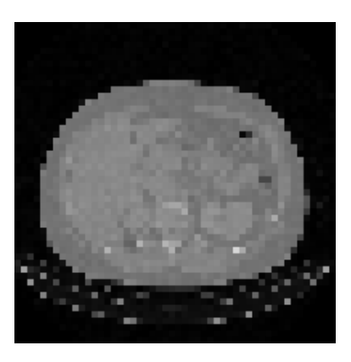
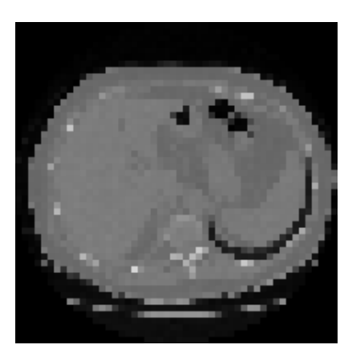
Computer Tomography Scan Classification

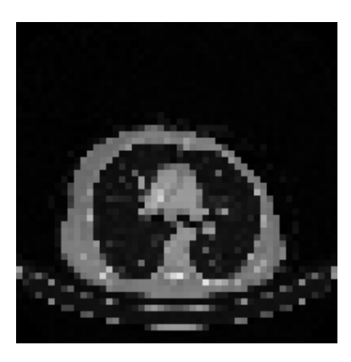
Darius Dima
Universitatea Bucuresti

1. Introduction

Motivation: Many lives could be saved by reducing the human error in medicine. Fortunately, with our current development in Artificial Intelligence and Machine Learning technologies, new methods have emerged to do just that. With rapid improvements in computer vision, computers are able to accurately identify a problem and help doctors prescribe the appropriate diagnostic. A couple of areas in medicine, for example radiology, has great room for improvement. Some experts believe, that one day AI-based systems could eventually replace radiologists. Others affirm that computers are there to assist physicians and not replace them entirely. As in other activities, technology helps achieve more work in less time and at a higher quality. We aim to investigate this technology in order to build future applications that will enhance medical assistance in the future. We believe that this technology is unlikely to alter the patient-physician relationship, and it will only enable the doctors to save more and more lives.
Proposal: Empowered by the development of deep neural network technology, we propose as our current goal to to offer computers the ability to distinguish between three types of blood vessels in the lungs from previously taken tomography scans. By achieving this, we'll be able to help medics to better diagnose the patients and save more lives. We will accomplish this by training a neural network to classify images into blood vessels: native, arterial and venous.


0 - native
1 - arterial
2 - venous


0 - native
1 - arterial
2 - venous

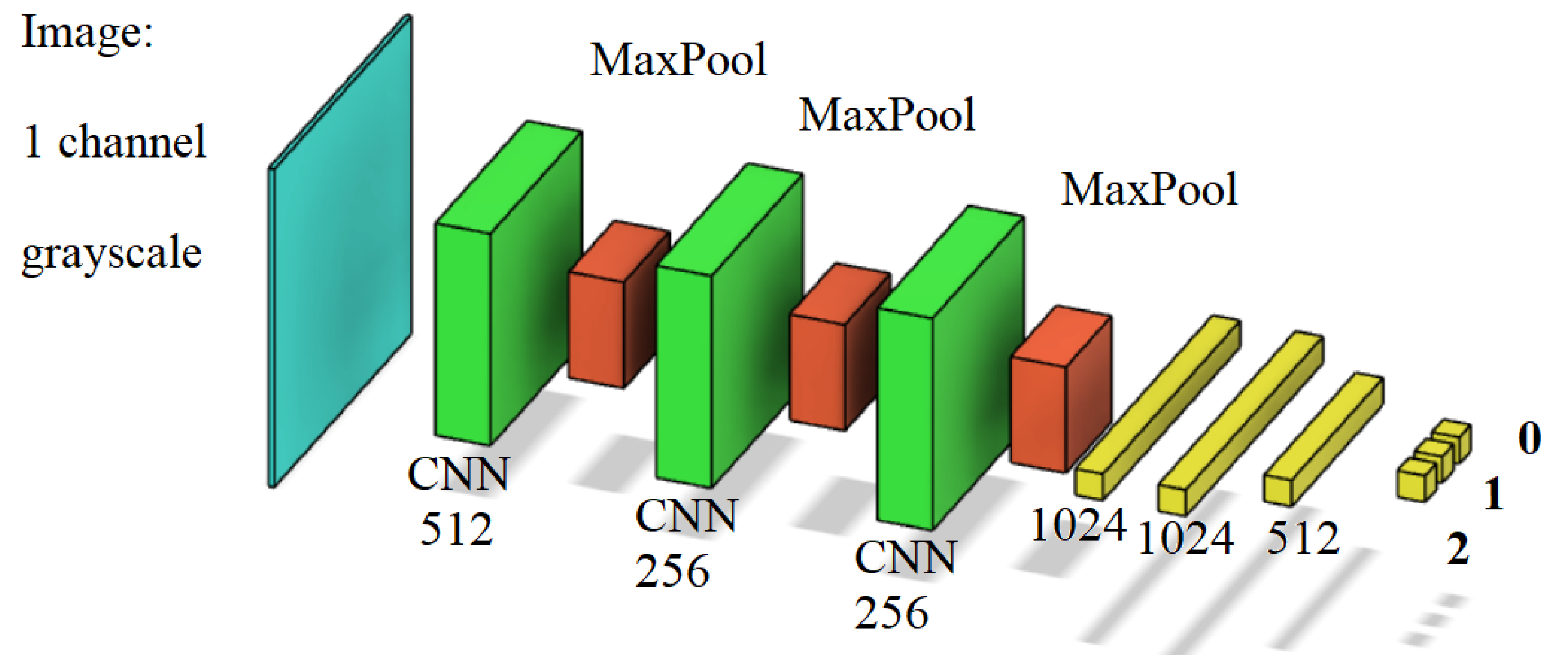

0 - native
1 - arterial
2 - venous

2. Dataset and Preprocessing

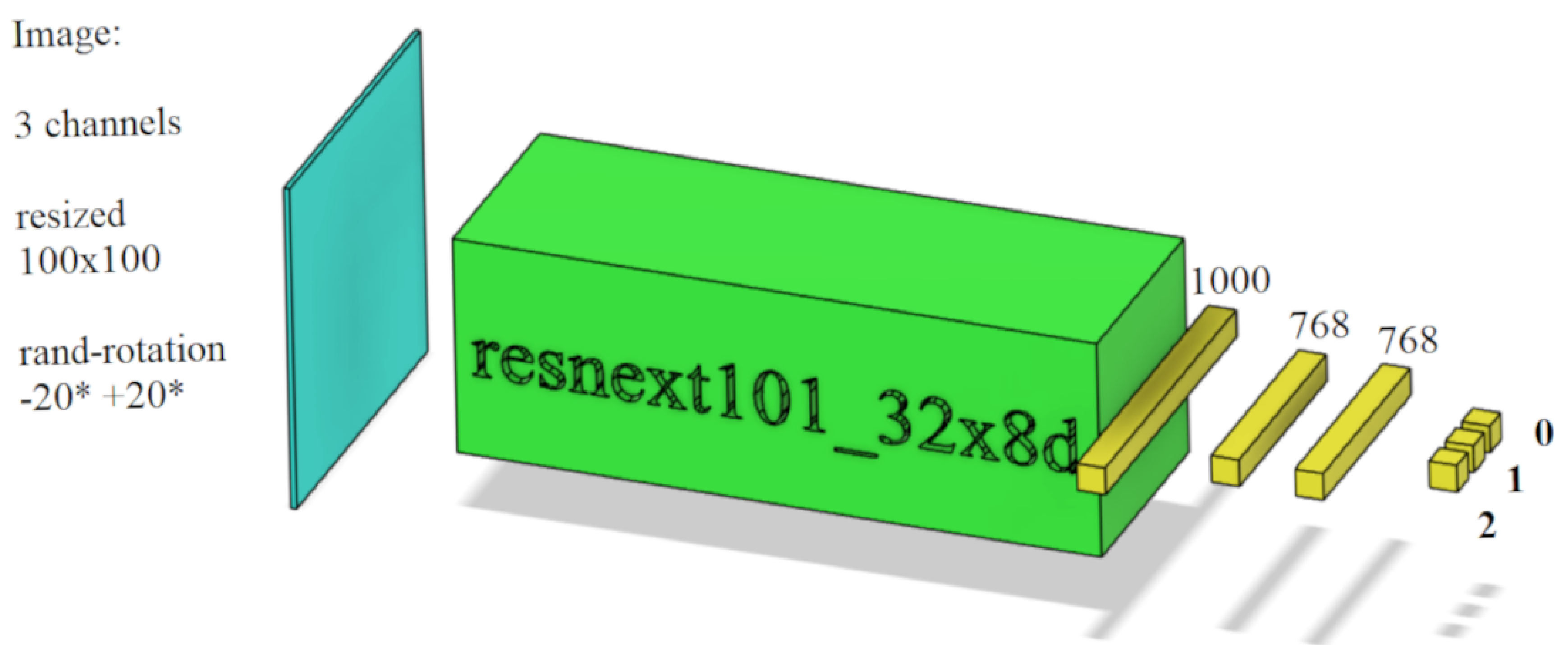
Dataset Description: We used the ai-unibuc-23-31-2021 dataset, provided on the kaggle competition. (link) It contains 23,400 total images, split into 3 sets: 15,000 for training, 4500 for validation and 3900 test images. Also, for each set of images, we had text file listing the image names and their labels, line by line with a comma separating the name from the label, except for the test, where the labels were missing. Each image has the size 50x50 pixels on 1 grayscale channel.
Data preprocessing: To prepare our dataset for for the Convolutional Neural Network, I normalized the images by calculating their mean and standard deviation. I've also scaled the images to 100x100, and made them into 3 channels by repeating the channel dimension 3 times. In addition, I applied a random rotation for the training set for the extracted features to be more general.
For svm approch I just had to reshape the image tensors in 1 dimension with 2500 pixel values.

3. Models

Baseline: We had to use our machine learning skills aquired in this semester to classify images in the dataset into the three categories, with an accuracy higher than 0.39384.
SVM: Support Vector Machine model performs poorly compared to neural network models
CNN model: I used an neural network model with 3 convolutional layers and 4 fully connected layers to obtain a better accuracy. I tried multiple parameters and different number of layers, but I finally settled to this one.
The hyperparameters which I empirically selected are: Adam optimizer, lr. 10^{-5}, batch size 64, and I trained it for 20 epoches. I used nn.Dropout modules with values 0.15 in between features and 0.2 in between linear layers for the model's regularisation.

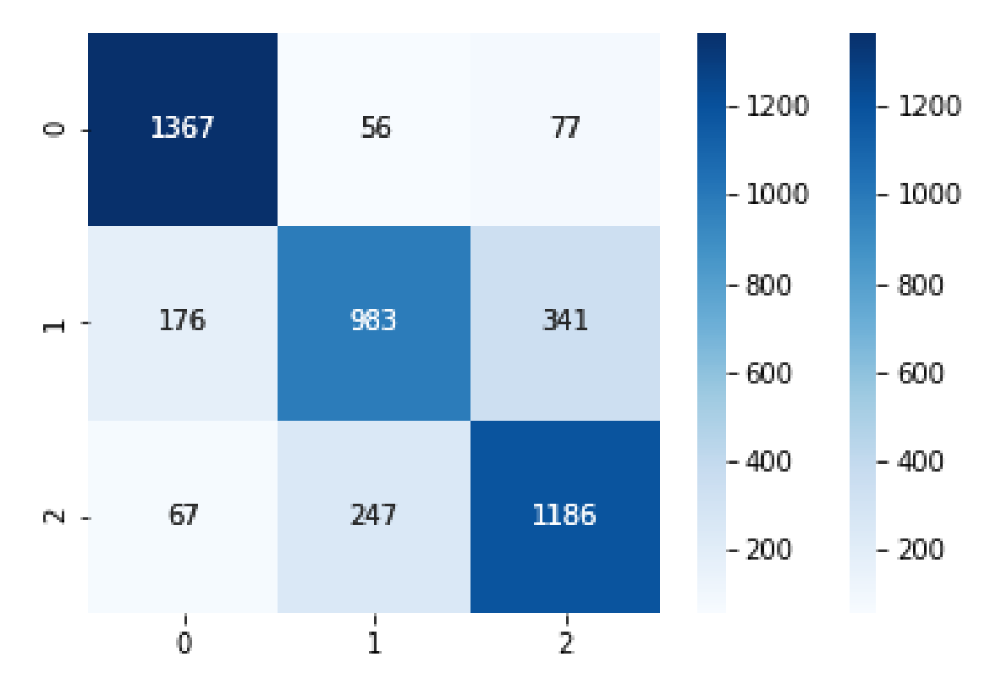


resnext101_32x8d model: I used the resnext101_32x8d model from torchvision.resnet library and modified it by adding 3 fully connected layers. This approch performed best so far.
The hyperparameters which I empirically selected are: Adam optimizer, varable learning rate, starting at lr. $4 * 10^{-5}$, and decreases by a factor of 0.8 every 2 epoches. Batch size 128, and I trained it for 31 epoches. I used nn.Dropout modules with values 0.2 in between the added linear layers for the model's regularisation.

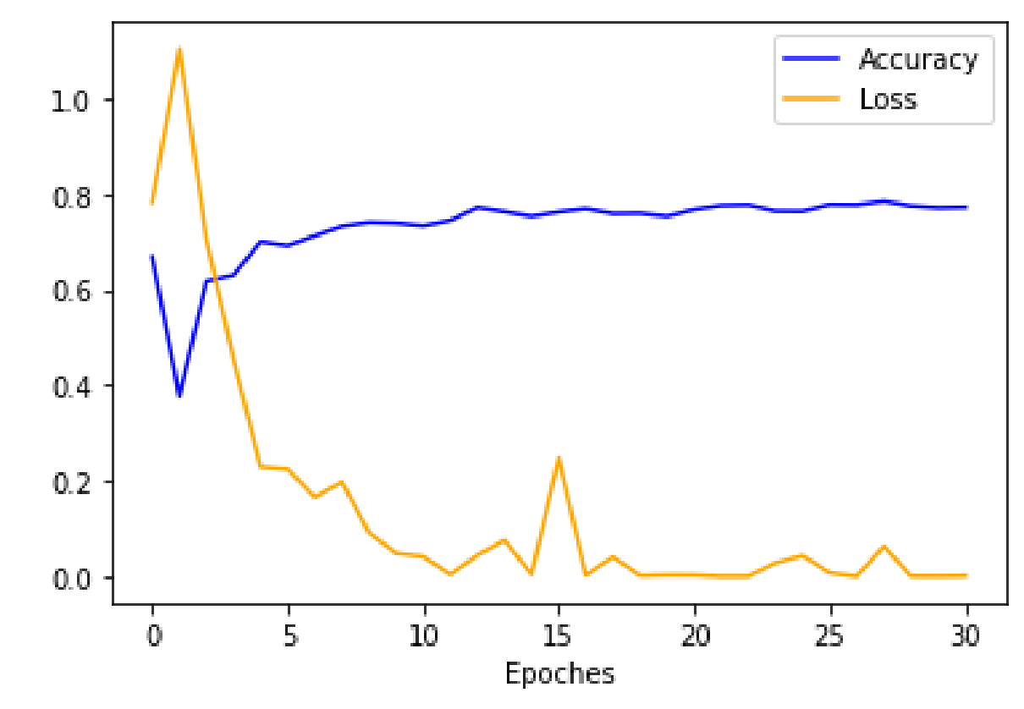


4. Results

Architecture	Input type	Training Loss	Validation Accuracy	Kaggle Accuracy
SUPPORT VECTOR MACHINE (SVM) MODEL	(1,50,50) tensor images	-	0.486	0.45230
CONVOLUTIONAL NEURAL NETWORK MODEL	(1,50,50) tensor images	0.41	0.715	0.64923
RESNEXT101_32X8D MODIFIED MODEL	(3,100,100) tensor images	0.00147	0.78578	0.74290



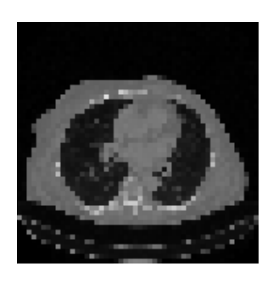
Confusion Matrix

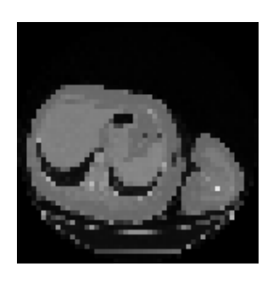


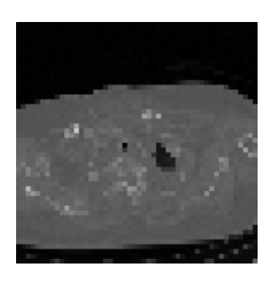
Accuracy on training set

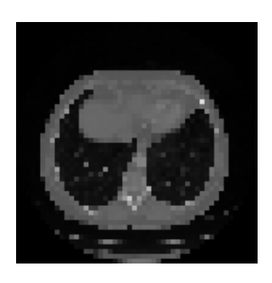
RESNEXT101_32X8D modified model
With the decrease in loss, the model gains in accuracy, as expected

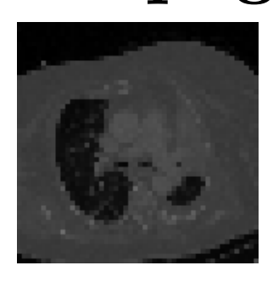
Missclassified examples:
9cdc1712-6ce.png - label: 1 pred: 2
80550e69-ead.png - label: 1 pred: 2
e9087030-61d.png - label: 1 pred: 2
9eb7a560-0c5.png - label: 2 pred: 1
5b0c836d-b98.png - label: 1 pred: 0
0153f8d1-9b1.png - label: 1 pred: 0
d970889f-4c6.png - label: 2 pred: 0
a80ce9e7-29b.png - label: 0 pred: 2

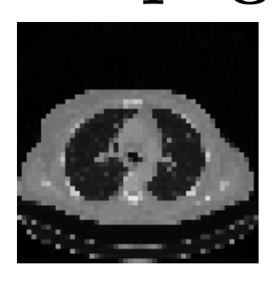

5b0c836d-b98.png

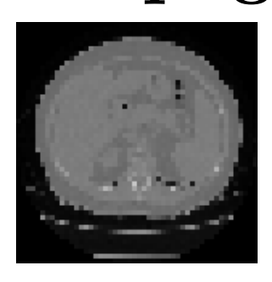

9eb7a560-0c5.png

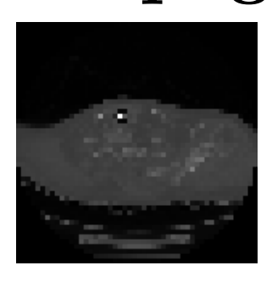

80550e69-ead.png


d970889f-4c6.png


9cdc1712-6ce.png


0153f8d1-9b1.png


a80ce9e7-29b.png


e9087030-61d.png

5. Conclusion

We have explored a support vector machine and two neural networks models. We conclude that models based on convolutions have a higher chance of learning useful information, thus they extracted futures play a major role for the accuracy of the models.
The most troublesome issues we've encountered came from the similarity of the images. Thus, we had to set up the convolution layers carefully, because images from different classes share some similar features.
Future work: I assume that giving multiple types of futures to the fully connected layers would help increase quality of the predictions. Thus, implementing inception blocks in our models is an option worth pursuing.