In [ ]:

```
%reload_ext autoreload
%autoreload 2
```

In [ ]:

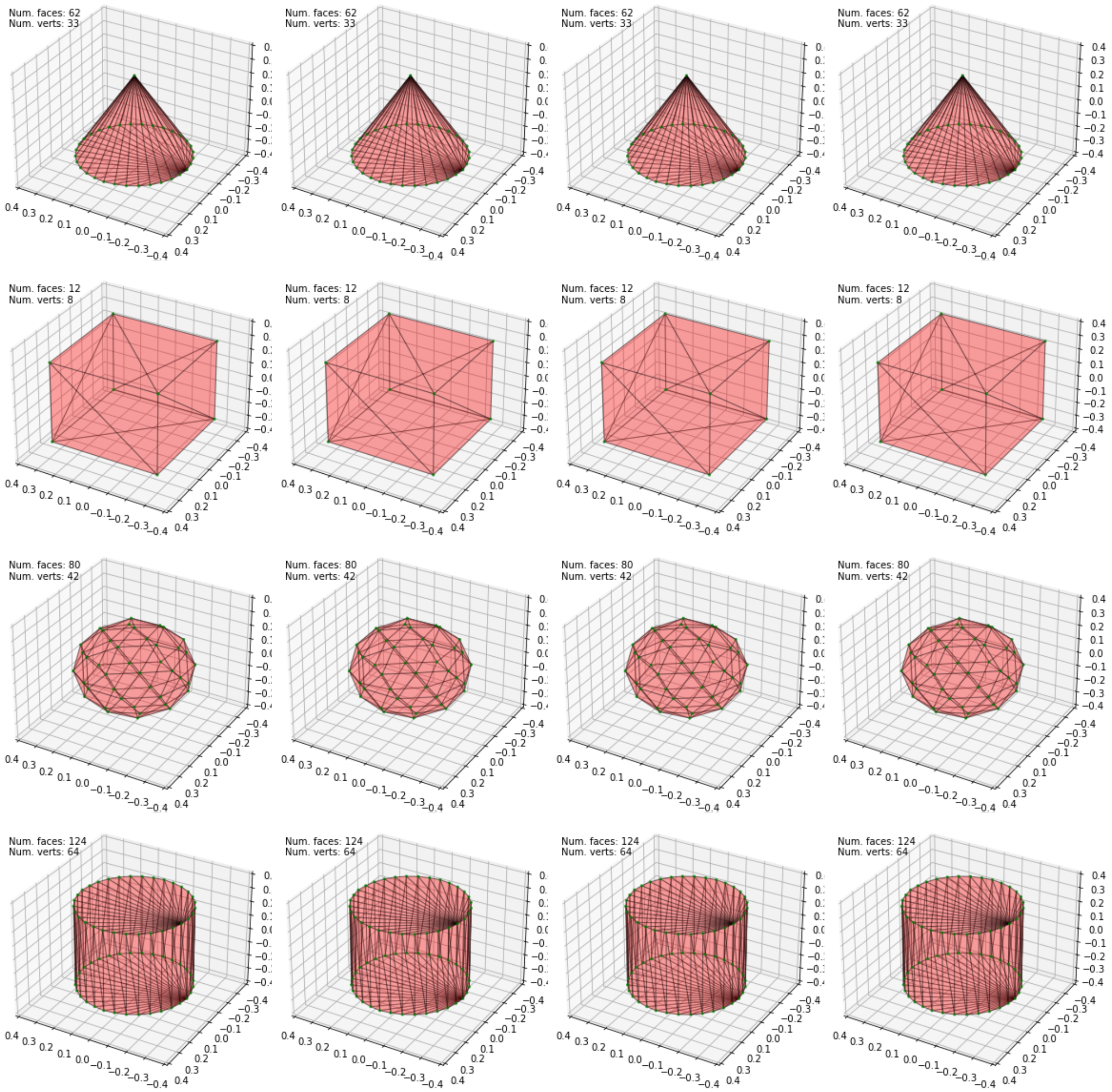
```
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
from torch.distributions.categorical import Categorical

from tqdm import tqdm

from polygen.modules.data_modules import PolygenDataModule, CollateMethod
from polygen.modules.vertex_model import ImageToVertexModel
from polygen.modules.face_model import FaceModel
import polygen.utils.data_utils as data_utils
```

In [ ]:

```
img_data_module = PolygenDataModule(data_dir = "image_meshes/",
                                collate_method = CollateMethod.IMAGES,
                                batch_size = 4,
                                training_split = 1.0,
                                val_split = 0.0,
                                use_image_dataset = True,
                                img_extension = "png",
                                apply_random_shift_vertices = False,
)

img_dataset = img_data_module.shapenet_dataset

face_data_module = PolygenDataModule(data_dir = "image_meshes/",
                                collate_method = CollateMethod.FACES,
                                batch_size = 4,
                                training_split = 1.0,
                                val_split = 0.0,
                                use_image_dataset = True,
                                img_extension = "png",
                                apply_random_shift_faces = False,
                                shuffle_vertices = False,
)

img_data_module.setup()
face_data_module.setup()

img_dataloader = img_data_module.train_dataloader()
face_dataloader = face_data_module.train_dataloader()
```

In [ ]:

```
mesh_list = []
for i in range(len(img_dataset)):
    mesh_dict = img_dataset[i]
    curr_verts, curr_faces = mesh_dict['vertices'], mesh_dict['faces']
    curr_verts = data_utils.dequantize_verts(curr_verts).numpy()
    curr_faces = data_utils.unflatten_faces(curr_faces.numpy())
    mesh_list.append({'vertices': curr_verts, 'faces': curr_faces})

data_utils.plot_meshes(mesh_list, ax_lims = 0.4)
```



In [ ]:

```
def load_models():
    img_decoder_config = {
        "hidden_size": 256,
        "fc_size": 1024,
        "num_layers": 5,
        'dropout_rate': 0.
    }

    face_transformer_config = {
        'hidden_size': 256,
        'fc_size': 1024,
        'num_layers': 3,
        'dropout_rate': 0.
    }

    img_model = ImageToVertexModel(
        decoder_config = img_decoder_config,
        max_num_input_verts = 800,
        quantization_bits = 8,
        learning_rate = 5e-4,
        gamma = 1.
    )

    face_model = FaceModel(encoder_config = face_transformer_config,
                           decoder_config = face_transformer_config,
                           class_conditional = False,
                           max_seq_length = 500,
                           quantization_bits = 8,
                           decoder_cross_attention = True,
                           use_discrete_vertex_embeddings = True,
                           learning_rate = 5e-4,
                           gamma = 1.,
                          )

    return img_model, face_model

def sample_and_plot(vertex_model, vertex_batch, face_model):
    with torch.no_grad():
        vertex_samples = vertex_model.sample(context = vertex_batch, num_sampl
                                        top_p = 0.95, recenter_verts = False,
        max_vertices = torch.max(vertex_samples["num_vertices"]).item()
        vertex_samples["vertices"] = vertex_samples["vertices"][:, :max_vertic
        vertex_samples["vertices_mask"] = vertex_samples["vertices_mask"][:, :
        face_samples = face_model.sample(context = vertex_samples, max_sample_
    mesh_list = []
    for i in range(vertex_samples["vertices"].shape[0]):
        num_vertices = vertex_samples["num_vertices"][i]
        vertices = vertex_samples["vertices"][i][:num_vertices].numpy()
        num_face_indices = face_samples['num_face_indices'][i]
        faces = data_utils.unflatten_faces(face_samples["faces"][i][:num_face_
        mesh_list.append({'vertices': vertices, 'faces': faces})
    data_utils.plot_meshes(mesh_list, ax_lims = 0.5)
```

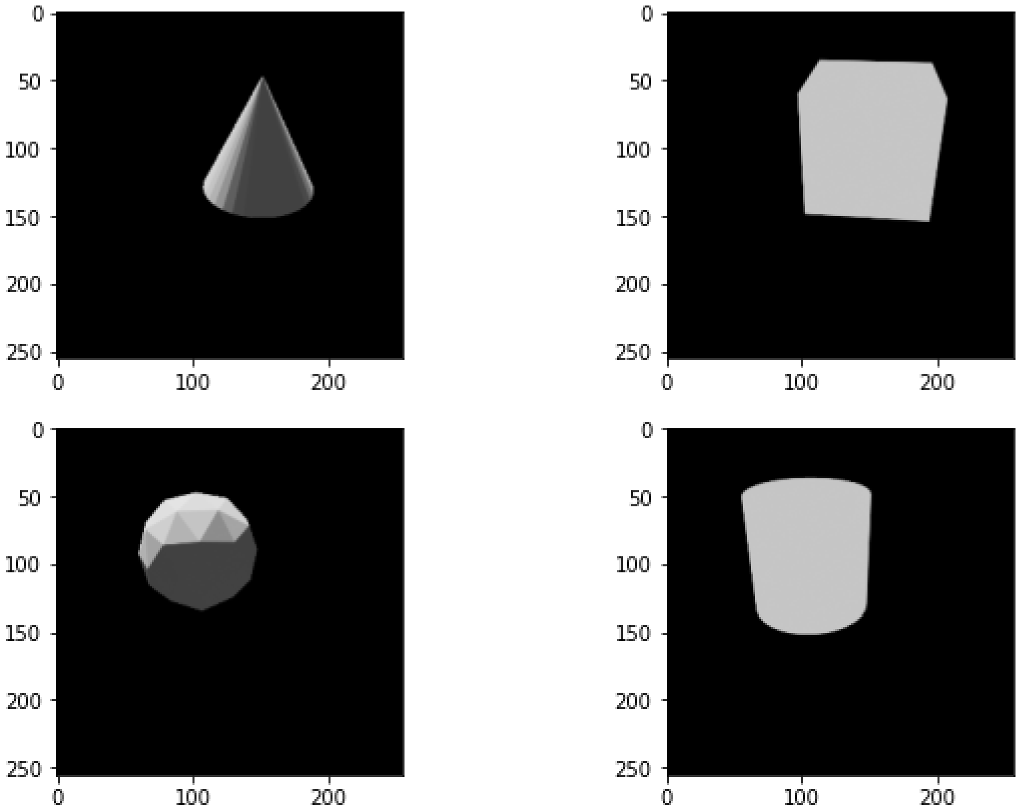
In [ ]:

```
vertex_model, face_model = load_models()
epochs = 500
vertex_model_optimizer = vertex_model.configure_optimizers()["optimizer"]
face_model_optimizer = face_model.configure_optimizers()["optimizer"]
for i in tqdm(range(epochs)):
    for j, (vertex_batch, face_batch) in enumerate(zip(img_dataloader, face_da
        vertex_model_optimizer.zero_grad()
        face_model_optimizer.zero_grad()

        vertex_logits = vertex_model(vertex_batch)
        face_logits = face_model(face_batch)

        vertex_pred_dist = Categorical(logits = vertex_logits)
        face_pred_dist = Categorical(logits = face_logits)

        vertex_loss = -torch.sum(vertex_pred_dist.log_prob(vertex_batch["verti
        face_loss = -torch.sum(face_pred_dist.log_prob(face_batch["faces"]) *

        vertex_loss.backward()
        face_loss.backward()

        vertex_model_optimizer.step()
        face_model_optimizer.step()
    if ((i + 1) % 50 == 0):
        print(f"Epoch {i + 1}: Vertex loss = {vertex_loss.item()}, Face loss =
```

```
 10%|█         | 50/500 [02:59<26:34,  3.54s/it]
Epoch 50: Vertex loss = 16.735572814941406, Face loss = 2.837860107421875
 20%|██        | 100/500 [05:53<22:12,  3.33s/it]
Epoch 100: Vertex loss = 7.459108352661133, Face loss = 1.207326889038086
 30%|███       | 150/500 [09:00<23:06,  3.96s/it]
Epoch 150: Vertex loss = 7.716439247131348, Face loss = 0.4701204299926758
 40%|████      | 200/500 [11:56<17:08,  3.43s/it]
Epoch 200: Vertex loss = 6.128841400146484, Face loss = 0.3149299621582031
 50%|█████     | 250/500 [14:50<13:50,  3.32s/it]
Epoch 250: Vertex loss = 6.904537200927734, Face loss = 1.575474739074707
 60%|██████    | 300/500 [17:46<11:33,  3.47s/it]
Epoch 300: Vertex loss = 2.8032970428466797, Face loss = 0.6678276062011719
 70%|███████   | 350/500 [20:43<09:00,  3.60s/it]
Epoch 350: Vertex loss = 0.2704658508300781, Face loss = 0.6448230743408203
 80%|████████  | 400/500 [23:43<05:56,  3.56s/it]
Epoch 400: Vertex loss = 0.28075599670410156, Face loss = 0.46938133239746094
 90%|█████████ | 450/500 [26:45<02:53,  3.48s/it]
Epoch 450: Vertex loss = 0.13967037200927734, Face loss = 0.3481259346008301
100%|██████████| 500/500 [29:55<00:00,  3.59s/it]
Epoch 500: Vertex loss = 0.1100454330444336, Face loss = 0.2909095287322998
```

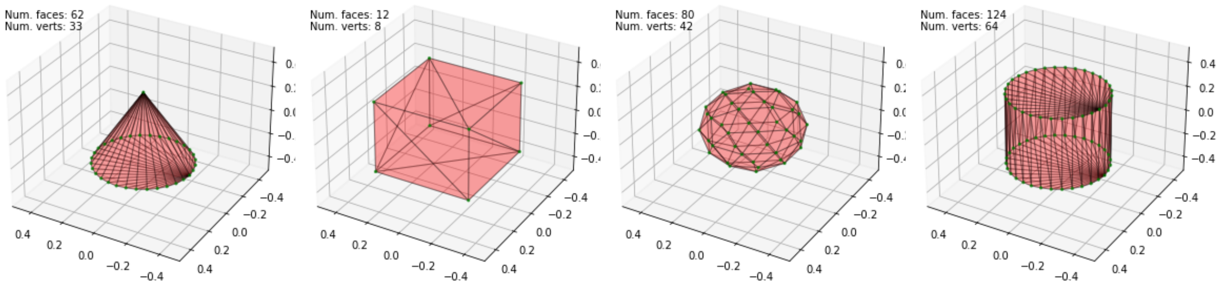
In [ ]:

```
# Construct custom batch with 1 image for each object
cone_context = img_dataset[0]
cube_context = img_dataset[4]
icosphere_context = img_dataset[8]
cylinder_context = img_dataset[12]
batch = [cone_context, cube_context, icosphere_context, cylinder_context]
fig = plt.figure(figsize = (10, 7))
for i, context in enumerate(batch):
    fig.add_subplot(2, 2, i + 1)
    curr_image = context["image"]
    plt.imshow(curr_image.permute(1, 2, 0)) #Going from [C, H, W] to [H, W, C]
```



In [ ]:

```
img_batch = img_data_module.collate_img_model_batch(batch)
img_batch = {"image": img_batch["image"]} #Removing vertices and faces from ba
sample_and_plot(vertex_model, img_batch, face_model)
```



In [ ]: