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).

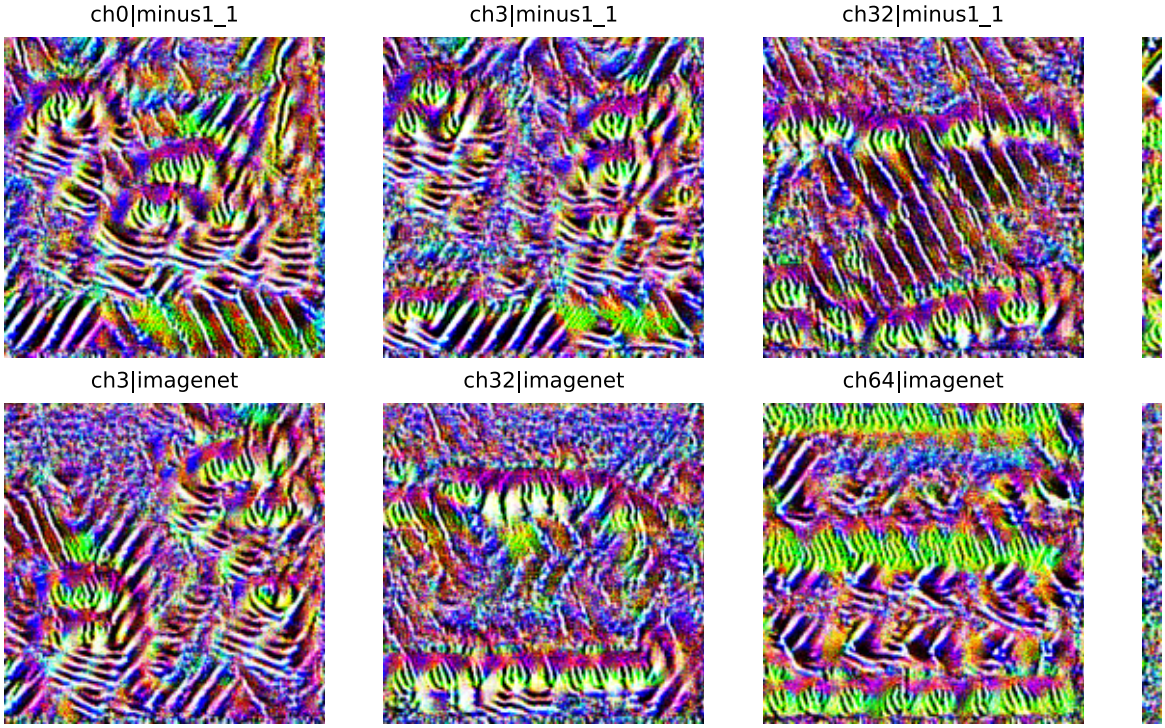
In [18]:
```
# --- 3.1 Feature visualization (self-contained implementation) ---
# This cell provides helper functions and synthesizes images to maximize selecte
# from the last Conv2d layer of each model using gradient ascent on the input im
%matplotlib inline
import matplotlib_inline
matplotlib_inline.backend_inline.set_matplotlib_formats('svg')
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100

import torch, torch.nn as nn, torch.nn.functional as F
import numpy as np, random
from PIL import Image, ImageFilter

# ---------- Helpers ----------
def tv_loss(img):
    x_diff = img[:,:,:,1:] - img[:,:,:,:-1]
    y_diff = img[:,:,1:,:] - img[:,:,:-1,:]
    return (x_diff.abs().mean() + y_diff.abs().mean())

def pil_blur_tensor(img_tensor, radius=0.5):
    arr = img_tensor.detach().cpu().squeeze(0).numpy()
    arr = np.transpose(arr, (1,2,0)) * 255.0
    img = Image.fromarray(np.clip(arr,0,255).astype('uint8'))
    img = img.filter(ImageFilter.GaussianBlur(radius=radius))
    arr = np.asarray(img).astype(np.float32) / 255.0
    arr = np.transpose(arr, (2,0,1))
```

```
    return torch.from_numpy(arr).unsqueeze(0).float().to(img_tensor.device)

IMAGENET_MEAN = torch.tensor([0.485,0.456,0.406]).view(1,3,1,1)
IMAGENET_STD  = torch.tensor([0.229,0.224,0.225]).view(1,3,1,1)

def preprocess_for_model(x, normalization='minus1_1', device='cpu'):
    x = x.to(device)
    if normalization == 'imagenet':
        return (x - IMAGENET_MEAN.to(device)) / IMAGENET_STD.to(device)
    elif normalization == 'minus1_1':
        return x * 2.0 - 1.0
    else:
        return x

def deprocess_img(x, normalization='minus1_1'):
    x = x.detach().cpu().clone()
    if normalization == 'imagenet':
        x = x * IMAGENET_STD + IMAGENET_MEAN
    elif normalization == 'minus1_1':
        x = (x + 1.0) / 2.0
    x = x.clamp(0,1)
    return x

def find_last_conv_module(model):
    last_name = None
    last_mod = None
    for name, mod in model.named_modules():
        if isinstance(mod, nn.Conv2d):
            last_name = name
            last_mod = mod
    return last_name, last_mod

def show_imgs(W_list, nc=8, titles=None):
    import matplotlib.pyplot as plt, numpy as np
    n = len(W_list)
    if n == 0:
        print("No images to show.")
        return
    nr = (n - 1) // nc + 1
    plt.figure(figsize=(nc*2, nr*2))
    for i, img in enumerate(W_list):
        plt.subplot(nr, nc, i+1)
        if isinstance(img, torch.Tensor):
            img = img.detach().cpu().numpy()
            if img.shape[0] == 3:
                img = np.transpose(img, (1,2,0))
        img = np.clip(img, 0, 1)
        plt.imshow(img)
        plt.axis('off')
        if titles is not None and i < len(titles):
            plt.title(titles[i], fontsize=8)
    plt.tight_layout()
    plt.show()

# ---------- Synthesis functions ----------
def synthesize_channel(model, target_module, channel_idx, img_size=(224,224), st
                       tv_weight=1e-4, l2_weight=1e-6, blur_every=30, blur_radiu
                       jitter=8, normalization='minus1_1', device=None, show_prc
    if device is None:
        device = next(model.parameters()).device if any(True for _ in model.para
```

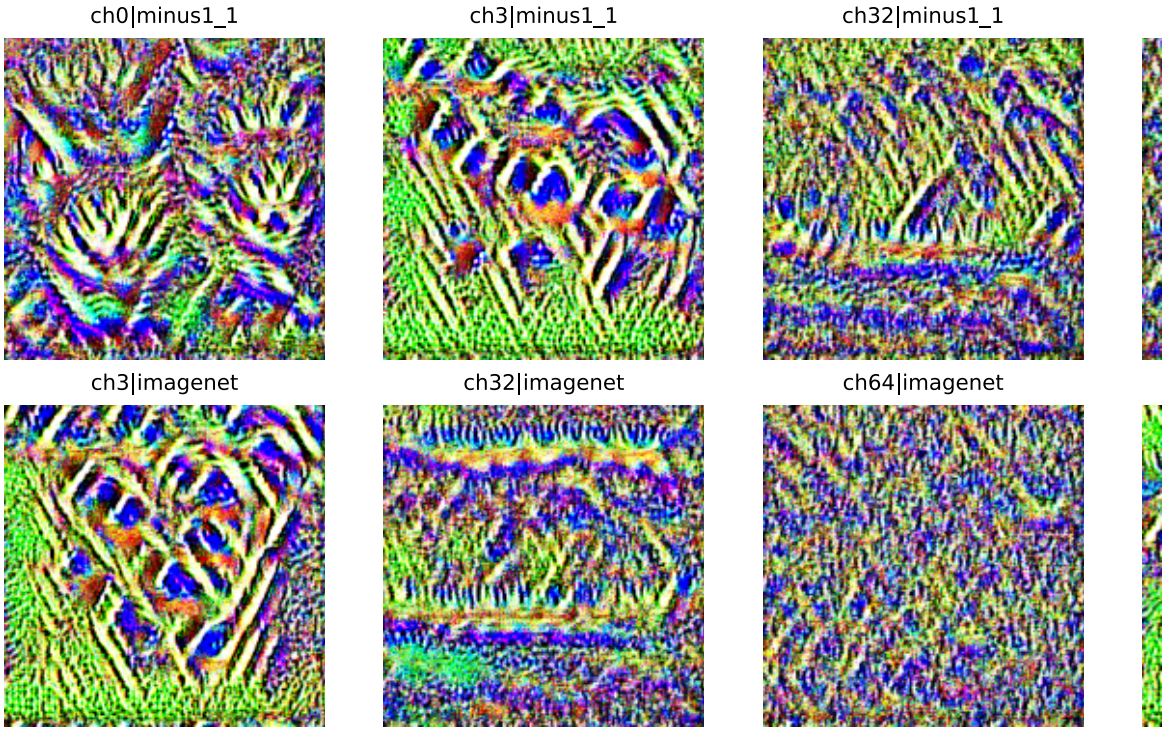
```
    if seed is not None:
        torch.manual_seed(seed); np.random.seed(seed); random.seed(seed)
    model.eval()
    H,W = img_size
    img = torch.randn(1,3,H,W, device=device) * 0.08 + 0.5
    img = img.clamp(0,1)
    img.requires_grad_(True)
    optimizer = torch.optim.Adam([img], lr=lr)
    activations = {}
    def hook_fn(module, inp, out):
        activations['val'] = out
    handle = target_module.register_forward_hook(hook_fn)
    try:
        for i in range(steps):
            optimizer.zero_grad()
            ox = random.randint(-jitter, jitter); oy = random.randint(-jitter, j
            img_j = torch.roll(img, shifts=(ox, oy), dims=(2,3)) if (ox or oy) e
            inp = preprocess_for_model(img_j, normalization=normalization, devic
            out = model(inp)
            act = activations.get('val', None)
            if act is None:
                # no activation captured; abort early
                break
            if channel_idx is None:
                score = act.pow(2).mean()
            else:
                if channel_idx >= act.shape[1]:
                    score = torch.tensor(0.0, device=device)
                else:
                    score = act[:, channel_idx].mean()
            loss = -score + tv_weight * tv_loss(img) + l2_weight * (img.norm()**
            loss.backward()
            optimizer.step()
            with torch.no_grad(): img.clamp_(0,1)
            if blur_every and (i % blur_every == 0) and i>0:
                with torch.no_grad():
                    img = pil_blur_tensor(img, radius=blur_radius).to(device)
                    img.requires_grad_(True)
                    optimizer = torch.optim.Adam([img], lr=lr)
        out_img = deprocess_img(preprocess_for_model(img, normalization=normaliz
        return out_img.detach().cpu()
    finally:
        try: handle.remove()
        except Exception: pass

def synthesize_class(model, class_idx, img_size=(224,224), steps=220, lr=0.14,
                     tv_weight=1e-4, l2_weight=1e-6, blur_every=30, blur_radius=
                     jitter=8, normalization='minus1_1', device=None, show_progr
    if device is None:
        device = next(model.parameters()).device if any(True for _ in model.para
    if seed is not None:
        torch.manual_seed(seed); np.random.seed(seed); random.seed(seed)
    model.eval()
    H,W = img_size
    img = torch.randn(1,3,H,W, device=device) * 0.08 + 0.5
    img = img.clamp(0,1); img.requires_grad_(True)
    optimizer = torch.optim.Adam([img], lr=lr)
    for i in range(steps):
        optimizer.zero_grad()
        ox = random.randint(-jitter, jitter); oy = random.randint(-jitter, jitte
```

```
        img_j = torch.roll(img, shifts=(ox, oy), dims=(2,3)) if (ox or oy) else
        inp = preprocess_for_model(img_j, normalization=normalization, device=de
        out = model(inp)
        logits = out[0] if isinstance(out, (list,tuple)) else out
        if class_idx >= logits.shape[1]:
            score = torch.tensor(0.0, device=logits.device)
        else:
            score = logits[:, class_idx].mean()
        loss = -score + tv_weight * tv_loss(img) + l2_weight * (img.norm()**2)
        loss.backward()
        optimizer.step()
        with torch.no_grad(): img.clamp_(0,1)
        if blur_every and (i % blur_every == 0) and i>0:
            with torch.no_grad():
                img = pil_blur_tensor(img, radius=blur_radius).to(device)
                img.requires_grad_(True)
                optimizer = torch.optim.Adam([img], lr=lr)
    out_img = deprocess_img(preprocess_for_model(img, normalization=normalizatic
    return out_img.detach().cpu()

# ---------- Runner: synthesize for last conv layer per model ----------
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print("Device for synthesis:", device)

for model_name, model in all_models.items():
    print(f"\n=== Model: {model_name} ===")
    model.to(device).eval()
    lname, lmod = find_last_conv_module(model)
    if lmod is None:
        print("No Conv2d found for", model_name)
        continue
    print("Last conv module:", lname, "out_channels:", getattr(lmod, 'out_channe
    ch_count = getattr(lmod, 'out_channels', 128)
    # choose representative channels
    chosen = [0, min(3, ch_count-1), max(0, ch_count//4), max(0, ch_count//2), m
    chosen = sorted(list(dict.fromkeys([c for c in chosen if c>=0])))[:6]
    imgs = []; titles = []
    for norm in ['minus1_1','imagenet']:
        for ch in chosen:
            try:
                im = synthesize_channel(model, lmod, ch, img_size=(224,224), ste
                                        tv_weight=1e-4, l2_weight=1e-6, blur_eve
                                        jitter=6, normalization=norm, device=dev
                imgs.append(im.squeeze(0)); titles.append(f"ch{ch}|{norm}")
            except Exception as e:
                print("Channel synth failed:", e)
    if imgs:
        show_imgs(imgs, nc=6, titles=titles)
    else:
        print("No images produced for", model_name)
```

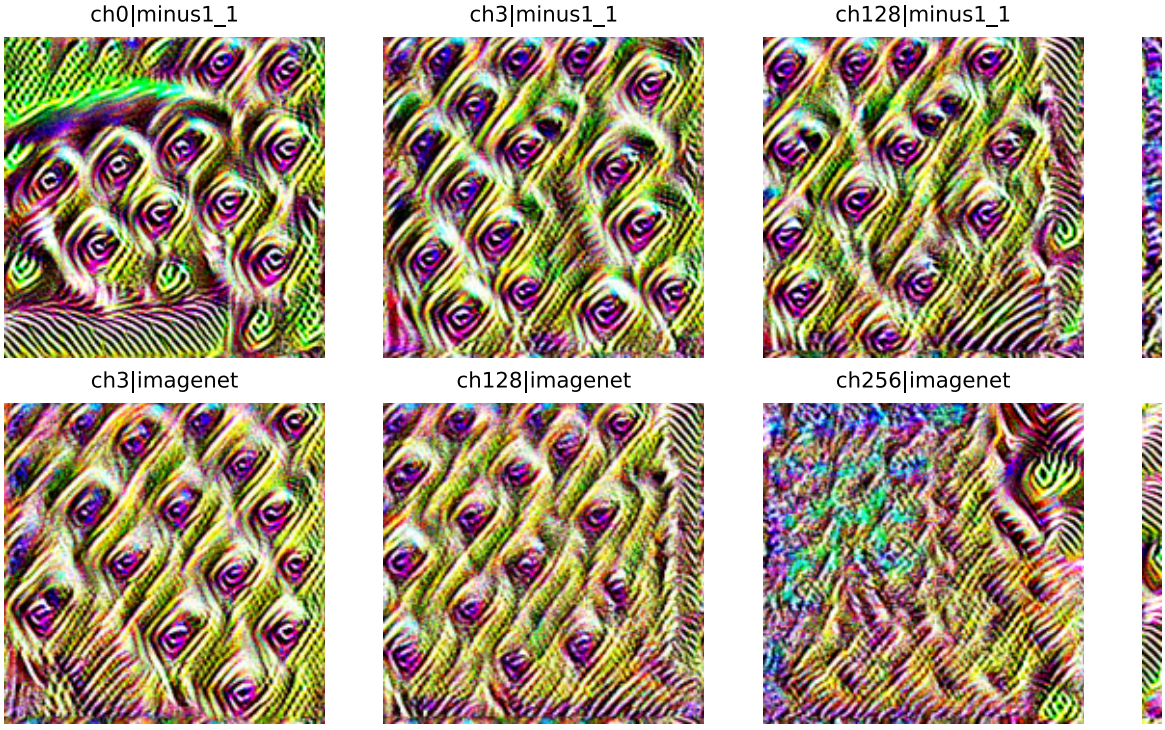
```
Device for synthesis: cuda:0

=== Model: CNN5 ===
Last conv module: conv5 out_channels: 128
```



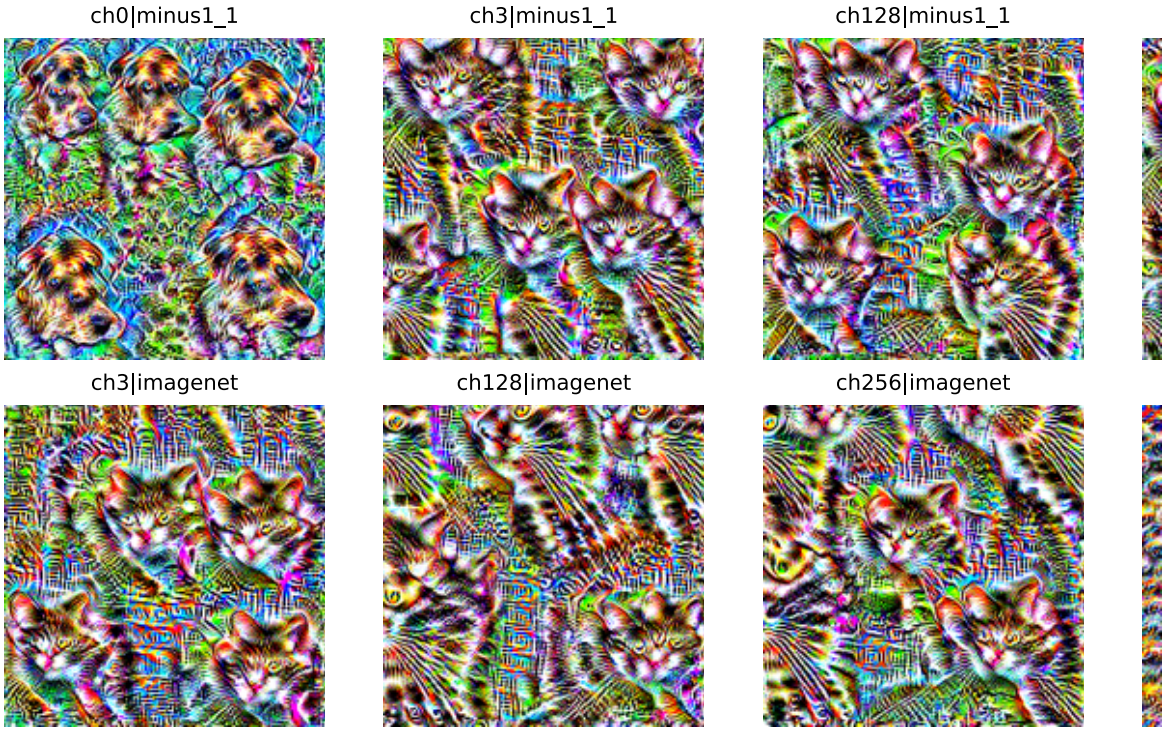

```
=== Model: CNN7 ===
Last conv module: conv7 out_channels: 128
```




```
=== Model: ResNet18 ===
Last conv module: layer4.1.conv2 out_channels: 512
```




```
=== Model: ResNet18p ===
Last conv module: layer4.1.conv2 out_channels: 512
```



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?

- **INSERT YOUR ANSWER HERE**
- ...
- ...

In [ ]:

Art!

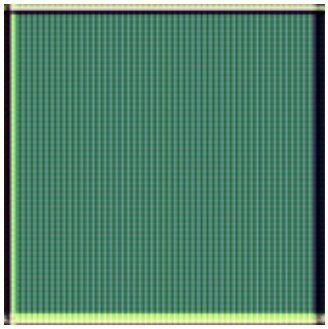
Now explore the feature visualizations of other parts of the network. See if you can find some interesting textures or patterns. What do they resemble?

In [19]:
```
# --- 4. Art! exploration (self-contained runner uses functions defined above) -
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print("Device:", device)

for model_name, model in all_models.items():
    print(f"\n--- Art visuals for {model_name} ---")
    model.to(device).eval()
    conv_names = [n for n,m in model.named_modules() if isinstance(m, torch.nn.C
    if not conv_names:
        print("No conv modules found for", model_name)
        continue
    k = min(4, len(conv_names))
    indices = [int(round(i * (len(conv_names)-1) / (k-1))) if k>1 else 0 for i i
    chosen_layers = [conv_names[i] for i in indices]
    print("Chosen layers:", chosen_layers)
    for lname in chosen_layers:
        module = None
```

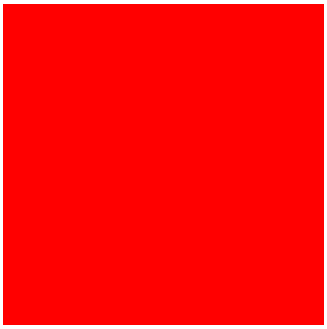
```
        for n,m in model.named_modules():
            if n == lname:
                module = m; break
        if module is None: continue
        ch_count = getattr(module, 'out_channels', 64)
        channels = [0, min(3, ch_count-1), max(1, ch_count//4), max(1, ch_count/
        channels = sorted(list(dict.fromkeys([c for c in channels if c>=0])))[:6
        imgs = []; titles = []
        for norm in ['minus1_1','imagenet']:
            for ch in channels:
                try:
                    im = synthesize_channel(model, module, ch, img_size=(224,224
                                            tv_weight=2e-4, l2_weight=1e-6, blur
                                            jitter=6, normalization=norm, device
                    imgs.append(im.squeeze(0)); titles.append(f"{lname}|ch{ch}|{
                except Exception:
                    pass
        if imgs:
            show_imgs(imgs, nc=6, titles=titles)

    # class logit synthesis
    imgs = []; titles = []
    for norm in ['minus1_1','imagenet']:
        for ci in [0,1]:
            try:
                im = synthesize_class(model, ci, img_size=(224,224), steps=220,
                                      tv_weight=1e-4, l2_weight=1e-6, blur_every
                                      jitter=8, normalization=norm, device=devic
                imgs.append(im.squeeze(0)); titles.append(f"class{ci}|{norm}")
            except Exception:
                pass
    if imgs:
        show_imgs(imgs, nc=6, titles=titles)
```

```
Device: cuda:0

--- Art visuals for CNN5 ---
Chosen layers: ['conv1', 'conv2', 'conv4', 'conv5']
```

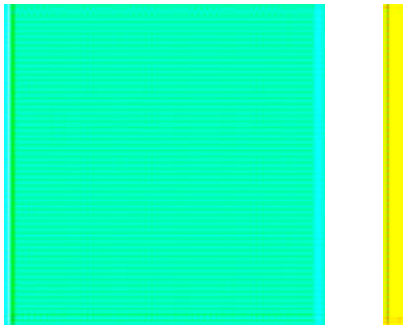
conv1|ch0|minus1_1



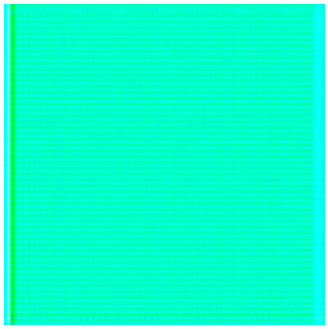
conv1|ch3|minus1_1



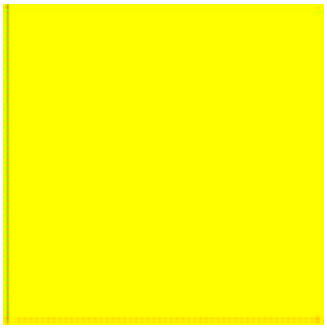
conv1|ch4|minus1_1



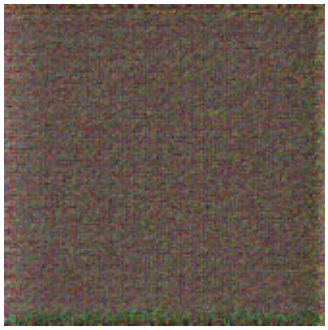
conv1|ch4|imagenet



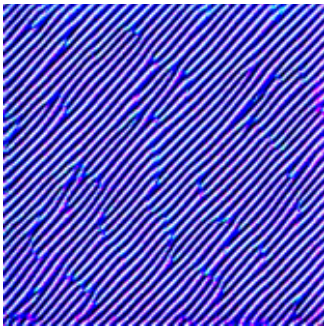
conv1|ch8|imagenet



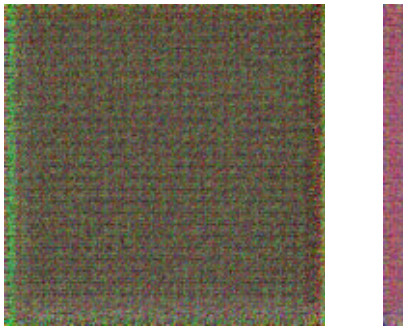
conv2|ch0|minus1_1



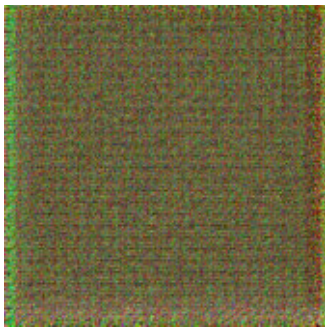
conv2|ch3|minus1_1



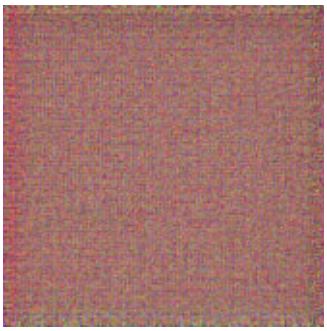
conv2|ch8|minus1_1



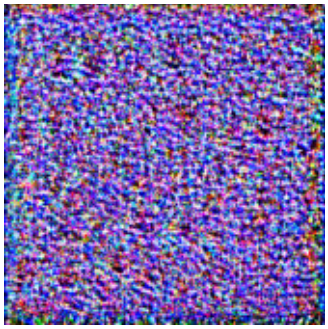
conv2|ch8|imagenet



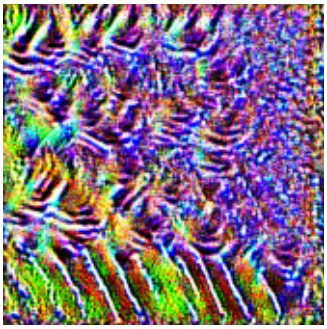
conv2|ch16|imagenet



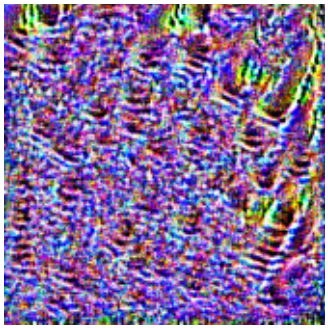
conv4|ch0|minus1_1



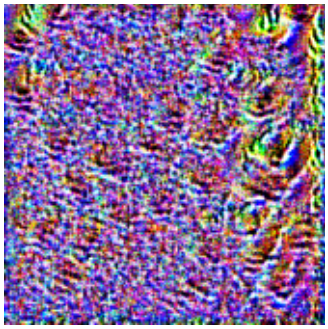
conv4|ch3|minus1_1



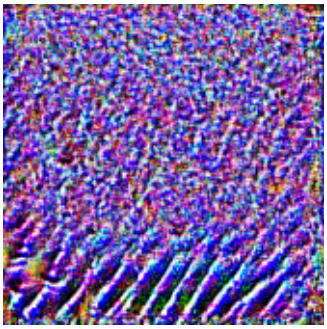
conv4|ch32|minus1_1



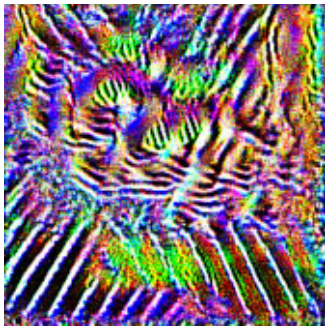
conv4|ch32|imagenet



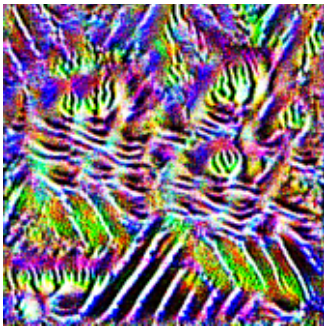
conv4|ch64|imagenet



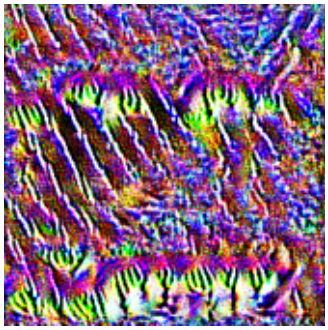
conv5|ch0|minus1_1



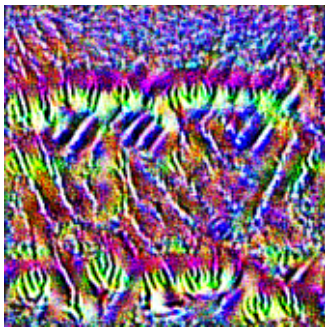
conv5|ch3|minus1_1



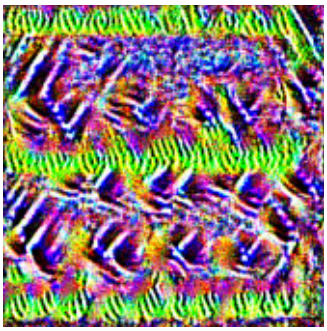
conv5|ch32|minus1_1



conv5|ch32|imagenet

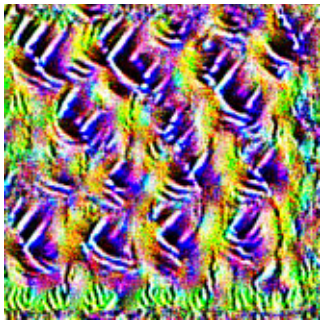


conv5|ch64|imagenet

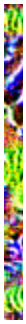


class0|minus1_1

class1|minus1_1

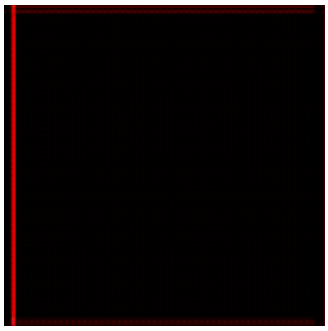


class0|imagenet

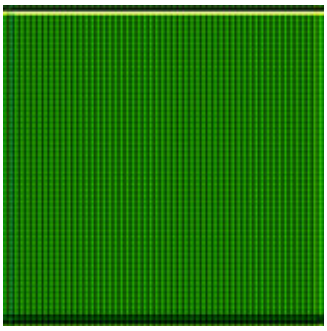


```
--- Art visuals for CNN7 ---
Chosen layers: ['conv1', 'conv3', 'conv5', 'conv7']
```

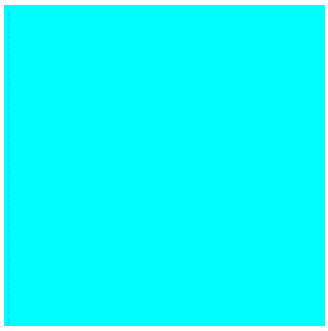
conv1|ch0|minus1_1

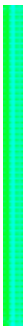


conv1|ch3|minus1_1

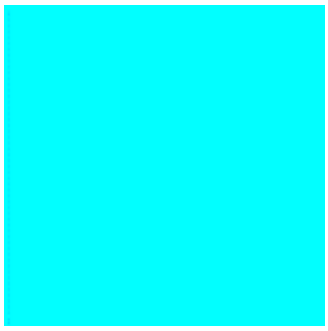


conv1|ch4|minus1_1

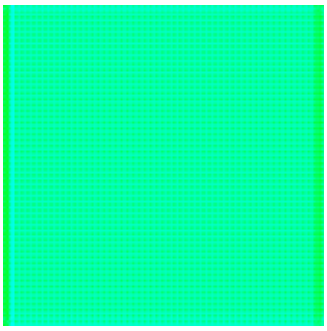




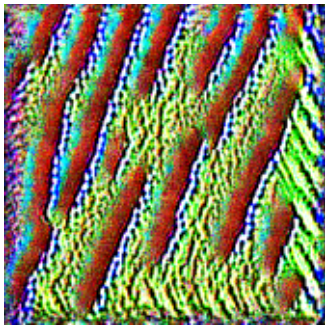
conv1|ch4|imagenet



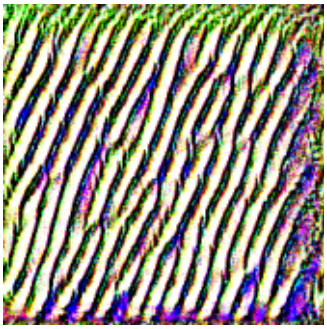
conv1|ch8|imagenet



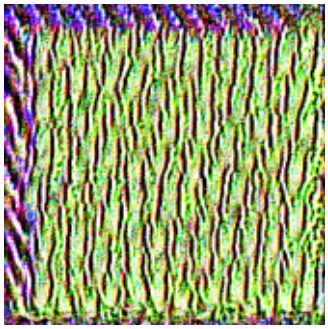
conv3|ch0|minus1_1



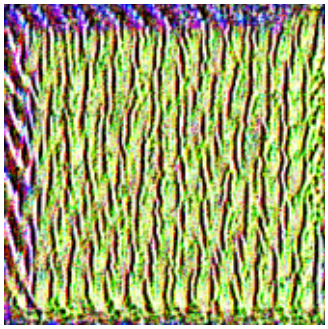
conv3|ch3|minus1_1



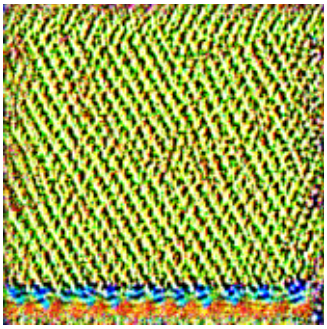
conv3|ch16|minus1_1



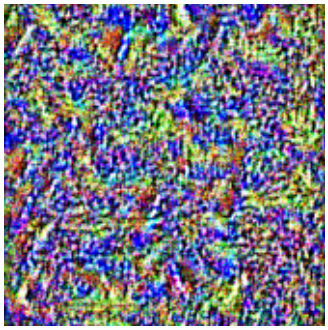
conv3|ch16|imagenet



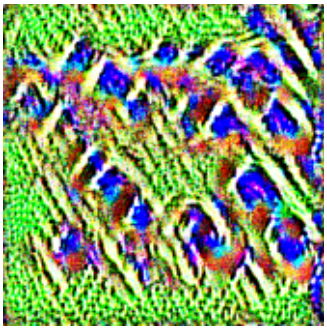
conv3|ch32|imagenet



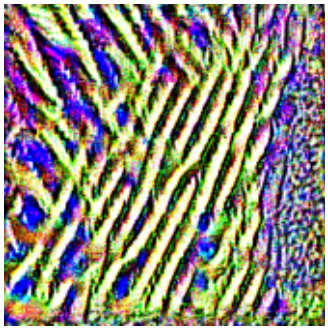
conv5|ch0|minus1_1



conv5|ch3|minus1_1

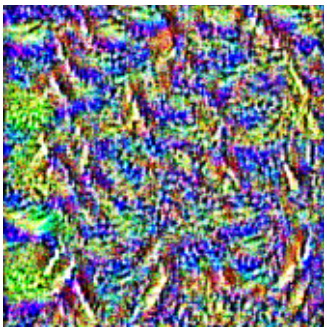


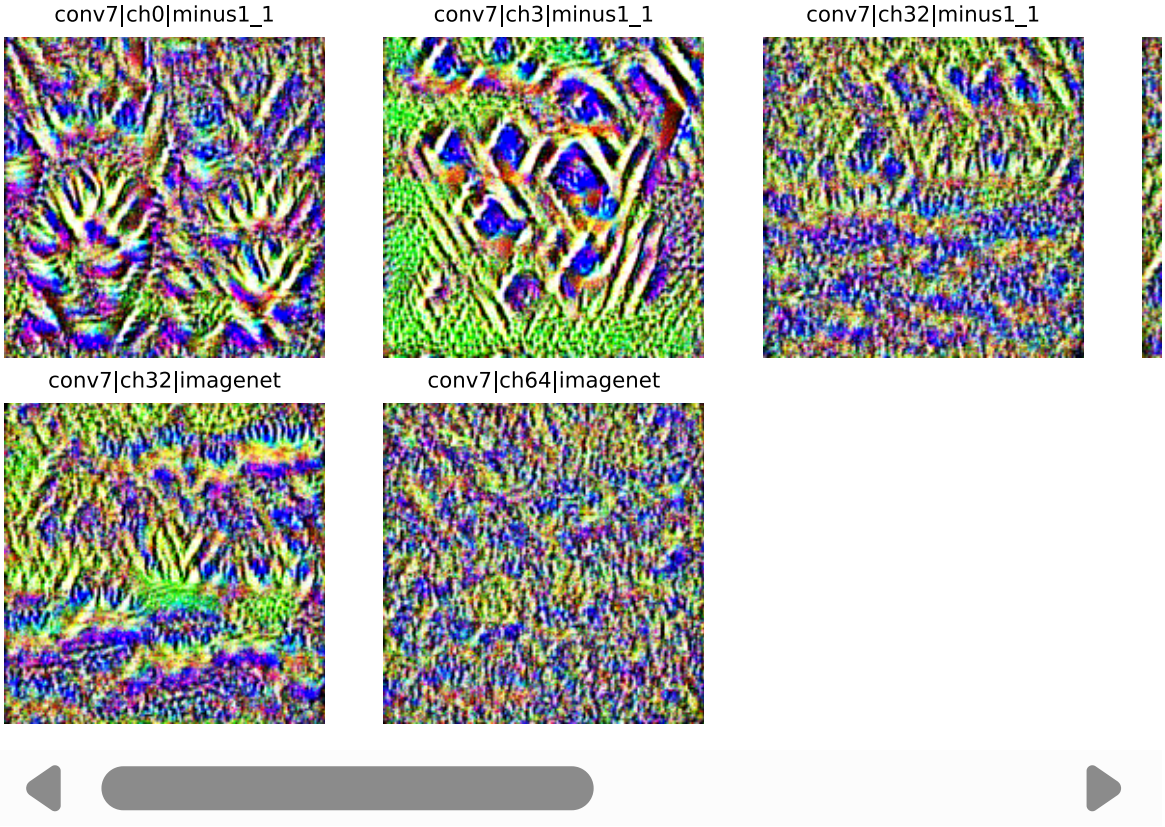
conv5|ch32|minus1_1

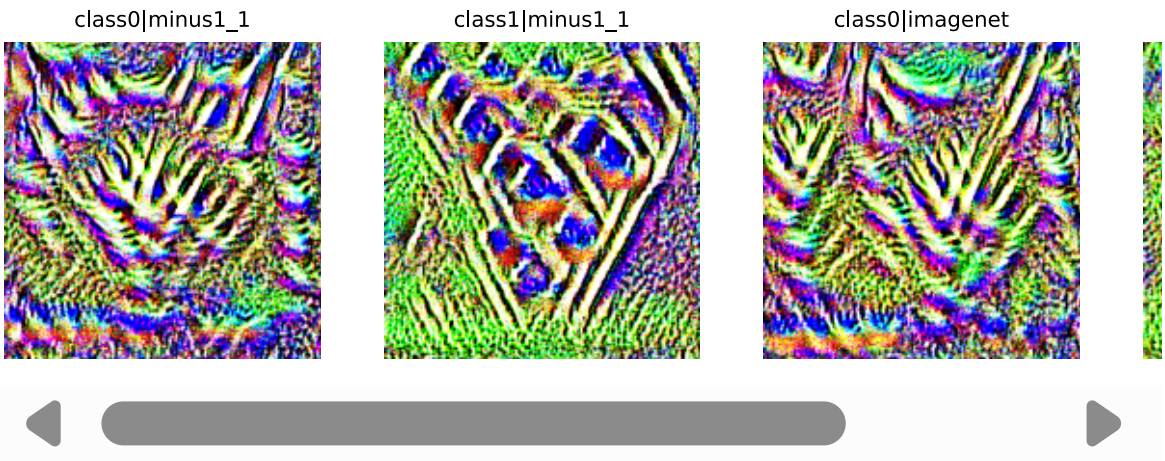


conv5|ch32|imagenet

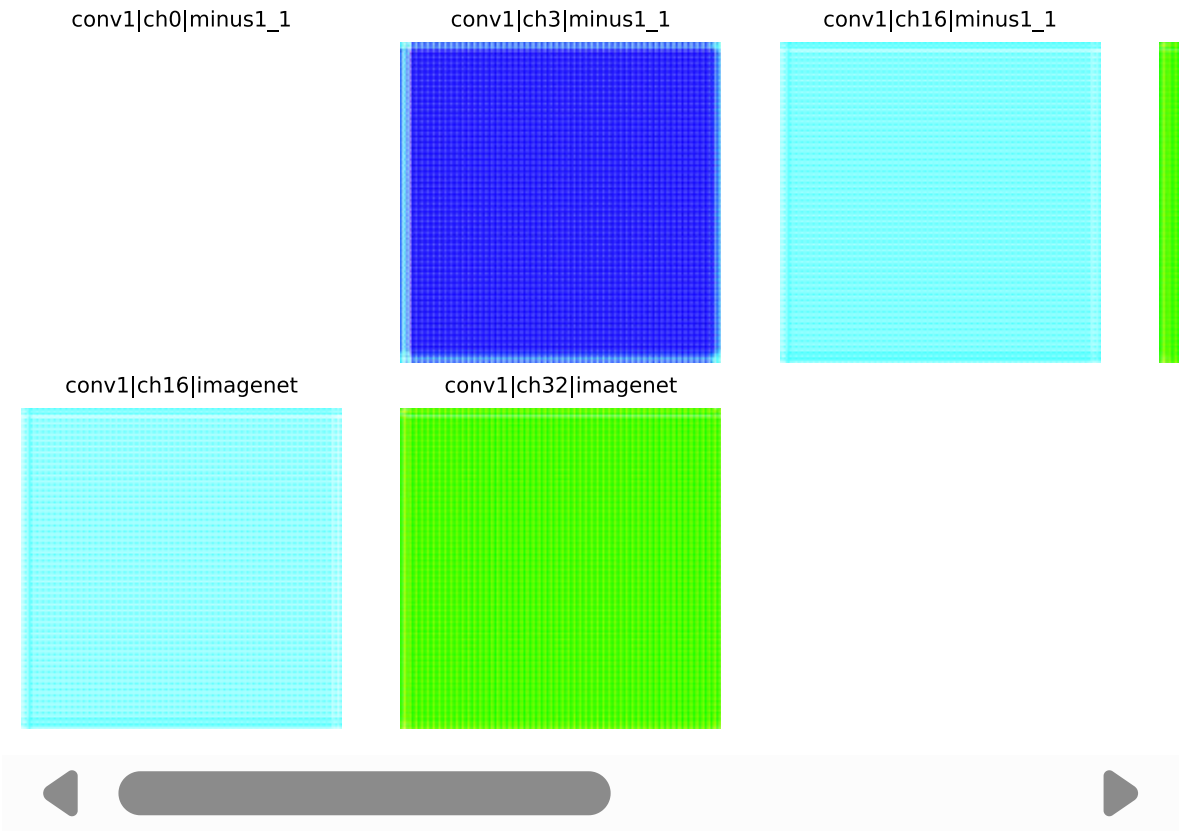
conv5|ch64|imagenet

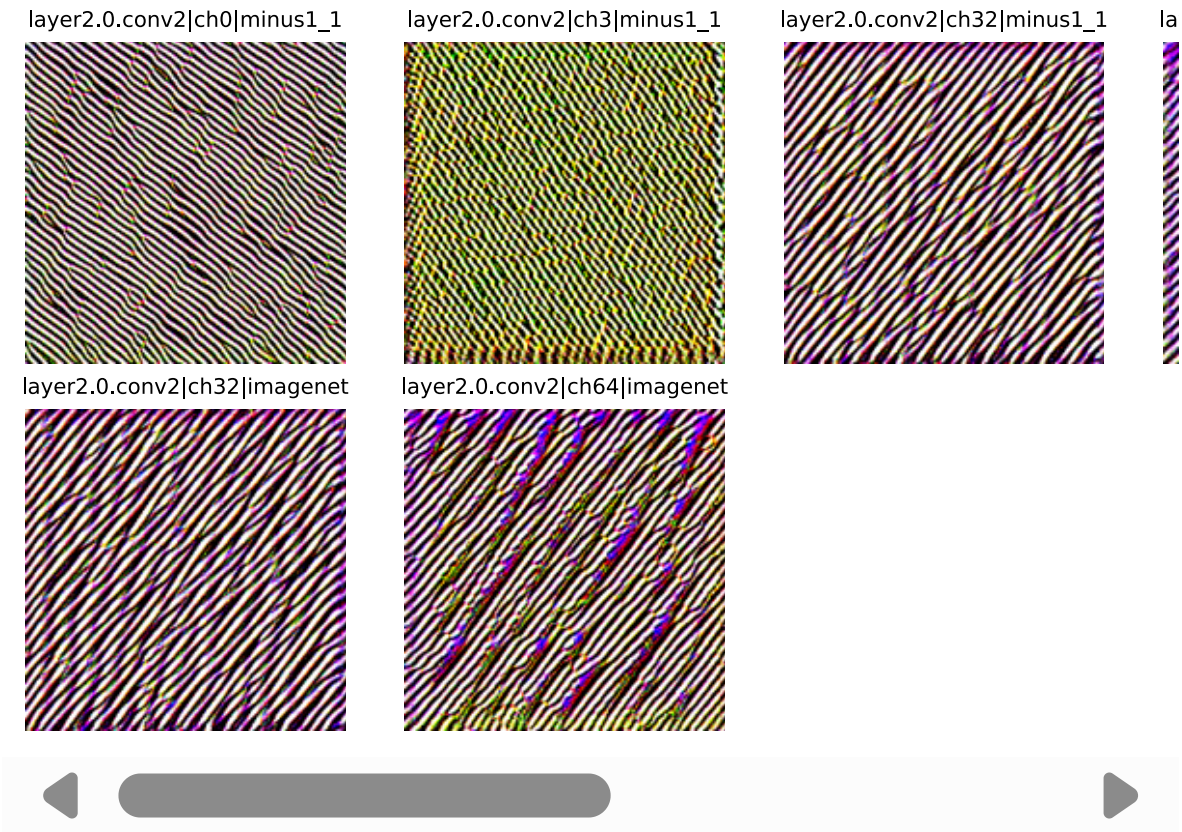




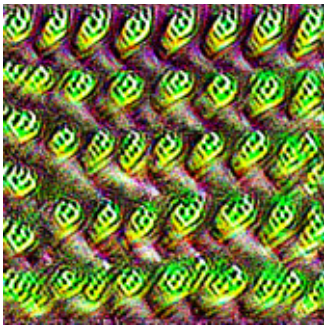



```
--- Art visuals for ResNet18 ---
Chosen layers: ['conv1', 'layer2.0.conv2', 'layer3.1.conv1', 'layer4.1.conv2']
```

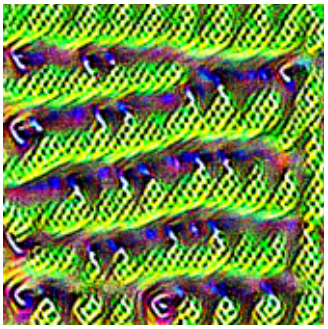

conv1|ch0|minus1_1
conv1|ch3|minus1_1
conv1|ch16|minus1_1
conv1|ch16|imagenet
conv1|ch32|imagenet


layer2.0.conv2|ch0|minus1_1
layer2.0.conv2|ch3|minus1_1
layer2.0.conv2|ch32|minus1_1
la
layer2.0.conv2|ch32|imagenet
layer2.0.conv2|ch64|imagenet

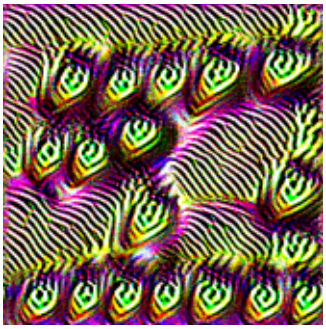
layer3.1.conv1|ch0|minus1_1



layer3.1.conv1|ch3|minus1_1

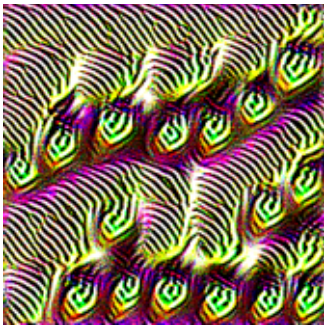


layer3.1.conv1|ch64|minus1_1

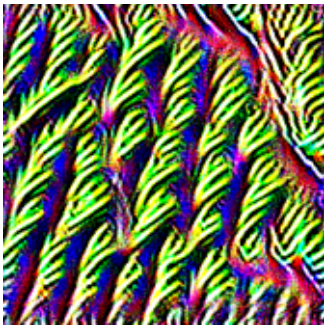


lay

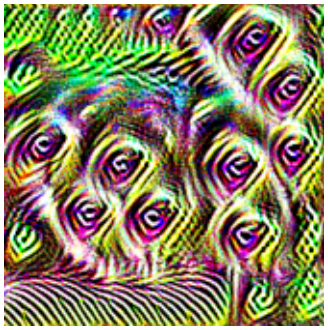
layer3.1.conv1|ch64|imagenet



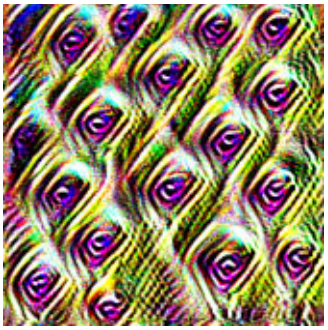
layer3.1.conv1|ch128|imagenet



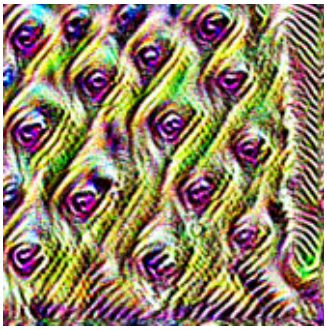
layer4.1.conv2|ch0|minus1_1



layer4.1.conv2|ch3|minus1_1

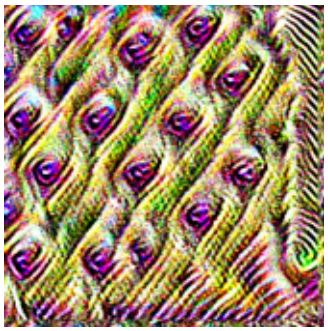


layer4.1.conv2|ch128|minus1_1

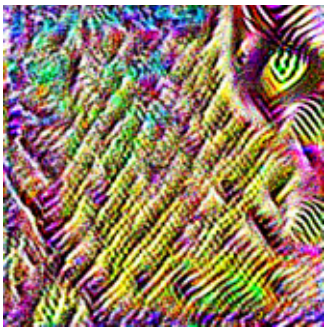


la

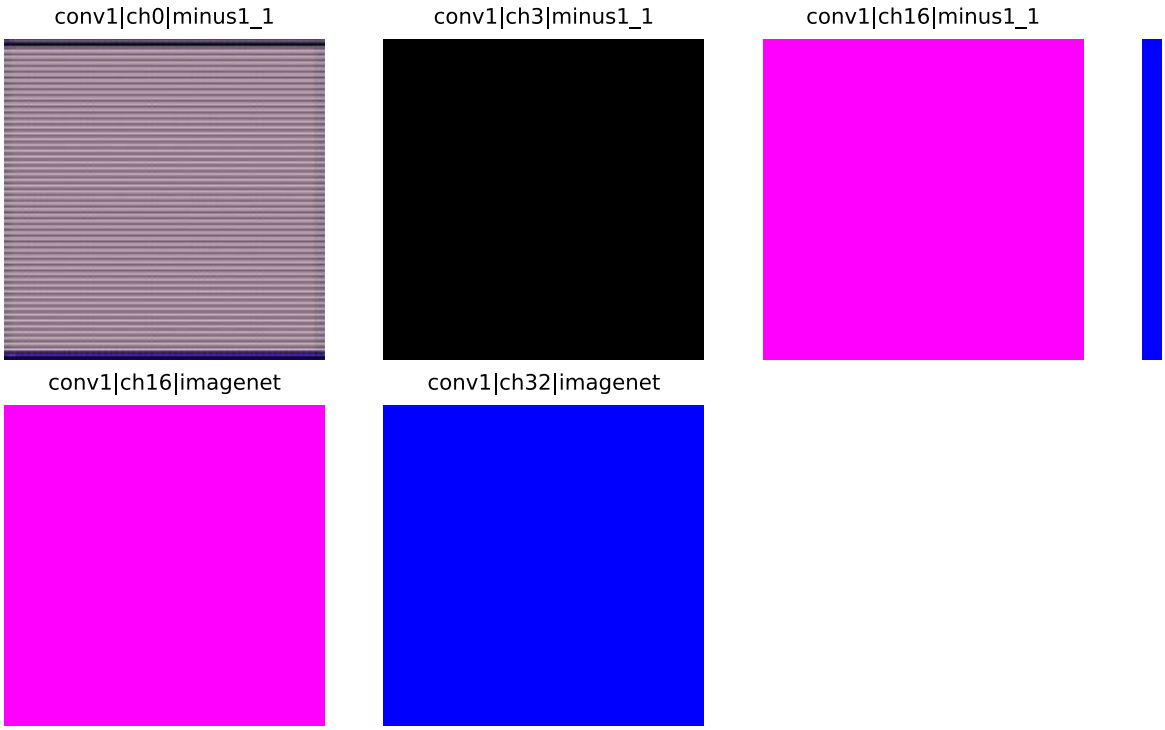
layer4.1.conv2|ch128|imagenet



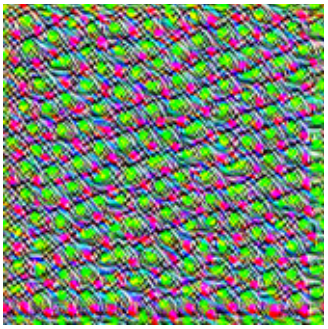
layer4.1.conv2|ch256|imagenet



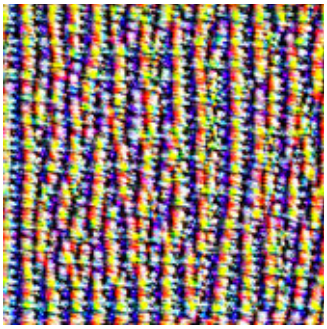

```
--- Art visuals for ResNet18p ---
Chosen layers: ['conv1', 'layer2.0.conv2', 'layer3.1.conv1', 'layer4.1.conv2']
```



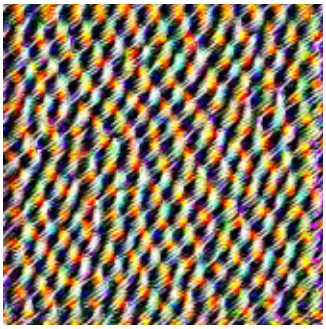
layer2.0.conv2|ch0|minus1_1



layer2.0.conv2|ch3|minus1_1

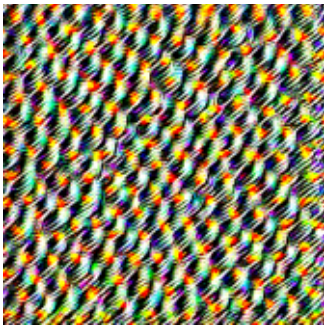


layer2.0.conv2|ch32|minus1_1

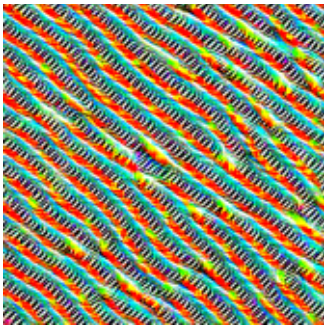


la

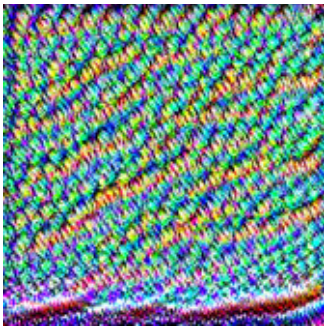
layer2.0.conv2|ch32|imagenet



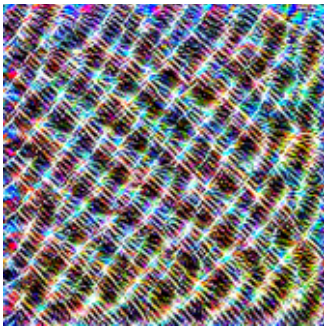
layer2.0.conv2|ch64|imagenet



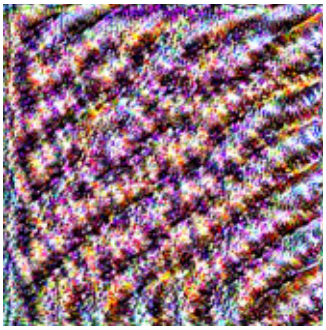
layer3.1.conv1|ch0|minus1_1



layer3.1.conv1|ch3|minus1_1

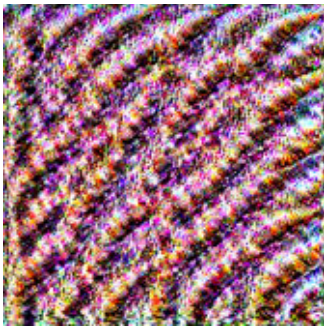


layer3.1.conv1|ch64|minus1_1

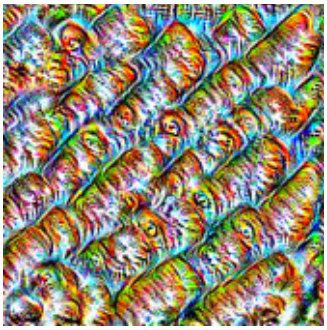


lay

layer3.1.conv1|ch64|imagenet



layer3.1.conv1|ch128|imagenet



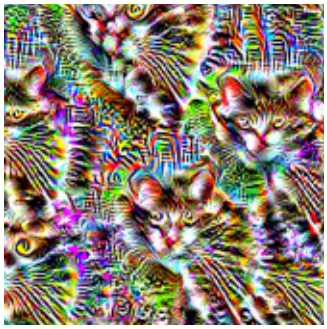
layer4.1.conv2|ch0|minus1_1

layer4.1.conv2|ch3|minus1_1

layer4.1.conv2|ch128|minus1_1

la

layer4.1.conv2|ch128|imagenet



layer4.1.conv2|ch256|imagenet

class0|minus1_1

class1|minus1_1

class0|imagenet

In [ ]:

In [ ]:

In [ ]:

In [ ]: