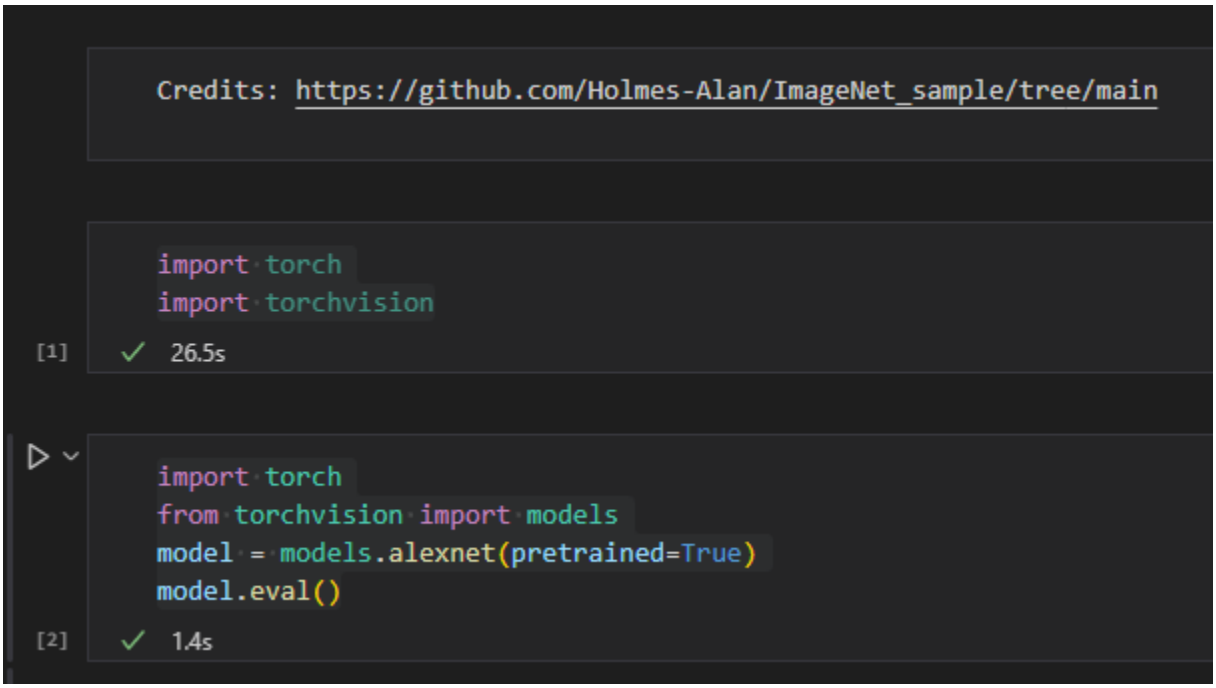
1. Report two examples when AlexNet cannot correctly predict the class of the input image. You need to upload your results (e.g., screenshots) and try to explain why it fails.

2. Take this image (link) as input and show your results. Can you explain your findings about it?

1.



```python
# sample execution (requires torchvision)
from PIL import Image
import matplotlib.pyplot as plt
from torchvision import transforms

filename = './there-are-two-animals-that-are-standing-woods-together-generative-ai_1034036-66762.jpg'

input_image = Image.open(filename)
plt.imshow(input_image)
plt.axis('off')

preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)

# normalize the output as probability
percentage = torch.nn.functional.softmax(output, dim=1)[0] * 100

# Load Imagenet Synsets
with open('./imagenet_synsets.txt', 'r') as f:
    synsets = f.readlines()

# len(synsets)==1001
# sysnets[0] == background
synsets = [x.strip() for x in synsets]
splits = [line.split(' ') for line in synsets]
key_to_classname = {spl[0]:' '.join(spl[1:]) for spl in splits}

with open('./imagenet_classes.txt', 'r') as f:
    class_id_to_key = f.readlines()

class_id_to_key = [x.strip() for x in class_id_to_key]

print('The best prediction:\n')
_, index = torch.max(output, 1)
classname = key_to_classname[class_id_to_key[index[0]]]
probability = percentage[index[0]].item()
print("'{}': {}% is a '{}'".format(filename, probability, classname))

print('\nTop 5 prediction:\n')
_, indices = torch.sort(output, descending=True)
for idx in indices[0][:5]:
    print("{}% is a '{}'".format(percentage[idx].item(), key_to_classname[class_id_to_key[idx]]))
```

✓ 0.2s

```
The best prediction:
```

```
The best prediction:

'./there-are-two-animals-that-are-standing-woods-together-generative-ai_1034036-66762.jpg': 29.15629768371582% is a 'hog, pig, grunter, squealer, Sus scrofa'

Top 5 prediction:

29.15629768371582% is a 'hog, pig, grunter, squealer, Sus scrofa'
13.493433952331543% is a 'wild boar, boar, Sus scrofa'
7.97960376739501953% is a 'red fox, Vulpes vulpes'
7.413844108581543% is a 'timber wolf, grey wolf, gray wolf, Canis lupus'
5.760602951049805% is a 'kit fox, Vulpes macrotis'
```

I got two images that are not the image given in number 2. One is a complex AI generated nature scene which has multiple animals in it. It clearly has a wolf in it and one would assume that the image would probably generate wolf as the first or sheep.

Why it most likely fails is because it is AI generated and it's too complex (no exact target). This could lead the pretrained AI to make false assumptions since its training data has been of presumably real life images (which don't have the features that AI image generators naturally make).

```python
# sample execution (requires torchvision)
from PIL import Image
import matplotlib.pyplot as plt
from torchvision import transforms

filename = './imageofpink.png'

input_image = Image.open(filename)
plt.imshow(input_image)
plt.axis('off')

preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)

# normalize the output as probability
percentage = torch.nn.functional.softmax(output, dim=1)[0] * 100

# Load Imagenet Synsets
with open('./imagenet_synsets.txt', 'r') as f:
    synsets = f.readlines()

# len(synsets)==1001
# sysnets[0] == background
synsets = [x.strip() for x in synsets]
splits = [line.split(' ') for line in synsets]
key_to_classname = {spl[0]:' '.join(spl[1:]) for spl in splits}

with open('./imagenet_classes.txt', 'r') as f:
    class_id_to_key = f.readlines()

class_id_to_key = [x.strip() for x in class_id_to_key]

print('The best prediction:\n')
_, index = torch.max(output, 1)
classname = key_to_classname[class_id_to_key[index[0]]]
probability = percentage[index[0]].item()
print("'{}': {}% is a '{}'".format(filename, probability, classname))

print('\nTop 5 prediction:\n')
_, indices = torch.sort(output, descending=True)
for idx in indices[0][:5]:
    print("{}% is a '{}'".format(percentage[idx].item(), key_to_classname[class_id_to_key[idx]]))
```

✓ 0.1s

```
    print("{}% is a '{}'".format(percentage[idx].item(), key_to_classname[class_id_to_key[idx]]))
```

✓ 0.1s

```
The best prediction:

'./imageofpink.png': 50.9422492980957% is a 'cock'

Top 5 prediction:

50.9422492980957% is a 'cock'
19.639593124389655% is a 'hen'
4.92436933517456055% is a 'ostrich, Struthio camelus'
3.61755371093755% is a 'baboon'
3.216555356979375% is a 'patas, hussar monkey, Erythrocebus patas'
```

The other image fails since it clearly depicts a pink llama. It was AI generated, so this could cause issue, but the main issue most likely is the fact that it's of a different colour than an actual llama. The AI is most likely not able to recognize that it's not a rooster because it doesn't know what a X coloured Y animal would look like due to the training data given to it. As a human you can tell if it's unnatural and can think outside of the box and not rely on things such as colour but only shape. While the AI might not.

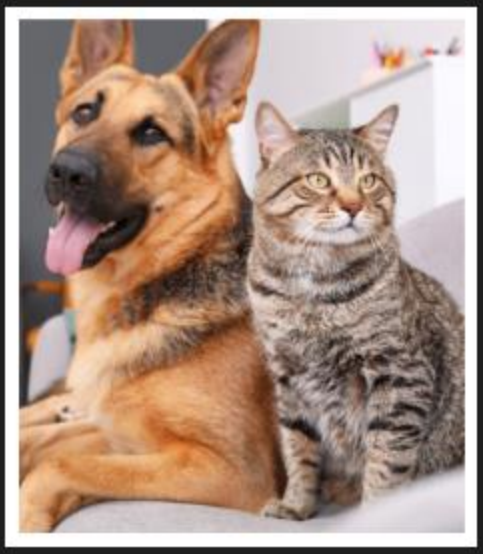
2.

```python
# sample execution (requires torchvision)
from PIL import Image
import matplotlib.pyplot as plt
from torchvision import transforms

filename = './living-with-cats-dogs-689902.jpg'

input_image = Image.open(filename)
plt.imshow(input_image)
plt.axis('off')

preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)

# normalize the output as probability
percentage = torch.nn.functional.softmax(output, dim=1)[0] * 100

# Load Imagenet Synsets
with open('./imagenet_synsets.txt', 'r') as f:
    synsets = f.readlines()

# len(synsets)==1001
# sysnets[0] == background
synsets = [x.strip() for x in synsets]
splits = [line.split(' ') for line in synsets]
key_to_classname = {spl[0]:' '.join(spl[1:]) for spl in splits}

with open('./imagenet_classes.txt', 'r') as f:
    class_id_to_key = f.readlines()

class_id_to_key = [x.strip() for x in class_id_to_key]

print('The best prediction:\n')
_, index = torch.max(output, 1)
classname = key_to_classname[class_id_to_key[index[0]]]
probability = percentage[index[0]].item()
print("'{}': {}% is a '{}'".format(filename, probability, classname))

print('\nTop 5 prediction:\n')
_, indices = torch.sort(output, descending=True)
for idx in indices[0][:5]:
    print("{}% is a '{}'".format(percentage[idx].item(), key_to_classname[class_id_to_key[idx]]))
```

[3] ✓ 4.4s

```
The best prediction:

'./living-with-cats-dogs-689902.jpg': 26.3090248107910l6% is a 'tiger cat'
```

```
The best prediction:

'./living-with-cats-dogs-689902.jpg': 26.3090248107910l6% is a 'tiger cat'

Top 5 prediction:

26.3090248107910l6% is a 'tiger cat'
13.073825836181l64% is a 'tabby, tabby cat'
12.5662622451782l3% is a 'German shepherd, German shepherd dog, German police dog, alsatian'
12.4810466766357l2% is a 'Egyptian cat'
8.6255874633789l6% is a 'tiger, Panthera tigris'
```



The image given via the link gives conflicting results. It shows an image of a german shepherd and a tabby cat. While it does recognize that these things might exist in the image, since it needs to give one result only it has to get creative or make a choice between the two. I assume that it thought tiger cat because the tabby cat has features similar to a tiger cat, and maybe it also tried to assume "tiger" and "cat" due to the orange colour from the german shepherd combined with the tabby cat's stripes. Since the AI model is trained to try and give only one answer it tries to take in everything instead of describing the scene (which some other AIs are able to do).