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.

```
# 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
```

```
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

```python
def show_imgs(W_list, nc=10, highlight_green=None, highlight_red=None, 
titles=None):
    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.

```
# 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,))
])
```

```
# 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)
```

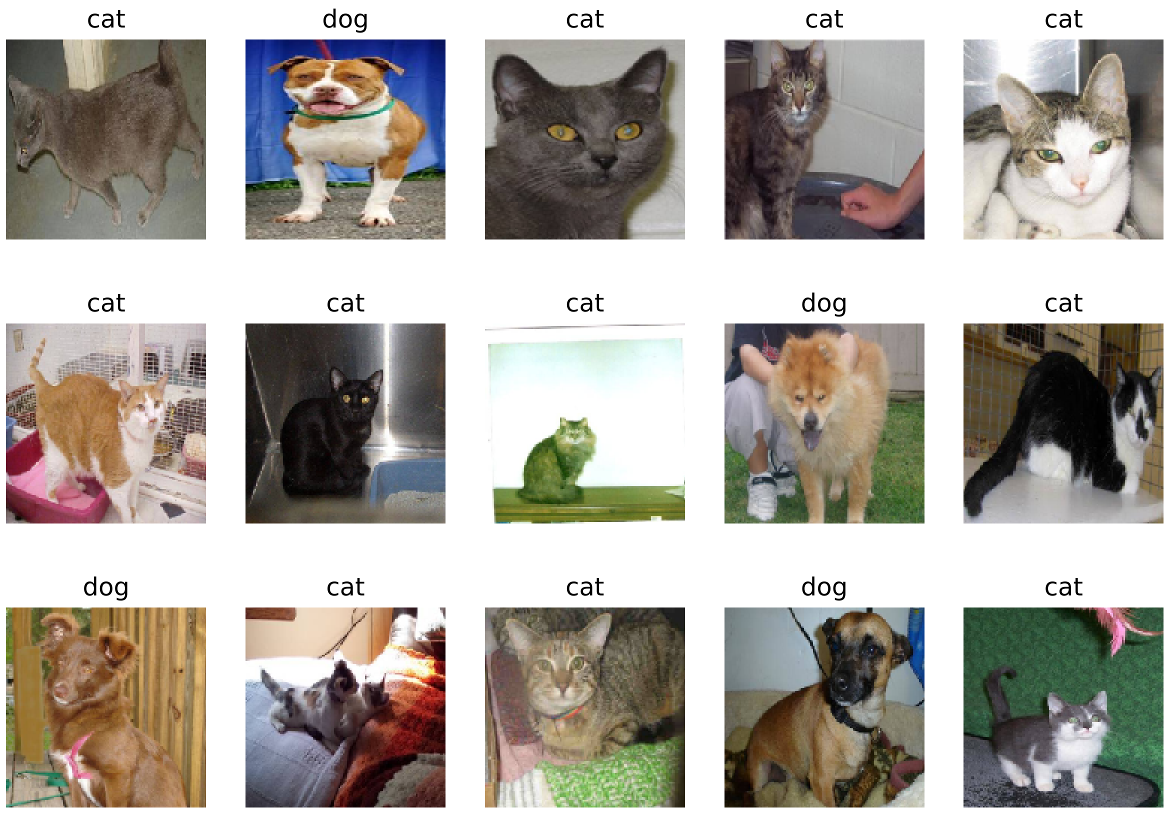
```
# collect validation data file names
img_files = os.listdir('valid/')
len(img_files)
img_files = list(filter(lambda x: x != 'valid', img_files))
def valid_path(p): return f"valid/{p}"
img_files = list(map(valid_path, img_files))[::10] # subsample to make it faster
```

```
# 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)
```

```
(500, 5)
```

Now let's view a few samples

```
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)

```
class CatAndDogNet4(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 3, out_channels = 16, kernel_size=
(5, 5), stride=2, padding=1)
        self.conv2 = nn.Conv2d(in_channels = 16, out_channels = 32, kernel_size=
(5, 5), stride=2, padding=1)
        self.conv3 = nn.Conv2d(in_channels = 32, out_channels = 64, kernel_size=
(3, 3), padding=1)
        self.conv4 = nn.Conv2d(in_channels = 64, out_channels = 128,
kernel_size=(3, 3), padding=1)
        self.conv5 = nn.Conv2d(in_channels = 128, out_channels = 128,
kernel_size=(3, 3), padding=1)
        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
```

```
# 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)

```
class CatAndDogNet6(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 3, out_channels = 16, kernel_size=
(5, 5), stride=2, padding=1)
        self.conv2 = nn.Conv2d(in_channels = 16, out_channels = 32, kernel_size=
(5, 5), stride=2, padding=1)
        self.conv3 = nn.Conv2d(in_channels = 32, out_channels = 64, kernel_size=
(3, 3), padding=1)
        self.conv4 = nn.Conv2d(in_channels = 64, out_channels = 128,
kernel_size=(3, 3), padding=1)
        self.conv5 = nn.Conv2d(in_channels = 128, out_channels = 128,
kernel_size=(3, 3), padding=1)
        self.conv6 = nn.Conv2d(in_channels = 128, out_channels = 128,
kernel_size=(3, 3), padding=1)
        self.conv7 = nn.Conv2d(in_channels = 128, out_channels = 128,
kernel_size=(3, 3), padding=1)
        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
```

```
# 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)

```
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()
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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)

```
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()
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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_running_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.

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

Evaluate each model on the validation data.

```
# 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[myname]:.2f}%')
```

```
.....
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`

```
!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-
packages (from torch-lucent) (2.8.0)
Requirement already satisfied: torchvision in /opt/conda/lib/python3.12/site-
packages (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
(from 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-
packages (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_64.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-
packages (from torch-lucent) (1.7.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.12/site-
packages (from torch>=1.5.0->torch-lucent) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in
/opt/conda/lib/python3.12/site-packages (from torch>=1.5.0->torch-lucent)
(4.14.1)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.12/site-
packages (from torch>=1.5.0->torch-lucent) (80.9.0)
Requirement already satisfied: sympy>=1.13.3 in /opt/conda/lib/python3.12/site-
packages (from torch>=1.5.0->torch-lucent) (1.14.0)
Requirement already satisfied: networkx in /opt/conda/lib/python3.12/site-
packages (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/lib/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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/python3.12/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-
packages (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/site-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/python3.12/site-packages (from coveralls->torch-lucent) (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/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-
packages (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/site-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/site-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-
packages (from ipython->torch-lucent) (0.19.2)
Requirement already satisfied: matplotlib-inline in
/opt/conda/lib/python3.12/site-packages (from ipython->torch-lucent) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.12/site-
packages (from ipython->torch-lucent) (4.9.0)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in
/opt/conda/lib/python3.12/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-
packages (from ipython->torch-lucent) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in
/opt/conda/lib/python3.12/site-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/site-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-
packages (from pytest->torch-lucent) (25.0)
Requirement already satisfied: pluggy<2,>=1.5 in /opt/conda/lib/python3.12/site-
packages (from pytest->torch-lucent) (1.6.0)
Requirement already satisfied: scipy>=1.8.0 in /opt/conda/lib/python3.12/site-
packages (from scikit-learn->torch-lucent) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in /opt/conda/lib/python3.12/site-
packages (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-
packages (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,
pytest-mock, coveralls, kornia, torch-lucent
•[2K   •[90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━•[0m •[32m9/9•[0m [torch-lucent]
[0m [kornia]e]
•[1A•[2KSuccessfully 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).

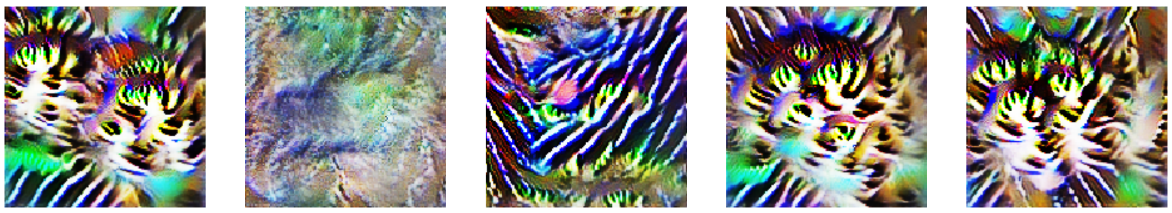
```
# Feature visualization on the last convolutional layer of each model.
```

