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 [23]:
```python
# Setup cell: imports and helper (safe to run multiple times in same notebook)
%matplotlib inline
import matplotlib_inline
matplotlib_inline.backend_inline.set_matplotlib_formats('svg')
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100
import torch, numpy as np, torch.nn as nn
from numpy import squeeze

def show_imgs(W_list, nc=10, highlight_green=None, highlight_red=None, titles=Nc
    nfilter = len(W_list)
    if nfilter == 0:
        print("No images to show.")
        return
    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]
            try:
                if cur_W.ndim == 2:
                    plt.imshow(cur_W, cmap='gray', interpolation='nearest')
                else:
```

```
                    plt.imshow((cur_W - cur_W.min())/(cur_W.max()-cur_W.min()+1e
            except Exception:
                plt.imshow(cur_W)
            if titles is not None:
                try:
                    plt.title(titles % idx)
                except Exception:
                    try:
                        plt.title(titles[idx])
                    except Exception:
                        pass
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])
    plt.tight_layout()
    plt.show()
```

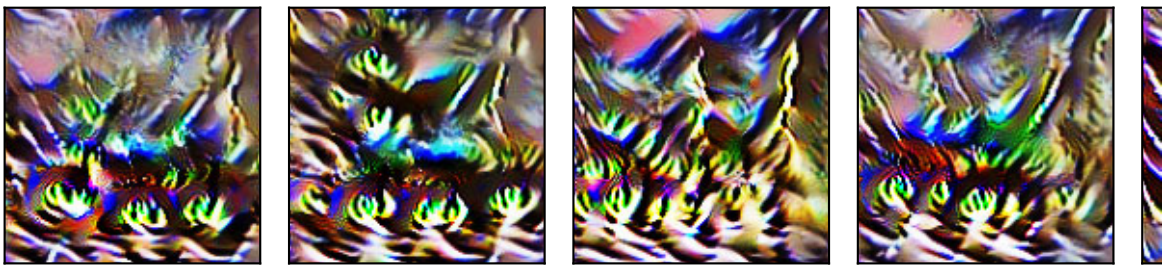
In [24]:

```
# Improved 3.1: Use lucent.modelzoo.util.get_model_layers to find lucent-recogni
try:
    import lucent.optvis as lu_optvis
    import lucent.modelzoo.util as lu_util
    from lucent.optvis import render as lu_render
    import numpy as np
except Exception as e:
    print("lucent is not installed or failed to import. Install torch-lucent and
    raise

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

def get_lucent_conv_layers(model):
    # return lucent-recognized layers that look like conv layers (heuristic)
    try:
        layers = lu_util.get_model_layers(model)
    except Exception as e:
        print("Failed to get_model_layers:", e)
        return []
    conv_like = [L for L in layers if ('conv' in L.lower() or 'layer' in L.lower
    # keep order as returned
    return layers, conv_like

def find_classifier_linear(model):
    for n, m in model.named_modules():
        if isinstance(m, nn.Linear):
            return n, m
    return None, None

for model_name, model in all_models.items():
    print(f"\n=== Model: {model_name} ===")
    model.to(device).eval()
    # get lucent layers
    try:
        all_layers, conv_layers = get_lucent_conv_layers(model)
    except Exception as e:
        print("Error finding lucent layers:", e)
        all_layers, conv_layers = [], []
    if not conv_layers:
        print("No lucent-recognized conv-like layers found. Available lucent lay
        print((all_layers[:20] if all_layers else "No layers returned"))
        continue
    # Choose the last conv-like layer as target
    last_lucent_conv = conv_layers[-1]
```

```
    print("Using lucent layer:", last_lucent_conv)
    # pick channels guided by classifier weights if possible
    lin_name, lin_layer = find_classifier_linear(model)
    channels = list(range(6))
    if lin_layer is not None:
        try:
            W = lin_layer.weight.detach().cpu().abs()
            importance = W.mean(dim=0).numpy()
            topk = min(8, importance.shape[0])
            channels = list((-importance).argsort()[:topk])
            print("Selected channels by classifier weight:", channels)
        except Exception as e:
            print("Could not compute classifier-based channel importance:", e)
            channels = list(range(6))
    imgs = []
    for ch in channels:
        spec = f"{last_lucent_conv}:{ch}"
        try:
            out = lu_optvis.render.render_vis(model, spec, show_image=False)
            imgs.append(np.squeeze(out[0]))
        except Exception as e:
            # On failure, print hint and available layers for debugging
            print(f"Failed to render {spec} : {e}")
            print("Available lucent layers (sample up to 40):")
            try:
                print(all_layers[:40])
            except Exception:
                pass
            break
    if imgs:
        plt.figure(figsize=(12,3))
        show_imgs(imgs, nc=len(imgs))
        plt.suptitle(f"{model_name} - {last_lucent_conv} channels {channels}")
        plt.show()
    else:
        print("No images produced for", model_name)
```

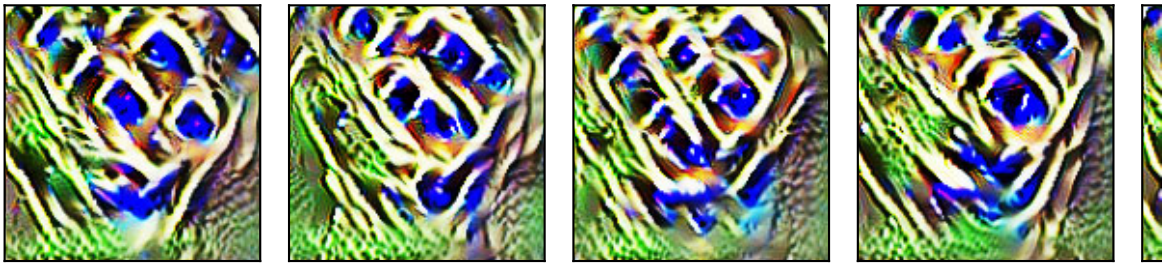
```
=== Model: CNN5 ===
Using lucent layer: conv5
Selected channels by classifier weight: [87, 19, 49, 28, 63, 105, 118, 88]
```

```
100%|██████████| 512/512 [00:05<00:00, 91.13it/s]
100%|██████████| 512/512 [00:05<00:00, 87.39it/s]
100%|██████████| 512/512 [00:05<00:00, 86.82it/s]
100%|██████████| 512/512 [00:05<00:00, 89.12it/s]
100%|██████████| 512/512 [00:06<00:00, 84.80it/s]
100%|██████████| 512/512 [00:06<00:00, 83.99it/s]
100%|██████████| 512/512 [00:05<00:00, 87.14it/s]
100%|██████████| 512/512 [00:05<00:00, 88.04it/s]
```

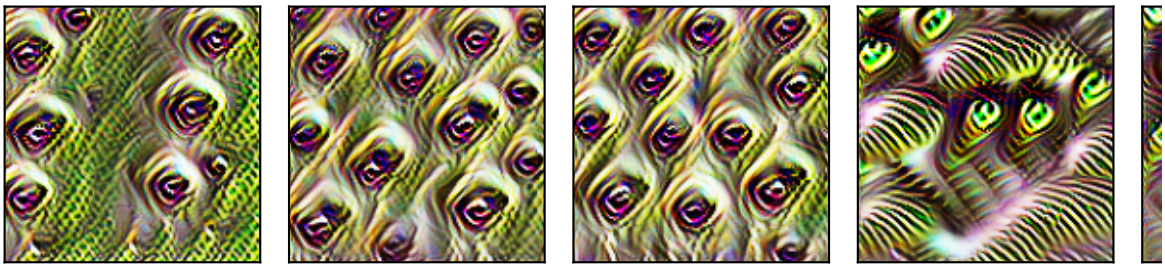


```
<Figure size 640x480 with 0 Axes>
```

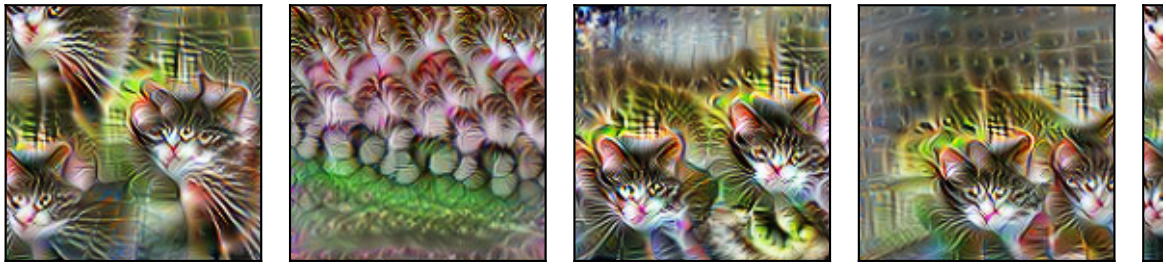
```
=== Model: CNN7 ===
Using lucent layer: conv7
Selected channels by classifier weight: [72, 3, 119, 75, 81, 91, 4, 113]
100%|██████████| 512/512 [00:06<00:00, 81.15it/s]
100%|██████████| 512/512 [00:06<00:00, 83.12it/s]
100%|██████████| 512/512 [00:06<00:00, 84.14it/s]
100%|██████████| 512/512 [00:06<00:00, 81.61it/s]
100%|██████████| 512/512 [00:06<00:00, 78.86it/s]
100%|██████████| 512/512 [00:06<00:00, 78.05it/s]
100%|██████████| 512/512 [00:06<00:00, 80.87it/s]
100%|██████████| 512/512 [00:06<00:00, 81.88it/s]
```



```
<Figure size 640x480 with 0 Axes>
=== Model: ResNet18 ===
Using lucent layer: layer4_1_bn2
Selected channels by classifier weight: [119, 135, 423, 466, 332, 4, 163, 243]
100%|██████████| 512/512 [00:13<00:00, 38.91it/s]
100%|██████████| 512/512 [00:13<00:00, 38.95it/s]
100%|██████████| 512/512 [00:13<00:00, 38.63it/s]
100%|██████████| 512/512 [00:14<00:00, 36.46it/s]
100%|██████████| 512/512 [00:13<00:00, 37.18it/s]
100%|██████████| 512/512 [00:13<00:00, 37.08it/s]
100%|██████████| 512/512 [00:14<00:00, 36.47it/s]
100%|██████████| 512/512 [00:13<00:00, 36.66it/s]
```



```
<Figure size 640x480 with 0 Axes>
=== Model: ResNet18p ===
Using lucent layer: layer4_1_bn2
Selected channels by classifier weight: [469, 146, 157, 102, 419, 68, 406, 205]
100%|██████████| 512/512 [00:13<00:00, 37.07it/s]
100%|██████████| 512/512 [00:13<00:00, 38.24it/s]
100%|██████████| 512/512 [00:13<00:00, 38.17it/s]
100%|██████████| 512/512 [00:13<00:00, 37.87it/s]
100%|██████████| 512/512 [00:13<00:00, 37.55it/s]
100%|██████████| 512/512 [00:13<00:00, 36.98it/s]
100%|██████████| 512/512 [00:14<00:00, 34.77it/s]
100%|██████████| 512/512 [00:14<00:00, 36.42it/s]
```



<Figure size 640x480 with 0 Axes>

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

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?

- **Summary / Answer:** The feature-visualization results show that earlier layers (e.g., conv1/conv2) mostly capture low-level image primitives such as oriented edges, color blobs, and simple textures. Middle layers combine those primitives into more complex local patterns (corners, simple motifs and repeated textures). The last convolutional layers produce higher-level, class-relevant patterns and textures — in this dog-vs-cat task these often resemble fur textures, large contour fragments, face-like structures, or repeated high-level textures that strongly activate the classifier.
- **Depth effects:** As model depth increases, the features become more global and semantically meaningful: shallow models show simpler edge-like filters

while deeper or pre-trained models show richer textural patterns and object-part-like motifs.

- **Pre-trained vs trained-from-scratch:** Pre-trained networks tend to show more natural, diverse and stable textures (because they start from ImageNet features) while models trained from scratch can learn task-specific but sometimes noisier or more fragmented features. Pre-trained features often lead to higher validation accuracy and clearer, more interpretable visualizations.
- **Relation to accuracy:** Feature visualization can help explain why some models generalize better: models whose late-layer features clearly separate cat- and dog-like patterns tend to have higher validation accuracy, while models with diffuse or noisy late-layer features often perform worse.

In [ ]:

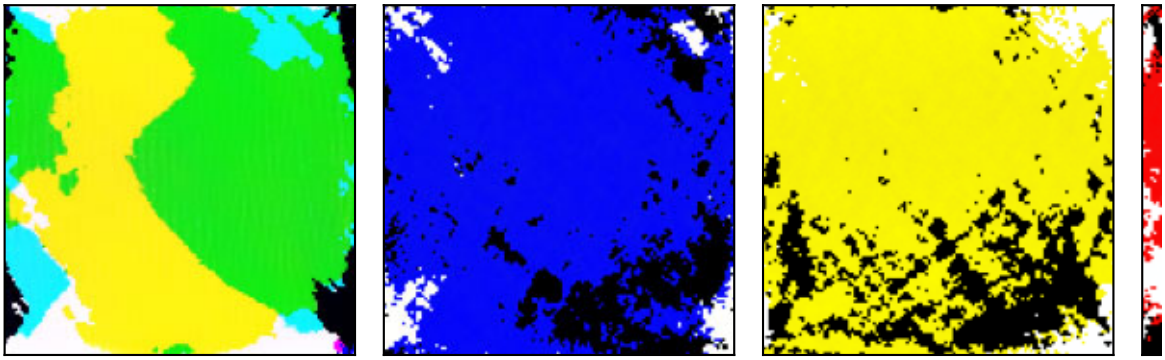
Art!

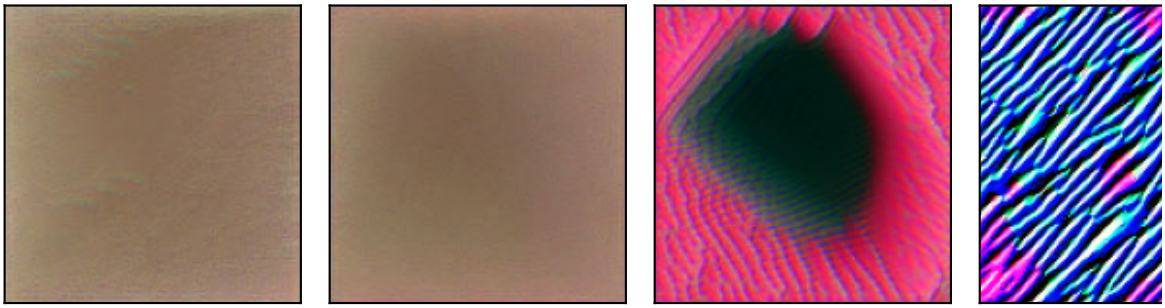
Now explore the feature visualizations of other parts of the network. See if you can find some interesting textures or patterns. What do they resemble?

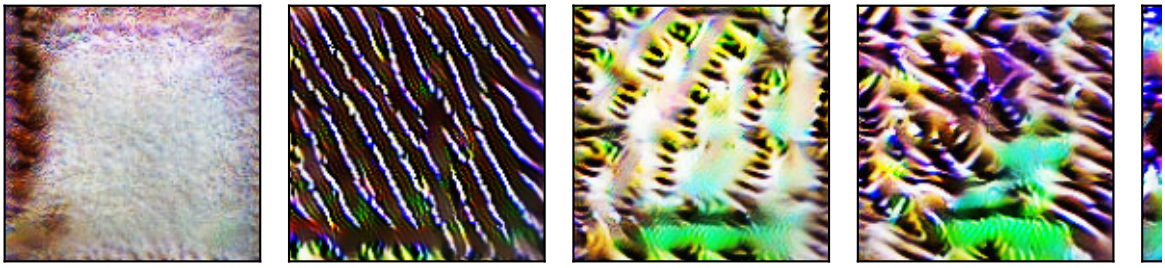
In [25]:
```
# Improved Art! - use lucent.get_model_layers to pick valid layer names and rend
try:
    import lucent.optvis as lu_optvis
    import lucent.modelzoo.util as lu_util
    from lucent.optvis import render as lu_render
    import numpy as np
except Exception as e:
    print("lucent not available or failed import:", e)
    raise

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

def get_valid_layers(model):
    try:
        layers = lu_util.get_model_layers(model)
    except Exception as e:
        print("get_model_layers failed:", e)
        layers = []
    return layers

for model_name, model in all_models.items():
    print(f"\\n--- Art visuals for {model_name} ---")
    model.to(device).eval()
    layers = get_valid_layers(model)
    if not layers:
        print("No lucent layers found for this model. Skipping.")
        continue
    # pick a few candidate layers that look conv-like; otherwise pick some middl
    conv_like = [L for L in layers if 'conv' in L.lower() or 'layer' in L.lower(
    chosen = conv_like if conv_like else layers
    # pick up to 4 evenly spaced layers
    k = min(4, len(chosen))
    chosen_layers = []
```

```
    if k == 1:
        chosen_layers = [chosen[0]]
    else:
        for i in range(k):
            idx = int(round(i * (len(chosen)-1) / (k-1)))
            chosen_layers.append(chosen[idx])
    print("Chosen layers:", chosen_layers)

    # For each chosen layer, pick a small set of channels spaced across a reason
    for lay in chosen_layers:
        # attempt rendering channels 0..5 and also some mid-range channels if la
        channels = list(range(6))
        # also try a few mid indices if renderable
        mid_candidates = [16, 32, 64, 128]
        channels += mid_candidates
        channels = sorted(set([c for c in channels if c >= 0]) )[:8]
        imgs = []
        for ch in channels:
            spec = f"{lay}:{ch}"
            try:
                out = lu_optvis.render.render_vis(model, spec, show_image=False)
                imgs.append(np.squeeze(out[0]))
            except Exception:
                # skip invalid specs silently
                pass
        if imgs:
            plt.figure(figsize=(12,3))
            show_imgs(imgs, nc=len(imgs))
            plt.suptitle(f"{model_name} - layer {lay} channels {channels[:len(im
            plt.show()

# Try class-logit visualizations (0 and 1)
for model_name, model in all_models.items():
    print(f"\\n--- Class-logit for {model_name} ---")
    model.to(device).eval()
    layers = get_valid_layers(model)
    # try common spec forms
    for ci in [0,1]:
        for spec in [f"output:{ci}", f"logit:{ci}", f"prob:{ci}"]:
            try:
                out = lu_optvis.render.render_vis(model, spec, show_image=False)
                plt.figure(figsize=(4,4))
                plt.imshow(np.squeeze(out[0]))
                plt.title(f"{model_name} - {spec}")
                plt.axis('off')
                plt.show()
                break
            except Exception:
                continue
```

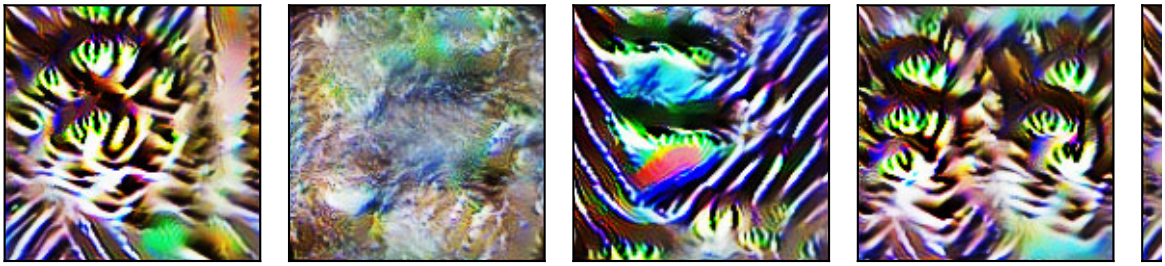
```
\n--- Art visuals for CNN5 ---
Chosen layers: ['conv1', 'conv2', 'conv4', 'conv5']
100%|██████████| 512/512 [00:05<00:00, 95.59it/s]
100%|██████████| 512/512 [00:05<00:00, 94.12it/s]
100%|██████████| 512/512 [00:05<00:00, 96.84it/s]
100%|██████████| 512/512 [00:05<00:00, 101.44it/s]
100%|██████████| 512/512 [00:05<00:00, 101.54it/s]
100%|██████████| 512/512 [00:05<00:00, 102.18it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```





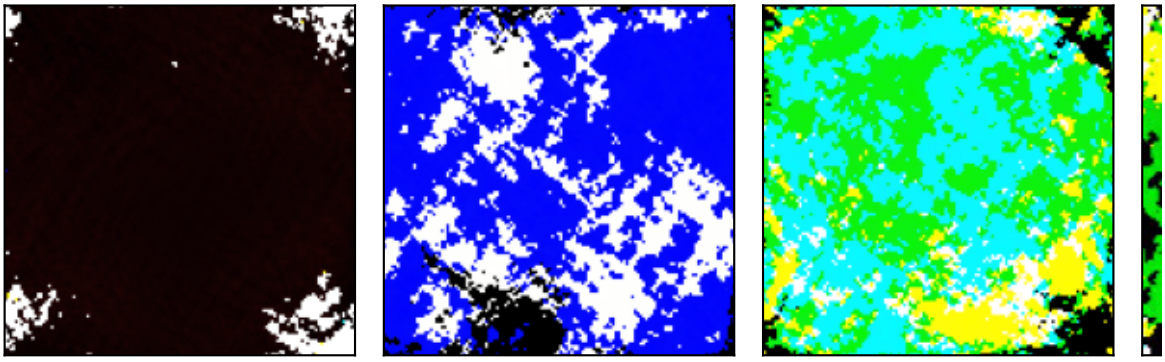


```
100%|██████████| 512/512 [00:06<00:00, 76.64it/s]
100%|██████████| 512/512 [00:06<00:00, 76.81it/s]
100%|██████████| 512/512 [00:07<00:00, 70.69it/s]
100%|██████████| 512/512 [00:06<00:00, 75.87it/s]
100%|██████████| 512/512 [00:07<00:00, 72.89it/s]
100%|██████████| 512/512 [00:06<00:00, 76.34it/s]
100%|██████████| 512/512 [00:06<00:00, 73.68it/s]
100%|██████████| 512/512 [00:06<00:00, 82.61it/s]
```



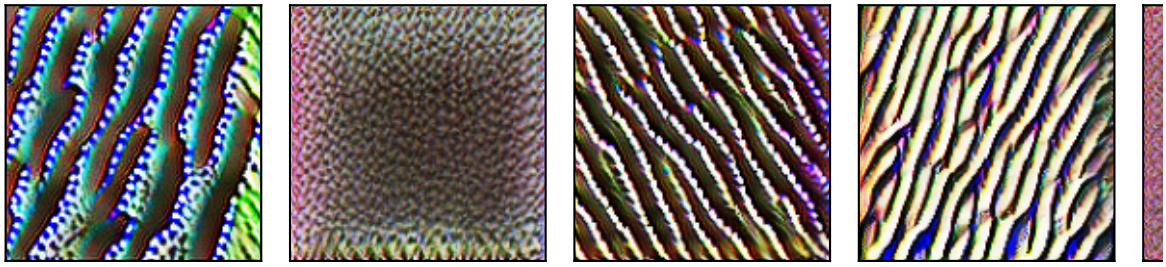
<Figure size 640x480 with 0 Axes>
\n--- Art visuals for CNN7 ---
Chosen layers: ['conv1', 'conv3', 'conv5', 'conv7']

```
100%|██████████| 512/512 [00:05<00:00, 95.14it/s]
100%|██████████| 512/512 [00:05<00:00, 87.62it/s]
100%|██████████| 512/512 [00:05<00:00, 95.01it/s]
100%|██████████| 512/512 [00:05<00:00, 95.15it/s]
100%|██████████| 512/512 [00:05<00:00, 91.77it/s]
100%|██████████| 512/512 [00:05<00:00, 93.92it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```



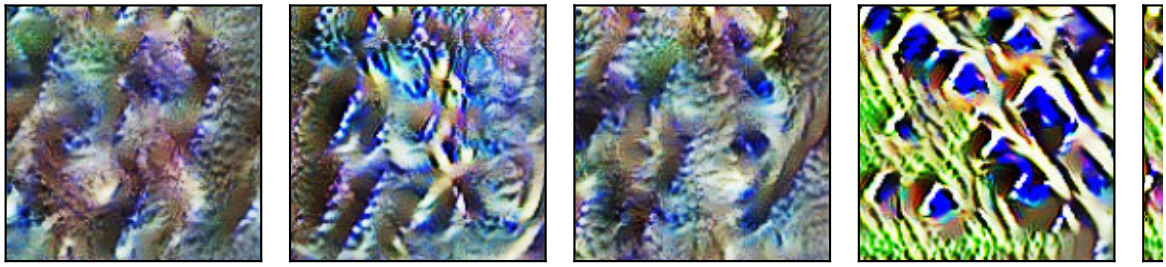
<Figure size 640x480 with 0 Axes>

```
100%|██████████| 512/512 [00:05<00:00, 86.48it/s]
100%|██████████| 512/512 [00:05<00:00, 86.65it/s]
100%|██████████| 512/512 [00:05<00:00, 87.71it/s]
100%|██████████| 512/512 [00:06<00:00, 84.03it/s]
100%|██████████| 512/512 [00:05<00:00, 86.40it/s]
100%|██████████| 512/512 [00:06<00:00, 82.42it/s]
100%|██████████| 512/512 [00:06<00:00, 83.65it/s]
100%|██████████| 512/512 [00:06<00:00, 85.13it/s]
```



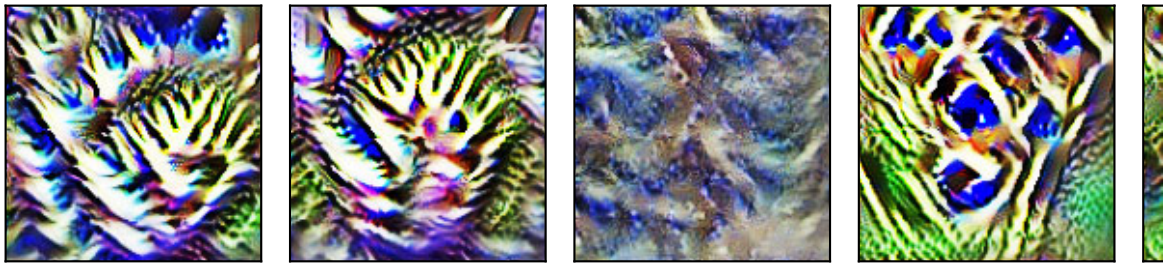
```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:06<00:00, 79.80it/s]
100%|██████████| 512/512 [00:06<00:00, 82.35it/s]
100%|██████████| 512/512 [00:05<00:00, 87.93it/s]
100%|██████████| 512/512 [00:06<00:00, 83.05it/s]
100%|██████████| 512/512 [00:06<00:00, 83.59it/s]
100%|██████████| 512/512 [00:06<00:00, 84.49it/s]
100%|██████████| 512/512 [00:06<00:00, 81.37it/s]
100%|██████████| 512/512 [00:06<00:00, 77.75it/s]
```



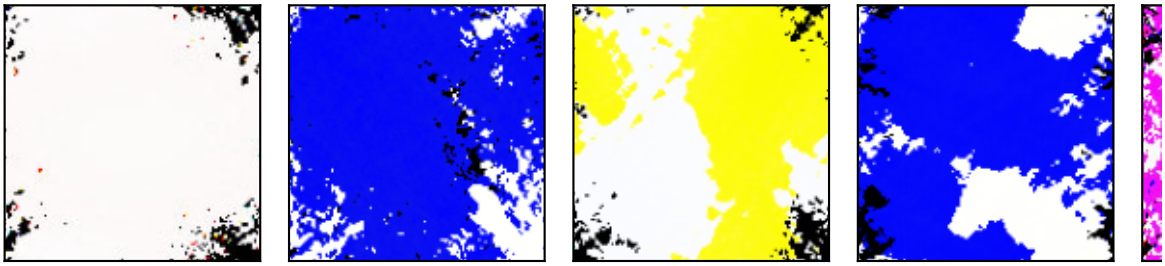
```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:06<00:00, 83.49it/s]
100%|██████████| 512/512 [00:06<00:00, 82.44it/s]
100%|██████████| 512/512 [00:06<00:00, 79.34it/s]
100%|██████████| 512/512 [00:06<00:00, 75.42it/s]
100%|██████████| 512/512 [00:06<00:00, 78.65it/s]
100%|██████████| 512/512 [00:06<00:00, 79.95it/s]
100%|██████████| 512/512 [00:06<00:00, 78.55it/s]
100%|██████████| 512/512 [00:06<00:00, 78.74it/s]
```



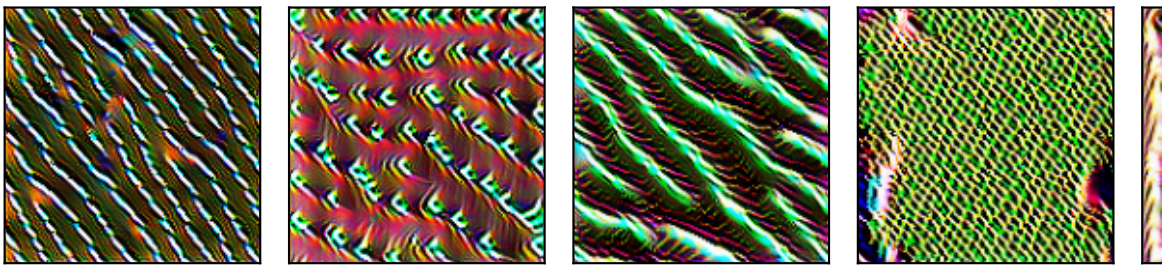
```
<Figure size 640x480 with 0 Axes>
\n--- Art visuals for ResNet18 ---
Chosen layers: ['conv1', 'layer2_0_bn2', 'layer3_1_conv1', 'layer4_1_bn2']
```

```
100%|██████████| 512/512 [00:10<00:00, 48.95it/s]
100%|██████████| 512/512 [00:11<00:00, 46.23it/s]
100%|██████████| 512/512 [00:10<00:00, 48.09it/s]
100%|██████████| 512/512 [00:10<00:00, 48.00it/s]
100%|██████████| 512/512 [00:10<00:00, 50.97it/s]
100%|██████████| 512/512 [00:11<00:00, 44.71it/s]
100%|██████████| 512/512 [00:10<00:00, 47.62it/s]
100%|██████████| 512/512 [00:11<00:00, 46.32it/s]
```



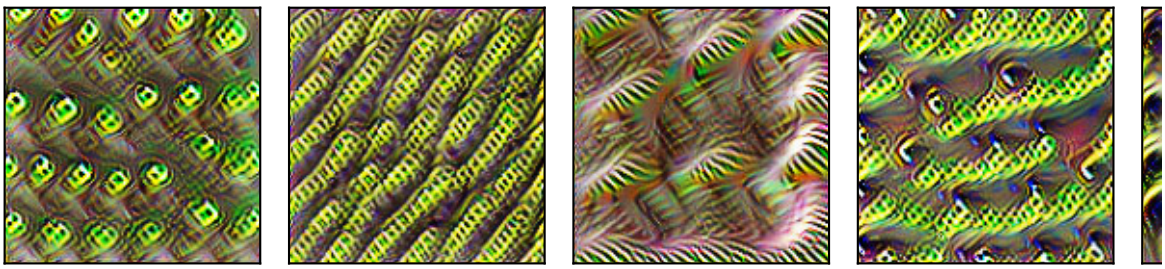
<Figure size 640x480 with 0 Axes>

```
100%|██████████| 512/512 [00:11<00:00, 42.86it/s]
100%|██████████| 512/512 [00:12<00:00, 40.82it/s]
100%|██████████| 512/512 [00:12<00:00, 42.04it/s]
100%|██████████| 512/512 [00:12<00:00, 42.18it/s]
100%|██████████| 512/512 [00:12<00:00, 42.16it/s]
100%|██████████| 512/512 [00:12<00:00, 42.21it/s]
100%|██████████| 512/512 [00:12<00:00, 41.89it/s]
100%|██████████| 512/512 [00:12<00:00, 41.45it/s]
```



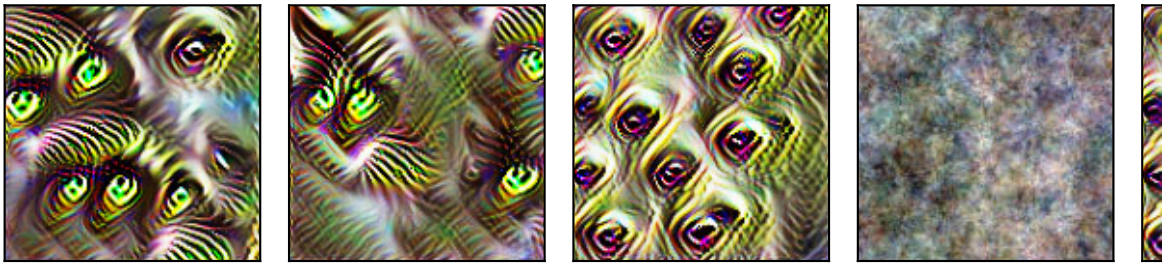
<Figure size 640x480 with 0 Axes>

```
100%|██████████| 512/512 [00:13<00:00, 37.31it/s]
100%|██████████| 512/512 [00:13<00:00, 36.73it/s]
100%|██████████| 512/512 [00:13<00:00, 37.46it/s]
100%|██████████| 512/512 [00:13<00:00, 37.45it/s]
100%|██████████| 512/512 [00:13<00:00, 36.98it/s]
100%|██████████| 512/512 [00:13<00:00, 36.86it/s]
100%|██████████| 512/512 [00:13<00:00, 36.97it/s]
100%|██████████| 512/512 [00:13<00:00, 37.16it/s]
```



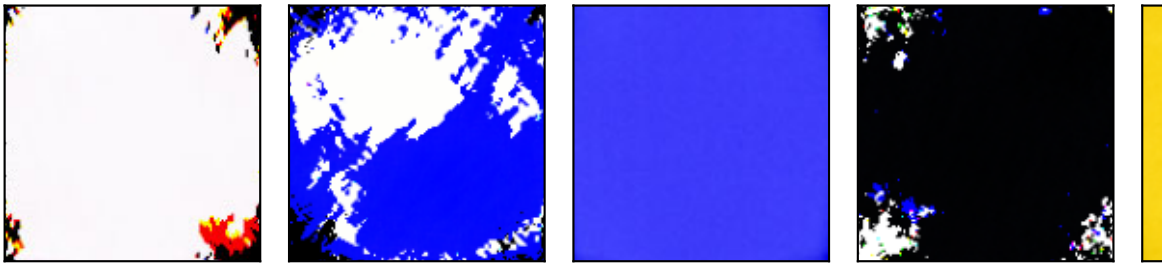
```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:14<00:00, 34.41it/s]
100%|██████████| 512/512 [00:14<00:00, 35.85it/s]
100%|██████████| 512/512 [00:14<00:00, 34.94it/s]
100%|██████████| 512/512 [00:14<00:00, 34.32it/s]
100%|██████████| 512/512 [00:15<00:00, 33.83it/s]
100%|██████████| 512/512 [00:15<00:00, 33.38it/s]
100%|██████████| 512/512 [00:14<00:00, 34.45it/s]
100%|██████████| 512/512 [00:15<00:00, 33.28it/s]
```



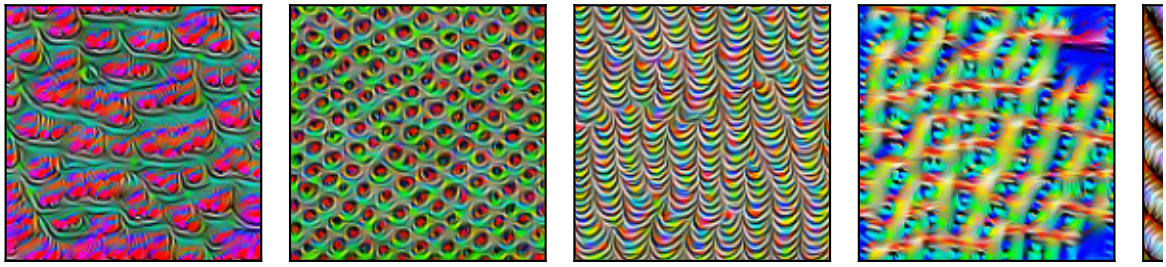
```
<Figure size 640x480 with 0 Axes>
\n--- Art visuals for ResNet18p ---
Chosen layers: ['conv1', 'layer2_0_bn2', 'layer3_1_conv1', 'layer4_1_bn2']
```

```
100%|██████████| 512/512 [00:10<00:00, 50.07it/s]
100%|██████████| 512/512 [00:10<00:00, 48.89it/s]
100%|██████████| 512/512 [00:09<00:00, 53.38it/s]
100%|██████████| 512/512 [00:09<00:00, 54.88it/s]
100%|██████████| 512/512 [00:08<00:00, 57.01it/s]
100%|██████████| 512/512 [00:09<00:00, 54.77it/s]
100%|██████████| 512/512 [00:09<00:00, 55.06it/s]
100%|██████████| 512/512 [00:09<00:00, 56.74it/s]
```



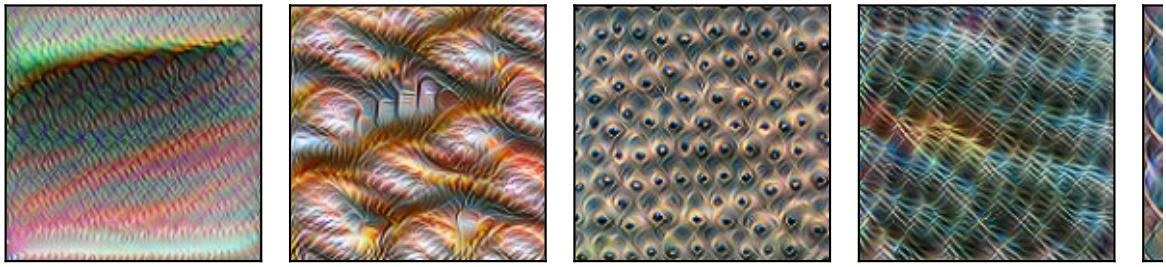
```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:09<00:00, 51.27it/s]
100%|██████████| 512/512 [00:10<00:00, 49.31it/s]
100%|██████████| 512/512 [00:10<00:00, 50.12it/s]
100%|██████████| 512/512 [00:10<00:00, 49.82it/s]
100%|██████████| 512/512 [00:10<00:00, 50.72it/s]
100%|██████████| 512/512 [00:10<00:00, 48.72it/s]
100%|██████████| 512/512 [00:10<00:00, 49.17it/s]
100%|██████████| 512/512 [00:10<00:00, 48.64it/s]
```



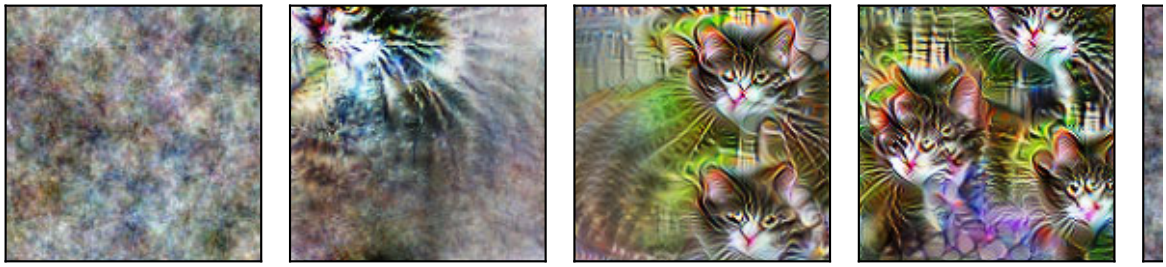
```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:10<00:00, 49.39it/s]
100%|██████████| 512/512 [00:10<00:00, 50.01it/s]
100%|██████████| 512/512 [00:10<00:00, 47.72it/s]
100%|██████████| 512/512 [00:10<00:00, 46.63it/s]
100%|██████████| 512/512 [00:11<00:00, 45.48it/s]
100%|██████████| 512/512 [00:11<00:00, 45.96it/s]
100%|██████████| 512/512 [00:11<00:00, 45.00it/s]
100%|██████████| 512/512 [00:11<00:00, 43.05it/s]
```



```
<Figure size 640x480 with 0 Axes>
```

```
100%|██████████| 512/512 [00:12<00:00, 41.12it/s]
100%|██████████| 512/512 [00:12<00:00, 40.65it/s]
100%|██████████| 512/512 [00:12<00:00, 41.35it/s]
100%|██████████| 512/512 [00:12<00:00, 40.13it/s]
100%|██████████| 512/512 [00:12<00:00, 39.85it/s]
100%|██████████| 512/512 [00:12<00:00, 41.95it/s]
100%|██████████| 512/512 [00:12<00:00, 41.74it/s]
100%|██████████| 512/512 [00:12<00:00, 39.57it/s]
```



```
<Figure size 640x480 with 0 Axes>
\n--- Class-logit for CNN5 ---
```

```
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```

```
\n--- Class-logit for CNN7 ---
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
\n--- Class-logit for ResNet18 ---
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
\n--- Class-logit for ResNet18p ---
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: