Chapter 1

Methodology

In this chapter we will discuss how we created the dataset used in our research, what the dataset contains and the purpose for which we created it.

1.1 Data collection

As discussed in Section **??** there is number of publicly available datasets online, and some which are restricted. Some of these datasets are difficult to access, and there is a need for more publicly available datasets which are collected for the purpose of deep learning. To assist the under-explored field of research within medical computer assisted analysis tools the datasets need to be large and well annotated. Some of the mentioned datasets lack adequately documented, annotated samples from a good source and is not well suited for our research. Thus, as a vital part of our research, we aim to produce a collection of well annotated and adequately big dataset that can be used not only in this study, but also contribute to the research community and have a impact on the research comparability in future. We achieve this by collecting medical data, sorting and annotating it and making the dataset publicly available and free for non-commercial, educational and research purposes.

1.1.1 Privacy, Legal and Ethics Issues

To obtain medical videos from a hospital in Norway is very difficult and not straight forward. All medical data is considered personal and is therefore strongly protected from unauthorized use and distribution by the **Pasientjournalloven??** legislation. A medical study conducted at 2 academic hospitals from May 2017 to September 2018 found that most patients are willing to share their data and biospecimens for research purposes [1]. Regardless of the patient opting in to share their data and biospecimens it is difficult for researchers to get their hands on it. We solved this problem by collaborating with a number of Norwegian hospitals and research teams working there. One of the research teams we collaborated with is Augere Medical (See Section **??** for more info), and through them we got in contact with Vestre Viken Hospital Trust, allowing our research team to download anonymous data from hospital systems and transfer it using secure media to our facility. Upon downloading the data we further stripped the metadata files for potential information regarding patients like time stamps, dates and camera equipment used.

1.1.2 Kvasir-PillCam

The dataset we used in our experiments consist of endoscopic videos collected from Bærum Hospital, a hospital in Vestre Viken Hospital Trust. Unlike Kvasir and Hyper-Kvasir datasets we have made the Kvasir-PillCam dataset for the purpose of this thesis. Initially we received 44 VCE videos, which were first reviewed by a clinician, whom selected thumbnails of region of interests of both lesions and normal findings. The videos were then exported from Vestre Viken Hospital Trust and re-encoded to an open source file format (MP4?), from proprietary Sony technology format (whats the name?) Prior to being exported the videos were anonymized by removing all metadata and renaming the files with randomly generated file names. A few videos had to be shortened to cut out images taken by the capsule before entering the mouth of the patient. After that the videos are

Class name	# samples	Percentage
Angiectasia	908	xx,x
Blood	658	xx,x
Erosion	525	xx,x
Erythematous	251	xx,x
Foreign Bodies	776	xx,x
Hematin	12	xx,x
Ileo-cecal valve	4704	xx,x
Normal	33,129	xx,x
Polyp	583	xx,x
Pylorus	1410	xx,x
Ulcer	759	xx,x
Unknown	190	xx,x

Table 1.1: Hyper-PillCam class names, corresponding amount of samples and the percentage of samples in each class.

uploaded to Augere Medical AS[1] tagging tool. Three MSc students went through all the frames of the videos in collaboration with an expert and labeled and marked findings with bounding boxes. When the student encountered images they were uncertain of the expert reviewed the case.

There are a total of XX different findings in the dataset. The findings are split in two different types; anatomical landmarks and pathological findings. Anatomical landmarks consist of two categories:

- Pylorus valve; where the capsule enters the small intestine from the stomach.
- Ileocecal valve; the transition from small intestine to large intestine.

Pathological findings consist of:

- Protruding lesions; polyps, varices and tumors.
- Excavated lesions; scars, ulcers, erosions, angiectasia, and dieulafoy lesions.
- Mocusa; edematous, scalloping, hemorrhagic, erythematous, and ulcerated mucosa.
- Normal; no pathological finding.

We have also included one category for foreign bodies which includes images with things found inside the small intestine which does not fit in any of the previously mentioned categories. This includes thing like other VCE-devices and tables. The images annotated with the *normal* tag is meant to be used for binary classification where all the others classes are combined into a single class for *finding* and one class for no findings. *Normal* images are hand picked from portions of video where there are no findings, and taken from each video to get a diverse category.

When all 44 videos have been precisely labeled the dataset is exported from Augere Medical tagging tool and split into folders for each class. The number of images per class are given in Table 1.1. In total we have 43,905 labeled images in 12 classes. The sample distribution across the classes is skewed depending on how many findings there are in the videos. Some findings occur often and some very rarely.

Imbalanced dataset pose a challenge for predictive algorithms as most learning algorithms are based on the assumption of an equal number of samples for each class. This results in models that have poor predictive performance, especially for minority class or classes. This is a great problem because in many medical datasets the minority class is the most important and therefore more sensitive for classification errors.

In addition to labeling the images the dataset also contain a JSON format file which stores coordinates for where in the frame the finding is located. The Kvasir-PillCam dataset will be an open-source dataset available for others scientists, and will later be grown to include more PillCam videos, both labeled and unlabeled samples.

Later, we received an additional XX videos. These videos were not annotated but used for unlabeled data. (and/or sample-videos?) These videos were processed exactly the same as previous videos, so that it will be compatible (**describe**) with the annotated dataset.

[1]`https://augere.md/`

1.2 Data pipeline

The preprocessing pipeline is implemented by using TensorFlows *data.Dataset* library. This was chosen over *datagenerator* due to the easy of use, but later became an issue due to the complexity and some diffuse runtime errors. The main benefits by using data.Dataset is that all data is handled in tensors and computation is automatically distributed to the GPU which enhance the load distribution of processing power.

The pipeline outline can be seen in Figure XX.

All the code for handling the data pipeline is managed in a separate script called pipeline.py. Psuedo code for the script is:

1.2.1 Data preprocessing

- When splitting the dataset it is important to ensure minority classes are represented in both training, validation and test dataset. Found that opencv resizes images a bit faster (1.2s vs 1.6 for pillow)
- What is the necessity of rotating the samples. Will they not always be 'square'?
- Normalization; any benefit of using L2 normalization in last dense layer when using a predefined model?

1.2.2 Splitting dataset

Initially the dataset was split into three; training data, test data, and validation data. This was done by using tf.data.Dataset core operations *take* and *skip*. The take function, when called upon, returns a sub-dataset with the same number of samples as the number it receives as a argument. Skip function returns a sub-dataset where it skips the number of elements as stated in input argument and return the remaining samples.

```
>>> dataset = tf.data.Dataset.range(10)
>>> dataset = dataset.skip(7)
>>> list(dataset.as_numpy_iterator())
[7, 8, 9]
```

However, because of the imbalanced dataset we have been working with this turned out to drop minority classes in some runs. This happened because the take and skip methods picks samples from the entire dataset as whole, and not per class. In Hyper-Kvasir the class with smallest number of samples is hemorrhoids with 6 samples, and depending on shuffling these samples would often all end up in one of the dataset splits and not be represented in the other two. To mitigate this issue I created a new script for pre-splitting dataset into sub folders for each, training, testing and validation dataset. The outline of this script is given below.

```
for every class name in directories:
  sort the images
  shuffle the filenames
  split the class into train, test and val
  for each split_ds in datasets:
    for filename in sub-split:
      resize and save the image
```

We split the data into 60% training data, and leave 15% for test data and validation data respectively. To make the split reproducible we first sort the data with alphabetically after file names then use seeded random to shuffle before splitting. In this step we also reduce the image dimensions to 256 by 256 pixels to make the data easier to use during the next preprocessing steps and to reduce dataset file size. In Hyper-Kvasir this size reduction correspond to a magnitude in file size reduction.

1.2.3 Sample the dataset or class weight

Found that when weighting the classes the model learns slower. Which means decay rate of learning rate must be lowered for it not to drop too soon. Also increase the buffer for early stopping of training.

1.2.4 Caching

We call this consumer / producer overlap, where the consumer is the GPU and the producer is the CPU.

1.2.5 Shuffle the data

1.2.6 Repeat

1.2.7 Data augmentation

1.2.8 Batching

1.2.9 Training

1.3 System implementation

1.3.1 Generating new pseudo labels

When we use the trained teacher model to generate new pseudo labels from the dataset we set a threshold for which the predicted image is saved if it is above the set value. Here we have two options, either to set the threshold low and extract a large dataset with high uncertainty, but get the most of our minority classes. Or we can set the threshold high, and gather a dataset with lower uncertainty, but might miss more samples for the minority classes.

After subtracting samples from the unlabeled dataset through predicting and saving images with a probability score higher than the threshold, we sort the images based on the probability scores. This is done so for majority classes with a large amount of samples can be squeezed to fit the minority classes and we can select the samples with the highest probability scores.

The sorted samples from the unlabeled dataset is then resampled based on the original distribution of samples in the training data. Since the original dataset is resampled that means we will pick out X samples for each class to keep the distribution even between classes during training.

Inspecting the pseudo labels

Since the unlabeled dataset don't contain labels, it is difficult to fully comprehend the results of the extracted data.

1.3.2 Learning rate

Many models train better if you gradually reduce the learning rate during training. We are using learning rate which follows a inverse time decay which means that the learning rate is decreased to 1/2 of the base rate at 10 epochs, 1/3 at 20 epochs and so on.

1.3.3 Hyper-parameter tuning

1.4 Summary