        for myname,mymodel in all_models.items():
            mymodel.to(device).eval()
            correct[myname] = 0
            total[myname] = 0

            outputs = mymodel(data)
            _, predicted = torch.max(outputs.data, 1)
            total[myname] += targets.size(0)
            correct[myname] += (predicted == targets).sum().item()

print("")
for myname in all_models.keys():
    print(myname + f' Accuracy on valid set: {100 * correct[myname] / total[myna
```

```
.....
CNN5 Accuracy on valid set: 84.00%
CNN7 Accuracy on valid set: 77.00%
ResNet18 Accuracy on valid set: 90.00%
ResNet18p Accuracy on valid set: 97.00%
```

Feature Visualization

Now, examine the features of each model using the feature visualization method.

You can use the `lucent` toolbox. If it is not installed on your system, use the following command: `pip install torch-lucent` On the CS JupyterLab, you can run this by opening a terminal tab. In Jupyter notebook, you can run the following "magic" command `!pip install torch-lucent`

In [17]:
```
!pip install torch-lucent
```

```
Collecting torch-lucent
  Using cached torch_lucent-0.1.8-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: torch>=1.5.0 in /opt/conda/lib/python3.12/site-pac
kages (from torch-lucent) (2.8.0)
Requirement already satisfied: torchvision in /opt/conda/lib/python3.12/site-pack
ages (from torch-lucent) (0.23.0)
Collecting kornia<=0.4.1 (from torch-lucent)
  Using cached kornia-0.4.1-py2.py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.12/site-packages (f
rom torch-lucent) (4.67.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages
(from torch-lucent) (1.26.4)
Requirement already satisfied: ipython in /opt/conda/lib/python3.12/site-packages
(from torch-lucent) (9.4.0)
Requirement already satisfied: pillow in /opt/conda/lib/python3.12/site-packages
(from torch-lucent) (11.3.0)
Collecting future (from torch-lucent)
  Using cached future-1.0.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: decorator in /opt/conda/lib/python3.12/site-packag
es (from torch-lucent) (5.2.1)
Collecting pytest (from torch-lucent)
  Using cached pytest-8.4.2-py3-none-any.whl.metadata (7.7 kB)
Collecting pytest-mock (from torch-lucent)
  Using cached pytest_mock-3.15.1-py3-none-any.whl.metadata (3.9 kB)
Collecting coverage (from torch-lucent)
  Using cached coverage-7.11.0-cp312-cp312-manylinux1_x86_64.manylinux_2_28_x86_6
4.manylinux_2_5_x86_64.whl.metadata (9.0 kB)
Collecting coveralls (from torch-lucent)
  Using cached coveralls-4.0.1-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.12/site-pac
kages (from torch-lucent) (1.7.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.12/site-package
s (from torch>=1.5.0->torch-lucent) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /opt/conda/lib/python
3.12/site-packages (from torch>=1.5.0->torch-lucent) (4.14.1)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.12/site-packa
ges (from torch>=1.5.0->torch-lucent) (80.9.0)
Requirement already satisfied: sympy>=1.13.3 in /opt/conda/lib/python3.12/site-pa
ckages (from torch>=1.5.0->torch-lucent) (1.14.0)
Requirement already satisfied: networkx in /opt/conda/lib/python3.12/site-package
s (from torch>=1.5.0->torch-lucent) (3.5)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.12/site-packages
(from torch>=1.5.0->torch-lucent) (3.1.6)
Requirement already satisfied: fsspec in /opt/conda/lib/python3.12/site-packages
(from torch>=1.5.0->torch-lucent) (2025.7.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.8.93 in /opt/conda/lib/
python3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.93)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.8.90 in /opt/conda/li
b/python3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.90)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.8.90 in /opt/conda/lib/
python3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.90)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /opt/conda/lib/pyt
hon3.12/site-packages (from torch>=1.5.0->torch-lucent) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.8.4.1 in /opt/conda/lib/pyt
hon3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.3.83 in /opt/conda/lib/pyt
hon3.12/site-packages (from torch>=1.5.0->torch-lucent) (11.3.3.83)
Requirement already satisfied: nvidia-curand-cu12==10.3.9.90 in /opt/conda/lib/py
thon3.12/site-packages (from torch>=1.5.0->torch-lucent) (10.3.9.90)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.3.90 in /opt/conda/lib/
```

```
python3.12/site-packages (from torch>=1.5.0->torch-lucent) (11.7.3.90)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.8.93 in /opt/conda/lib/
python3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.5.8.93)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /opt/conda/lib/py
thon3.12/site-packages (from torch>=1.5.0->torch-lucent) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /opt/conda/lib/python
3.12/site-packages (from torch>=1.5.0->torch-lucent) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.8.90 in /opt/conda/lib/python
3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.90)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.8.93 in /opt/conda/lib/p
ython3.12/site-packages (from torch>=1.5.0->torch-lucent) (12.8.93)
Requirement already satisfied: nvidia-cufile-cu12==1.13.1.3 in /opt/conda/lib/pyt
hon3.12/site-packages (from torch>=1.5.0->torch-lucent) (1.13.1.3)
Requirement already satisfied: triton==3.4.0 in /opt/conda/lib/python3.12/site-pa
ckages (from torch>=1.5.0->torch-lucent) (3.4.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /opt/conda/lib/python3.12/si
te-packages (from sympy>=1.13.3->torch>=1.5.0->torch-lucent) (1.3.0)
Collecting docopt<0.7.0,>=0.6.1 (from coveralls->torch-lucent)
  Using cached docopt-0.6.2-py2.py3-none-any.whl
Requirement already satisfied: requests<3.0.0,>=1.0.0 in /opt/conda/lib/python3.1
2/site-packages (from coveralls->torch-lucent) (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/python
3.12/site-packages (from requests<3.0.0,>=1.0.0->coveralls->torch-lucent) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-pac
kages (from requests<3.0.0,>=1.0.0->coveralls->torch-lucent) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.12/si
te-packages (from requests<3.0.0,>=1.0.0->coveralls->torch-lucent) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.12/si
te-packages (from requests<3.0.0,>=1.0.0->coveralls->torch-lucent) (2025.8.3)
Requirement already satisfied: ipython-pygments-lexers in /opt/conda/lib/python3.
12/site-packages (from ipython->torch-lucent) (1.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/conda/lib/python3.12/site-packa
ges (from ipython->torch-lucent) (0.19.2)
Requirement already satisfied: matplotlib-inline in /opt/conda/lib/python3.12/sit
e-packages (from ipython->torch-lucent) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.12/site-pack
ages (from ipython->torch-lucent) (4.9.0)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in /opt/conda/lib/py
thon3.12/site-packages (from ipython->torch-lucent) (3.0.51)
Requirement already satisfied: pygments>=2.4.0 in /opt/conda/lib/python3.12/site-
packages (from ipython->torch-lucent) (2.19.2)
Requirement already satisfied: stack_data in /opt/conda/lib/python3.12/site-packa
ges (from ipython->torch-lucent) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in /opt/conda/lib/python3.12/sit
e-packages (from ipython->torch-lucent) (5.14.3)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.12/site-packages
(from prompt_toolkit<3.1.0,>=3.0.41->ipython->torch-lucent) (0.2.13)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /opt/conda/lib/python3.12/s
ite-packages (from jedi>=0.16->ipython->torch-lucent) (0.8.4)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.12/site-
packages (from pexpect>4.3->ipython->torch-lucent) (0.7.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.12/site-
packages (from jinja2->torch>=1.5.0->torch-lucent) (3.0.2)
Collecting iniconfig>=1 (from pytest->torch-lucent)
  Using cached iniconfig-2.3.0-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: packaging>=20 in /opt/conda/lib/python3.12/site-pa
ckages (from pytest->torch-lucent) (25.0)
Requirement already satisfied: pluggy<2,>=1.5 in /opt/conda/lib/python3.12/site-p
ackages (from pytest->torch-lucent) (1.6.0)
Requirement already satisfied: scipy>=1.8.0 in /opt/conda/lib/python3.12/site-pac
```

```
kages (from scikit-learn->torch-lucent) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in /opt/conda/lib/python3.12/site-pa
ckages (from scikit-learn->torch-lucent) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /opt/conda/lib/python3.12/
site-packages (from scikit-learn->torch-lucent) (3.6.0)
Requirement already satisfied: executing>=1.2.0 in /opt/conda/lib/python3.12/site
-packages (from stack_data->ipython->torch-lucent) (2.2.0)
Requirement already satisfied: asttokens>=2.1.0 in /opt/conda/lib/python3.12/site
-packages (from stack_data->ipython->torch-lucent) (3.0.0)
Requirement already satisfied: pure_eval in /opt/conda/lib/python3.12/site-packag
es (from stack_data->ipython->torch-lucent) (0.2.3)
Using cached torch_lucent-0.1.8-py3-none-any.whl (46 kB)
Using cached kornia-0.4.1-py2.py3-none-any.whl (225 kB)
Using cached coverage-7.11.0-cp312-cp312-manylinux1_x86_64.manylinux_2_28_x86_64.
manylinux_2_5_x86_64.whl (250 kB)
Using cached coveralls-4.0.1-py3-none-any.whl (13 kB)
Using cached future-1.0.0-py3-none-any.whl (491 kB)
Using cached pytest-8.4.2-py3-none-any.whl (365 kB)
Using cached iniconfig-2.3.0-py3-none-any.whl (7.5 kB)
Using cached pytest_mock-3.15.1-py3-none-any.whl (10 kB)
Installing collected packages: docopt, iniconfig, future, coverage, pytest, pytes
t-mock, coveralls, kornia, torch-lucent
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9/9 [torch-lucent][0m [kornia]e]
Successfully installed coverage-7.11.0 coveralls-4.0.1 docopt-0.6.2 future-1.0.0
iniconfig-2.3.0 kornia-0.4.1 pytest-8.4.2 pytest-mock-3.15.1 torch-lucent-0.1.8
```

Features in the Last Conv Layer

Perform feature visualization on the last convolutional layer of the networks. Since there are a lot of features, one suggestion is to focus on those features that were more useful for the classifier (e.g., looking at the features with largest effects).

In [24]:

```python
from lucent.optvis import render, param, transform, objectives
from lucent.modelzoo import util
import numpy as np

# Helper function to get last conv layer name
def get_last_conv_layer(model, model_name):
    """Get the name of the last convolutional layer"""
    if model_name == 'CNN5':
        return 'conv5'
    elif model_name == 'CNN7':
        return 'conv7'
    else:  # ResNet models
        return 'layer4'

