LUS Images classification with uncertainty detection and image similarity

Davide Modolo (#229297)

Abstract
The proposed multi-stage model for predicting LUS image scores is built using three main components: a multi-class frame classifier, an uncertainty detection model, and a similarity module. The idea is to retrieve similar images and analyze them when the initial prediction is uncertain. The entire project is currently available on GitHub [1].

Introduction

This project aimed to develop an alternative way to predict LUS images scores. The first idea that came to my mind was to build something that could be used by doctors, retrieving similar images when the score of a specific frame was not sure to "help" with the decision.

The existing methodology consists in scoring all 14 different spots and summing their values. If the result is $< 24/42$, the patient can be left going home because it indicates a low probability of worsening.

As explained in the article by S. Roy et al. [2], LUS images are scored as:

- 0: no artifact in the picture;
- 1: at least one vertical artifact (B-line);
- 2: small consolidation below the pleural surface;
- 3: wider hyperechogenic area ($< 50\%$) below the pleural surface.

Frames are taken from videos taken using ultrasound probes and are taken in a maximum of 14 different spots (6 on the front and 8 on the back of the patient), as explained in the article by G. Soldati et al. [3].

My proposed model consists of three main components:

- A multi-class frame classifier that predicts the score of individual LUS images.
- An uncertainty detection model that evaluates the confidence of the initial prediction.
- A similarity module that retrieves similar images and analyzes them when the first model is not confident.

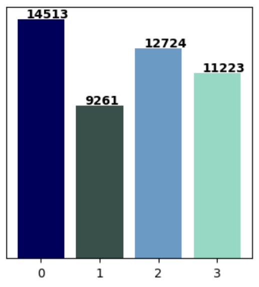
With this multi-stage approach, I aim to improve the accuracy of LUS image scoring and provide technicians with a more reliable tool for diagnosis.

Unfortunately, we will see this idea is probably not effective and the results are not encouraging.

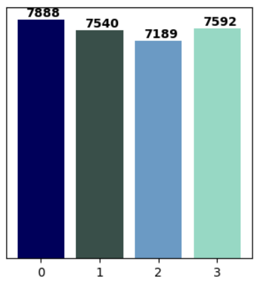
1. Data

We have been given a partial dataset from the San Matteo hospital, consisting of 11 patients for a total of ∼47k frames.

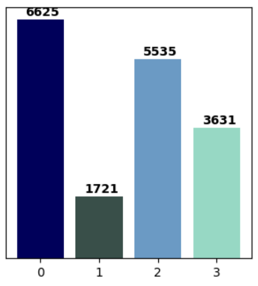
The dataset score distribution is shown in Figure 1a; at a first glance it could seem to be almost balanced (with only the score 1 that has fewer frames), but in reality many patients are inherently unbalanced (the score distribution for each patient is shown in Figure 2).



(a) Entire dataset



(b) Train set



(c) Test set

Figure 1. Score distribution in the dataset

1.1 Augmentation

Using the raw dataset got me overfitting even during the first epoch. To address this issue I implemented some transformations taken from the article [2].

In specific, each transformation is activated with a probability of 50%. The set of my augmentation function is:

- affine transformations (translation = $\pm 15\%$, rotation = $\pm 15°$, scaling $\pm 45\%$, and shearing = $\pm 4.5°$)
- multiplication with a constant ($\pm 45\%$)

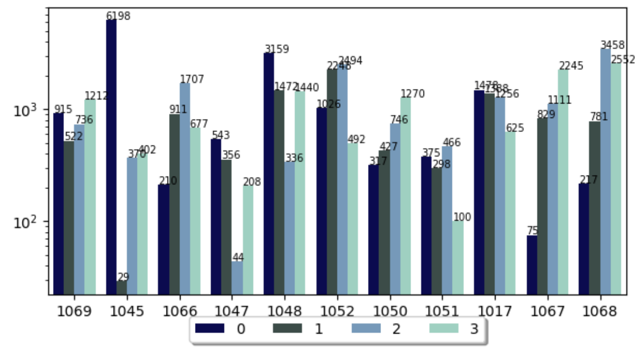


Figure 2. Number of frames for each score for each patient (log scale for better visualization).

- Gaussian blurring ($\sigma = 3/4$)
- horizontal flipping ($p = 0.5$)

1.2 Data splitting

Having 11 patients available, my idea was to use 8 of them to train the model and the remaining 3 for testing. This was due to the fact that using a portion of the frames for a patient in test and another in train easily leads to overfitting. Even dividing by exams would not be effective since different exams for the same patients still have a big correlation.

The first attempt was to test with a *k-fold* approach and then choose the best configuration, but having 165 combinations with $\sim 4h$ per combination was unfeasible.

So, to balance the dataset I computed the standard deviation within scores for each 8-patient combination and selected the one with the lowest std (Figure 3), resulting in the division shown in Figure 1b; the problem now was with the test set, that resulted to be very unbalanced (Figure 1c). After different attempts to balance both sets, I decided to just select an equal number of images for each score from the training patients set of frames to use in the `test_model` method (still, confusion matrices on this report are built using the entire available test set).

2. Multi-class classifiers

The first module of my project consists in a deep learning classifier that predicts the score from a frame.

Different pre-trained models have been tested with several different values for my hyperparameters. The training part has been made several times in order to find a model that didn't overfit in the first epoch or didn't stuck in a local minimum that always gave one single score.

Frames are very similar, using models too big could get overfitting and using models too small could get no good generalization capability.

2.1 ResNet18

ResNet (Residual Network) is a network introduced by K. He et al. [4] trained on the ImageNet dataset [5].

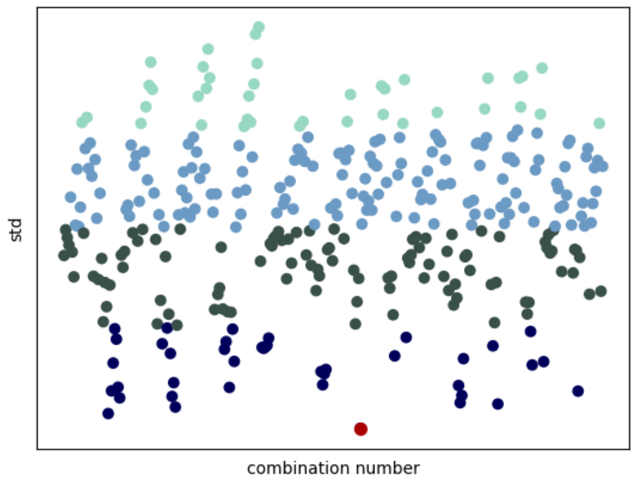


Figure 3. Standard deviation within the number of frames per score of every combination of 8 patients, the red one is the minimum (and so, it is the selected combination).

There are different versions of this model based on the number of layers. Looking for a "small" model, ResNet18 was the smallest one and so it has been selected for testing.

After many runs, I was able to achieve an accuracy of $\sim 56.65\%$ in my test set before overfitting. The confusion matrix on the test set can be seen in Figure 4, resulting in an accuracy class-wise that can be seen in Table 1.

0	1	2	3
55.52%	66.88%	40.43%	77.03%

Table 1. TODO: Recumpute Accuracy class-wise of the fine-tuned ResNet18.

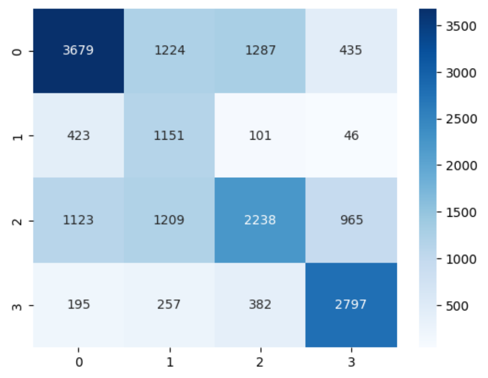


Figure 4. Confusion matrix of the fine-tuned ResNet18.

2.2 VGG16

VGG (Visual Geometry Group) is a convolutional Neural Network built by K. Simonyan, A. Zisserman [6]. It has been trained on a subset of the ImageNet dataset.

Similarly to ResNet, VGG is available with 16 and 18 layers. For the same reasons as above, VGG16 has been selected and tested.

Independently from my fine-tuning tries, VGG16 started memorizing the training data even in the first epoch (even having fewer parameters than ResNet18).

2.3 SqueezeNet

SqueezeNet is a model developed by F. N. Iandola et al. [7] in 2016.

Following the idea to find a compact model, I found this variation of AlexNet that is still capable of very good performance while requiring fewer parameters. It has been trained on ImageNet.

SqueezeNet gave me the best results in the early stage of the project, but after refining the fine-tuning of the ResNet, I decided to not use it.

2.4 Built-from-scratch model

I even tried building from scratch a Convolutional Neural Network. I tried different combinations of Convolutional layers but results were very poor, resulting in a path I didn't follow deeper.

3. Binary classifiers

The second goal of this project was to develop a mechanism to determine the confidence of the multi-class frame classifier in its predictions.

Initially, I explored the possibility of using a threshold-based approach; however, during the in-class presentation, we realized that a more sophisticated approach would be interesting to try to capture the behaviour of the model in both correct and incorrect predictions. In addition, I observed that the maximum softmax values were very similar, indicating that a simple threshold approach would not be effective.

As a result, I developed two binary classification models that use the softmax values of the first model to evaluate the confidence in its predictions.

To train these models, the data has been built by using the trained ResNet18 model on the training set to then save the output values with their correctness (T/F); I also had to balance the new dataset since even a bit of unbalance in it would led to having one and only one output value.

3.1 Deep model

For this approach, I built a simple neural network with four inputs and two outputs. It uses one dense layer with a Sigmoid activation function to make predictions (ReLU has also been tested but resulted in worse performance). Any layer or complexity I added resulted in a very bad performance, with this configuration I was able to achieve the 54.66% in one of many runs.

3.2 Support Vector Classifier

From the sci-kit learn library a linear model that tries to find the best hyperplane that separates the two classes. The SVC model can be trained quickly and its performance was similar to the DL model I created, but I found it more prone to "overfitting" since even a dataset that was a little unbalanced in the class distribution resulted in 50% (so random choice). 56.34% acc. I found out that the accuracy was not that good, in fact, it was very good at finding confidence in classes 0 and 2, but not so good in 1 and 3.

3.3 Four SVCs

To investigate further the behaviour of the SVC, I decided also to test one different SVC for each class with the idea of: "Depending on the class predicted by the first classifier, I call one of the four SVCs trying to have a more specific approach". Needless to say, the results seemed fine until I tried it in the last model: it didn't work. The average accuracy by class is 54.2%, slower than the single SVC approach, but it seemed to be more balanced across the classes (we will see, still it works worse than the single SVC).

4. Image similarity

The third module of my project consists of an image similarity model. As briefly said before, this idea was born from the definition of diagnosis that has some degree of subjectivity. Of course, thanks to the scoring mechanism proposed and cited in Introduction, this effect is mitigated. Still, I was interested in showing similar images when the score prediction had low confidence and I developed the third module.

In the first presentation held in class, my idea was to use a Near Duplicate Image Search with Locality Sensitive Hashing, but in practice, I didn't implement the LSH part since the results were not good in the first place. As discussed in one of the last lectures, I also decided to try t-SNE.

4.1 Near Duplicate Image Search

- WHAT IS THIS
    - HOW I DID IT using `annoy` library.
    - RESULTS

4.2 t-SNE

bla bla bla

4.2.1 Embedding

bla bla bla

Embedding with no t-SNE While working on the embedding-based t-SNE, I decided to try using the embeddings as they were, using the cosine similarity to find the closest *n* images. "It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction" [8].

4.2.2 Raw Images

bla bla bla

4.2.3 Behavior

bla bla bla Even if this representation is purely based on the first classifier softmax values (and not on visually similar images) it got the best results.

5. Performance analysis

For the binary class, performance in the final model was very similar so I decided to use the DeepL one since it had higher accuracy on its own.

Cut from above the cm of ResNet and table

Add both SVC and DL cm

Add table for 4 SVC

Add plots of all the 3 tsne approaches.

Add CM of the final model

Add new table i prepared

6. Conclusions

Very very slow due to image similarity.

Bad performance maybe to: binary gets wrong when is correct. Or binary gets 0 and the t-SNE is wrong in exacly those.

slow tsne low acc

6.1 Future works

bla bla bla

References

[1] GitHub repository with the project. [Online]. Available: https://github.com/davidemodolo/Lung-Ultrasound-Image-Classifier

[2] S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, C. Saltori, I. Huijben, N. Chennakeshava, F. Mento, A. Sentelli, E. Peschiera, R. Trevisan, G. Maschietto, E. Torri, R. Inchingolo, A. Smargiassi, G. Soldati, P. Rota, A. Passerini, R. J. G. van Sloun, E. Ricci, and L. Demi, "Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound," *IEEE Transactions on Medical Imaging*, vol. 39, no. 8, pp. 2676–2687, 2020.

[3] G. Soldati, A. Smargiassi, R. Inchingolo, D. Buonsenso, T. Perrone, D. F. Briganti, S. Perlini, E. Torri, A. Mariani, E. E. Mossolani, F. Tursi, F. Mento, and L. Demi, "Proposal for International Standardization of the Use of Lung Ultrasound for Patients With COVID-19: A Simple, Quantitative, Reproducible Method," *J Ultrasound Med*, vol. 39, no. 7, pp. 1413–1419, Jul 2020.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] ImageNet site. [Online]. Available: https://www.image-net.org/

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[7] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," 2016.

[8] Cosine Similarity. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/cosine-similarity

[9] Stylish Article Template. [Online]. Available: https://www.latextemplates.com/template/stylish-article

7. Data

The template I used for this report can be found on *latextemplates.com* [9].