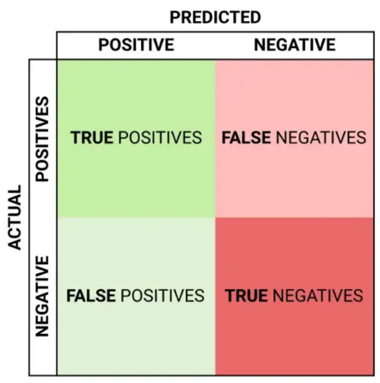
Medical Image Multi-Label Classification

Bianca-Oana Guita
511

I. Introduction

For this competition, we were given a task to classify medical images having a multi–label environment. We would train a deep learning classification model on magnetic resonance pictures and predict for each one of the train images the correct label. There are two labels (0 and 1) assigned to three different classes in a multi-label environment. For example, an image can be categorized as one or more classes.

For evaluation purposes, F1 Score was used on the test set, based on precision (TruePositives/(TruePositives + FalseNegatives)) and recall (TruePositives/(TruePositives + FalseNegatives)).



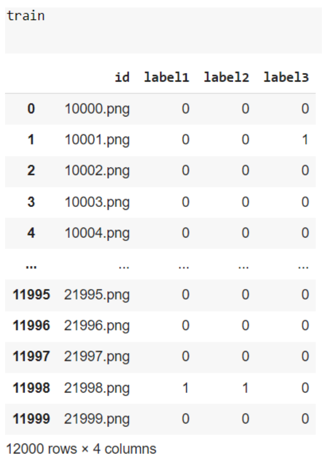
In our case, having a multi-label task, the average F1 score for every class in particular is calculated.

$$\frac{2(P * R)}{P + R}$$

II. Dataset

We were provided with a dataset that contains a total of 20000 images and other csv files:

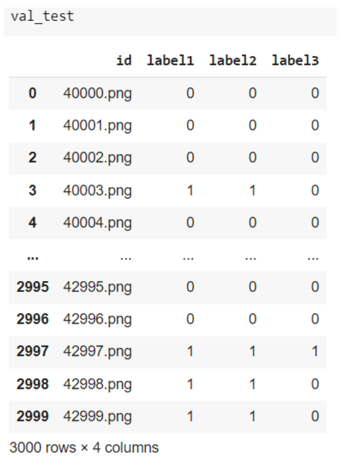
- 12,000 training images
- 3,000 validation images
- 5,000 test images
- *train_labels.csv* - having the names of the training file images and their labels
- *val_labels.csv* - having the names of the validation file images and their labels
- *sample_submission.txt* - having the names of the test file images and the correct format of the predictions' submission.



train

	id	label1	label2	label3
0	10000.png	0	0	0
1	10001.png	0	0	1
2	10002.png	0	0	0
3	10003.png	0	0	0
4	10004.png	0	0	0
...	...	...	...	...
11995	21995.png	0	0	0
11996	21996.png	0	0	0
11997	21997.png	0	0	0
11998	21998.png	1	1	0
11999	21999.png	0	0	0

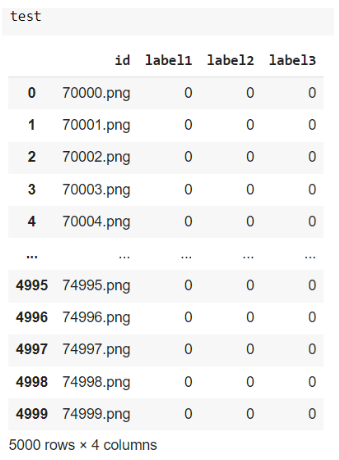
12000 rows × 4 columns



val_test

	id	label1	label2	label3
0	40000.png	0	0	0
1	40001.png	0	0	0
2	40002.png	0	0	0
3	40003.png	1	1	0
4	40004.png	0	0	0
...	...	...	...	...
2995	42995.png	0	0	0
2996	42996.png	0	0	0
2997	42997.png	1	1	1
2998	42998.png	1	1	0
2999	42999.png	1	1	0

3000 rows × 4 columns



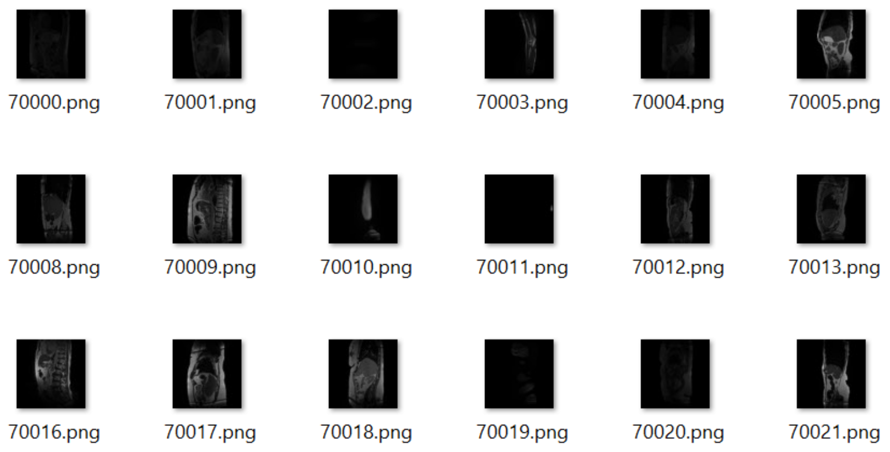
test

	id	label1	label2	label3
0	70000.png	0	0	0
1	70001.png	0	0	0
2	70002.png	0	0	0
3	70003.png	0	0	0
4	70004.png	0	0	0
...	...	...	...	...
4995	74995.png	0	0	0
4996	74996.png	0	0	0
4997	74997.png	0	0	0
4998	74998.png	0	0	0
4999	74999.png	0	0	0

5000 rows × 4 columns

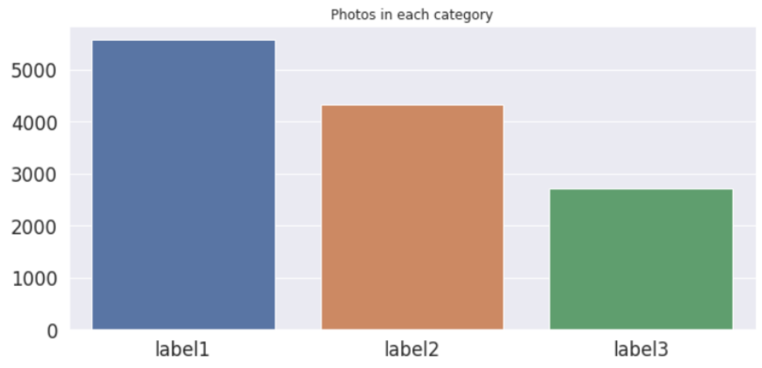
Each first column ('id) gives information about the file name. The second, third and fourth column represent the binary label linked to the first, second and third class.

The images would look like this:

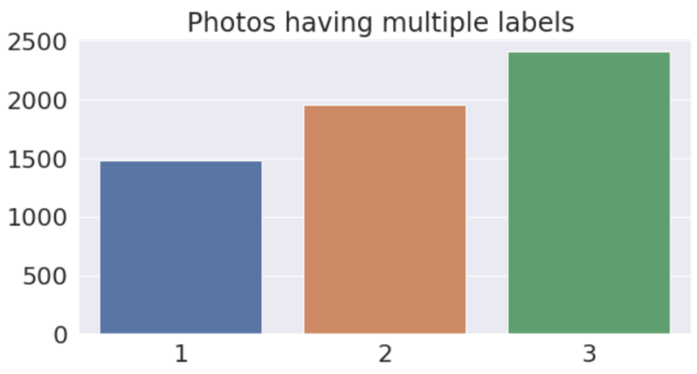


III. Preprocessing data and visualizations

To have a better idea of how our data is represented, I illustrated the partition of the images in each category. For training images we can find more pictures under class 1.



Representing the graph regarding the number of images that can be found in multiple classes, we notice that the images in class 3 are more likely to be also found in class 1 or 2 as well.



For getting the data ready for the models, I resized them (224x224) changing their dimension, converted it to tensor, then normalized with a mean of 0.485 and a standard deviation of 0.229. I left the images greyscale, as they are, having one input channel. I created the data loaders and they're all ready to be modeled.

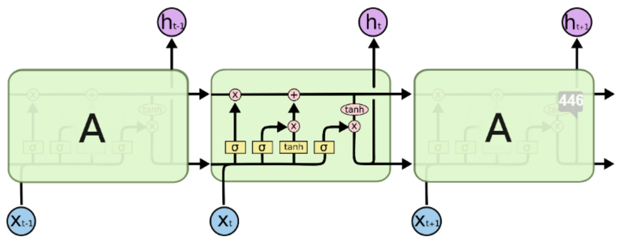
IV. Models

All of that with 2 Dense Layers								
Model	**Layers**	**Batch_size**	**Pooling**	**Epochs**	**Optimizer**	**Loss Functior**	**Accuracy**	
CNN	2	48	MaxPool2d	12	Adam	Cross Entropy	12.73	
CNN	2	48	MaxPool2d	12	Adam	BCE	54.87	
CNN	2	64	MaxPool2d	12	Adam	BCE	57.12	
CNN	2	64	AvgPool2d	15	Adam	BCE	57.32	
CNN	2,out_ch=512	48	MaxPool2d	15	Adam	BCE	68.70	
CNN	2,out_ch=512	64	MaxPool2d	15	Adam	BCE	71.30	
CNN	3	64	AvgPool2d	15	Adam	BCE	71.77	
CNN	2	128	AvgPool2d	15	Adam	BCE	73.50	
Trying with 1 Dense Layer								
CNN	2	128	AvgPool2d	15	Adam	BCE	73.87	84.97
VGG-custom	21	48	AvgPool2d	15	Adam	BCE	74.10	
VGG-custom	14	48	MaxPool2d	15	Adam	BCE	74.27	89.04
VGG-custom	14	64	MaxPool2d	15	Adam	BCE	74.33	88.93
LSTM	1	64	-		Adam	F1	0.3	

As we can see in the table below, I mostly worked with convolutional neural networks and tried different parameters for convolutions, tried several pooling and loss function options. Even though Cross Entropy is used specially for multi-class classification, I chose BCE loss in the end, as I wanted to generate results as probabilities (using the sigmoid function). The Sigmoid function couldn't be used for two output features as CrossEntropyLoss gives. If I were to choose Cross Entropy, I should've used softmax, but since it is only suitable for multi-class classification, and we have a multi-label type of classification, it doesn't give meaningful result probabilities.

Firstly, on each layer I applied MaxPool2d which calculates the biggest value from each layer, then switched to AvgPool2d to test it, as it preserves the average values of the map of features. Also added the Batch Normalization mechanism that improves the quality of the neural network, stabilizing the distribution of hidden channels. For the activation function I opted for Rectified Linear Unit (ReLU) which solves the issue of gradient vanishing and makes the model learn easier, faster and better. For the fully-connected layers, I used 2 layers of Linear, at first, then I noticed that with only one dense layer, the model works better. The linear function is specially organized to compute the linear equation between the input and ponders. For the optimization algorithm, I went for Adam, as it gave me best results. It is an extension to SGD, that is more efficient, it requires less memory, it suits tasks having large datas or params. It has a learning rate that performs better on the problem of sparse gradients .

The VGG-custom architecture models appeared to work a little bit better on my dataset. Couldn't say the same with the LSTM, unfortunately. Long short-term memory can be classified as a recurring neural network and is capable of processing both single data units like pictures, and whole sequences of data, like videos. Basically, the RNNs address the problem of remembering previous events. The networks have loops, making the info persist. LSTMs can be seen as RNNs but more special. They have a chain structure, but the module that repeats itself has four neural layers, instead of just one.



The repeating module in an LSTM contains four interacting layers.