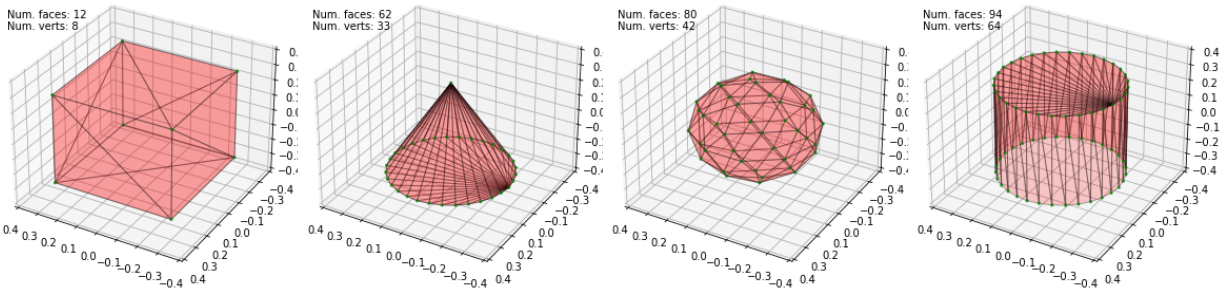
In [ ]:
```
%reload_ext autoreload
%autoreload 2
```

In [ ]:
```
"""Important imports"""
import glob

import torch

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

In [ ]:
```
"""Get dataset ready"""
data_dir = "meshes/"
all_files = glob.glob(data_dir + "/*.obj")
label_dict = {}
for i, mesh_file in enumerate(all_files):
    label_dict[mesh_file] = i

vertex_data_module = PolygenDataModule(data_dir = data_dir, collate_method = C
                                       training_split = 1.0, val_split = 0.0,
                                       label_dict = label_dict, apply_random_

face_data_module = PolygenDataModule(data_dir = data_dir, collate_method = Col
                                       training_split = 1.0, val_split = 0.0,
                                       label_dict = label_dict, apply_random_

vertex_data_module.setup()
face_data_module.setup()

vertex_dataloader = vertex_data_module.train_dataloader()
face_dataloader = face_data_module.train_dataloader()

vertex_batch = next(iter(vertex_dataloader))
face_batch = next(iter(face_dataloader))
```

In [ ]:
```
"""plot dataset"""
dataset = vertex_data_module.shapenet_dataset
mesh_list = []
for i in range(len(dataset)):
    mesh_dict = 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 [ ]:

```
"""Load models"""
def load_models():
    transformer_config = {
        "hidden_size": 128,
        "fc_size": 512,
        "num_layers": 3,
        "dropout_rate": 0.
    }

    vertex_model = VertexModel(decoder_config=transformer_config, class_condit
                               max_num_input_verts=250, quantization_bits=8)

    face_model = FaceModel(encoder_config=transformer_config, decoder_config=t
                           max_seq_length=500, quantization_bits=8, decoder_c
    return vertex_model, face_model
```

In [ ]:

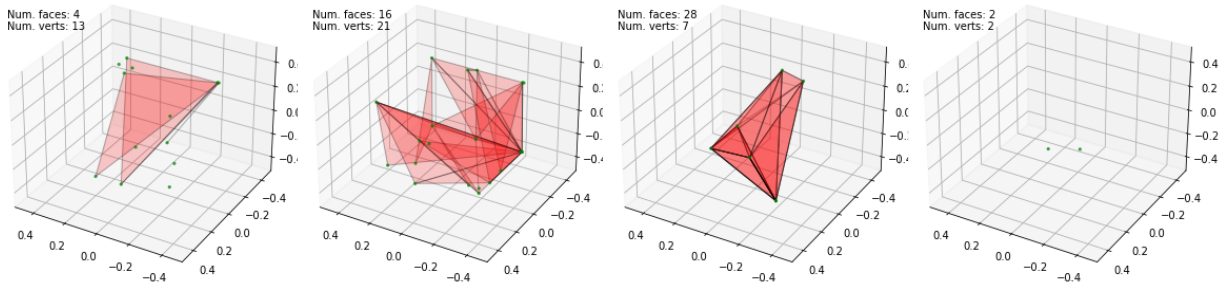
```
def sample_and_plot(vertex_model, vertex_batch, face_model):
    with torch.no_grad():
        vertex_samples = vertex_model.sample(context = vertex_batch, num_sampl
                                             top_p = 0.95, recenter_verts = False,
        max_vertices = torch.max(vertex_samples["num_vertices"]).item()
        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 [ ]:
```
def sample_and_plot_vertices(vertex_model, vertex_batch):
    with torch.no_grad():
        vertex_samples = vertex_model.sample(context = vertex_batch, num_sampl
                                             max_sample_length = 200, top_p = 0

    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()
        mesh_list.append({'vertices': vertices})

    data_utils.plot_meshes(mesh_list, ax_lims = 0.5)
```

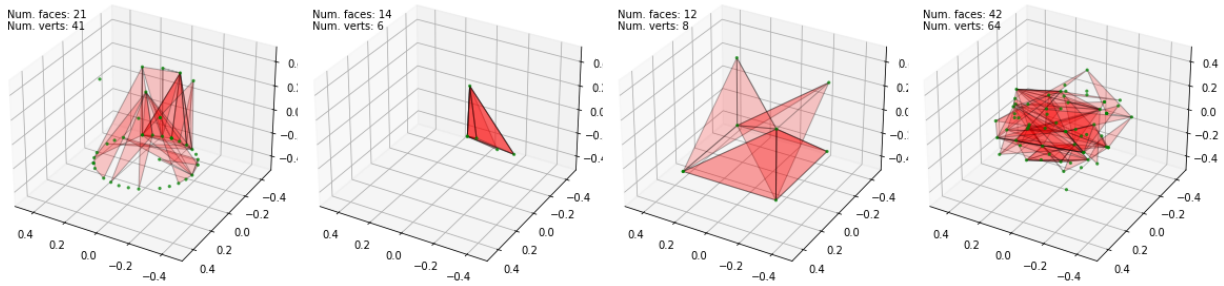
In [ ]:
```
def sample_and_plot_faces(face_model, face_batch):
    with torch.no_grad():
        face_samples = face_model.sample(context = face_batch, max_sample_leng
    mesh_list = []
    for i in range(face_samples["faces"].shape[0]):
        curr_faces = face_samples["faces"][i]
        num_face_indices = face_samples['num_face_indices'][i]
        curr_faces = data_utils.unflatten_faces(curr_faces[:num_face_indices].
        vertices = face_batch["vertices"][i].numpy()
        mesh_list.append({'vertices': vertices, 'faces': curr_faces})

    data_utils.plot_meshes(mesh_list, ax_lims = 0.5)
```

In [ ]:
```
"""Joint Train Vertex and Face Model"""
vertex_model, face_model = load_models()
epochs = 1000
vertex_model_optimizer = vertex_model.configure_optimizers()["optimizer"]
face_model_optimizer = face_model.configure_optimizers()["optimizer"]

for i in range(epochs):
    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 = torch.distributions.categorical.Categorical(logits=vert
    face_pred_dist = torch.distributions.categorical.Categorical(logits = face

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

    vertex_loss.backward()
    face_loss.backward()

    vertex_model_optimizer.step()
    face_model_optimizer.step()

    if (i + 1) % 100 == 0:
        print(f"Epoch {i + 1}: Vertex loss = {vertex_loss.item()}, Face loss =
        sample_and_plot(vertex_model, vertex_batch, face_model)
```

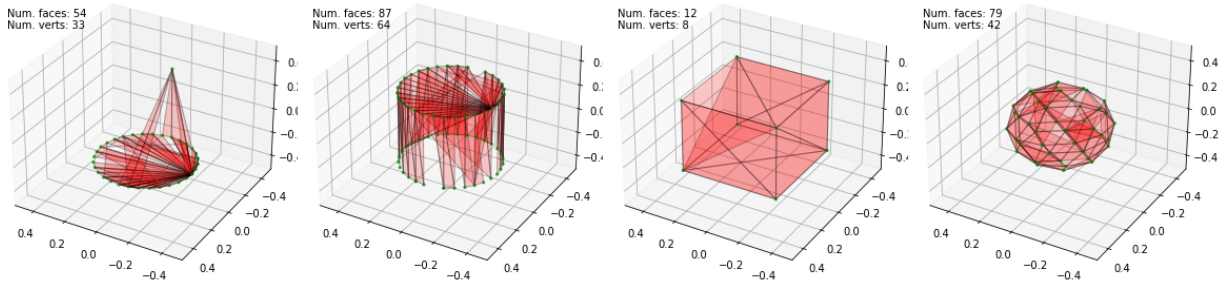
```
/nethome/aahluwalia30/anaconda3/envs/polygen-env/lib/python3.7/site-packages/t
orch/autograd/__init__.py:132: UserWarning: CUDA initialization: Found no NVID
IA driver on your system. Please check that you have an NVIDIA GPU and install
ed a driver from http://www.nvidia.com/Download/index.aspx (Triggered internal
ly at  /opt/conda/conda-bld/pytorch_1607370156314/work/c10/cuda/CUDAFunctions.
cpp:100.)
  allow_unreachable=True)  # allow_unreachable flag
Epoch 100: Vertex loss = 703.0980224609375, Face loss = 2274.43603515625
```



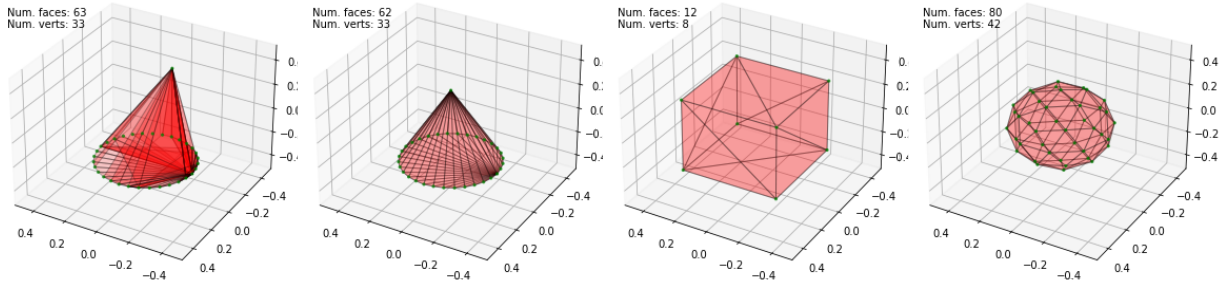

```
Epoch 200: Vertex loss = 62.88766098022461, Face loss = 488.6462097167969
```



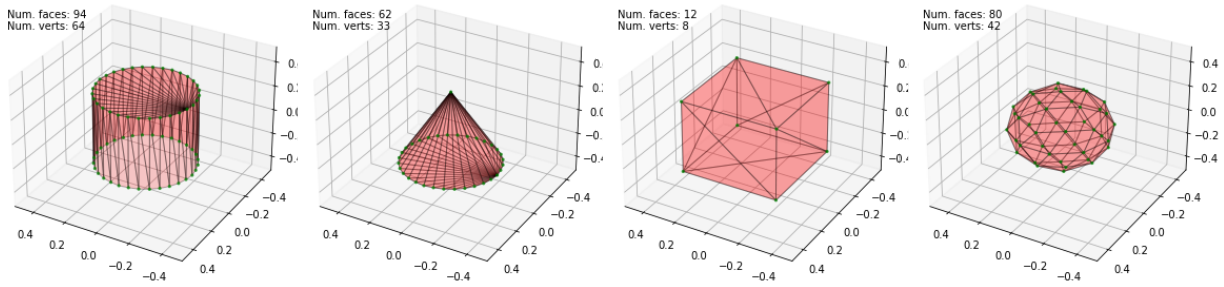

```
Epoch 300: Vertex loss = 12.346000671386719, Face loss = 54.125797271728516
```



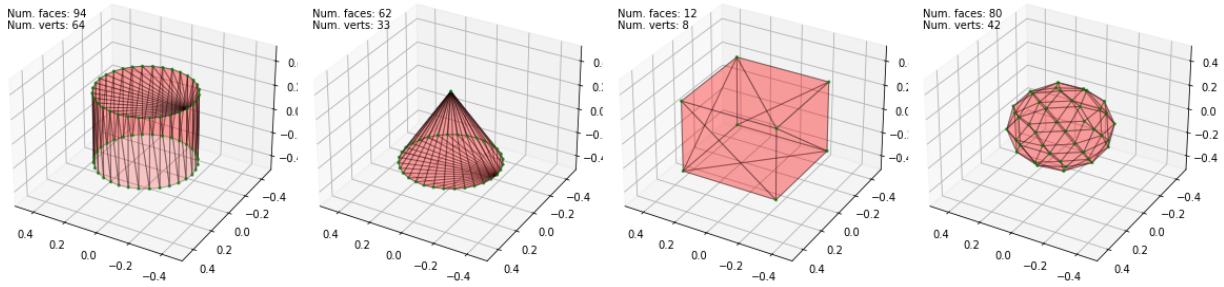

```
Epoch 400: Vertex loss = 4.828774452209473, Face loss = 14.66518497467041
```



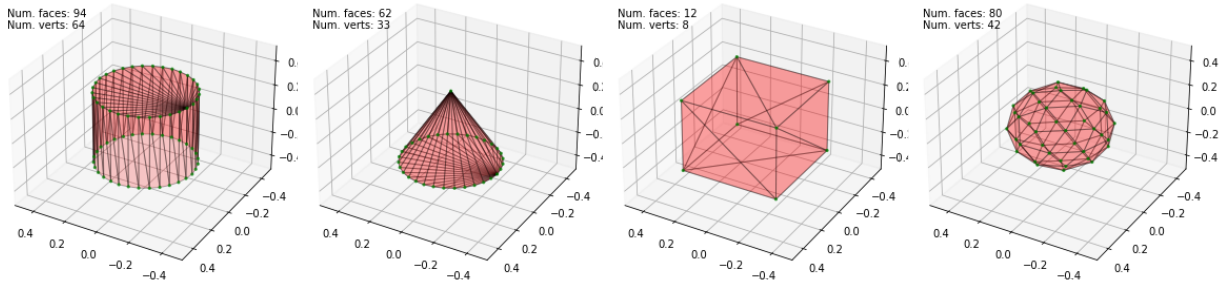

```
Epoch 500: Vertex loss = 2.8105287551879883, Face loss = 7.2781982421875
```



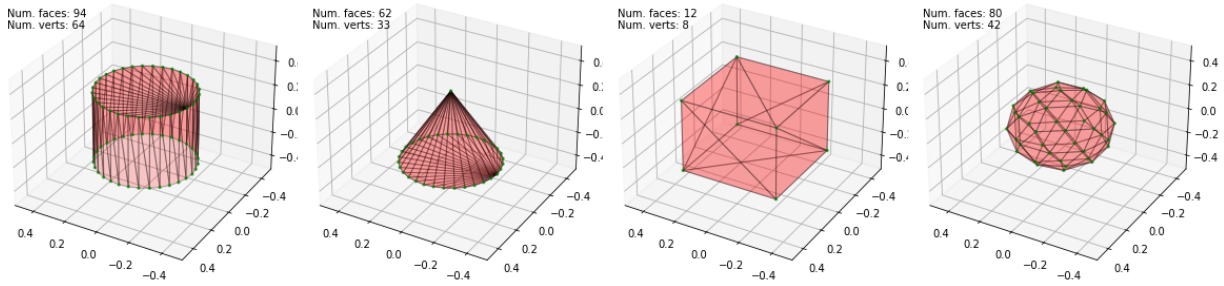
Epoch 600: Vertex loss = 1.8900117874145508, Face loss = 4.429965972900391



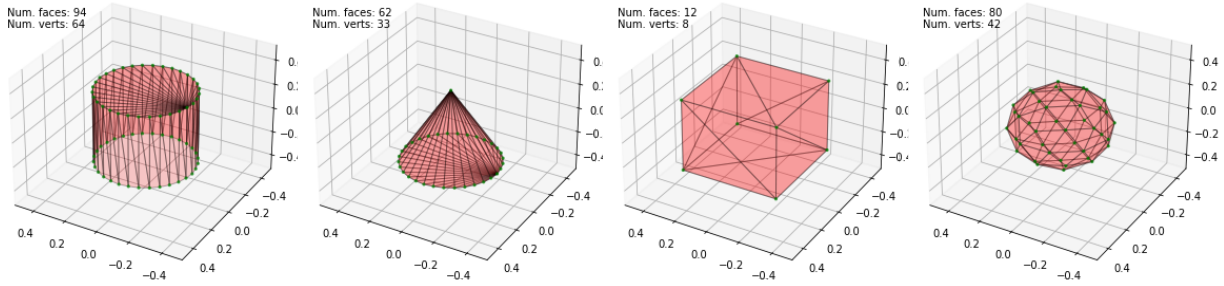
Epoch 700: Vertex loss = 1.364004135131836, Face loss = 2.933084487915039



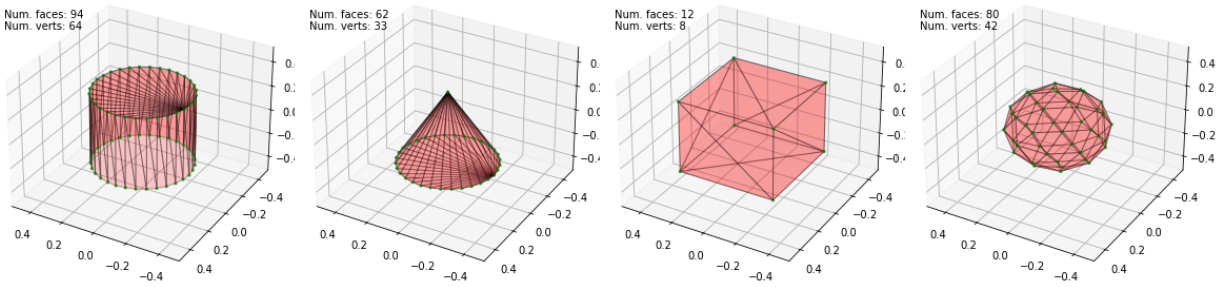
Epoch 800: Vertex loss = 1.0395793914794922, Face loss = 3.3917274475097656



Epoch 900: Vertex loss = 0.8192691802978516, Face loss = 1.661198616027832



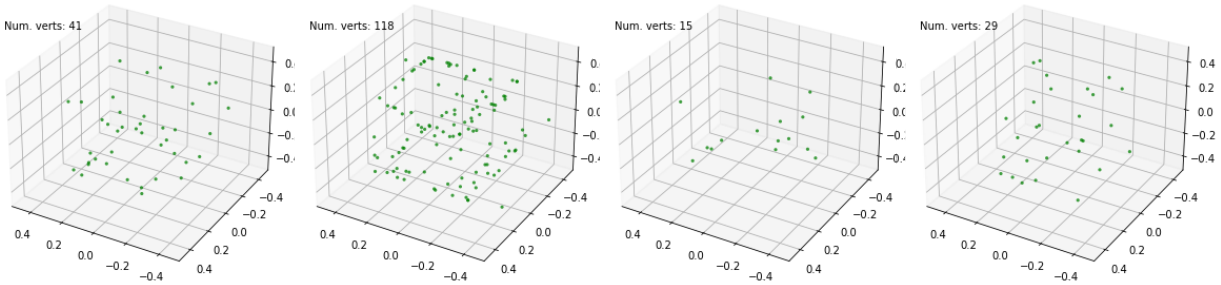
Epoch 1000: Vertex loss = 0.6636209487915039, Face loss = 1.3942031860351562



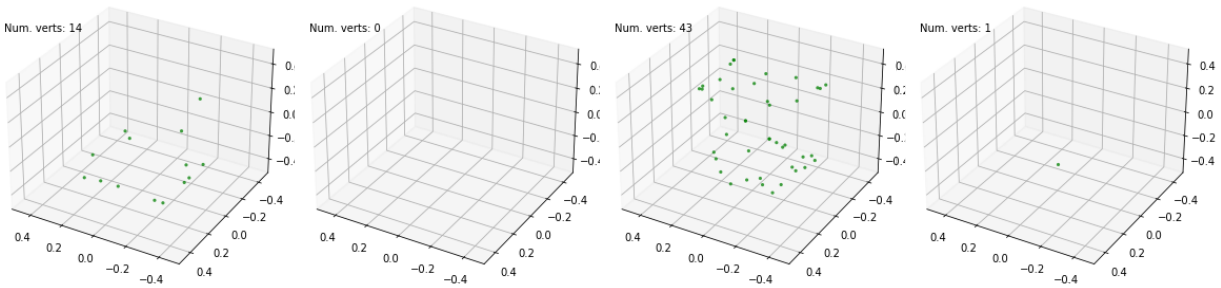
In [ ]:
```
vertex_model, face_model = load_models()
```

In [ ]:
```
"""Train Vertex Model"""
epochs = 500
vertex_model_optimizer = vertex_model.configure_optimizers()["optimizer"]

for i in range(epochs):
    vertex_model_optimizer.zero_grad()
    vertex_logits = vertex_model(vertex_batch)
    vertex_pred_dist = torch.distributions.categorical.Categorical(logits=vert
    vertex_loss = -torch.sum(vertex_pred_dist.log_prob(vertex_batch["vertices_
    vertex_loss.backward()
    vertex_model_optimizer.step()

    if (i + 1) % 50 == 0:
        print(f"Epoch {i + 1}: Vertex Loss = {vertex_loss.item()}")
        sample_and_plot_vertices(vertex_model, vertex_batch)
```

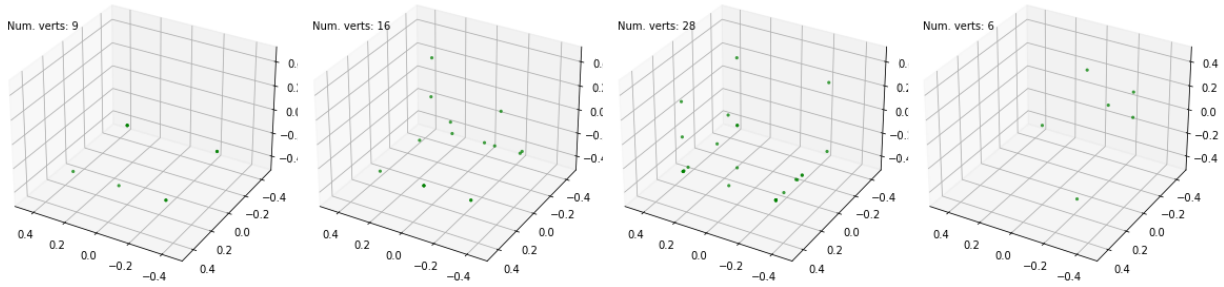
```
Epoch 50: Vertex Loss = 1240.1552734375
```



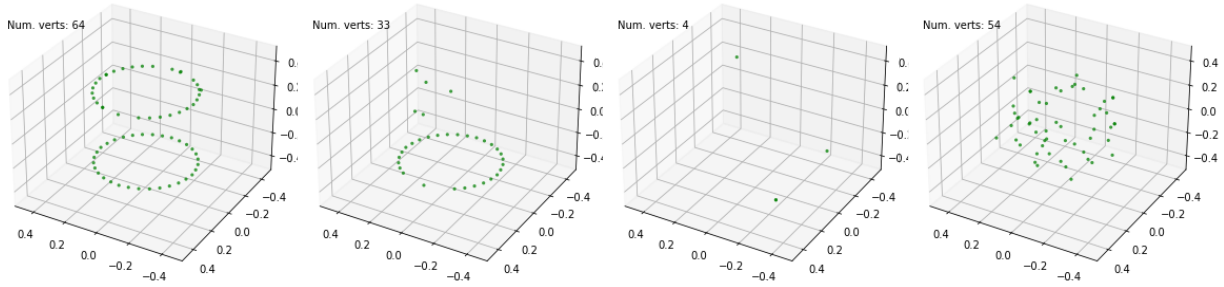

```
Epoch 100: Vertex Loss = 707.2379150390625
```



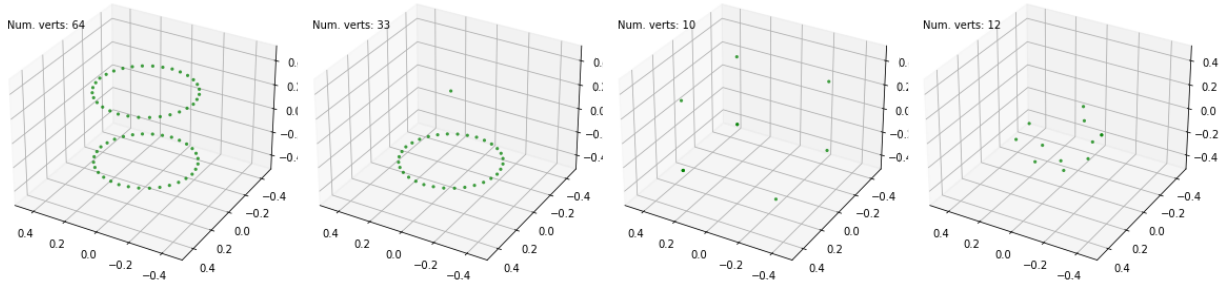

```
Epoch 150: Vertex Loss = 209.0985107421875
```



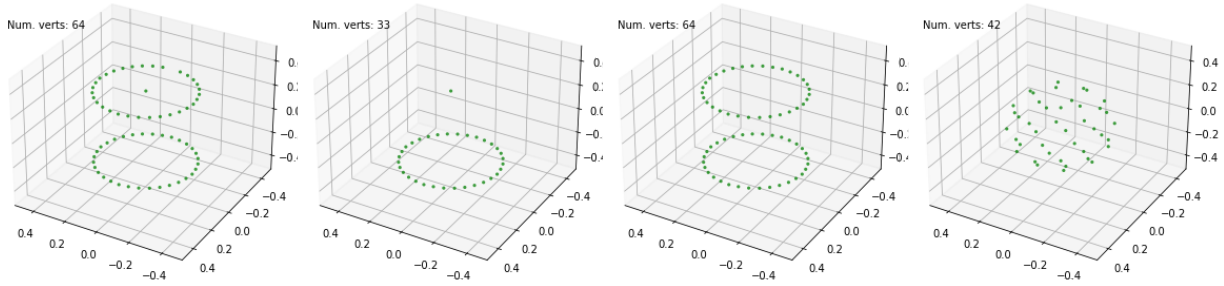
Epoch 200: Vertex Loss = 48.60784149169922



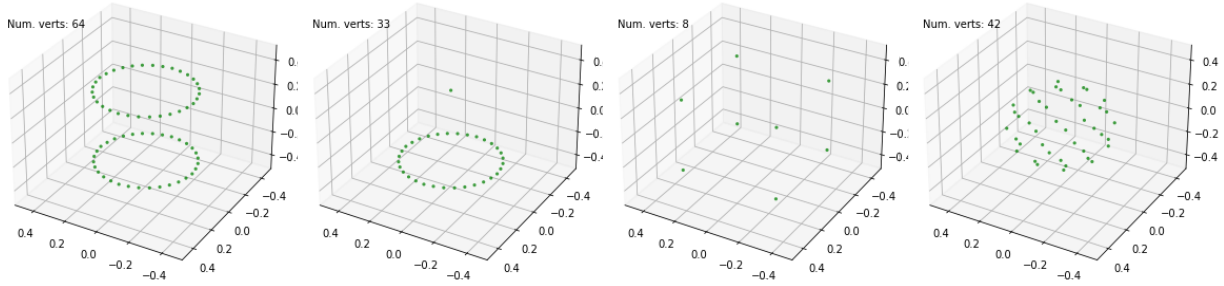
Epoch 250: Vertex Loss = 18.96495819091797



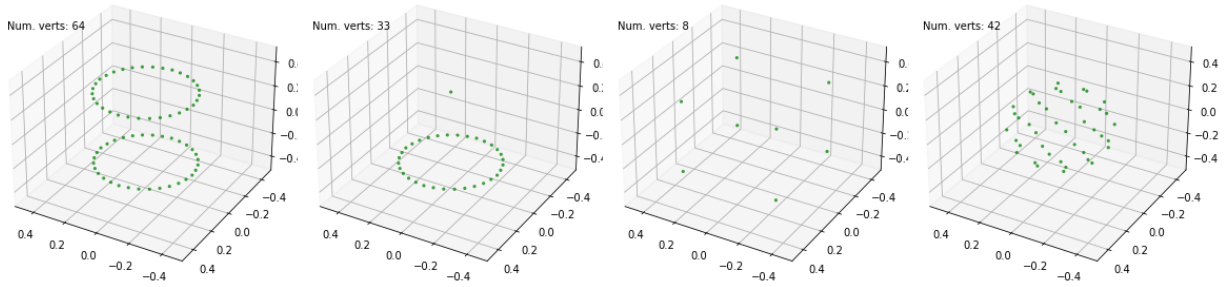
Epoch 300: Vertex Loss = 10.349289894104004



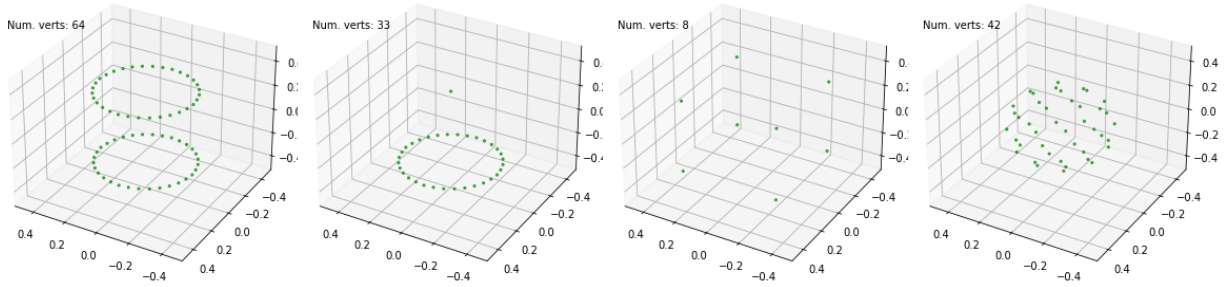
Epoch 350: Vertex Loss = 6.691725730895996



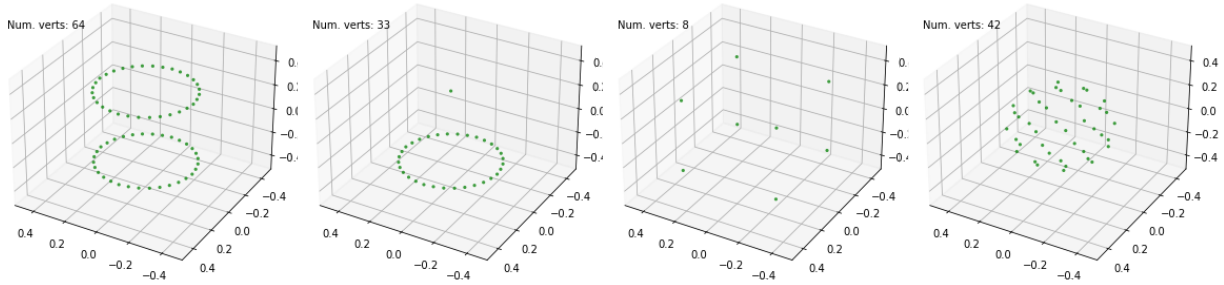
Epoch 400: Vertex Loss = 4.746973037719727




```
Epoch 450: Vertex Loss = 3.4720287322998047
```



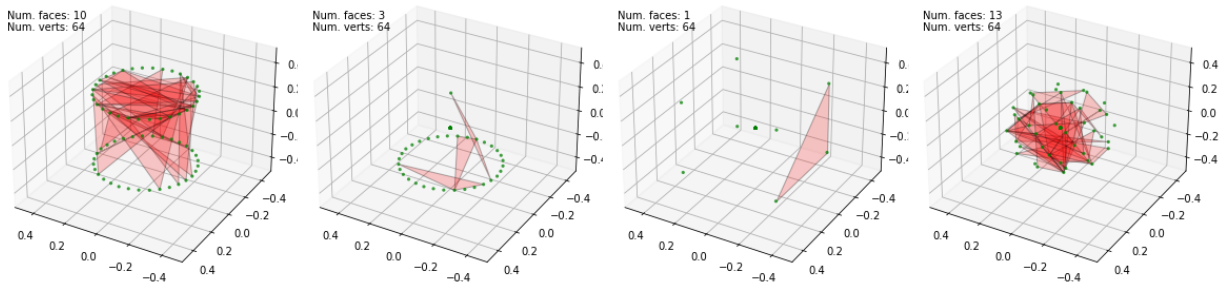

```
Epoch 500: Vertex Loss = 2.7114200592041016
```



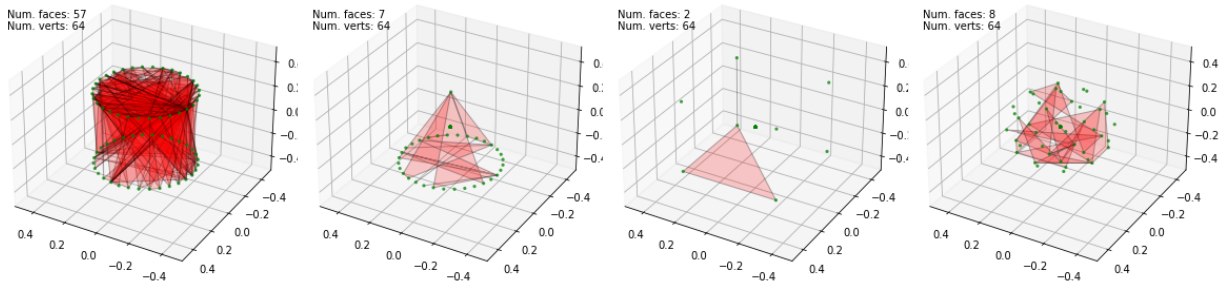
In [ ]:

```
"""Train Face Model"""
epochs = 500
face_model_optimizer = face_model.configure_optimizers()["optimizer"]
for i in range(epochs):
    face_model_optimizer.zero_grad()
    face_logits = face_model(face_batch)
    face_pred_dist = torch.distributions.categorical.Categorical(logits = face
    face_loss = -torch.sum(face_pred_dist.log_prob(face_batch["faces"])* face_
    face_loss.backward()
    face_model_optimizer.step()

    if (i + 1) % 50 == 0:
        print(f"Epoch {i + 1}: Face Loss = {face_loss.item()}")
        sample_and_plot_faces(face_model, face_batch)
```

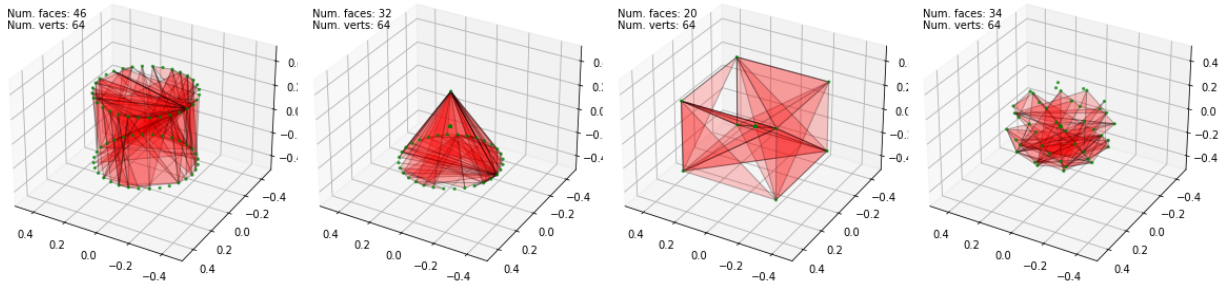
```
Epoch 50: Face Loss = 2934.786376953125
```



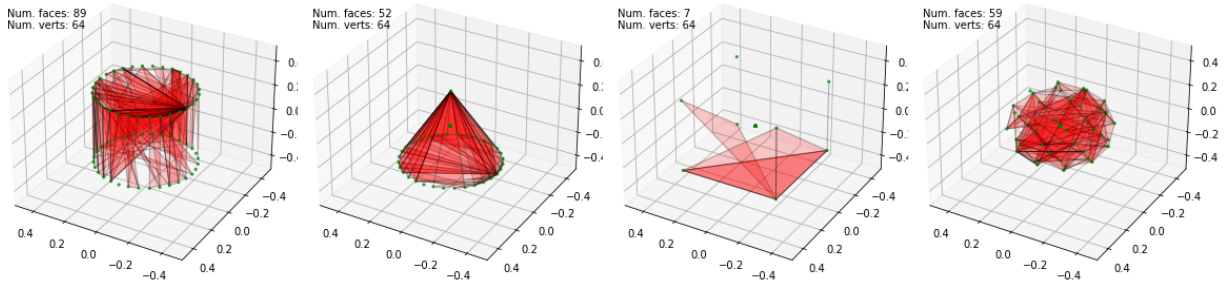

```
Epoch 100: Face Loss = 2222.126708984375
```



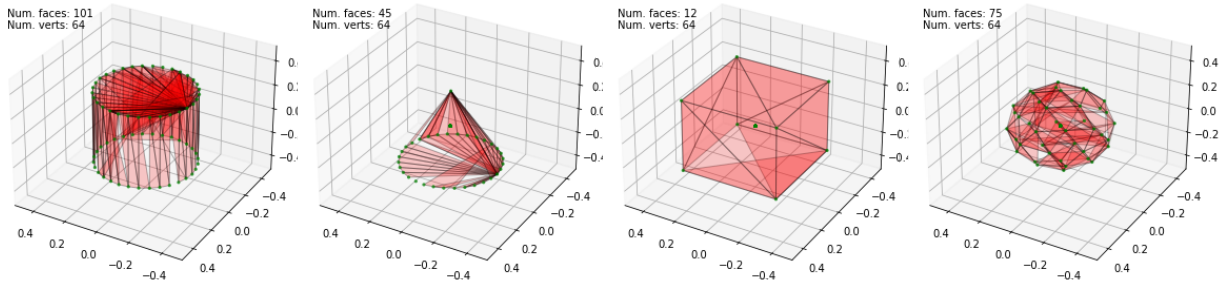
Epoch 150: Face Loss = 1217.81689453125



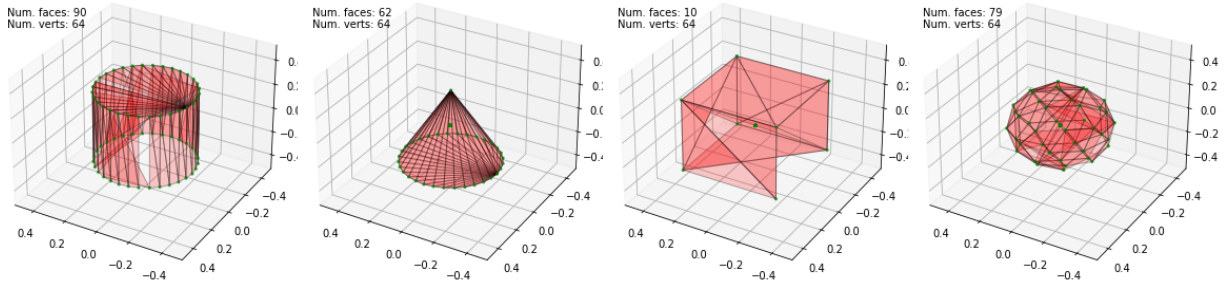
Epoch 200: Face Loss = 463.4594421386719



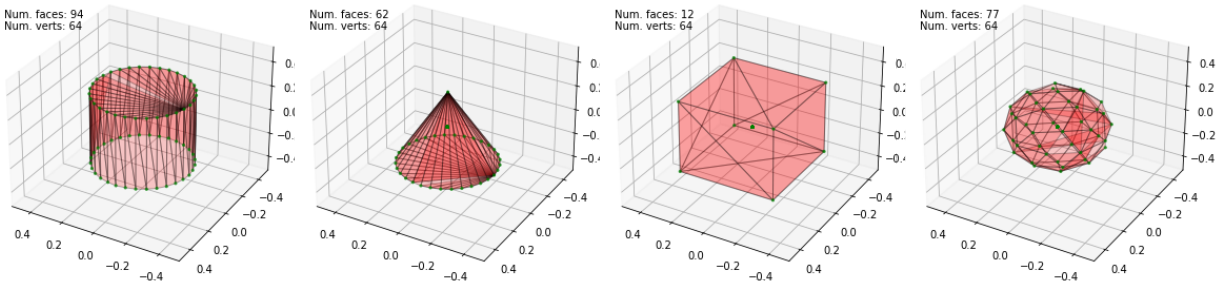
Epoch 250: Face Loss = 135.11849975585938



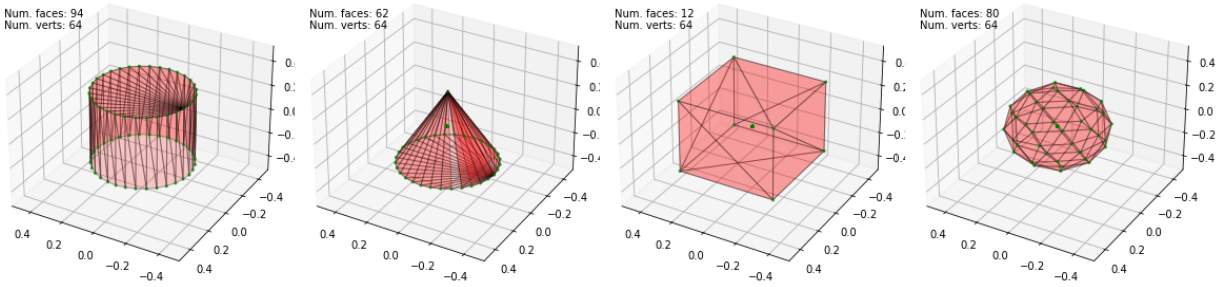
Epoch 300: Face Loss = 45.82008743286133



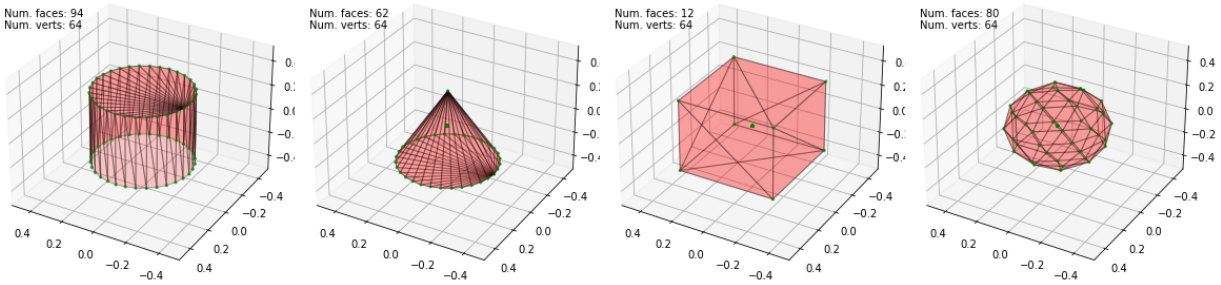
Epoch 350: Face Loss = 19.490257263183594



Epoch 400: Face Loss = 11.451800346374512



Epoch 450: Face Loss = 8.232046127319336



Epoch 500: Face Loss = 6.607647895812988