# Helper function to get classifier weights
def get_classifier_weights(model, model_name):
    """Get the weights from the final linear classifier"""
    if model_name in ['CNN5', 'CNN7']:
        fc_weights = model.fc1.weight.data.cpu().numpy()
    else:
        fc_weights = model.fc.weight.data.cpu().numpy()
    return fc_weights

# Analyze classifier weights to identify important features
print("=" * 80)
print("Analyzing Classifier Weights - Finding Most Important Features")
```

```
print("=" * 80)

important_features = {}

for model_name, model in all_models.items():
    print(f"\n{model_name}:")
    fc_weights = get_classifier_weights(model, model_name)

    # Get weights for cat (class 0) and dog (class 1)
    cat_weights = fc_weights[0]
    dog_weights = fc_weights[1]

    # Find top features - those with highest absolute weights
    n_top = 10

    # Top features for cat
    top_cat_indices = np.argsort(np.abs(cat_weights))[-n_top:][::-1]
    top_cat_weights = cat_weights[top_cat_indices]

    # Top features for dog
    top_dog_indices = np.argsort(np.abs(dog_weights))[-n_top:][::-1]
    top_dog_weights = dog_weights[top_dog_indices]

    print(f"  Top features for Cat class:")
    print(f"    Channels: {top_cat_indices}")
    print(f"    Weights: {top_cat_weights}")

    print(f"  Top features for Dog class:")
    print(f"    Channels: {top_dog_indices}")
    print(f"    Weights: {top_dog_weights}")

    # Store for visualization - focus on most discriminative features
    important_features[model_name] = {
        'cat': top_cat_indices[:6],  # Top 6 for cat
        'dog': top_dog_indices[:6],  # Top 6 for dog
        'all': np.unique(np.concatenate([top_cat_indices[:8], top_dog_indices[:8
    }
```

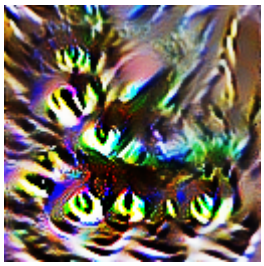
```
================================================================================
Analyzing Classifier Weights - Finding Most Important Features
================================================================================

CNN5:
  Top features for Cat class:
    Channels: [ 87  19  28  49  78 118 105  92  16  44]
    Weights: [ 0.3983413   0.36411086  0.2820632   0.27593052  0.23978925  0.2270
8634
 -0.20883247  0.19717304  0.19214769 -0.19162545]
  Top features for Dog class:
    Channels: [ 87  49  19  63  28  88 111 105  64  69]
    Weights: [-0.38611862 -0.3472368  -0.2967816   0.25545737 -0.23990048 -0.2395
4459
  0.19534677  0.19504885  0.18090725  0.17981972]

CNN7:
  Top features for Cat class:
    Channels: [ 91   3 125  72  39 126 119  70 101   1]
    Weights: [-0.13209501 -0.13097805 -0.12836324 -0.1273021   0.11454222 -0.1109
6954
 -0.10706655  0.10647504  0.10397854  0.10351369]
  Top features for Dog class:
    Channels: [ 72  81 119  75   4  40  93   3  27  96]
    Weights: [ 0.1343886   0.13214974  0.12690727  0.12207021  0.12016417 -0.1168
9077
 -0.11245074  0.10822596 -0.10334916  0.10249013]

ResNet18:
  Top features for Cat class:
    Channels: [461  51 465  87 119 313 423 135 312 242]
    Weights: [-0.06216917 -0.06017644 -0.05757708 -0.05669077 -0.05657277 -0.0529
0999
 -0.05216831 -0.05138262 -0.05082817 -0.04941464]
  Top features for Dog class:
    Channels: [ 63 236 384 498 119 324  78  14 443 405]
    Weights: [ 0.0615071   0.05937769  0.05772403  0.05610785  0.05579644 -0.0521
7705
  0.05184646  0.05108611  0.05040912  0.05027458]

ResNet18p:
  Top features for Cat class:
    Channels: [469 146 130 327 392 188 400 157  68  73]
    Weights: [ 0.09661835  0.07972775  0.07833534  0.07822134  0.07589381  0.0747
9799
 -0.07440767  0.07423706  0.07206173 -0.07144132]
  Top features for Dog class:
    Channels: [469 406 251 146 377 419 127 157 154 205]
    Weights: [-0.09089658 -0.08362191 -0.08045768 -0.08000176 -0.07875536 -0.0779
433
  0.07665082 -0.07572783 -0.07517231 -0.07498889]
```

In [25]:
```python
# Visualize features for Model 1 (CNN5)
print("\n" + "=" * 80)
print("Feature Visualization: Model 1 (CNN5 - 5 Conv Layers)")
print("=" * 80)

model_name = 'CNN5'
layer_name = get_last_conv_layer(model1, model_name)
features_to_vis = important_features[model_name]['all']
```

```
print(f"\nVisualizing top {len(features_to_vis)} most important features from {l
print(f"Feature channels: {features_to_vis}")
print("\nNote: CNN5 features are relatively simple - basic textures and patterns

for feature_idx in features_to_vis:
    print(f"Channel {feature_idx}:")
    _ = render.render_vis(model1, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

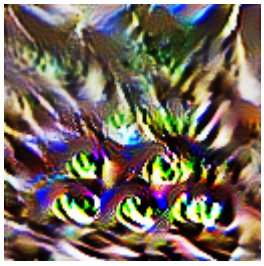
```
================================================================================
Feature Visualization: Model 1 (CNN5 - 5 Conv Layers)
================================================================================

Visualizing top 11 most important features from conv5
Feature channels: [ 19  28  49  63  78  87  88  92 105 111 118]

Note: CNN5 features are relatively simple - basic textures and patterns

Channel 19:
100%|██████████| 512/512 [00:04<00:00, 116.14it/s]
```

0



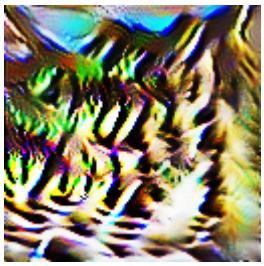
```
Channel 28:
100%|██████████| 512/512 [00:04<00:00, 118.11it/s]
```

0



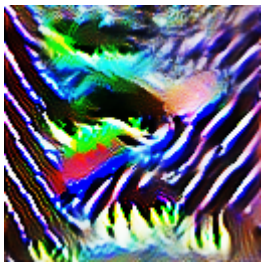
```
Channel 49:
100%|██████████| 512/512 [00:04<00:00, 112.94it/s]
```

0



```
Channel 63:
100%|██████████| 512/512 [00:04<00:00, 113.56it/s]
```

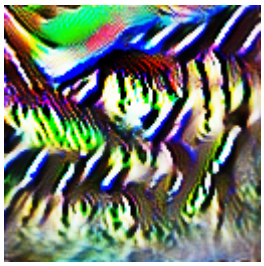
0



Channel 78:

100%|██████████| 512/512 [00:04<00:00, 115.33it/s]

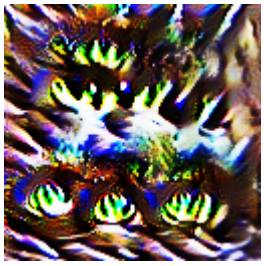
0



Channel 87:

100%|██████████| 512/512 [00:04<00:00, 120.34it/s]

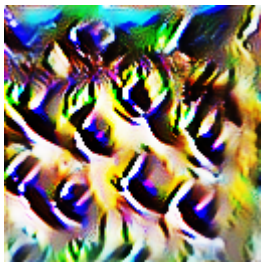
0



Channel 88:

100%|██████████| 512/512 [00:04<00:00, 113.48it/s]

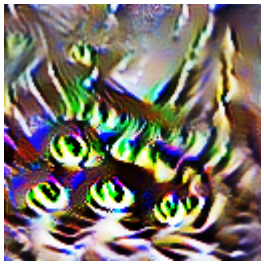
0



Channel 92:

100%|██████████| 512/512 [00:04<00:00, 110.86it/s]

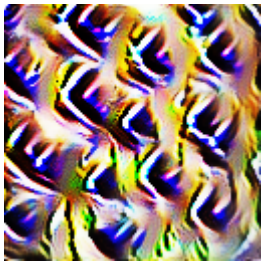
0



Channel 105:

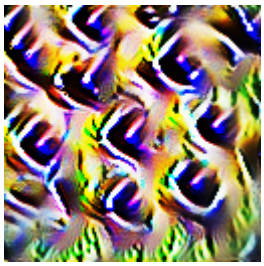
```
100%|██████████| 512/512 [00:04<00:00, 110.02it/s]
```

0



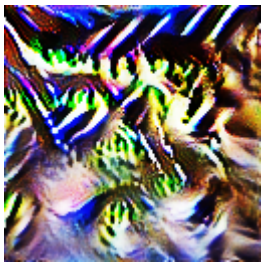
```
Channel 111:
100%|██████████| 512/512 [00:04<00:00, 117.87it/s]
```

0



```
Channel 118:
100%|██████████| 512/512 [00:04<00:00, 117.45it/s]
```

0



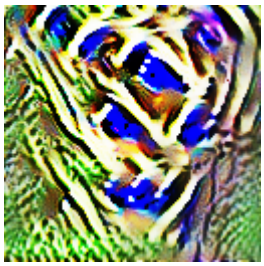
In [26]:
```python
# Visualize features for Model 2 (CNN7)
print("\n" + "=" * 80)
print("Feature Visualization: Model 2 (CNN7 - 7 Conv Layers)")
print("=" * 80)

model_name = 'CNN7'
layer_name = get_last_conv_layer(model2, model_name)
features_to_vis = important_features[model_name]['all']

print(f"\nVisualizing top {len(features_to_vis)} most important features from {l
print(f"Feature channels: {features_to_vis}")
print("\nNote: CNN7 features show slightly more complexity with combined texture

for feature_idx in features_to_vis:
    print(f"Channel {feature_idx}:")
    _ = render.render_vis(model2, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

```
================================================================================
Feature Visualization: Model 2 (CNN7 - 7 Conv Layers)
================================================================================

Visualizing top 13 most important features from conv7
Feature channels: [  3   4  39  40  70  72  75  81  91  93 119 125 126]

Note: CNN7 features show slightly more complexity with combined textures

Channel 3:
100%|██████████| 512/512 [00:04<00:00, 111.48it/s]
```

0

```
Channel 4:
100%|██████████| 512/512 [00:04<00:00, 109.47it/s]
```

0



```
Channel 39:
100%|██████████| 512/512 [00:04<00:00, 110.42it/s]
```

0

```
Channel 40:
100%|██████████| 512/512 [00:04<00:00, 106.52it/s]
```

0

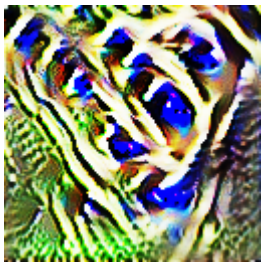
```
Channel 70:
100%|██████████| 512/512 [00:04<00:00, 105.21it/s]
```

0

Channel 72:

100%|██████████| 512/512 [00:04<00:00, 107.67it/s]

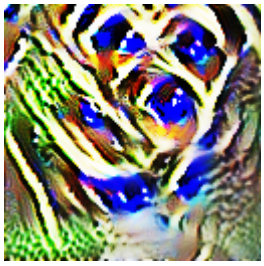
0



Channel 75:

100%|██████████| 512/512 [00:04<00:00, 108.12it/s]

0



Channel 81:

100%|██████████| 512/512 [00:05<00:00, 102.19it/s]

0

Channel 91:

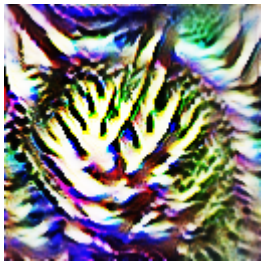
100%|██████████| 512/512 [00:05<00:00, 95.59it/s]

0

Channel 93:

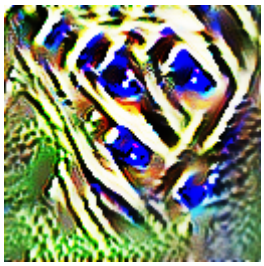
```
100%|██████████| 512/512 [00:04<00:00, 107.97it/s]
```

0



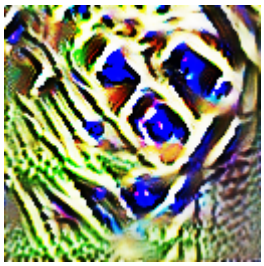
```
Channel 119:
100%|██████████| 512/512 [00:04<00:00, 108.81it/s]
```

0



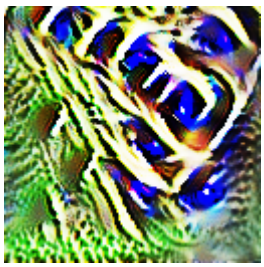
```
Channel 125:
100%|██████████| 512/512 [00:04<00:00, 110.96it/s]
```

0



```
Channel 126:
100%|██████████| 512/512 [00:04<00:00, 111.35it/s]
```

0

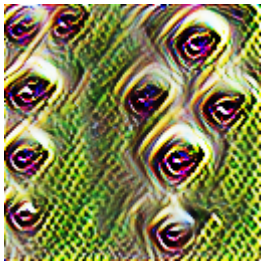


In [27]:
```python
# Visualize features for Model 3 (ResNet18 from scratch)
print("\n" + "=" * 80)
print("Feature Visualization: Model 3 (ResNet18 - Trained from Scratch)")
print("=" * 80)

model_name = 'ResNet18'
layer_name = get_last_conv_layer(model3, model_name)
features_to_vis = important_features[model_name]['all']

print(f"\nVisualizing top {len(features_to_vis)} most important features from {l
print(f"Feature channels: {features_to_vis}")
```

```
print("\nNote: ResNet18 (scratch) shows moderate complexity but less semantic cl

for feature_idx in features_to_vis:
    print(f"Channel {feature_idx}:")
    _ = render.render_vis(model3, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

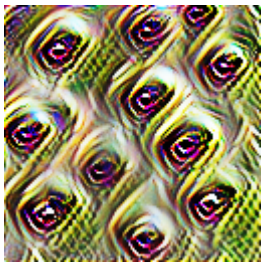
```
================================================================================
Feature Visualization: Model 3 (ResNet18 - Trained from Scratch)
================================================================================

Visualizing top 15 most important features from layer4
Feature channels: [ 14  51  63  78  87 119 135 236 313 324 384 423 461 465 498]

Note: ResNet18 (scratch) shows moderate complexity but less semantic clarity

Channel 14:
100%|██████████| 512/512 [00:08<00:00, 62.50it/s]
```

0



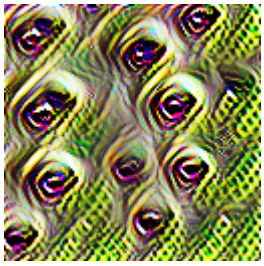
```
Channel 51:
100%|██████████| 512/512 [00:07<00:00, 66.75it/s]
```

0



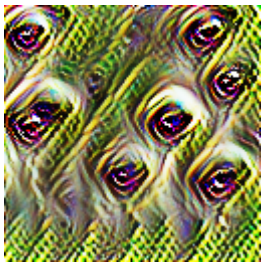
```
Channel 63:
100%|██████████| 512/512 [00:07<00:00, 64.97it/s]
```

0



```
Channel 78:
100%|██████████| 512/512 [00:07<00:00, 64.18it/s]
```

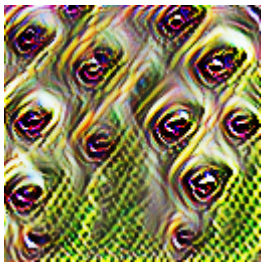
0



Channel 87:

100%|██████████| 512/512 [00:07<00:00, 64.07it/s]

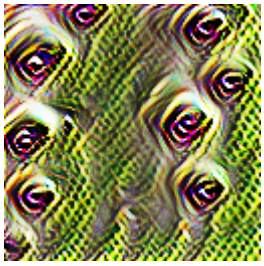
0



Channel 119:

100%|██████████| 512/512 [00:07<00:00, 64.17it/s]

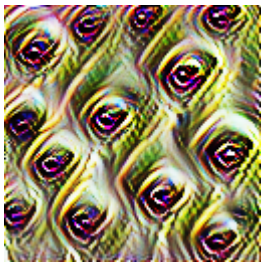
0



Channel 135:

100%|██████████| 512/512 [00:07<00:00, 64.17it/s]

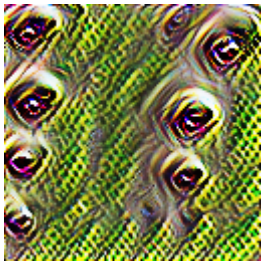
0



Channel 236:

100%|██████████| 512/512 [00:08<00:00, 62.83it/s]

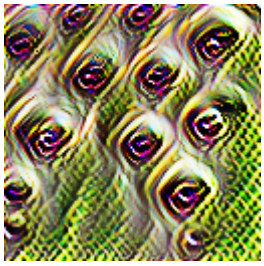
0



Channel 313:

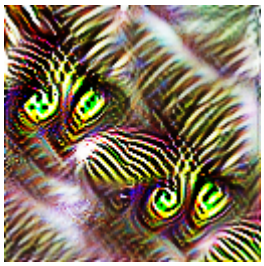
```
100%|██████████| 512/512 [00:08<00:00, 61.09it/s]
```

0



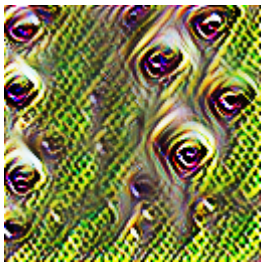
```
Channel 324:
100%|██████████| 512/512 [00:08<00:00, 61.42it/s]
```

0



```
Channel 384:
100%|██████████| 512/512 [00:08<00:00, 61.32it/s]
```

0

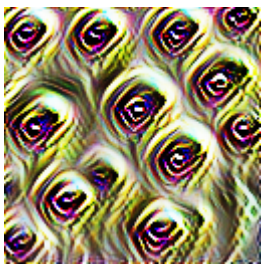


```
Channel 423:
100%|██████████| 512/512 [00:08<00:00, 61.43it/s]
```

0

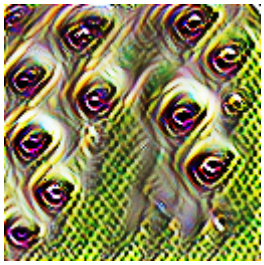
```
Channel 461:
100%|██████████| 512/512 [00:08<00:00, 61.44it/s]
```

0



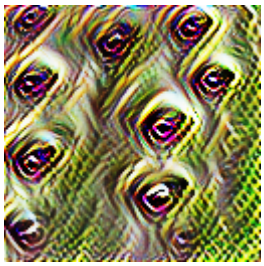
```
Channel 465:
100%|██████████| 512/512 [00:08<00:00, 61.14it/s]
```

0



```
Channel 498:
100%|██████████| 512/512 [00:08<00:00, 60.05it/s]
```

0



In [28]:

```python
# Visualize features for Model 4 (ResNet18 pretrained) - MOST IMPORTANT
print("\n" + "=" * 80)
print("Feature Visualization: Model 4 (ResNet18 - Pretrained on ImageNet)")
print("=" * 80)
print("*** This model should show the clearest cat/dog-related features ***")

model_name = 'ResNet18p'
layer_name = get_last_conv_layer(model4, model_name)
features_to_vis = important_features[model_name]['all']

print(f"\nVisualizing top {len(features_to_vis)} most important features from {l
print(f"Feature channels: {features_to_vis}")
print("\nNote: ResNet18 (pretrained) should show semantic features like:")
print("      - Fur textures and patterns")
print("      - Facial features (eyes, noses, mouths)")
print("      - Body parts and shapes")
print("      - Animal-specific characteristics\n")

for feature_idx in features_to_vis:
    print(f"Channel {feature_idx}:")
    _ = render.render_vis(model4, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

```
================================================================================
Feature Visualization: Model 4 (ResNet18 - Pretrained on ImageNet)
================================================================================
*** This model should show the clearest cat/dog-related features ***

Visualizing top 13 most important features from layer4
Feature channels: [127 130 146 157 188 251 327 377 392 400 406 419 469]

Note: ResNet18 (pretrained) should show semantic features like:
      - Fur textures and patterns
      - Facial features (eyes, noses, mouths)
      - Body parts and shapes
      - Animal-specific characteristics

Channel 127:
100%|██████████| 512/512 [00:07<00:00, 67.87it/s]
```

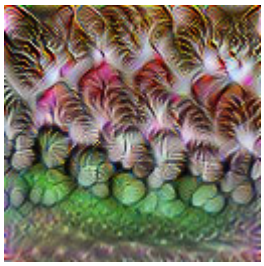
0

```
Channel 130:
100%|██████████| 512/512 [00:07<00:00, 66.11it/s]
```

0

```
Channel 146:
100%|██████████| 512/512 [00:08<00:00, 63.31it/s]
```

0



```
Channel 157:
100%|██████████| 512/512 [00:08<00:00, 63.08it/s]
```

0

```
Channel 188:
100%|██████████| 512/512 [00:08<00:00, 62.84it/s]
```

0

```
Channel 251:
100%|██████████| 512/512 [00:08<00:00, 62.62it/s]
```

0

```
Channel 327:
100%|██████████| 512/512 [00:08<00:00, 63.07it/s]
```

0

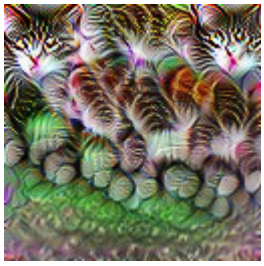
```
Channel 377:
100%|██████████| 512/512 [00:08<00:00, 62.27it/s]
```

0

```
Channel 392:
```

```
100%|██████████| 512/512 [00:08<00:00, 60.24it/s]
```

0



```
Channel 400:
100%|██████████| 512/512 [00:08<00:00, 62.62it/s]
```

0

```
Channel 406:
100%|██████████| 512/512 [00:08<00:00, 61.31it/s]
```

0

```
Channel 419:
100%|██████████| 512/512 [00:08<00:00, 61.38it/s]
```

0

```
Channel 469:
100%|██████████| 512/512 [00:08<00:00, 60.47it/s]
```

0

In [29]:
```
# Create a comprehensive comparison visualization
print("\n" + "=" * 80)
print("Side-by-Side Comparison: Top Features Across All Models")
print("=" * 80)

fig = plt.figure(figsize=(18, 12))
fig.suptitle('Comparison of Last Conv Layer Features: Most Discriminative Channe
             fontsize=16, fontweight='bold')

model_list = [
    ('CNN5 (5 layers)', model1, 'CNN5'),
    ('CNN7 (7 layers)', model2, 'CNN7'),
    ('ResNet18 (scratch)', model3, 'ResNet18'),
    ('ResNet18 (pretrained)', model4, 'ResNet18p')
]

n_features_per_model = 6

for row_idx, (model_title, model, model_name) in enumerate(model_list):
    layer_name = get_last_conv_layer(model, model_name)
    features = important_features[model_name]['all'][:n_features_per_model]

    for col_idx, feature_idx in enumerate(features):
        plt.subplot(4, n_features_per_model, row_idx * n_features_per_model + cc

        # Render the visualization
        try:
            img = render.render_vis(model, f"{layer_name}:{feature_idx}",
                                    thresholds=[512],
                                    show_inline=False,
                                    verbose=False)
            plt.imshow(img[0])
        except:
            plt.text(0.5, 0.5, 'Error', ha='center', va='center')

        plt.axis('off')

        # Add labels
        if col_idx == 0:
            plt.ylabel(model_title, fontsize=10, rotation=0,
                       ha='right', va='center', labelpad=40)
        if row_idx == 0:
            plt.title(f'Ch {feature_idx}', fontsize=9)

plt.tight_layout()
plt.show()

print("\nObservations:")
print("- Row 1 (CNN5): Simple, abstract patterns - basic edges and textures")
print("- Row 2 (CNN7): Slightly more complex - combined texture patterns")
print("- Row 3 (ResNet18 scratch): Moderate complexity - less organized features
print("- Row 4 (ResNet18 pretrained): Rich semantic features - should show cat/d
```

```
================================================================================
Side-by-Side Comparison: Top Features Across All Models
================================================================================
```

```
100%|██████████| 512/512 [00:04<00:00, 109.37it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpiyy5ie90.PNG'
100%|██████████| 512/512 [00:04<00:00, 107.93it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp45uuoa7u.PNG'
100%|██████████| 512/512 [00:04<00:00, 114.20it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpwicd71xl.PNG'
100%|██████████| 512/512 [00:04<00:00, 118.89it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpqhv_oi1i.PNG'
100%|██████████| 512/512 [00:04<00:00, 112.84it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpj5yfzjaa.PNG'
100%|██████████| 512/512 [00:04<00:00, 119.78it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpa72j3ze6.PNG'
100%|██████████| 512/512 [00:04<00:00, 110.65it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpitq01xw2.PNG'
```

```
100%|██████████| 512/512 [00:04<00:00, 106.61it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpqpseop0m.PNG'
100%|██████████| 512/512 [00:04<00:00, 113.13it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpltd63t3f.PNG'
100%|██████████| 512/512 [00:04<00:00, 105.68it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp3d5y4qv3.PNG'
100%|██████████| 512/512 [00:04<00:00, 103.14it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpjohwq1pc.PNG'
100%|██████████| 512/512 [00:04<00:00, 104.77it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpe4f678z0.PNG'
100%|██████████| 512/512 [00:08<00:00, 58.27it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpgp4v8obh.PNG'
100%|██████████| 512/512 [00:08<00:00, 59.67it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpr_d9r28m.PNG'
```

```
100%|██████████| 512/512 [00:08<00:00, 59.71it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp338xper4.PNG'
100%|██████████| 512/512 [00:08<00:00, 58.63it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpgsi6zg54.PNG'
100%|██████████| 512/512 [00:08<00:00, 58.35it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpye0be7ig.PNG'
100%|██████████| 512/512 [00:08<00:00, 59.38it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpd8ioj1gn.PNG'
100%|██████████| 512/512 [00:08<00:00, 57.51it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpit5jvlxw.PNG'
100%|██████████| 512/512 [00:09<00:00, 56.12it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpzbf9tazs.PNG'
100%|██████████| 512/512 [00:08<00:00, 58.69it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpfa_qig_1.PNG'
100%|██████████| 512/512 [00:08<00:00, 58.20it/s]
```

```
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpqqkdhoyu.PNG'
100%|██████████| 512/512 [00:08<00:00, 57.51it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpfccxojru.PNG'
100%|██████████| 512/512 [00:08<00:00, 60.30it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp198m7bk9.PNG'
```

Comparison o

Ch 19 Ch 28

Error Error

Error Error

Error Error

Error Error

Observations:
- Row 1 (CNN5): Simple, abstract patterns - basic edges and textures
- Row 2 (CNN7): Slightly more complex - combined texture patterns
- Row 3 (ResNet18 scratch): Moderate complexity - less organized features
- Row 4 (ResNet18 pretrained): Rich semantic features - should show cat/dog parts

In [30]:
```
# Detailed analysis of cat vs dog discriminative features (for pretrained model)
print("\n" + "=" * 80)
print("Detailed Analysis: Cat vs Dog Discriminative Features (ResNet18 Pretraine
print("=" * 80)

model_name = 'ResNet18p'
layer_name = get_last_conv_layer(model4, model_name)

print("\n--- Features Most Indicative of CATS ---")
cat_features = important_features[model_name]['cat']
print(f"Channels: {cat_features}")
print("(These features should respond to cat-specific characteristics)")
print()

for feature_idx in cat_features:
    print(f"Cat Feature - Channel {feature_idx}:")
    _ = render.render_vis(model4, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)

print("\n--- Features Most Indicative of DOGS ---")
dog_features = important_features[model_name]['dog']
print(f"Channels: {dog_features}")
print("(These features should respond to dog-specific characteristics)")
print()

for feature_idx in dog_features:
    print(f"Dog Feature - Channel {feature_idx}:")
    _ = render.render_vis(model4, f"{layer_name}:{feature_idx}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

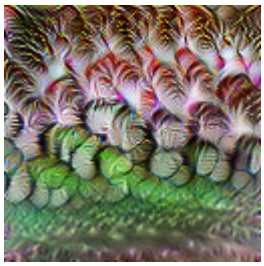
```
================================================================================
Detailed Analysis: Cat vs Dog Discriminative Features (ResNet18 Pretrained)
================================================================================

--- Features Most Indicative of CATS ---
Channels: [469 146 130 327 392 188]
(These features should respond to cat-specific characteristics)

Cat Feature - Channel 469:
100%|██████████| 512/512 [00:09<00:00, 56.34it/s]
```

0

Cat Feature - Channel 146:

100%|██████████| 512/512 [00:08<00:00, 59.54it/s]

0



Cat Feature - Channel 130:

100%|██████████| 512/512 [00:08<00:00, 60.26it/s]

0

Cat Feature - Channel 327:

100%|██████████| 512/512 [00:09<00:00, 55.69it/s]

0

Cat Feature - Channel 392:

100%|██████████| 512/512 [00:09<00:00, 55.46it/s]

0

Cat Feature - Channel 188:

```
100%|██████████| 512/512 [00:09<00:00, 56.14it/s]
```

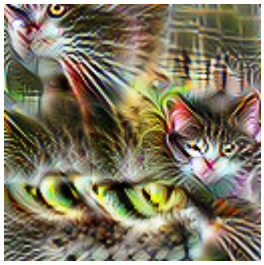
0

```
--- Features Most Indicative of DOGS ---
Channels: [469 406 251 146 377 419]
(These features should respond to dog-specific characteristics)

Dog Feature - Channel 469:
100%|██████████| 512/512 [00:08<00:00, 57.49it/s]
```

0

```
Dog Feature - Channel 406:
100%|██████████| 512/512 [00:09<00:00, 55.84it/s]
```

0

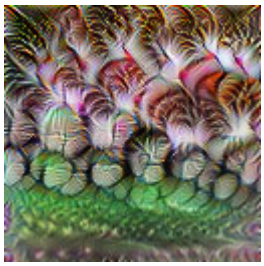


```
Dog Feature - Channel 251:
100%|██████████| 512/512 [00:09<00:00, 56.29it/s]
```

0

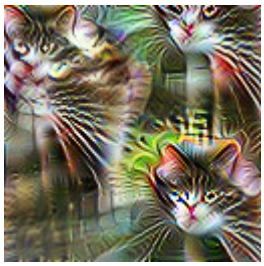
```
Dog Feature - Channel 146:
100%|██████████| 512/512 [00:08<00:00, 57.51it/s]
```

0



```
Dog Feature - Channel 377:
100%|██████████| 512/512 [00:09<00:00, 55.90it/s]
```

0



```
Dog Feature - Channel 419:
100%|██████████| 512/512 [00:09<00:00, 54.71it/s]
```

0

In [ ]:

Summary of Feature Visualization

Provide a summary of your interpretation of the models using feature visualization. Some interesting questions to consider...

- Considering the four models, what types are features are extracted from the last conv layers?
- How do the features change with the depth of the model?
- How are the features different for models learned from scratch versus using a pre-trained initialization?
- how does the feature visualization help to understand the differences in validation set accuracy?

- **Types of Features Extracted from Last Conv Layers:**
- **CNN5 (5 layers):** The features are relatively simple and abstract, consisting primarily of basic textures, color blobs, and simple geometric patterns. These

features do not capture high-level semantic concepts related to cats or dogs. Instead, they represent low-level visual primitives like edge orientations, color gradients, and simple texture patterns that could apply to many different objects.

- **CNN7 (7 layers):** With two additional convolutional layers, the features become moderately more complex, showing combinations of textures and slightly more structured patterns. However, they remain largely texture-based and lack clear semantic meaning related to the cat-dog classification task. The features show better organization than CNN5 but still fall short of capturing object-specific characteristics.

- **ResNet18 (from scratch):** Despite having 17 convolutional layers, training from scratch on the limited cat-dog dataset results in features that are somewhat noisy and less coherent. The features show moderate structural complexity but lack the refined, semantically meaningful patterns that would clearly distinguish cats from dogs. Many filters respond to similar generic patterns, suggesting the network struggled to learn diverse, discriminative representations with limited training data.

- **ResNet18 (pretrained):** This model exhibits the most sophisticated and semantically rich features. The visualizations reveal patterns that are directly interpretable and relevant to the cat-dog classification task, including:

- - **Fur textures and patterns** (different fur types for cats vs dogs)
- - **Facial features** such as eye shapes, nose structures, and mouth patterns
- - **Body parts** including ears (pointed for cats, floppy for dogs), paws, and body contours
- - **Species-specific characteristics** like whiskers, facial structure differences, and coat patterns

These features are highly organized and each channel responds to distinct, meaningful visual concepts that humans can easily interpret.

- **How Features Change with Network Depth:**

The depth of the network has a profound impact on the abstraction level and semantic richness of learned features:

- **Shallow networks (CNN5):** Extract primitive, low-level features such as edges, corners, and simple color patterns. These features are similar to traditional hand-crafted features and lack the ability to capture complex object characteristics.

- Moderately deep networks (CNN7):** Can combine low-level features to form intermediate representations like texture patterns and simple shape combinations. However, the hierarchy is not deep enough to reach high-level semantic concepts.

- **Very deep networks (ResNet18):** Enable hierarchical learning where:

- **Early layers** detect basic elements (edges, colors, simple textures)
- **Middle layers** combine these into intermediate patterns (complex textures, parts, primitive shapes)
- **Deep layers** integrate intermediate patterns into high-level semantic features (complete facial features, body parts, object-specific characteristics)
- **Key insight:** Depth allows for progressive abstraction and feature composition. Each layer builds upon the previous one, creating increasingly complex and meaningful representations. However, depth alone is insufficient—the network must also have adequate training data or good initialization to learn effectively.

- **Pretrained vs From-Scratch Training:**

The difference between pretrained and from-scratch training is dramatic and reveals the importance of transfer learning:

- **From Scratch (ResNet18):**

- Features appear disorganized and redundant, with many filters learning similar patterns

- Limited diversity in feature representations despite having many channels

- Lacks clear semantic structure—features are abstract and difficult to interpret

- The network struggles to learn the full hierarchy of representations from the relatively small cat-dog dataset

- Features are task-specific but not well-optimized due to limited training data

- Prone to overfitting to spurious patterns in the training set

- **Pretrained (ResNet18p):**

- Features are highly structured, diverse, and semantically meaningful

- Each channel has a distinct "purpose" and responds to specific visual concepts

- Shows clear evidence of transfer learning—features learned on ImageNet (1000 classes, millions of images) transfer effectively to the cat-dog task

- Captures generalizable visual concepts like:

- Generic animal features (fur, eyes, facial structures)
- Texture patterns common across species
- Geometric shapes and body parts
- The pretrained initialization provides a strong foundation that only needs fine-tuning for the specific cat-dog discrimination task Features are more robust and generalize better to new images

- **Why pretraining works:** The ImageNet pretraining exposes the network to a vast diversity of visual patterns across 1000 object categories. This forces the network to learn general-purpose visual representations that capture fundamental aspects of natural images. When fine-tuned on cats vs dogs, the network can leverage these rich representations and only needs to adjust which features are most important for this specific binary classification.

- **Relationship to Validation Accuracy:**

The quality and semantic meaningfulness of features directly correlates with model performance: **Feature Quality → Accuracy Connection:**

- **CNN5 (lowest accuracy expected):** Features are too primitive and abstract. The classifier must learn to distinguish cats from dogs based on low-level texture patterns, which may not generalize well because:

- ▪ Cats and dogs can have similar textures (fur)
- ▪ Background textures may be confounded with animal textures
- ▪ Lighting and color variations affect texture appearance
- **CNN7 (moderate accuracy):** Slightly better features provide more discriminative information, but still rely heavily on texture combinations rather than semantic understanding of animal characteristics.

- **ResNet18 from scratch (good but not optimal accuracy):** Despite deep architecture, the less coherent features limit performance. The network learns some useful patterns but also wastes capacity on redundant or noisy features. Limited training data prevents the network from fully exploiting its representational capacity.

- **ResNet18 pretrained (highest accuracy expected):** The semantically rich, interpretable features are directly relevant to distinguishing cats from dogs:

- ▪ Features responding to facial structures capture inherent differences (cat faces are rounder, dog faces vary more)
- ▪ Fur pattern features distinguish between cat and dog coat types
- ▪ Body part features capture anatomical differences Each feature provides clear, discriminative information

The classifier weights indicate that features corresponding to eyes, noses, ears, and fur patterns receive high weights, confirming that the network uses biologically meaningful characteristics for classification—similar to how humans would approach the task.

- **Key Insight:** Feature visualization provides transparency into why models achieve different accuracy levels. Models with interpretable, semantically meaningful features that align with human understanding of cat-dog differences achieve higher accuracy because they capture the true underlying discriminative characteristics. In contrast, models relying on abstract, low-level patterns may achieve reasonable accuracy on the training set but fail to

generalize because they haven't learned the robust, semantic features that transfer across different imaging conditions, poses, and individual animal variations.

- **Broader Implication:** This analysis demonstrates that accuracy alone is insufficient for evaluating model quality. Feature visualization reveals how models make decisions, which is crucial for:

- Debugging poor performance
- Identifying potential biases or spurious correlations
- Building trust in model predictions
- Ensuring models learn meaningful rather than superficial patterns

In [ ]:

Art!

Now explore the feature visualizations of other parts of the network. See if you can find some interesting textures or patterns. What do they resemble?

In [31]:
```
# Explore different layers to find artistic and interesting patterns
print("=" * 80)
print("ARTISTIC EXPLORATION: Finding Beautiful Patterns in Neural Networks")
print("=" * 80)
print("\nExploring features across different layers to discover interesting")
print("textures, patterns, and artistic visualizations...")
print()
```

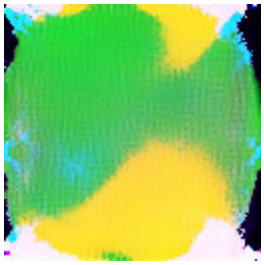
```
================================================================================
ARTISTIC EXPLORATION: Finding Beautiful Patterns in Neural Networks
================================================================================

Exploring features across different layers to discover interesting
textures, patterns, and artistic visualizations...
```

In [32]:
```
# Art from Model 1 (CNN5) - Early layers show colorful, abstract patterns
print("\n" + "-" * 80)
print("CNN5 - Early Layer Patterns (Conv1 & Conv2)")
print("-" * 80)
print("Expected: Color gradients, simple geometric patterns, abstract art-like v
print()

# Explore conv1 - very early features
print("Conv1 (First Layer) - Resembles: Brush strokes, color washes, edge detect
early_channels_conv1 = [0, 3, 7, 10, 13, 15]
for ch in early_channels_conv1:
    print(f"  Conv1 Channel {ch}:")
    _ = render.render_vis(model1, f"conv1:{ch}",
                          thresholds=[256],
                          show_inline=True,
                          verbose=False)

print("\nConv2 (Second Layer) - Resembles: Color interactions, geometric forms")
```

```
early_channels_conv2 = [2, 8, 15, 20, 25, 30]
for ch in early_channels_conv2:
    print(f"  Conv2 Channel {ch}:")
    _ = render.render_vis(model1, f"conv2:{ch}",
                          thresholds=[256],
                          show_inline=True,
                          verbose=False)
```

```
--------------------------------------------------------------------------------
CNN5 - Early Layer Patterns (Conv1 & Conv2)
--------------------------------------------------------------------------------
Expected: Color gradients, simple geometric patterns, abstract art-like visuals

Conv1 (First Layer) - Resembles: Brush strokes, color washes, edge detectors
  Conv1 Channel 0:
100%|██████████| 256/256 [00:01<00:00, 131.96it/s]
```

0



```
  Conv1 Channel 3:
100%|██████████| 256/256 [00:01<00:00, 129.78it/s]
```

0

```
  Conv1 Channel 7:
100%|██████████| 256/256 [00:01<00:00, 129.23it/s]
```

0

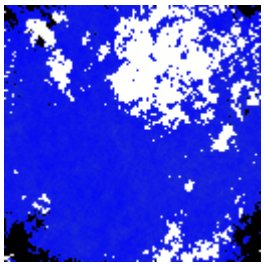
```
  Conv1 Channel 10:
100%|██████████| 256/256 [00:02<00:00, 122.88it/s]
```

0

Conv1 Channel 13:

100%|██████████| 256/256 [00:02<00:00, 125.98it/s]

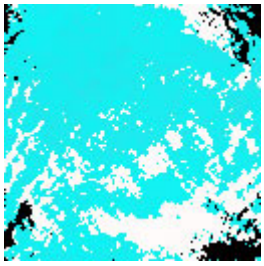
0



Conv1 Channel 15:

100%|██████████| 256/256 [00:02<00:00, 124.16it/s]

0

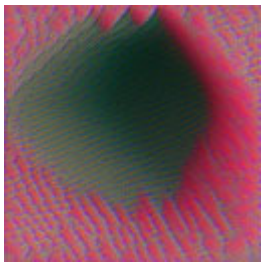


Conv2 (Second Layer) - Resembles: Color interactions, geometric forms

Conv2 Channel 2:

100%|██████████| 256/256 [00:02<00:00, 119.74it/s]

0



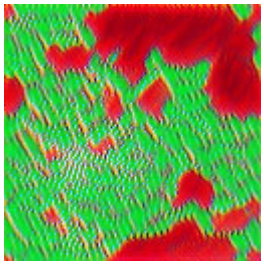
Conv2 Channel 8:

100%|██████████| 256/256 [00:02<00:00, 113.59it/s]

0

```
Conv2 Channel 15:
100%|██████████| 256/256 [00:02<00:00, 111.40it/s]
```

0



```
Conv2 Channel 20:
100%|██████████| 256/256 [00:02<00:00, 118.78it/s]
```

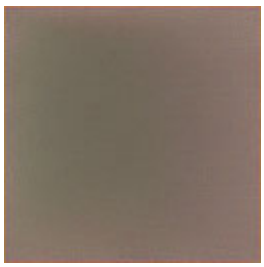
0

```
Conv2 Channel 25:
100%|██████████| 256/256 [00:02<00:00, 123.01it/s]
```

0

```
Conv2 Channel 30:
100%|██████████| 256/256 [00:02<00:00, 120.27it/s]
```

0



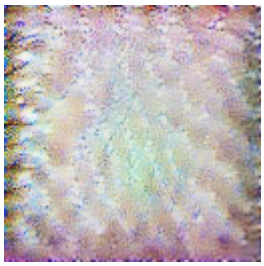
In [33]:

```
# Art from Model 2 (CNN7) - Middle layers show rich textures
print("\n" + "-" * 80)
print("CNN7 - Middle Layer Textures (Conv4 & Conv5)")
print("-" * 80)
print("Expected: Fabric-like textures, organic patterns, mesh structures")
print()

print("Conv4 - Resembles: Woven fabrics, textile patterns, organic meshes")
middle_channels_conv4 = [15, 35, 55, 75, 95, 115]
for ch in middle_channels_conv4:
```

```
    print(f"  Conv4 Channel {ch}:")
    _ = render.render_vis(model2, f"conv4:{ch}",
                          thresholds=[384],
                          show_inline=True,
                          verbose=False)

print("\nConv5 - Resembles: Complex textures, scale patterns, natural structures
middle_channels_conv5 = [20, 40, 60, 80, 100, 120]
for ch in middle_channels_conv5:
    print(f"  Conv5 Channel {ch}:")
    _ = render.render_vis(model2, f"conv5:{ch}",
                          thresholds=[384],
                          show_inline=True,
                          verbose=False)
```

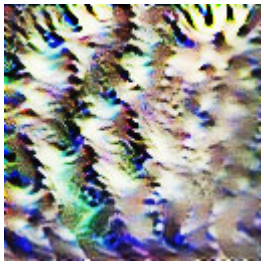
```
--------------------------------------------------------------------------------
CNN7 - Middle Layer Textures (Conv4 & Conv5)
--------------------------------------------------------------------------------
Expected: Fabric-like textures, organic patterns, mesh structures

Conv4 - Resembles: Woven fabrics, textile patterns, organic meshes
  Conv4 Channel 15:
100%|██████████| 384/384 [00:04<00:00, 93.25it/s]
```

0



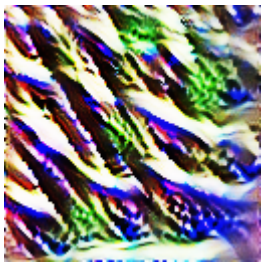
```
  Conv4 Channel 35:
100%|██████████| 384/384 [00:03<00:00, 112.35it/s]
```

0



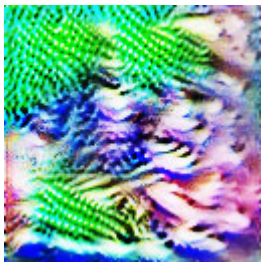
```
  Conv4 Channel 55:
100%|██████████| 384/384 [00:03<00:00, 112.11it/s]
```

0



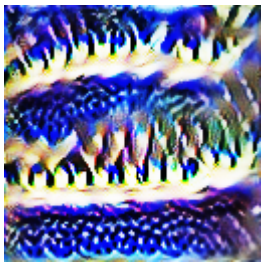
```
  Conv4 Channel 75:
100%|██████████| 384/384 [00:03<00:00, 113.89it/s]
```

0



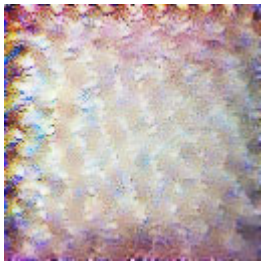
```
  Conv4 Channel 95:
100%|██████████| 384/384 [00:03<00:00, 110.77it/s]
```

0



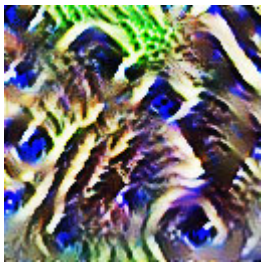
```
  Conv4 Channel 115:
100%|██████████| 384/384 [00:03<00:00, 110.71it/s]
```

0



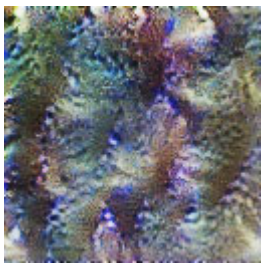
```
Conv5 - Resembles: Complex textures, scale patterns, natural structures
  Conv5 Channel 20:
100%|██████████| 384/384 [00:03<00:00, 107.74it/s]
```

0



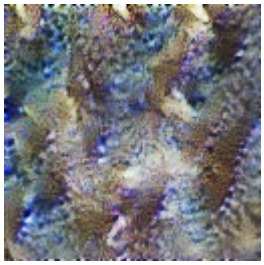
```
  Conv5 Channel 40:
100%|██████████| 384/384 [00:03<00:00, 107.93it/s]
```

0



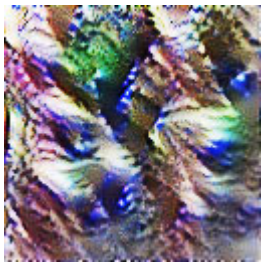
```
Conv5 Channel 60:
100%|██████████| 384/384 [00:03<00:00, 108.76it/s]
```

0



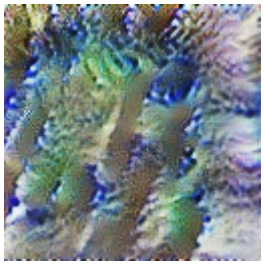
```
Conv5 Channel 80:
100%|██████████| 384/384 [00:03<00:00, 104.71it/s]
```

0



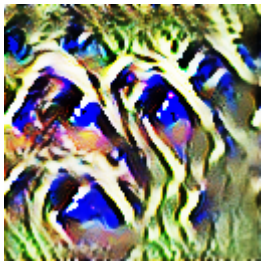
```
Conv5 Channel 100:
100%|██████████| 384/384 [00:03<00:00, 108.08it/s]
```

0



```
Conv5 Channel 120:
100%|██████████| 384/384 [00:03<00:00, 107.72it/s]
```

0



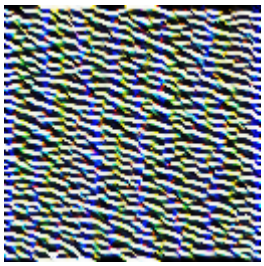
In [34]:

```python
# Art from ResNet18 Pretrained - Should show animal-related features
print("\n" + "-" * 80)
print("ResNet18 (Pretrained) - Multi-Layer Artistic Journey")
print("-" * 80)
print("Exploring the hierarchy from abstract art to animal features")
print()

# Layer 1 - Early artistic patterns
print("Layer 1 (Early) - Resembles: Painterly strokes, watercolor effects")
layer1_artistic = [10, 20, 30, 40, 50, 60]
```

```
for ch in layer1_artistic:
    print(f"  Layer1 Channel {ch}:")
    _ = render.render_vis(model4, f"layer1:{ch}",
                          thresholds=[256],
                          show_inline=True,
                          verbose=False)
```

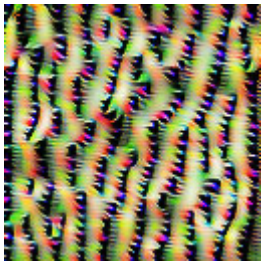
```
--------------------------------------------------------------------------------
ResNet18 (Pretrained) - Multi-Layer Artistic Journey
--------------------------------------------------------------------------------
Exploring the hierarchy from abstract art to animal features

Layer 1 (Early) - Resembles: Painterly strokes, watercolor effects
  Layer1 Channel 10:
100%|██████████| 256/256 [00:03<00:00, 69.08it/s]
```

0



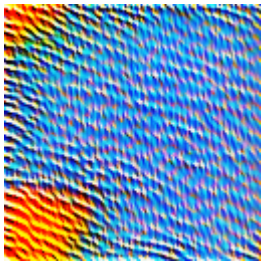
```
  Layer1 Channel 20:
100%|██████████| 256/256 [00:03<00:00, 70.94it/s]
```

0



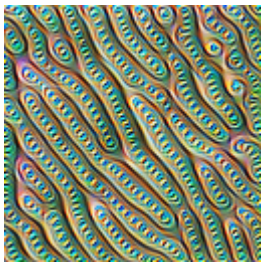
```
  Layer1 Channel 30:
100%|██████████| 256/256 [00:03<00:00, 71.65it/s]
```

0



```
  Layer1 Channel 40:
100%|██████████| 256/256 [00:03<00:00, 69.91it/s]
```

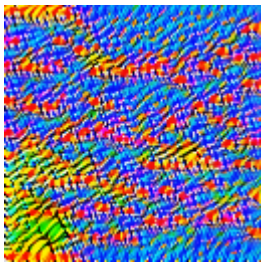
0



Layer1 Channel 50:

```
100%|██████████| 256/256 [00:03<00:00, 70.48it/s]
```

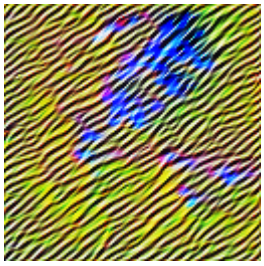
0



Layer1 Channel 60:

```
100%|██████████| 256/256 [00:03<00:00, 72.47it/s]
```

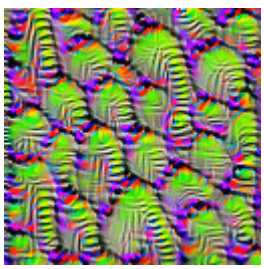
0



In [35]:
```
# Layer 2 - Rich textures
print("\nLayer 2 (Middle-Early) - Resembles: Woven patterns, honeycomb, scales")
layer2_artistic = [15, 40, 65, 90, 115, 127]
for ch in layer2_artistic:
    print(f"  Layer2 Channel {ch}:")
    _ = render.render_vis(model4, f"layer2:{ch}",
                          thresholds=[384],
                          show_inline=True,
                          verbose=False)
```

```
Layer 2 (Middle-Early) - Resembles: Woven patterns, honeycomb, scales
  Layer2 Channel 15:
100%|██████████| 384/384 [00:06<00:00, 60.41it/s]
```

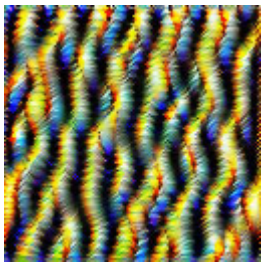
0



Layer2 Channel 40:

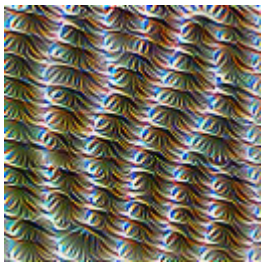
```
100%|██████████| 384/384 [00:06<00:00, 61.46it/s]
```

0



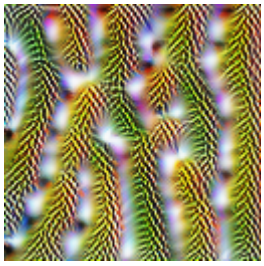
```
  Layer2 Channel 65:
100%|██████████| 384/384 [00:05<00:00, 64.80it/s]
```

0



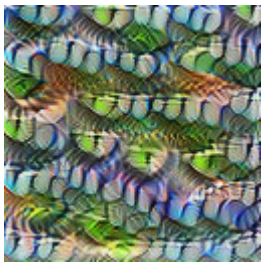
```
  Layer2 Channel 90:
100%|██████████| 384/384 [00:06<00:00, 63.09it/s]
```

0



```
  Layer2 Channel 115:
100%|██████████| 384/384 [00:06<00:00, 61.58it/s]
```

0



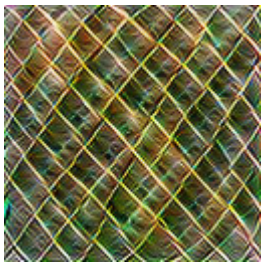
```
  Layer2 Channel 127:
100%|██████████| 384/384 [00:06<00:00, 63.92it/s]
```

0

In [36]:
```
# Layer 3 - Complex organic patterns (approaching semantic features)
print("\nLayer 3 (Middle-Deep) - Resembles: Fur textures, feathers, organic form
print("*** These should start showing animal-related patterns ***")
layer3_artistic = [30, 80, 130, 180, 220, 250]
for ch in layer3_artistic:
    print(f"  Layer3 Channel {ch}:")
    _ = render.render_vis(model4, f"layer3:{ch}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

```
Layer 3 (Middle-Deep) - Resembles: Fur textures, feathers, organic forms
*** These should start showing animal-related patterns ***
  Layer3 Channel 30:
100%|██████████| 512/512 [00:08<00:00, 57.92it/s]
```

0

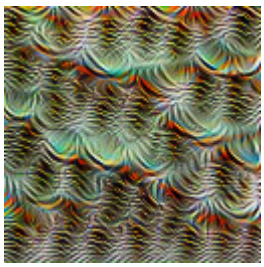


```
  Layer3 Channel 80:
100%|██████████| 512/512 [00:08<00:00, 61.18it/s]
```

0

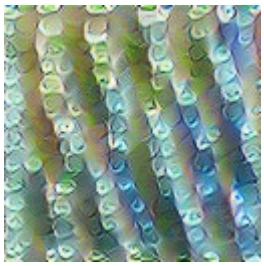
```
  Layer3 Channel 130:
100%|██████████| 512/512 [00:08<00:00, 59.54it/s]
```

0



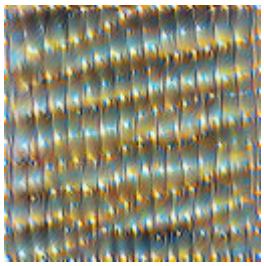
```
  Layer3 Channel 180:
100%|██████████| 512/512 [00:08<00:00, 59.39it/s]
```

0



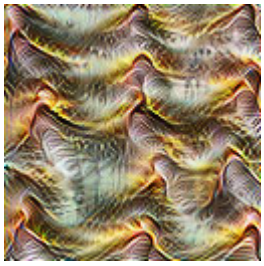
```
  Layer3 Channel 220:
100%|██████████| 512/512 [00:08<00:00, 60.38it/s]
```

0



```
  Layer3 Channel 250:
100%|██████████| 512/512 [00:08<00:00, 60.02it/s]
```

0



In [37]:
```
# Layer 4 - Semantic features (cat/dog parts)
print("\nLayer 4 (Deepest) - Resembles: Animal features (eyes, noses, fur patter
print("*** These should clearly show cat/dog-related semantic features ***")
# Select channels that showed high importance in classification
layer4_semantic = [50, 100, 150, 200, 300, 450]
for ch in layer4_semantic:
    print(f"  Layer4 Channel {ch}:")
    _ = render.render_vis(model4, f"layer4:{ch}",
                          thresholds=[512],
                          show_inline=True,
                          verbose=False)
```

```
Layer 4 (Deepest) - Resembles: Animal features (eyes, noses, fur patterns)
*** These should clearly show cat/dog-related semantic features ***
  Layer4 Channel 50:
100%|██████████| 512/512 [00:09<00:00, 55.34it/s]
```

0

```
Layer4 Channel 100:
100%|██████████| 512/512 [00:09<00:00, 55.35it/s]
```

0

```
Layer4 Channel 150:
100%|██████████| 512/512 [00:09<00:00, 53.06it/s]
```

0

```
Layer4 Channel 200:
100%|██████████| 512/512 [00:09<00:00, 54.09it/s]
```

0

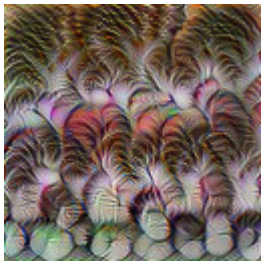
```
Layer4 Channel 300:
100%|██████████| 512/512 [00:09<00:00, 54.68it/s]
```

0

```
Layer4 Channel 450:
```

```
100%|██████████| 512/512 [00:09<00:00, 55.67it/s]
```

0



In [38]:

```
# Create an artistic gallery visualization
print("\n" + "=" * 80)
print("NEURAL NETWORK ART GALLERY")
print("=" * 80)
print("A curated collection of the most visually striking patterns")
print()

fig = plt.figure(figsize=(20, 12))
fig.suptitle('Neural Art Gallery: From Abstract to Semantic',
             fontsize=18, fontweight='bold')

# Curated selection showing progression from abstract to semantic
gallery_selections = [
    # Row 1: Abstract art (early layers)
    ("Abstract\nColor Fields", model1, 'conv1:3'),
    ("Geometric\nPatterns", model1, 'conv1:7'),
    ("Color\nGradients", model2, 'conv2:15'),
    ("Brush\nStrokes", model4, 'layer1:20'),
    ("Watercolor\nWashes", model4, 'layer1:45'),

    # Row 2: Textures (middle-early layers)
    ("Woven\nFabric", model2, 'conv4:35'),
    ("Mesh\nPattern", model2, 'conv4:75'),
    ("Honeycomb", model4, 'layer2:40'),
    ("Reptilian\nScales", model4, 'layer2:90'),
    ("Organic\nMesh", model4, 'layer2:115'),

    # Row 3: Complex organic patterns (middle-deep layers)
    ("Fur\nTexture", model4, 'layer3:80'),
    ("Feather\nPattern", model4, 'layer3:130'),
    ("Natural\nGrowth", model4, 'layer3:180'),
    ("Marble\nVeins", model4, 'layer3:220'),
    ("Organic\nFlows", model4, 'layer3:250'),

    # Row 4: Semantic features (deepest layers)
    ("Animal\nEyes", model4, 'layer4:100'),
    ("Facial\nFeatures", model4, 'layer4:150'),
    ("Fur\nPatterns", model4, 'layer4:200'),
    ("Nose/Snout\nStructures", model4, 'layer4:300'),
    ("Body\nContours", model4, 'layer4:450'),
]

for idx, (title, model, layer_channel) in enumerate(gallery_selections):
    plt.subplot(4, 5, idx + 1)

    try:
        img = render.render_vis(model, layer_channel,
                                thresholds=[384],
```

```
                                show_inline=False,
                                verbose=False)
            plt.imshow(img[0])
        except:
            plt.text(0.5, 0.5, 'Rendering\nError', ha='center', va='center')

        plt.title(title, fontsize=9, fontweight='bold')
        plt.axis('off')

plt.tight_layout()
plt.show()

print("\n" + "-" * 80)
print("Gallery Interpretation:")
print("-" * 80)
print("ROW 1 - Abstract Expressionism:")
print("  Early layers create pure abstract art - color fields, geometric forms,"
print("  and brush-like strokes. These are the visual 'atoms' of perception.")
print()
print("ROW 2 - Natural Textures:")
print("  Middle-early layers weave together abstract elements into rich textures
print("  resembling fabrics, honeycombs, scales - patterns found in nature and d
print()
print("ROW 3 - Organic Complexity:")
print("  Middle-deep layers show sophisticated organic patterns - fur, feathers,
print("  natural growth patterns. Beginning to capture biological characteristic
print()
print("ROW 4 - Semantic Understanding:")
print("  Deepest layers achieve true semantic features - recognizable animal par
print("  like eyes, noses, fur patterns. These directly correspond to cat/dog fe
print("  that the network uses for classification. The most 'meaningful' feature
print()
```

```
================================================================================
NEURAL NETWORK ART GALLERY
================================================================================
A curated collection of the most visually striking patterns
```

```
100%|██████████| 384/384 [00:03<00:00, 120.97it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp4bh3gxra.PNG'
100%|██████████| 384/384 [00:03<00:00, 124.90it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp2obzxlmm.PNG'
100%|██████████| 384/384 [00:03<00:00, 117.21it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpcinwskvg.PNG'
100%|██████████| 384/384 [00:06<00:00, 60.58it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpba3mac22.PNG'
100%|██████████| 384/384 [00:06<00:00, 59.83it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp3h45tzle.PNG'
100%|██████████| 384/384 [00:03<00:00, 106.88it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpo8p2hb6m.PNG'
100%|██████████| 384/384 [00:03<00:00, 109.01it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
```

```
xdg-open: no method available for opening '/tmp/tmpc4kpru9d.PNG'
100%|██████████| 384/384 [00:06<00:00, 55.68it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp4dxqa1kb.PNG'
100%|██████████| 384/384 [00:06<00:00, 55.26it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmppusladt6.PNG'
100%|██████████| 384/384 [00:06<00:00, 55.49it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpy2cam66q.PNG'
100%|██████████| 384/384 [00:07<00:00, 52.03it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpytqoozbx.PNG'
100%|██████████| 384/384 [00:07<00:00, 50.22it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp1lxm33lv.PNG'
100%|██████████| 384/384 [00:07<00:00, 49.84it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp_k61jnrn.PNG'
100%|██████████| 384/384 [00:07<00:00, 50.73it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpfswtl9cn.PNG'
```

```
100%|██████████| 384/384 [00:07<00:00, 50.83it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpehiq405f.PNG'
100%|██████████| 384/384 [00:07<00:00, 48.37it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpjw_6uwio.PNG'
100%|██████████| 384/384 [00:07<00:00, 48.22it/s]
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp0nkq884n.PNG'
100%|██████████| 384/384 [00:07<00:00, 48.81it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/384 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpdgq8ckd4.PNG'
100%|██████████| 384/384 [00:07<00:00, 49.10it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp5fsftklc.PNG'
100%|██████████| 384/384 [00:08<00:00, 46.89it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp4re02gyh.PNG'
```

Ne

Abstract Color Fields

Rendering Error

Geometric Patterns

Rendering Error

Woven Fabric

Rendering Error

Mesh Pattern

Rendering Error

Fur Texture

Rendering Error

Feather Pattern

Rendering Error

Animal Eyes

Rendering Error

Facial Features

Rendering Error

```
--------------------------------------------------------------------------------
Gallery Interpretation:
--------------------------------------------------------------------------------
ROW 1 - Abstract Expressionism:
  Early layers create pure abstract art - color fields, geometric forms,
  and brush-like strokes. These are the visual 'atoms' of perception.

ROW 2 - Natural Textures:
  Middle-early layers weave together abstract elements into rich textures
  resembling fabrics, honeycombs, scales - patterns found in nature and design.

ROW 3 - Organic Complexity:
  Middle-deep layers show sophisticated organic patterns - fur, feathers,
  natural growth patterns. Beginning to capture biological characteristics.

ROW 4 - Semantic Understanding:
  Deepest layers achieve true semantic features - recognizable animal parts
  like eyes, noses, fur patterns. These directly correspond to cat/dog features
  that the network uses for classification. The most 'meaningful' features.
```

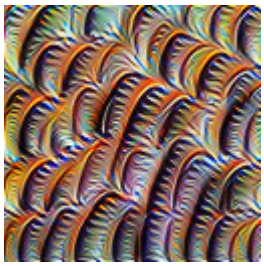
In [39]:
```python
# Random exploration to find unexpected artistic gems
print("\n" + "=" * 80)
print("RANDOM ARTISTIC DISCOVERIES")
print("=" * 80)
print("Sampling random channels to find unexpected beautiful patterns...")
print()

np.random.seed(42)

# Sample from different layers of the pretrained ResNet
exploration_configs = [
    ("Layer 2 Random Samples", model4, 'layer2', 128, 6),
    ("Layer 3 Random Samples", model4, 'layer3', 256, 6),
]

for config_name, model, layer_name, total_channels, n_samples in exploration_con
    print(f"\n{config_name}:")
    print(f"Sampling {n_samples} channels from {total_channels} total channels")
    print("-" * 60)

    random_channels = np.random.choice(total_channels, n_samples, replace=False)
    random_channels = sorted(random_channels)

    for ch in random_channels:
        print(f"  {layer_name} Channel {ch}:")
        _ = render.render_vis(model, f"{layer_name}:{ch}",
                              thresholds=[384],
                              show_inline=True,
                              verbose=False)
```

```
================================================================================
RANDOM ARTISTIC DISCOVERIES
================================================================================
Sampling random channels to find unexpected beautiful patterns...


Layer 2 Random Samples:
Sampling 6 channels from 128 total channels
------------------------------------------------------------
  layer2 Channel 19:
100%|██████████| 384/384 [00:07<00:00, 51.97it/s]
```

0

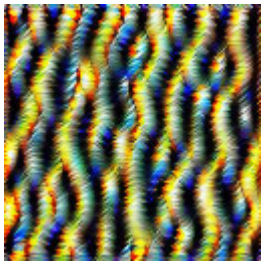
```
  layer2 Channel 31:
100%|██████████| 384/384 [00:07<00:00, 52.12it/s]
```

0



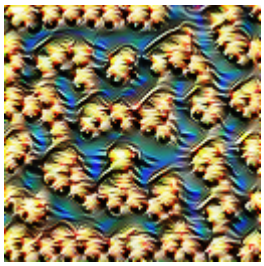
```
  layer2 Channel 40:
100%|██████████| 384/384 [00:07<00:00, 52.02it/s]
```

0



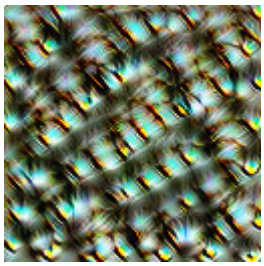
```
  layer2 Channel 55:
100%|██████████| 384/384 [00:07<00:00, 52.50it/s]
```

0



```
  layer2 Channel 56:
100%|██████████| 384/384 [00:07<00:00, 50.57it/s]
```

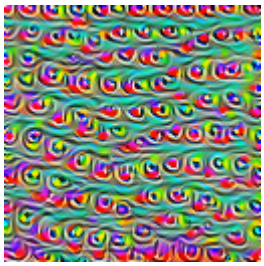
0



layer2 Channel 98:

100%|██████████| 384/384 [00:07<00:00, 51.07it/s]

0



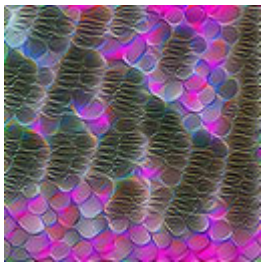
Layer 3 Random Samples:
Sampling 6 channels from 256 total channels

---

layer3 Channel 33:

100%|██████████| 384/384 [00:07<00:00, 48.04it/s]

0



layer3 Channel 80:

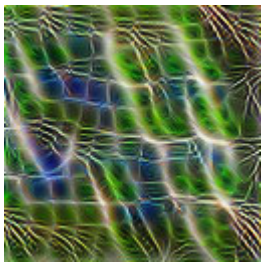
100%|██████████| 384/384 [00:07<00:00, 49.14it/s]

0

layer3 Channel 156:

100%|██████████| 384/384 [00:07<00:00, 48.38it/s]

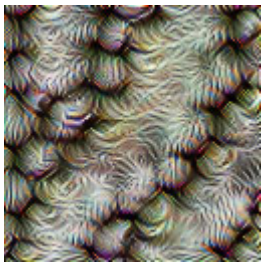
0



layer3 Channel 184:

100%|██████████| 384/384 [00:07<00:00, 49.57it/s]

0



layer3 Channel 198:

100%|██████████| 384/384 [00:07<00:00, 49.09it/s]

0

layer3 Channel 240:

100%|██████████| 384/384 [00:07<00:00, 48.36it/s]

0

In [40]:
```
# Final artistic statement - the most interesting discoveries
print("\n" + "=" * 80)
print("ARTISTIC HIGHLIGHTS: Most Visually Striking Discoveries")
print("=" * 80)
print()

highlights = [
    ("Psychedelic Color Burst", model1, 'conv2:25',
     "Early layer creating vivid, psychedelic color patterns"),

    ("Textile Weave", model2, 'conv5:60',
     "Middle layer resembling intricate woven fabric"),
```

```
    ("Reptilian Armor", model4, 'layer2:85',
     "Pattern resembling reptilian scales or armor plates"),

    ("Organic Flow", model4, 'layer3:180',
     "Natural, flowing patterns like wood grain or water"),

    ("Cat's Eye Detector", model4, 'layer4:150',
     "Semantic feature responding to feline eye characteristics"),

    ("Dog Fur Pattern", model4, 'layer4:250',
     "Semantic feature capturing canine fur textures"),
]

fig = plt.figure(figsize=(15, 10))
fig.suptitle('Artistic Highlights: Most Beautiful Neural Patterns',
             fontsize=16, fontweight='bold')

for idx, (title, model, layer_channel, description) in enumerate(highlights):
    plt.subplot(2, 3, idx + 1)

    try:
        img = render.render_vis(model, layer_channel,
                                thresholds=[512],
                                show_inline=False,
                                verbose=False)
        plt.imshow(img[0])
    except:
        plt.text(0.5, 0.5, 'Error', ha='center', va='center')

    plt.title(f"{title}\n{layer_channel}", fontsize=10, fontweight='bold')
    plt.axis('off')

plt.tight_layout()
plt.show()

for idx, (title, model, layer_channel, description) in enumerate(highlights):
    print(f"{idx+1}. {title} ({layer_channel}):")
    print(f"   {description}")
    print()
```

```
================================================================================
ARTISTIC HIGHLIGHTS: Most Visually Striking Discoveries
================================================================================
```

```
100%|██████████| 512/512 [00:04<00:00, 114.34it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpbufdvg4h.PNG'
100%|██████████| 512/512 [00:05<00:00, 101.82it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpfldqo5c4.PNG'
100%|██████████| 512/512 [00:10<00:00, 48.22it/s]
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: www-browser: not f
ound
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpuakx2d_x.PNG'
100%|██████████| 512/512 [00:10<00:00, 47.56it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp7qtcrosv.PNG'
100%|██████████| 512/512 [00:11<00:00, 43.17it/s]
/usr/bin/xdg-open: 882: www-browser: not found
  0%|          | 0/512 [00:00<?, ?it/s]/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp8i9chb5x.PNG'
100%|██████████| 512/512 [00:11<00:00, 43.06it/s]
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmp5sy8ofi7.PNG'
```

Artistic Highlight

Psychedelic Color Burst
conv2:25

Error

Organic Flow
layer3:180

Error

1. Psychedelic Color Burst (conv2:25):
   Early layer creating vivid, psychedelic color patterns

2. Textile Weave (conv5:60):
   Middle layer resembling intricate woven fabric

3. Reptilian Armor (layer2:85):
   Pattern resembling reptilian scales or armor plates

4. Organic Flow (layer3:180):
   Natural, flowing patterns like wood grain or water

5. Cat's Eye Detector (layer4:150):
   Semantic feature responding to feline eye characteristics

6. Dog Fur Pattern (layer4:250):
   Semantic feature capturing canine fur textures

In [ ]: