Machine Learning: How can active learning solve the lack of labeled images for lung disease recognition

Koen Desplenter KOEN.DESPLENTER@UGENT.BE
Quinten Vervynck QUINTEN.VERVYNCK@UGENT.BE

Abstract

In this paper we'll use pool-based active machine learning to try to reduce the amount of labeled images that are needed for lung disease recognition. We'll test and compare different query strategies, and experiment with enhancing the images.

1. Introduction

Respiratory diseases account for three of the top ten global causes of death according to World Health Organization in 2019 (Organizaton, 2020). Together these diseases claimed 7.7 million lives, accounting for 13.9 percent of global deaths in 2019. These statistics don't even include the impact of COVID-19 that emerged in late December of 2019. As COVID has shown since 2020, lung diseases can have a serious impact on our health. Medical imaging is a very important part in helping to cure patients or to tackle their symptoms. So this a great field to do research in, as it could have great benefits to our society.

1.1. Problem statement

Lung scans are an easy and non evasive way to check for certain diseases. However the differences between the images of a sick person and a healthy one are often very subtle and hard to identify. Next to that it is very time consuming and strongly dependent on the skills of the doctor, to identify the disease of the patient.
Machine learning based classification models can help assist and speedup this disease-identification process. We developed a classification model with active learning that can identify three types of lung diseases, namely healthy, COVID or pneumonia infected lungs.
The reason we choose to add COVID is because it is closely related to pneumonia. We will work with X-ray images since this is the most reliable and most used imaging technique for lung diseases(De Wever et al., 2011).

1.2. Research question

Our research looked into how active learning can create competitive models for lung disease recognition with less labeled data. We also looked into how the querying strategy might impact the total amount of images needed as well as the classification quality. At last we also experimented with the image quality, we enhanced the images, hoping to reduce the amount of labeled data needed to achieve the same results.

2. Prestudy

Before jumping into the project, we conducted a prestudy. We searched for recent papers about lung disease classification in an effort to get a broad perspective of what has already been done. Next to that we also looked for datasets that contained enough high quality images that we could use to train and test our model.
We also read the paper from the project description about active learning and researched more on this topic. Since this is the main matter of our research, we will discuss this topic separately in the next chapter.
In this chapter we'll share the main thoughts of the most interesting papers we've read, and showcase the dataset we will use throughout this paper.

2.1. Related work

Due to the global pandemic, a lot of research can be found on COVID-19 detection in lung scans. One of the first papers we stumbled upon, is a paper that used a VGG16-based Deep Learning model for the identification of pneumonia and COVID-19 in torso radiographs (Civit-Masot et al., 2020). The paper claims a high sensitivity in the identification of COVID-19, around 100%, and with a high degree of specificity. According to the researchers are the AUCs on ROC curves greater than 0.9 for all classes considered.
VGG-16 (Simonyan & Zisserman, 2014) is a type of convolutional neural network that is considered to be one of the best computer vision models to date. The architecture has some very small convolutional layers of

3x3 and consists of a total of 16 weight layers. VGG16 takes images of size 224 x 224 with 3 RGB channels as an input.
The dataset(Wangg, 2020) used in the paper contains 5887 images of which 1965 normal images, 3723 pneumonia images and only 199 covid images. It should be noted that there is high class inbalance in this dataset. One might wonder how the researchers were able to train such an accurate covid classification model with this few images.

Another interesting paper(Goyal & Singh, 2021) we discovered in PubMed Central took a whole different approach to their classification model. In contradiction to most of the image classification models, this paper makes use of soft computing methods such as artificial neural network (ANN), support vector machine (SVM), K-nearest neighbour (KNN), ensemble classifier and deep learning classifier. Their main motivation to differ from the normal approach is the fact that the computational complexity of CNN is higher due to the high-dimensional feature space.
One of the main takeaways from this paper is their critical view on earlier research. The researchers state the importance of image quality enhancement and the lack thereof in other papers. They also propose the use of a region of interest, ROI, in order to restrict feature extraction to the lungs. The paper also warns about the use of models that are pre-trained on irrelevant datasets like ImageNet. These are all things we kept in mind in our project.

2.2. Description of the dataset

With the knowledge we got from the papers, we started looking for a dataset that has good class balance, contains multiple thousands of images and preferably had ROI's for every image.
Our dataset (Kaggle, 2020) contains 15152 CXR images that were collected from various Universities and hospitals. Even though it contains a lot of images, there is a big class imbalance for the pneumonia class as can be seen in the table 2.1 below. The biggest advantage of this dataset is that it also includes a mask for the lungs that marks the region of interest. We used this mask to extract the lungs and darken the rest of the image. On the downside, the dataset does not contain any additional information about the patient such as age, gender or the stage of the disease.

	Covid19	Normal	Pneumonia
# images	3615	10192	1345

Table 1. Summary table of dataset

To get rid of this class imbalance, we took 1345 images from each class, so these 4035 (3*1345) images were the actual images used throughout the paper. Notice that this doesn't form any problem, as we're trying to reduce the amount of labeled data anyway. This dataset was then split into a training and test dataset with 3430 and 605 images respectively.

In order to be able to use the images with the vgg-16 model, we had to downscale the images (only by a few pixels) in order for them to have the correct input dimensions. Note that we've also done this for the corresponding masks, as they otherwise wouldn't reveal the lungs correctly.

We also looked into image pre-processing since it was stated important in research papers. We found a Github repository (Eduardo Garcia Misiuk, 2019) for X-ray image enhancement based on contrast limited adaptive histogram equalization, CLAHE. We setup some code to perform the processing in parallel on our local machines for 4035 images (1345 of each class). Afterwards we layed the masks on top of the images to darken out everything but the lungs. And so we've now got 2 datasets to compare. First we have the dataset with the original images with masks applied. And second we have the enhanced versions of the images, again with the masks applied.

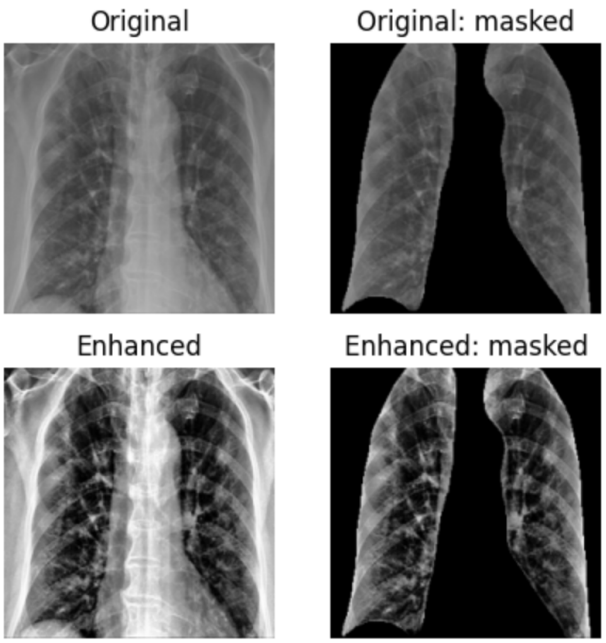

Figure 1. Pre-processing steps on X-ray images

3. Active learning

Active learning is a machine learning method that sits between unsupervised and supervised learning. The method

can ask an oracle to label unlabeled data in order to learn from it. The goal is to use as little as possible labeled data since it's often expensive and time consuming to label them.
Within active learning, there are three different types of which we'll use the pool-based variant. Next to that, in pool-based active learning, there are a lot of different query strategies to select which data needs to be labeled in order to learn as efficiently as possible.

3.1. Types of active learning

There are 3 main types of active learning, in this section we'll briefly discuss them.

3.1.1. Query synthesis

In this type of active learning, the model generates the data to be labeled. This technique is not useful for image recognition since it might generate images that don't resemble a real class, and thus can't be labeled properly.

3.1.2. Selective sampling

In this scenario, the unlabeled data comes as a stream. For each data point in the stream the learner decides whether it needs to be labeled (and thus trained on) or if it can be discarded.

3.1.3. Pool-based active learning

We'll discuss this type of active learning a bit more extensive, because it is the one that we'll use in our experiments. In pool-based active learning, the model has access to a pool of all the unlabeled data. Every data point in the pool is examined according to the query strategy (which may use the current machine learning model), evaluating the informativeness. The most informative data point(s) are then selected to be labeled by the oracle and the labeled, informative data points are then used in training. This loop continues until a certain accuracy threshold is met or a fixed number of iterations is achieved.

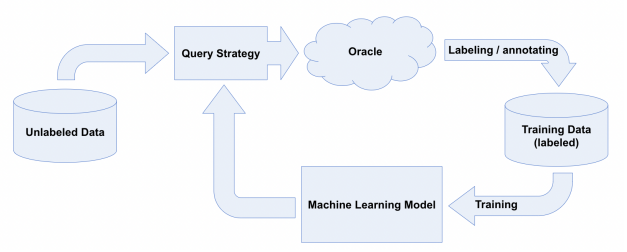

Figure 2. Pool-based active learning

3.2. Query strategies

In this section we'll discuss the different query strategies we compared. Note that there are more query strategies out there than the ones we list below, but these are the ones we've experimented with.

3.2.1. Random selection

The random selection strategy, randomly selects datapoints from the dataset to be labeled. The model isn't used in this strategy and therefore doesn't try to select datapoints that might be useful to the model. This resembles the ordinary machine learning way of training and should be considered a baseline for the other query strategies.

3.2.2. Least confidence

With the least confidence query we want to select those datapoints where the model is the least confident that the label it predicted is the correct one.
For every unlabeled datapoint in the dataset, we let the model run and we get 3 probabilities. Each probability corresponds to the probability that this datapoint is of a certain class. Naturally we take the highest probability of those 3, that would be the predicted class of the datapoint. The probability assigned to that class is the confidence that the model has in that prediction.
Once we've done that for every unlabeled datapoint in the dataset, we sort those datapoints lowest-confidence first. We then simply select the top n images, since the model has the least confidence in those predictions.

$$\phi_{LC}(x) = 1 - P_\theta(y^*|x) \tag{1}$$

3.2.3. Smallest margin

For the smallest margin strategy we start by predicting the same three class probabilities for every datapoint. Next the query tries to find all the datapoints that have a small margin between their first and second most likely class probability.
This can easily be done by calculating the difference between the highest and second to highest probabilities for all the datapoints. Finally the differences are sorted from small to large and the first n datapoints are selected.

$$\phi_{SM}(x) = P_\theta(y_1^*|x) - P_\theta(y_2^*|x) \tag{2}$$

3.2.4. Maximum entropy

The maximum entropy query is a bit more complicated in a sense that for every datapoint, every class probability gets multiplied with the logarithm of itself. Next, for all the datapoints the multiplications are summed up and multiplied with minus one. Finally the sums are sorted from small to

large and the last n datapoints are selected.

$$\phi_{ENT}(x) = -\sum_{y} P_\theta(y|x) log_2 P_\theta(y|x) \quad (3)$$

4. Experiment setup

In this chapter, we'll discuss the setup of our experiment. We will shortly explain how we used Tensorflow and what kind of code we developed to automate our experiments. Next we'll experimentally discover the best epochs & batch size for our dataset.

4.1. Tensorflow

We intended to use the scikit-activeml(Scikit, 2021) library for the active learning we stumbled upon a series of problems. Most of these problems were related to the fact that the library is still in beta release and not fully implemented. In order to overcome this, we switched to the TensorFlow(Google-Brain-Team, 2015) library. A huge benefit of TensorFlow is the availability of the VGG-16 model. This allowed us to import the model structure without pretrainend weights and guaranteed the support for other TensorFlow functions.
Tensorflow also supports graphical cards that greatly reduced our training time. This was very important since the HPC was unavailable for a few weeks so all experiments had to be run locally.

4.2. Experiment setup

In order to structure all the code we created a library that contained the most import classes and functions. With regard to handing in the code, everything is collected in a single notebook. This section will discuss the report notebook in a logical order to explain the setup of the experiment.

The first big part of the notebook is about the dataset.
The first function of the pipeline initializes the dataset by downloading it from Kaggle and unpacking the zip file. A subset, the first 1345 of every class, of the images is then selected, pre-processed (resizing and applying masks) and saved into .npy files. In the next step the enhanced images are created and, once again, saved into .npy files. These functions improved portability, allowing us to set up the experiments on any device quickly and uniformly. The .npy files are used in the Dataset class to load the dataset, preventing us from running the first two steps every time. This is some very welcome functionality, as enhancing the images takes a very long time. At last, we define a Dataset class, which provides functionality to get insights into the dataset and to create the train and test data sets.

The next big step in the pipeline is the declaration of the active learning through a model, a data class and the query strategies. The model class holds the VGG-16 model and allows for training and evaluating it. The data class handles the labeling of the data, to simulate the pool based approach. It is what simulates the labeling of images, by moving the images selected by a query strategy from the unlabeled data pool to the training data set. The query strategies return the selected indexes for a given model according to their strategy as discussed in section 3.2.

The final step of the pipeline is running the experiments themselves and creating the result graphs, these results will be discussed in chapter 5.

4.3. Experiments

As a first experiment we trained a model using active learning with every query strategy on both the original dataset and the enhanced dataset. As we train a new model in every iteration of the active learning loop, we can evaluate each of those models on our test data set. We then save the model and its accuracy on the test dataset as well as the query strategy, how many images were used, and whether the images where enhanced or not.

The results made us question whether having a fixed amount of epochs and batch size for every model is actually a good idea. As the amount of epochs and batch size could be optimized for the training data used. But as the size of the training data varies every iteration, and the data itself is chosen by some query strategy. We can only conclude that there is no way of knowing in advance which combination of epochs and batch size will be best for the training data in some iteration. So in our next experiment we conduct a small scale (because of the hpc being down) grid search in every iteration of the active learning loop, in order to get a relatively optimized combination of epochs and batch size. This is only possible because we hopefully don't need a lot of labeled data to train our models on which is the purpose of active learning.

In order to do this, we had to split off some samples from our train dataset, into a validation dataset. We'll use this validation dataset to evaluate our model during the grid search, and then pick out the best performing model. The test dataset will only be used to evaluate the best performing model and report the accuracy. This is done because the test dataset is not allowed to interfere with the construction of the model, and it may only be used to test the generalization of our model. This is why we needed a validation dataset, which we are able to use during construction of our model.

5. Results

This section will first discuss the overall results of the four queries on both the original and enhanced dataset. Afterwards we'll look at the performance of all the queries separately. All accuracy graphs in the first sections are made with the active learning model by querying a total of 512 images from the training dataset. Only the section discussing the results with the grid search uses a total of 256 images, which already took a very long time to run on our own laptops.

5.1. Original dataset

In order to get a quick first impression of how the different query strategies are performing, all of them are set out in figure three for the original dataset. The random query, our

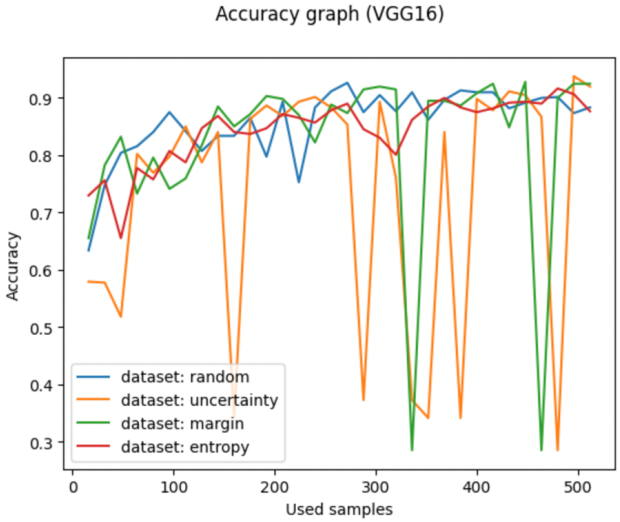

Figure 3. Query strategies on original dataset

baseline, had a maximum classification accuracy on the test set of 92,5%. All other query strategies scored very similar with least confidence query coming in at 93,7%.
There are some huge drops in accuracy at random times for both least confidence and smallest margin sampling. This strange behaviour needs some more attention and will be further discussed in the coming sections.
Another clear thing to see on the graph is that all four queries have very similar curves apart from those drops. This means that all four queries perform almost equally as good for any given number of used samples.

5.2. Enhanced dataset

In order to see whether there is a difference between the original dataset and the enhanced version, a similar graph is made in figure four for the enhanced dataset.

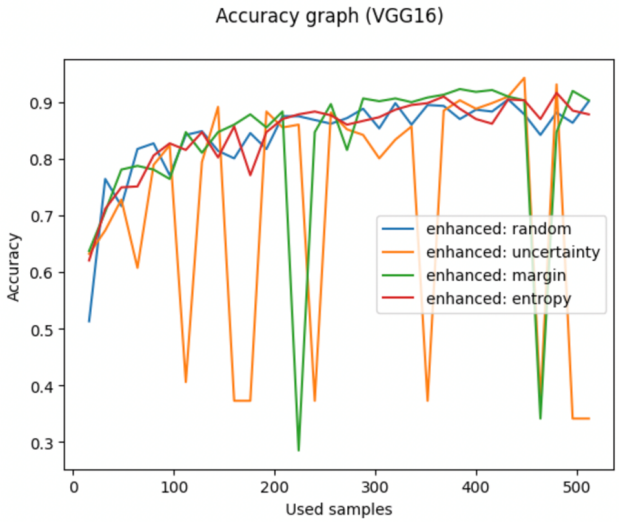

Figure 4. Query strategies on enhanced dataset

The least confidence query achieved the highest classification accuracy of all queries with an accuracy of 94,2%. This is only 0,5% higher than for the original dataset, which is not significant. The smallest margin and max entropy query scored marginally worse and as good respectively. The random query scored a maximum accuracy of 90,4%, which is two percent down from the original dataset. This highlights the fact that the random query is only random and has unpredictable performance. The query might have taken different pictures resulting in higher or lower accuracies. All of this does not indicate that there is a clear performance difference between the two datasets being used. Again there is a lot of similarity amongst the curves when not looking at the drops. By now it has also become clear that the drops always have more or less the same range of numbers being [0,3; 0,4].

5.3. Accuracy drops

As discussed before there are some random accuracy drops for the least confidence and smallest margin query on both datasets. A first field of interest we looked at to explain this strange behaviour was class imbalance. However there was no clear difference between the class balances from these two queries in comparison to the others. In fact the queried images have a relatively equal class balance as can be seen in the histograms below (figure 5).

So our next thought was that the fixed number of epoch and fixed batch size was a bad idea, and we conducted a grid search in every iteration. The results are described in the following section.

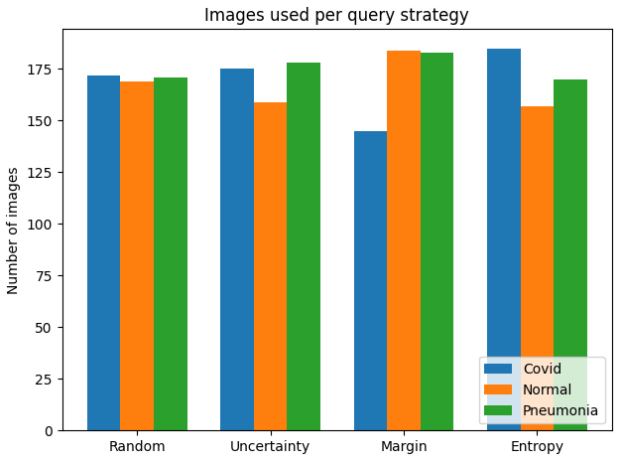

Figure 5. Histograms of the image classes selected by each query strategy

5.4. Grid search

In this section we discuss the results achieved when running a mini grid search in every active learning cycle. Due to the hpc being down, we had to make our grid search very small. It consisted of training a model with both 8 and 16 epochs combined with a batch size of 8 or 16. This results in training only 4 models in every iteration, which is indeed a very small grid search. Nevertheless we did see some improvement over not using a grid search in every iteration. To further reduce the execution time, we also only used a maximum total of 256 images per query strategy.

The first graph (figure 6) shows the results when using the original dataset. And the second graph (figure 7) shows the

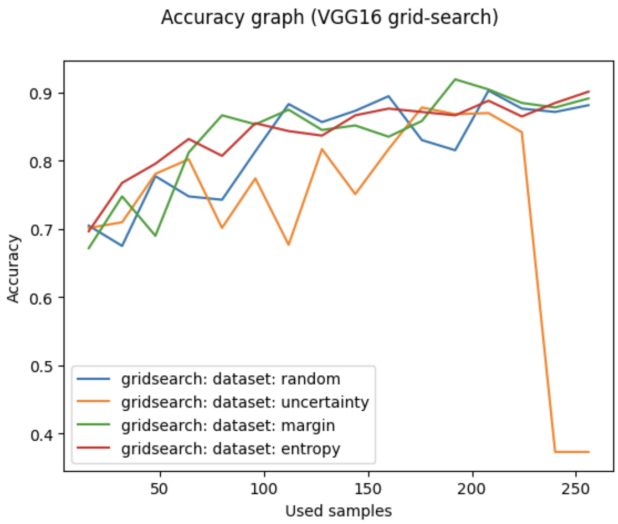

Figure 6. Query strategies on original dataset with grid search in every iteration

results on the enhanced dataset. The random querying strategy yielded a whopping maximum of 90,2% accuracy, on only 208 images. But the winner was the smallest margin strategy with 91,9% accuracy, on just 192 images.

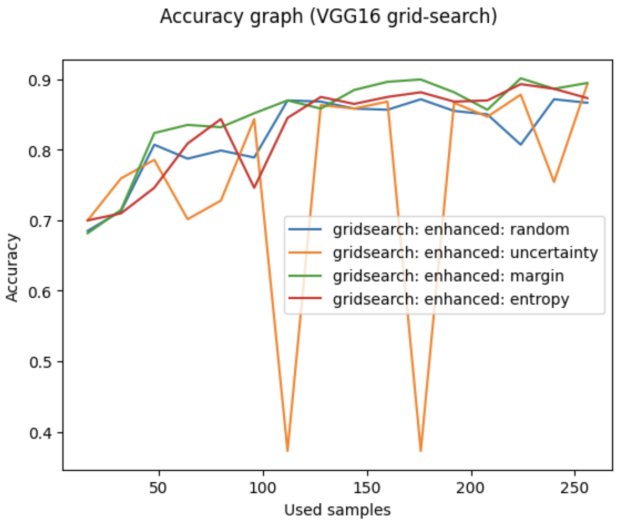

Figure 7. Query strategies on enhanced dataset with grid search in every iteration

But if we look at the accuracy drops, the main reason we started doing these grid searches, we can only say that it hasn't made it worse. We never experienced more accuracy drops, and only 2 cases where we had less accuracy drops.

6. Discussion

To summarize the findings of the research we can state that there is no clear difference in classification quality among the different query strategies. We think this is partially because of the underlying vgg-16 model, which is on its own very good in classifying images.

There is however a huge difference in the stability of the query strategies, where random and max entropy are much more stable than least confidence or smallest margin querying. It remains unclear why exactly but it became clear that class imbalance is not the reason. Further research might dig deeper into this and look at what exactly happens at the step that results in the huge drop. Which exact images were used and why these images might throw off the VGG-16 model.

As shown, it is very important to search for a good combination of the batch size and the number of epochs. This might significantly reduce the amount of images needed for training to achieve the same accuracy. The vgg-16 model is quite complex, and thus we think that finding good parameters becomes so important that you risk learning nothing when picking non-optimal parameters. With a bigger grid

search, research might also be able to reveal if this has an impact on query stability.
The main conclusion is that active learning in this case did not reduce the amount of labeled images that were needed since no query outperformed random at any point in the learning. There is also no clear improvement when enhancing the images, some query strategies performed better on the enhanced dataset, others performed worse. And the most important take of all, picking the right parameters for your model is of uttermost importance.

References

Civit-Masot, Javier, Luna-Perejón, Francisco, Domínguez Morales, Manuel, and Civit, Anton. Deep learning system for covid-19 diagnosis aid using x-ray pulmonary images. *Applied Sciences*, 10(13), 2020. ISSN 2076-3417. doi: 10.3390/app10134640. URL https://www.mdpi.com/2076-3417/10/13/4640.

De Wever, W., Coolen, J., and Verschakelen, J.A. Imaging techniques in lung cancer. *Breathe*, 7(4):338–346, 2011. ISSN 1810-6838. doi: 10.1183/20734735.022110. URL https://breathe.ersjournals.com/content/7/4/338.

Eduardo Garcia Misiuk, André Almada, Im Choyoung. X-ray images enhancement. https://github.com/asalmada/x-ray-images-enhancement, 2019. Accessed: 16.11.2022.

Google-Brain-Team. Tensorflow. https://www.tensorflow.org/, 2015. Accessed: 20.11.2022.

Goyal, Shimpy and Singh, Rajiv. Detection and classification of lung diseases for pneumonia and covid-19 using machine and deep learning techniques. *J Ambient Intell Humaniz Comput*, pp. 1–21, September 2021.

Kaggle. Covid-19 radiography database. https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database, 2020. Accessed: 13.11.2022.

Organizaton, World Health. The top 10 causes of death. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death, 2020. Accessed: 30.11.2022.

Scikit. Scikit-activeml. https://github.com/scikit-activeml/scikit-activeml, 2021. Accessed: 28.10.2022.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

Wangg, Linda. Covid-19 and pneumonia scans dataset. https://public.roboflow.ai/classification/covid-19-and-pneumonia-scans, 2020. Accessed: 12.11.2022.