Report 1: Project Overview
Christopher Clark

1. Introduction:

In this project, we will be exploring the use Conformal Uncertainty Quantification (CUQ) in classification and segmentation. In particular, we are going to delve into the *SegFormer* model from HuggingFace, found here: https://huggingface.co/docs/transformers/v4.28.1/model_doc/segformer. This is a pre-trained model which can be called on for semantic segmentation and/or classification. To this end, we will introduce the relevant datasets used to fine-tune the model and which we will be using to develop our intuition for CUQ.

The dataset used for classification fine-tuning is ImageNet-1k, which can be found here: https://huggingface.co/datasets/imagenet-1k

For semantic segmentation, we have the following:
1. ADE20K, https://groups.csail.mit.edu/vision/datasets/ADE20K/
2. Cityscapes, https://www.cityscapes-dataset.com/
3. COCO-stuff, https://github.com/nightrome/cocostuff

In the ImageNet-1k, we have 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. The images are in some snese "natural", i.e., images of everyday objects one might encounter throughout their life. This is a common dataset to test models against for this reason.

In ADE20k, we have 27,574 images (25,574 for training and 2,000 for testing) spanning 365 different scenes, 707,868 unique objects from 3,688 categories, along with their WordNet definition and hierarchy, amd 193,238 annotated object parts and parts of parts.

In CityScapes, we have 30 classes and 25,000 annotated images, 5,000 with fine annotations and 20,000 with courser ones. The images are taken from 50 cities in daytime and throughout the year.

In COCO-Stuff, we have 164K images which have been hand-annotated for what is in each image. There are 80 thing classes, 91 stuff classes and 1 class 'unlabeled for a total of 172 classes of segmented objects.

We would like to test our pre-trained SegFormer model on these datasets and learn about this model's uncertainty in its classifications and segmentations.

2. General Pipeline:

In this section, we will talk a bit about the general data pipeline.
To begin, we will talk a bit about the SegFormer model. The original paper is *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers* and can be found here: https://arxiv.org/abs/2105.15203.

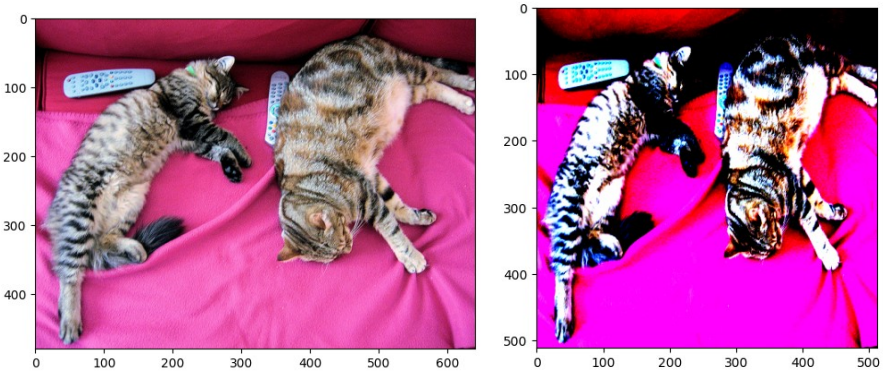
From the HuggingFace website, "The model consists of a hierarchical Transformer encoder and a lightweight all-MLP decode head to achieve great results on image segmentation benchmarks such as

ADE20K and Cityscapes." Of course, we can also call it with a classification head. There are six variants (essentially, different sizes) in the repo, with titles such as "MiT-bn", where "n" ranges from 0 to 5. On the above page, we can see their general structure, size, and performance on the ImageNet-1k dataset.

Of particular interest to us, when we load in a model, we can also load in the image processing technique used to train the model. Thus, we can determine some of the particulars of a dataset. For example, let's see the means and standard deviations across the channels for the ImageNet-1k dataset.

SegformerImageProcessor {"do_normalize": true, "do_reduce_labels": false, "do_rescale": true, "do_resize": true, "image_mean": [0.485, 0.456, 0.406],
"image_processor_type": "SegformerImageProcessor",
  "image_std": [0.229, 0.224, 0.225],
  "resample": 2,
  "rescale_factor": 0.00392156862745098,
  "size": {"height": 512, "width": 512}}

With this, we can easily transform any image so that it fits the data distribution on which the model was trained. For example, we take the image on the left and transform it with the SegformerImageProcessor above to get:



3. Models

Fortunately, HuggingFace makes it easy to load in our models with commands such as:

```
seg_model = SegformerForSemanticSegmentation.from_pretrained(pretrained_checkpoint, num_labels = 10)
clf_model = SegformerForImageClassification.from_pretrained(pretrained_checkpoint)
```

From the above we can easily get our outputs:

Shape of the raw segmentation model output: torch.Size([1, 10, 128, 128])
Shape of the upsampled logits: torch.Size([1, 10, 512, 512])

Shape of the raw classification model output: torch.Size([1, 1000])
Label corresponding to the argmax: tabby, tabby cat

So, with these tools, we can explore the models' behavior and attempt to quantify their uncertainty.