```python
# This block attempts to use the lucent toolbox (torch-lucent) to visualize a few
channels
# from the last convolutional layer for each model in the `all_models`
dictionary.
# If lucent is not installed or a layer name cannot be found automatically, a
helpful message is printed.
try:
    import lucent.optvis as lu_optvis
    from lucent.optvis import render as lu_render
    import numpy as np
except Exception as e:
    print("lucent is not available. Install it with `pip install torch-lucent`
to run feature visualization.")
    print("Exception:", e)
    raise

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

# Candidate layer name patterns to try for the "last conv layer"
candidate_layer_names = ["conv5", "conv_layers_5", "features.conv5", "layer4",
"features.4", "conv_layers_7", "conv5_relu"]

for model_name, model in all_models.items():
    print(f"\n=== Feature visualization for model: {model_name} ===")
    model.to(device).eval()
    found = False
    # Try each candidate to see if lucent can render it
    for cand in candidate_layer_names:
        for ch in range(0,5):  # test first 5 channels
            layer_spec = f"{cand}:{ch}"
            try:
                out = lu_optvis.render.render_vis(model, layer_spec,
show_image=False)
                # If no exception, we found a usable layer spec
                found = True
                break
            except Exception:
                found = False
        if found:
            chosen_layer = cand
            break
    if not found:
        print(f"Could not find a lucent-friendly layer name for model
'{model_name}'. You can try other names.")
        continue

    # Render a small set of channels from the chosen layer
    imgs = []
    channels_to_show = list(range(0,5))
    for ch in channels_to_show:
        spec = f"{chosen_layer}:{ch}"
        try:
            out = lu_optvis.render.render_vis(model, spec, show_image=False)
            imgs.append(np.squeeze(out[0]))
        except Exception as e:
            print(f"Failed to render {spec}: {e}")
    if imgs:
        plt.figure(figsize=(12,3))
        show_imgs(imgs, nc=len(imgs))
        plt.suptitle(f"{model_name} - layer {chosen_layer} channels
{channels_to_show}")
        plt.show()
    else:
        print("No images produced for this model.")
```

```
=== Feature visualization for model: CNN5 ===
```

```
100%|██████████| 512/512 [00:05<00:00, 91.03it/s]
100%|██████████| 512/512 [00:05<00:00, 98.60it/s]
100%|██████████| 512/512 [00:05<00:00, 100.44it/s]
100%|██████████| 512/512 [00:05<00:00, 97.82it/s]
100%|██████████| 512/512 [00:05<00:00, 94.58it/s]
100%|██████████| 512/512 [00:05<00:00, 96.16it/s]
```

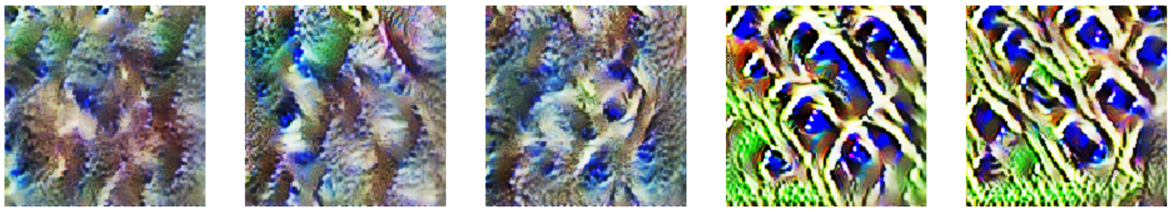
CNN5 - layer conv5 channels [0, 1, 2, 3, 4]



```
=== Feature visualization for model: CNN7 ===
```

```
100%|██████████| 512/512 [00:05<00:00, 94.78it/s]
100%|██████████| 512/512 [00:05<00:00, 95.91it/s]
100%|██████████| 512/512 [00:05<00:00, 94.74it/s]
100%|██████████| 512/512 [00:05<00:00, 95.61it/s]
100%|██████████| 512/512 [00:05<00:00, 93.84it/s]
100%|██████████| 512/512 [00:05<00:00, 94.06it/s]
```

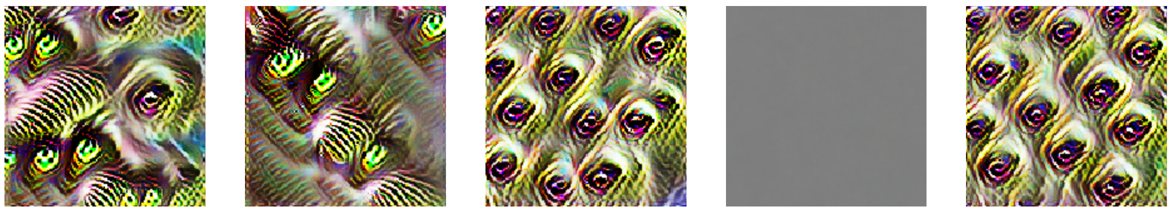
CNN7 - layer conv5 channels [0, 1, 2, 3, 4]



```
=== Feature visualization for model: 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]
  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]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
100%|██████████| 512/512 [00:10<00:00, 48.37it/s]
100%|██████████| 512/512 [00:10<00:00, 48.59it/s]
100%|██████████| 512/512 [00:10<00:00, 48.45it/s]
100%|██████████| 512/512 [00:10<00:00, 47.65it/s]
100%|██████████| 512/512 [00:10<00:00, 49.38it/s]
100%|██████████| 512/512 [00:10<00:00, 47.13it/s]
```

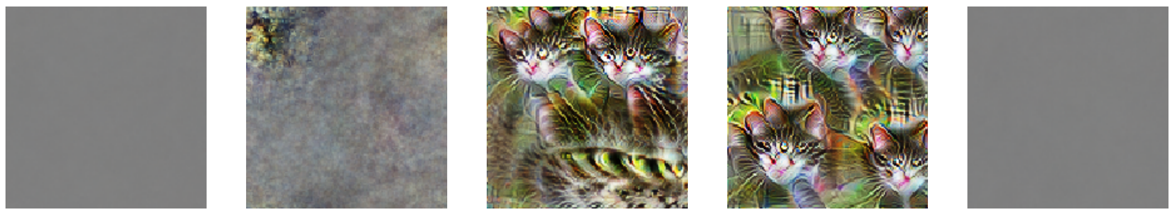
ResNet18 - layer layer4 channels [0, 1, 2, 3, 4]



```
=== Feature visualization for model: 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]
  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]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
100%|██████████| 512/512 [00:10<00:00, 50.06it/s]
100%|██████████| 512/512 [00:10<00:00, 48.83it/s]
100%|██████████| 512/512 [00:10<00:00, 50.01it/s]
100%|██████████| 512/512 [00:10<00:00, 49.63it/s]
100%|██████████| 512/512 [00:10<00:00, 49.18it/s]
100%|██████████| 512/512 [00:10<00:00, 48.28it/s]
```

ResNet18p - layer layer4 channels [0, 1, 2, 3, 4]



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.

Art!

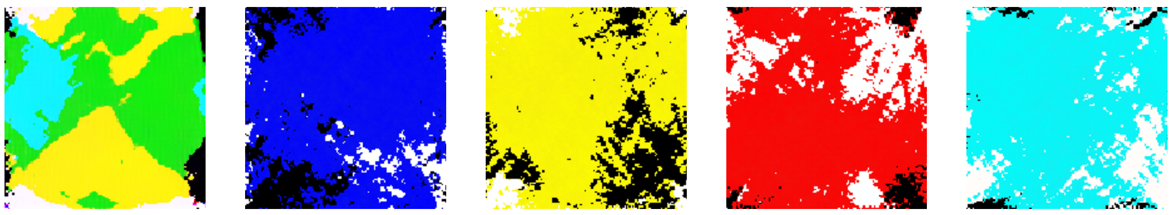
Now explore the feature visualizations of other parts of the network. See if you can find some interesting textures or patterns. What do they resemble?

```
# Art! - explore visualizations of other layers and class logits.
# Try a few earlier layers and also try to visualize the class logits directly
(if supported).
try:
    import lucent.optvis as lu_optvis
except Exception as e:
    print("lucent not available; install with `pip install torch-lucent`.")
    raise
```

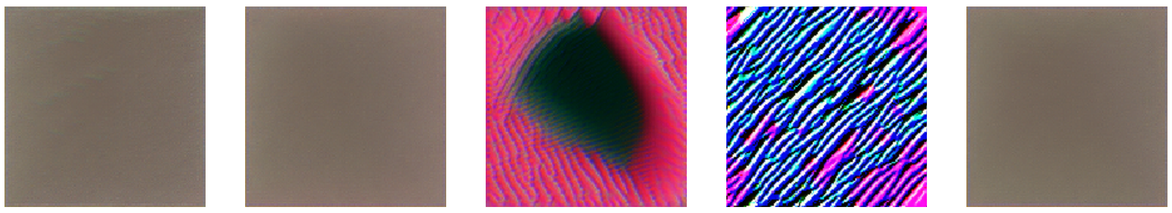
```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

# Layers we want to explore (common names used in this notebook's models)
layers_to_try = ["conv1", "conv2", "conv3", "conv4", "conv5", "conv_layers_2",
"conv_layers_5"]

for model_name, model in all_models.items():
    print(f"\n--- Art visualizations for model: {model_name} ---")
    model.to(device).eval()
    for lay in layers_to_try:
        imgs = []
        for ch in range(0,5):  # show first 5 channels (if available)
            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 if that layer spec is not valid for this model / lucent
mapping
                break
        if imgs:
            plt.figure(figsize=(12,3))
            show_imgs(imgs, nc=len(imgs))
            plt.suptitle(f"{model_name} - layer {lay} (channels 0..
{len(imgs)-1})")
            plt.show()

# Optionally produce a visualization that maximizes a final class logit (if model
mapping supports it).
for model_name, model in all_models.items():
    print(f"\n--- Class-logit visualization for: {model_name} ---")
    model.to(device).eval()
    # Try "output" or "logits" style specs commonly used by lucent
    class_specs = ["logit:0", "prob:0", "output:0"]
    for cs in class_specs:
        try:
            out = lu_optvis.render.render_vis(model, cs, show_image=False)
            plt.figure(figsize=(4,4))
            plt.imshow(np.squeeze(out[0]))
            plt.title(f"{model_name} - {cs}")
            plt.axis('off')
            plt.show()
            break
        except Exception:
            continue
```

```
--- Art visualizations for model: CNN5 ---
```

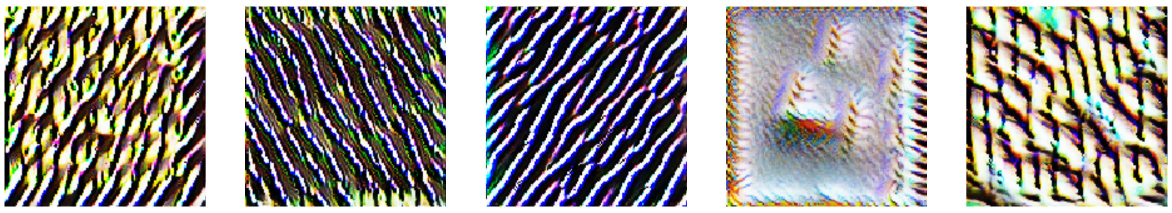
```
100%|██████████| 512/512 [00:04<00:00, 108.30it/s]
100%|██████████| 512/512 [00:04<00:00, 108.49it/s]
100%|██████████| 512/512 [00:04<00:00, 113.72it/s]
100%|██████████| 512/512 [00:04<00:00, 107.31it/s]
100%|██████████| 512/512 [00:04<00:00, 108.07it/s]
```


CNN5 - layer conv1 (channels 0..4)

```
100%|██████████| 512/512 [00:04<00:00, 109.89it/s]
100%|██████████| 512/512 [00:04<00:00, 104.78it/s]
100%|██████████| 512/512 [00:05<00:00, 96.26it/s]
100%|██████████| 512/512 [00:05<00:00, 100.95it/s]
100%|██████████| 512/512 [00:04<00:00, 106.87it/s]
```

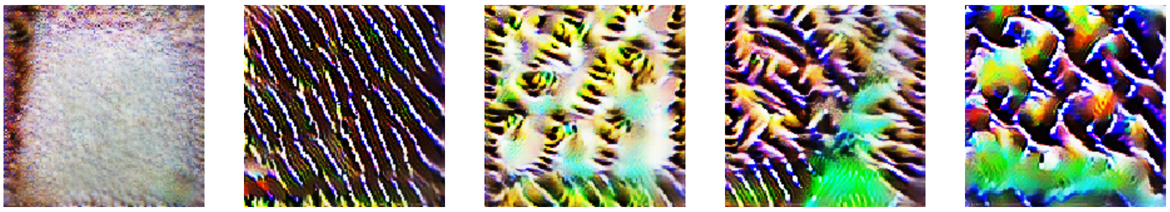

CNN5 - layer conv2 (channels 0..4)

```
100%|██████████| 512/512 [00:05<00:00, 99.33it/s]
100%|██████████| 512/512 [00:04<00:00, 102.57it/s]
100%|██████████| 512/512 [00:05<00:00, 99.43it/s]
100%|██████████| 512/512 [00:05<00:00, 97.29it/s]
100%|██████████| 512/512 [00:05<00:00, 101.45it/s]
```


CNN5 - layer conv3 (channels 0..4)

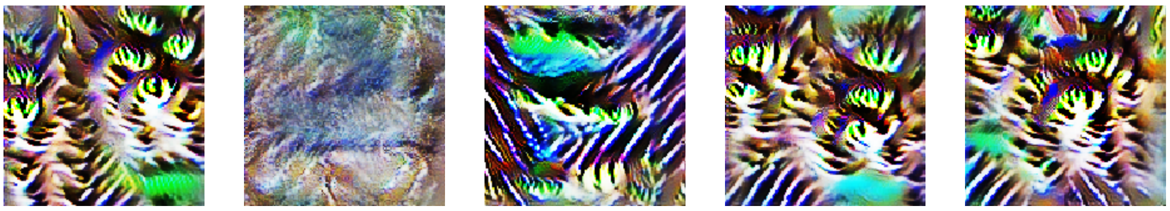
```
100%|██████████| 512/512 [00:05<00:00, 96.53it/s]
100%|██████████| 512/512 [00:05<00:00, 96.19it/s]
100%|██████████| 512/512 [00:05<00:00, 89.35it/s]
100%|██████████| 512/512 [00:05<00:00, 94.19it/s]
100%|██████████| 512/512 [00:05<00:00, 93.95it/s]
```

CNN5 - layer conv4 (channels 0..4)



```
100%|██████████| 512/512 [00:05<00:00, 94.50it/s]
100%|██████████| 512/512 [00:05<00:00, 96.47it/s]
100%|██████████| 512/512 [00:05<00:00, 94.57it/s]
100%|██████████| 512/512 [00:05<00:00, 91.55it/s]
100%|██████████| 512/512 [00:05<00:00, 90.46it/s]
```

CNN5 - layer conv5 (channels 0..4)

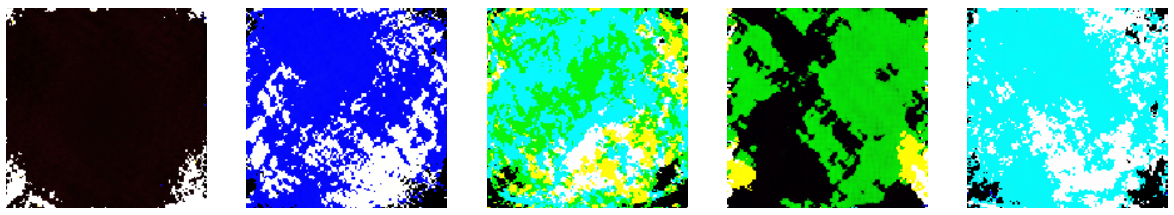


```
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```

```
--- Art visualizations for model: CNN7 ---
```

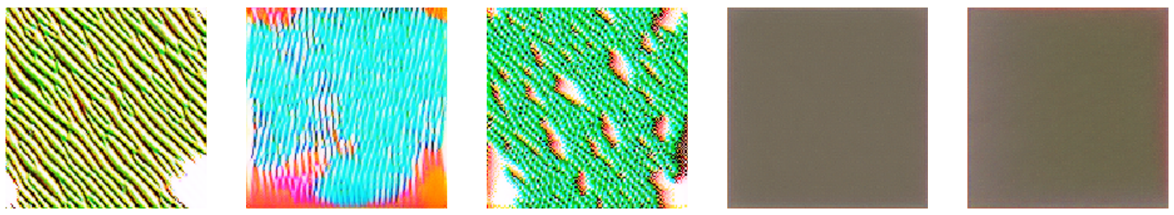
```
100%|██████████| 512/512 [00:04<00:00, 105.35it/s]
100%|██████████| 512/512 [00:04<00:00, 104.43it/s]
100%|██████████| 512/512 [00:04<00:00, 108.26it/s]
100%|██████████| 512/512 [00:05<00:00, 97.46it/s]
100%|██████████| 512/512 [00:05<00:00, 102.00it/s]
```

CNN7 - layer conv1 (channels 0..4)



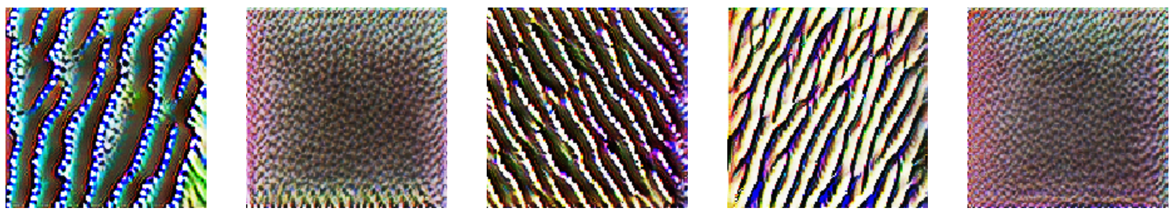
```
100%|██████████| 512/512 [00:05<00:00, 101.41it/s]
100%|██████████| 512/512 [00:04<00:00, 103.56it/s]
100%|██████████| 512/512 [00:05<00:00, 98.39it/s]
100%|██████████| 512/512 [00:05<00:00, 100.98it/s]
100%|██████████| 512/512 [00:05<00:00, 101.21it/s]
```

CNN7 - layer conv2 (channels 0..4)



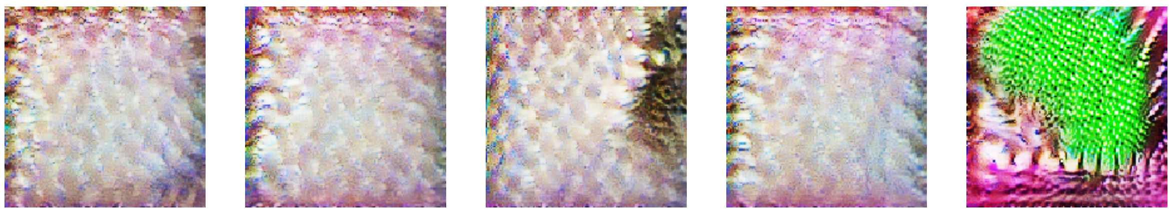
```
100%|██████████| 512/512 [00:05<00:00, 101.23it/s]
100%|██████████| 512/512 [00:05<00:00, 97.51it/s]
100%|██████████| 512/512 [00:05<00:00, 94.73it/s]
100%|██████████| 512/512 [00:05<00:00, 96.65it/s]
100%|██████████| 512/512 [00:05<00:00, 99.16it/s]
```

CNN7 - layer conv3 (channels 0..4)



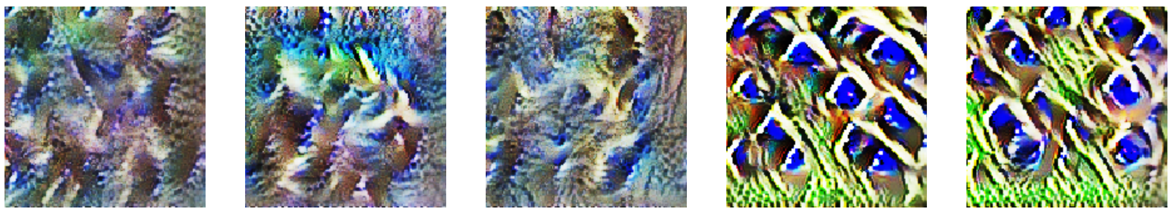
```
100%|██████████| 512/512 [00:05<00:00, 92.90it/s]
100%|██████████| 512/512 [00:05<00:00, 95.38it/s]
100%|██████████| 512/512 [00:05<00:00, 93.84it/s]
100%|██████████| 512/512 [00:05<00:00, 94.20it/s]
100%|██████████| 512/512 [00:05<00:00, 95.90it/s]
```

CNN7 - layer conv4 (channels 0..4)



```
100%|██████████| 512/512 [00:05<00:00, 91.98it/s]
100%|██████████| 512/512 [00:05<00:00, 90.67it/s]
100%|██████████| 512/512 [00:05<00:00, 89.16it/s]
100%|██████████| 512/512 [00:05<00:00, 87.73it/s]
100%|██████████| 512/512 [00:05<00:00, 92.50it/s]
```

CNN7 - layer conv5 (channels 0..4)

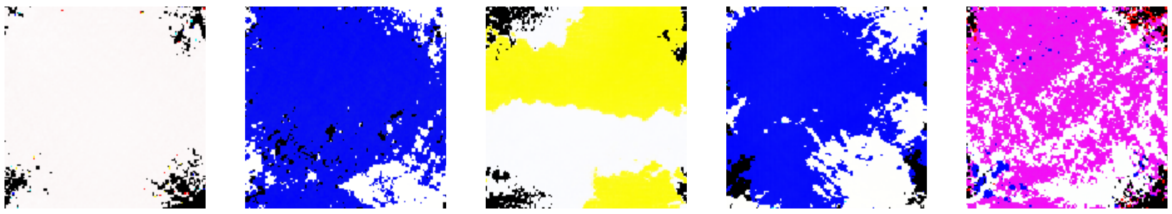


```
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```

```
--- Art visualizations for model: ResNet18 ---
```

```
100%|██████████| 512/512 [00:07<00:00, 72.39it/s]
100%|██████████| 512/512 [00:07<00:00, 69.06it/s]
100%|██████████| 512/512 [00:07<00:00, 68.00it/s]
100%|██████████| 512/512 [00:07<00:00, 70.57it/s]
100%|██████████| 512/512 [00:07<00:00, 69.25it/s]
```

ResNet18 - layer conv1 (channels 0..4)

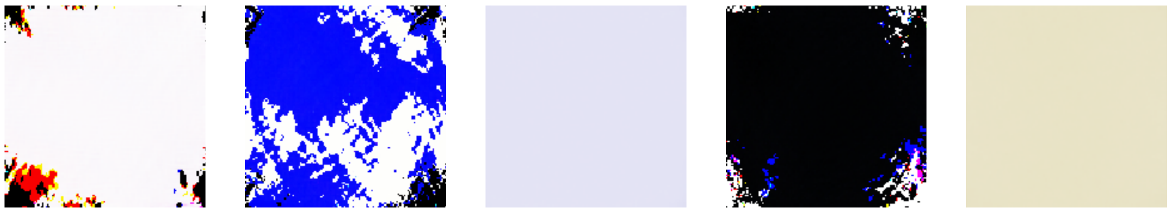


```
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
  0%|          | 0/512 [00:00<?, ?it/s]
```

```
--- Art visualizations for model: ResNet18p ---
```

```
100%|██████████| 512/512 [00:07<00:00, 70.67it/s]
100%|██████████| 512/512 [00:07<00:00, 70.85it/s]
100%|██████████| 512/512 [00:07<00:00, 69.66it/s]
100%|██████████| 512/512 [00:07<00:00, 69.09it/s]
100%|██████████| 512/512 [00:07<00:00, 68.91it/s]
```

ResNet18p - layer conv1 (channels 0..4)



```
  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]
```

```
--- Class-logit visualization for: CNN5 ---
```

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

```
--- Class-logit visualization for: CNN7 ---
```

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

```
--- Class-logit visualization for: ResNet18 ---
```

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

```
--- Class-logit visualization for: ResNet18p ---
```

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