Image Segmentation and Classification on Spine_Dataset

Yogesh Kalakoti
Department of Biochemical Engineering and Biotechnology
Indian Institute of Technology Delhi
Delhi, India
bez187512@iitd.ac.in

Aayush Singha Roy
School of Information and Technology
Indian Institute of Technology Delhi,
Delhi, India
siy197557@iitd.ac.in

I. Introduction

This is the first assignment towards the fulfillment of our course “Advanced Machine Learning (ELL888)”, being taught by Dr.A.Prathosh.

II. Spine Dataset

The Training Dataset consisted X-ray images in two classes namely “Normal” and Damaged” of two different views AP and Lateral. After removing missing values there were in total of 626 images which contained 3 segmented masks of AP view and 5 of Lateral View. The task was of image segmentation and then classifying new images to one of the classes.

A. Image Segmentation

Pixel level binary classification task was performed on the image and the ground truth provided. Among the many architectures available, we chose to go with U-Net as it suited our problem. We used a helper package already there in keras as keras-unet where we could customize this U-Net according to our problem. The reference to this is https://github.com/karolzak/keras-unet#Customizable-U-Net Eight different models were prepared to segment the eight different type of masks. Dice score was used as a validation metric to test the performance of our models.

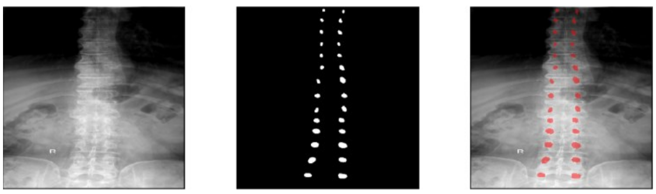


Figure 2: Representation of data

TensorFlow backend was used to train the models while keeping 30% of the data as test samples. Binary cross-entropy was used as a loss metric train the model along with Stochastic Gradient Descent (SGD) as an optimizer. Each segmentation model was run for 500 epochs and the runtime was about 40sec per epoch. Though the model was not saturated and had some more capacity to learn and improve, the computational constraints didn’t allow us to train our models beyond it.

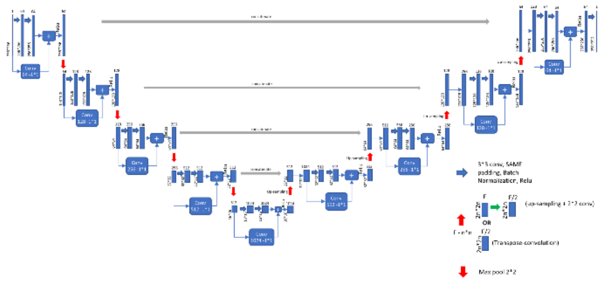


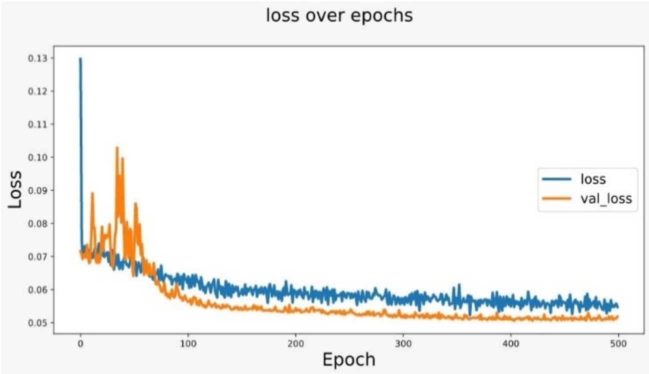
Figure 1: Architecture of U-Net

The table below summarizes the performance of the trained segmented models on training dataset as well as test dataset.

S.no.	Pose	Dice score (Avg)	Dice score (Avg) -test
1.	Ap_Pedicle	0.35	0.30
2.	Ap_Spinous_Process	-	-
3.	Ap_Vertebra	0.45	0.34
4.	Lat_Anterior_Vertebral_Line	0.12	0.08
5.	Lat_Disk_Height	-	-
6.	Lat_Posterior_Vertebral_Line	-	-
7.	Lat_Spinous_Process	0.23	0.20
8.	Lat_Vertebra	0.28	0.28

Table 1:Model performance for each semantic segmentation task

Due to some hardware bottlenecks and ambiguity in data, we were not able to train the semantic segmentation models for all of the eight type of masks. However, for the models that we were able to train, performed as expected and had significant room for improvement.



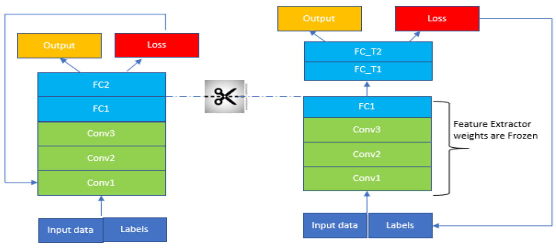
B. *Classification*

1) Classify1.ipynb

Here first we iterate through the files and form a numpy array of masks and labels 0 for Normal and 1 for Damaged. Here we take each mask independent of the other for classification classes. For example, for a Normal AP view, Ap_Pedicle is labeled as 0 and Ap_Spinous Process as 0.

By iterating through the files and resizing the images to (64,64) form an array of shape (5008,64,64,1). The conversion of labels in categorical form is also done.

Then we have used Pre-trained ResNet50 model. ResNet-50 is 50 layers deep network and has over 23 million trainable parameters. In tuning parameters, we have kept Dropout very high that is to 0.7 and used Adam optimizer with binary_crossentropy as loss for this binary classification problem.



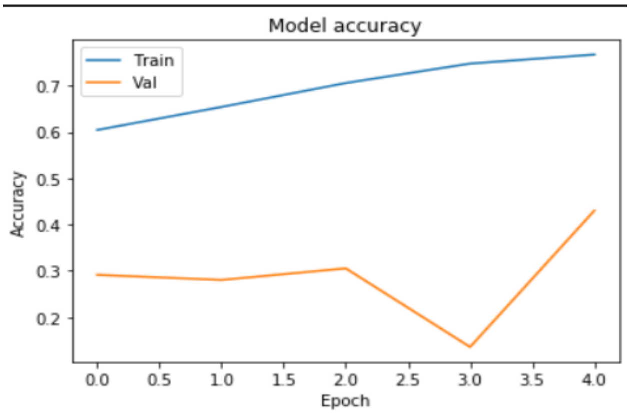
In ResNet50 we keep the include_top="false" so as to add a dense layer for our binary classification task.

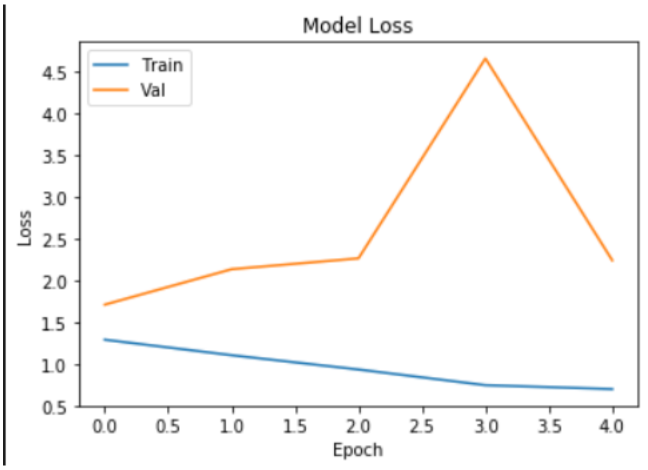
We fit the model with validation_split = 0.2 and ran it for only 5 epochs and a batch_size of 128.

```
Train on 4006 samples, validate on 1002 samples
Epoch 1/5
4006/4006 [==============================] - 1009s 252ms/step - loss: 1.2875 - acc: 0.6041 - val_loss: 1.7070 - val_acc: 0.2914
Epoch 2/5
4006/4006 [==============================] - 2445s 610ms/step - loss: 1.1019 - acc: 0.6538 - val_loss: 2.1317 - val_acc: 0.2804
Epoch 3/5
4006/4006 [==============================] - 911s 227ms/step - loss: 0.9307 - acc: 0.7054 - val_loss: 2.2614 - val_acc: 0.3054
Epoch 4/5
4006/4006 [==============================] - 7775s 2s/step - loss: 0.7431 - acc: 0.7474 - val_loss: 4.6550 - val_acc: 0.1357
Epoch 5/5
4006/4006 [==============================] - 2958s 738ms/step - loss: 0.6965 - acc: 0.7668 - val_loss: 2.2376 - val_acc: 0.4301
```

And after that the accuracy and loss graph was plotted we see a good jump in val_accuracy although it would improve more by increasing the number of epochs.

The fluctuation in both val_loss and val_accuracy was due to that many images were being given a classification score near 0.5. For example if after 1 epoch for an image the score was [0.53,0.47] the next epoch it became [0.48,0.52] leading to misclassification or classification and henceforth fluctuation in val_loss and val_accuracy.





After this we save the model as a h5 file namely my_model.h5.
'

2) test.ipynb

Next for predicting the labels of the test Images we use the tests masks. And calculate the score for each class for each particular masks. Hence now we have scores of both the classes of each of the 8 segmented masks of each patients both AP and Lateral view.

Here first also resize the image to (64,64) and load the model for prediction.

And for the deciding the final class we take the class which has the highest score over each of the segmented maks.

For example, for test Image Test ID (10) and the image 'Lat_Anterior_Posterior_Line.png'

[0.520282 0.479718]