Segmentation and Classification of Spine X-Rays

Ashish Gupta
Department of Mathematics
Indian Institute of Technology, Delhi
2017MT10520
ashish.gupta0925@gmail.com

Sakshi Taparia
Department of Mathematics
Indian Institute of Technology, Delhi
2017MT10748
sakshitaparia123@gmail.com

Abstract

The first half of the draft documents segmentation of X-ray images and the second half experiments with segmented images as well as original X-rays to perform binary classification of spines into "normal" and "damaged" categories. *U-Net* inspired architectures outperform all other models for segmentation of biomedical images because of their ability to learn from limited data. A DICE score approximately between 60-80 is achieved in 6 out of 8 segmentation cases. Further, training *ResNet-50* on X-ray images, coupled with transfer learning and feature extraction by pre-training on imagenet data set achieves classification accuracy of over 85%. Since pre-trained models have already learnt low-level features like edges and textures, they do not need to be re-learned during model fine-tuning.

1 Introduction

The data set consists of *X-ray images* of patients who visited the Indian Spinal Injuries Centre seeking treatment. The precise reason behind choosing X rays is that they are low cost as compared to MRI or CT Scan. The collected images of the patients focus on the thoracic and lumbar regions of the *vertebral column.* For each patient in the data set, we have two X-Ray images highlighting two different views, namely *AP and Lateral.*

1.1 Description

The fundamental labels for the data set are "normal" and "damaged". The data set comprises of 2 folders containing 328 "damaged" and 350 "normal" images of the spine. Each folder named as the patient ID contains two X-ray folders for the AP and the Lateral view.

For the AP view, we have 3 classes - Vertebra, Spinous Process, and Pedicle. Equivalently, for the lateral view, we have 5 classes that include Vertebra, Spinous Process, Disk Height, Anterior Vertebral Line, and Posterior Vertebral Line.

1.2 Data Analysis

The missing images provided separately have been merged into the original training data. Further, certain images are outside the designated folder. The name tags of the images are also non-identical. For instance, LAT X-ray image can be "lat.png", "LAT.png" or "LAT .png" in different patient folders. All such cases have been handled separately while taking the input.

Finally, the folder with **patient ID 169** is not considered for segmentation due to missing images.

Moreover, the X-ray images of different patients vary in size as documented in table 1.

Image Type	Min Height	Max Height	Min Width	Max Width
Normal	1714	3184	1211	2703
Damaged	2341	4280	1096	3520

Table 1: Image size variations

Some possible methods to standardize the image sizes are [1]:

Up-sizing images: When few pixels are scaled to higher resolution, it is necessary to "create" new pixels, to occupy the spaces that will appear. This can generate incorrect information.

Downsizing images: This method does not create any incorrect information and also reduces the input size. However, excessive down sampling might lead to significant information loss.

Padding with zeros: Since the back pixels of the segmented images do not capture any useful information, this method does not serve any specific benefit.

Since the width and height of the input images provided to U-net architecture should be divisible by 16 , we have downsized all images to a standard size of **512 * 256** pixels for model testing. However, for generating results are required, the size has been taken as 224 * 224 after replacing up-sampling in U-Net with transpose convolution.

1.3 Tasks

We are required to make a segmentation network to generate different kind of **segmentation** images corresponding to each X-ray. Specifically, we generate 3 segmented images for AP view and 5 segmented images for LAT view for all the patients. Next we are required to create a classifier that will **classify** the images into 2 categories - "normal" and "damaged”.

2 X-Ray Segmentation

Image segmentation is a computer vision technique used to understand a given image at a pixel level. Semantic segmentation, a category of image segmentation, essentially means a pixel level classification of an image i.e. each image pixel belongs to one particular class [2].

2.1 Related Work

Basic image segmentation techniques involve threshold methods, edge detection techniques, region based techniques, clustering techniques, watershed techniques, partial differential equation based and artificial neural network based techniques [3].

The state of the art approaches used for building semantic segmentation models are [4]:

- U-Net: Convolutional Networks for Biomedical Image Segmentation
- The One Hundred Layers Tiramisu: Fully Convolutional Dense-Nets for Semantic Segmentation
- Deep-Lab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRF
- Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation
- FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation
- Gated-SCNN: Gated Shape CNN for Semantic Segmentation

In biomedical image processing, it’s very crucial to get a class label for every cell in the image. The biggest challenge here is that thousands of images for training are not easily accessible.

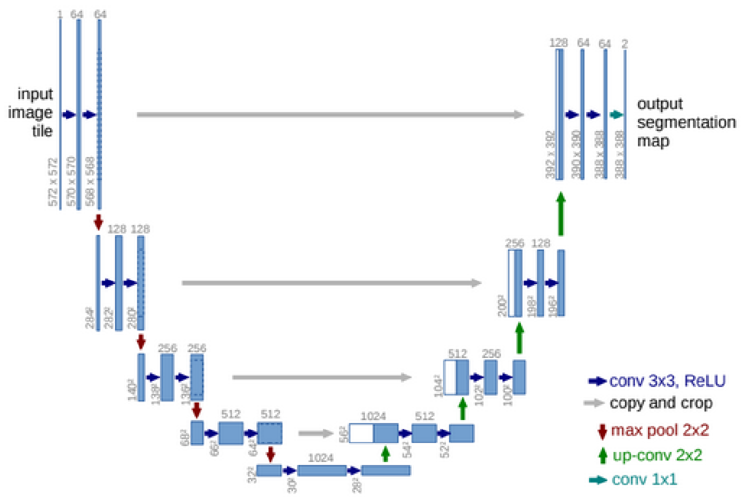


Figure 1: U-Net Architecture

U-Net outperforms other state of art models for biomedical image segmentation as this model uses data augmentation by applying elastic deformations on the available data[5]. The network architecture is made up of a contracting path on the left and an expansive path on the right.

The model builds upon the fully convolutional layer and modifies it to work on a few training images as well as yield more precise segmentation[6].

2.2 Experiments

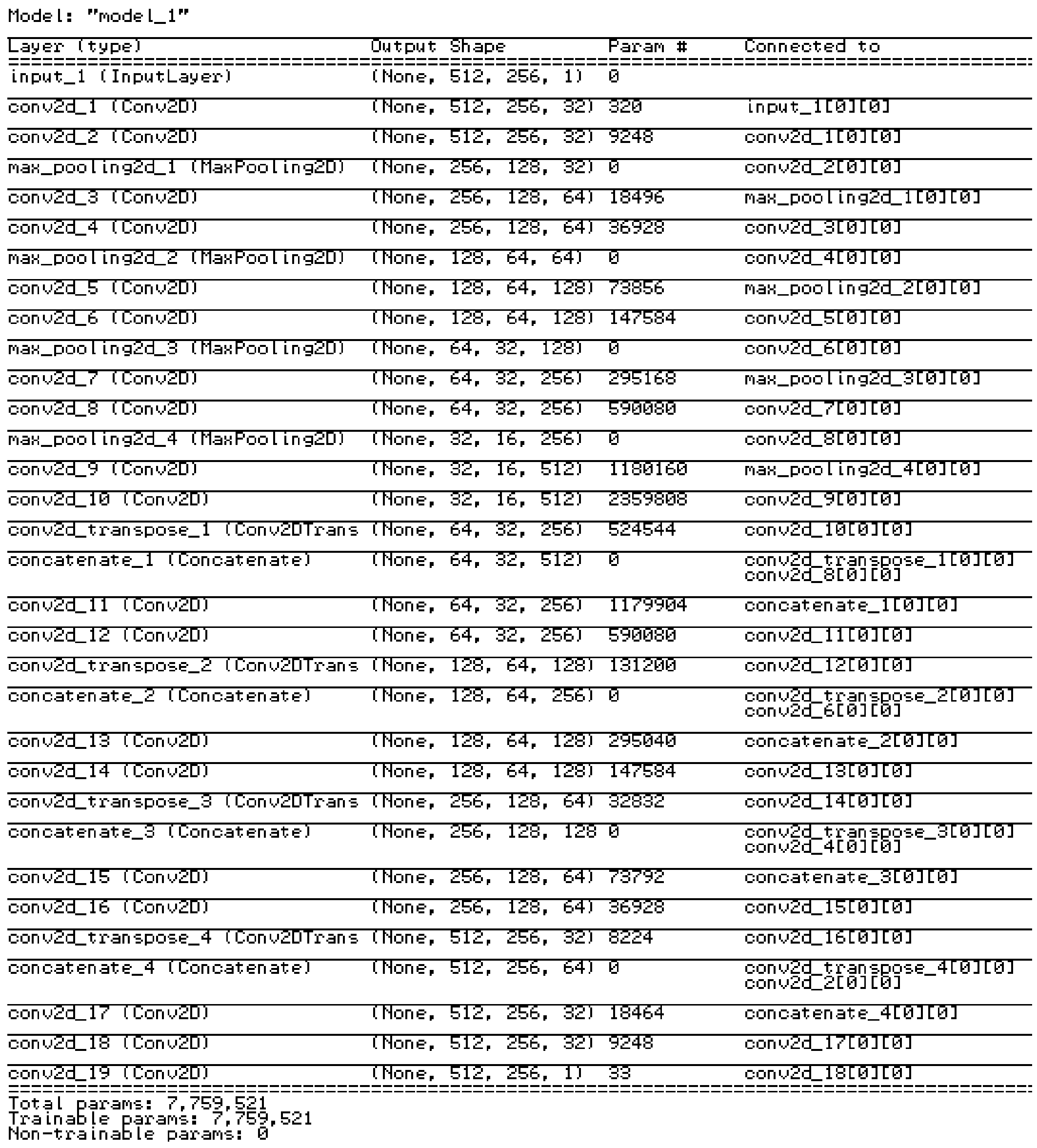
Motivation	Model	Loss	Result	Inference
Pixel-wise cross entropy loss (commonly used for segmentation) compares pixel individually to a target vector.	U-Net on the AP Pedicle image	Pixel-wise cross entropy loss and DICE metric	Segmented image is a black mask.	For images with unbalanced classes, regular cross entropy loss does not work well [7].
Intersection over Union metric for binary data measures overlap between two samples under the range [0,1] where 1 denotes perfect and complete overlap.	U-Net on the AP Pedicle image	Jaccard Index and DICE metric	Segmented image is a black mask.	Modifications are also required in the model architecture apart from the loss functions.
Modification in the U-net architecture is needed to allow learning	U-Net with batch normalization on the AP Pedicle image.	Jaccard Index and DICE metric	Segmented image is mostly black.	For deep networks, batch norm with convolutions might play an important role. However, this is not helpful with U-net.

Table 2: Experiments on X-ray segmentation

The Dice coefficient used as a validation metric is essentially a measure of overlap between two samples. This measure ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap. Further, Jaccard index is an equivalent of Intersection over union (IoU) metric which is differentiable and can be used as a loss function for training the model.

2.3 Methodology

Our final loss function is designed to give extra importance to learning the white pixels and the final training model used follows a modified U-Net inspired architecture.



```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 512, 256, 1)  0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 512, 256, 32) 320         input_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 512, 256, 32) 9248        conv2d_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 256, 128, 32) 0           conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 256, 128, 64) 18496       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 256, 128, 64) 36928       conv2d_3[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 128, 64, 64)  0           conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 128, 64, 128) 73856       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 128, 64, 128) 147584      conv2d_5[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 64, 32, 128)  0           conv2d_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 64, 32, 256)  295168      max_pooling2d_3[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 64, 32, 256)  590080      conv2d_7[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 32, 16, 256)  0           conv2d_8[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 32, 16, 512)  1180160     max_pooling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 32, 16, 512)  2359808     conv2d_9[0][0]
__________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 64, 32, 256)  524544      conv2d_10[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 64, 32, 512)  0           conv2d_transpose_1[0][0]
                                                                 conv2d_8[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 64, 32, 256)  1179904     concatenate_1[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 64, 32, 256)  590080      conv2d_11[0][0]
__________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 128, 64, 128) 131200      conv2d_12[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 128, 64, 256) 0           conv2d_transpose_2[0][0]
                                                                 conv2d_6[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 128, 64, 128) 295040      concatenate_2[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 128, 64, 128) 147584      conv2d_13[0][0]
__________________________________________________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 256, 128, 64) 32832       conv2d_14[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 256, 128, 128 0           conv2d_transpose_3[0][0]
                                                                 conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 256, 128, 64) 73792       concatenate_3[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 256, 128, 64) 36928       conv2d_15[0][0]
__________________________________________________________________________________________________
conv2d_transpose_4 (Conv2DTrans (None, 512, 256, 32) 8224        conv2d_16[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 512, 256, 64) 0           conv2d_transpose_4[0][0]
                                                                 conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 512, 256, 32) 18464       concatenate_4[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 512, 256, 32) 9248        conv2d_17[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 512, 256, 1)  33          conv2d_18[0][0]
==================================================================================================
Total params: 7,759,521
Trainable params: 7,759,521
Non-trainable params: 0
```

Figure 2: u-Net Inspired Architecture [8]

The custom loss function used is inspired by Jaccard Index where the numerator is the **square of intersection** value between actual and predicted image instead of just the intersection value.

Loss = [(intersection)^2 + smooth] / [sum - intersection + smooth]

Batch size has been set to 1 due to restrictions on resource allocation by Kaggle. The training images have been shuffled randomly. The model trains on the first 610 images and validates on the remaining 68 images. The model requires approximately 100 iterations in every case as shown in the graphs below.

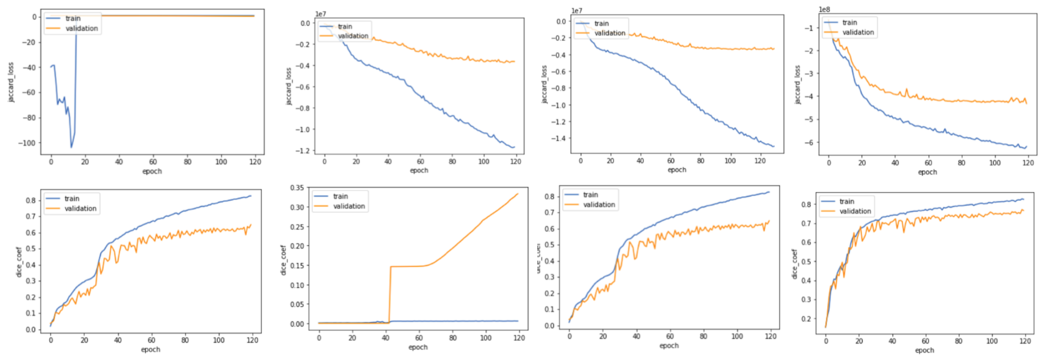


Figure 3: AP Vertebra, AP Spinous Process, AP Pedicle and LAT Vertebra (left to right)

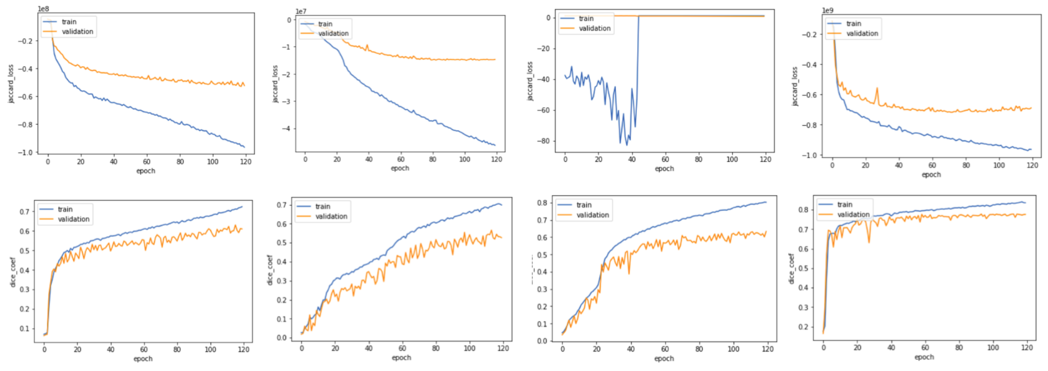


Figure 4: LAT Disk Height, Spinous Process, Posterior line and Anterior line (left to right).

2.4 Results

The DICE scores obtained between the predictions on the test data and the masks provided for the test data are:

Image Type	DICE Score
AP Vertebra	0.7711
AP Pedicle	0.6163
AP Spinous Process	0.5710
LAT Vertebra	0.7619
LAT Disk Height	0.6336
LAT Spinous Process	0.6020
LAT Anterior Vertebral Line	0.4463
Lat Posterior Vertebral Line	0.5449

Table 3: Segmentation accuracy

For the LAT Anterior Vertebral Line and Lat Posterior Vertebral Line, segmentation leads to an almost black image. This is because the line to be detected is extremely faint and further information loss occurs due to down sampling.

2.5 Discussion

The aim of this kernel is to develop a robust deep learning model from scratch on this limited amount of data. Some noteworthy features of the new architecture [9] are:

2.5.1 No dropout

Dropout is a regularization technique for reducing over fitting in neural networks by preventing complex co-adaptations on training data. Since validation and training accuracies in our case do not differ significantly, over fitting is unlikely and dropout is not needed.

2.5.2 No batch normalization

For deep networks, batch norm with convolutions might play an important role. However, it is not helpful in case of the architecture implemented by us.

2.5.3 Transpose convolution instead of up-sampling

Preserving the image dimensions throughout can be computationally expensive. One popular approach is to down sample the spatial resolution of the input, develop low resolution feature mappings and then up sample the feature representations into a full-resolution segmentation map.

There are different approaches for up sampling. Just like the pooling operations summarize a local area with a single value (average or maximum), the "unpooling" operations distribute a single value into a higher resolution.

However, transpose convolutions are by far the most popular approach as they allow for us to develop a learned up sampling. A typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position. On the contrary, a transpose convolution will take a single value from the low resolution feature map and multiply all of the weights in our filter by this value, projecting those weighted values into the output feature map.

2.5.4 No data augmentation

Data augmentation is a powerful technique which helps in improving the robustness of a model. It can generate different samples of under - sampled class in order to try to balance the overall distribution. However, the usefulness and type of data augmentation required depends on the problem domain. in this case it does not work.

2.5.5 Custom Loss Function

The custom loss function inspired by Jaccard index is biased to detect white pixels as white instead of detecting black pixels black i.e. it gives lot of weight to increasing the intersection of white pixels between the predicted and actual images.

2.5.6 Further suggestions

Some papers propose the use of residual connections, dense blocks and dilated convolutions still following a U-Net structure to improve the current model [10].

Increasing the size of input images and modifying the data by cropping images to capture only relevant parts can be helpful if the white pixels are too less to detect. Loss function can also be modified.

3 Spine Classification

Different input combinations out of the 10 input images per patient including AP and LAT X-rays, 3 AP - segmented images and 5 LAT - segmented images, is used to determine whether the patient's spine belongs to "damaged" (1) or "normal" (0) class.

3.1 Related Work

The state of the art models used for classification are [11]:

- VGG-16 and 19
- ResNet-50
- Inception-V3
- Xception

The **VGG network** architecture is characterized by its simplicity. It uses only 3*3 convolutional layers stacked on top of each other in increasing depth. Reduction of volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a soft max classifier layer.

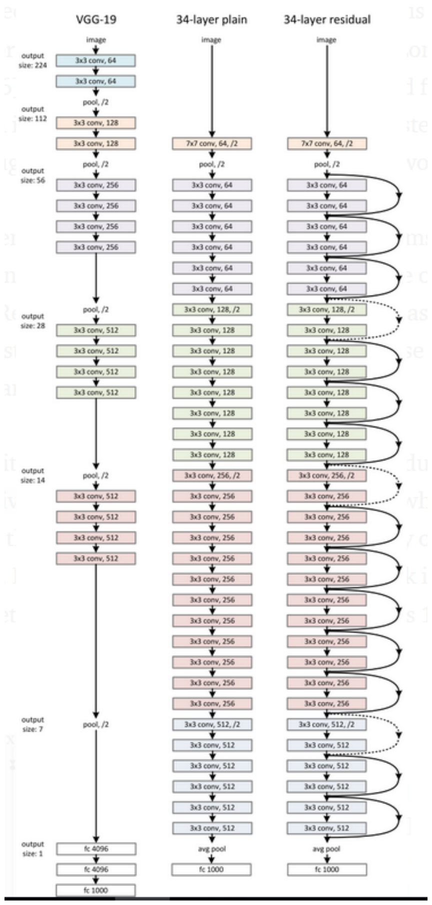


Figure 5: ResNet-50 Architecture

Unlike traditional sequential network architectures such as AlexNet, OverFeat or VGG - the **ResNet** model is an “exotic architecture” that relies on micro-architecture modules, also called network-in-network (NIN) architectures coupled with convolution and pooling layers. The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers. ResNet architecture demonstrates how extremely deep networks can be trained using standard Stochastic Gradient Descent and a reasonable initialization function through the use of **residual modules.**

The **inception net** module acts as a “multi-level feature extractor” by computing 1*1, 3*3, and 5*5 convolutions within the same module of the network — the output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network.

Inception is created to serve the purpose of **reducing the computational burden** of deep neural nets while obtaining state-of-art performance. The weights for Inception V3 are smaller than both VGG and ResNet, coming in at 96MB. While Inception focuses on computational cost, ResNet focuses on computational accuracy.

3.2 Experiments

Since the data set is balanced with respect to the two classes, therefore the binary cross entropy loss and accuracy metric works well.

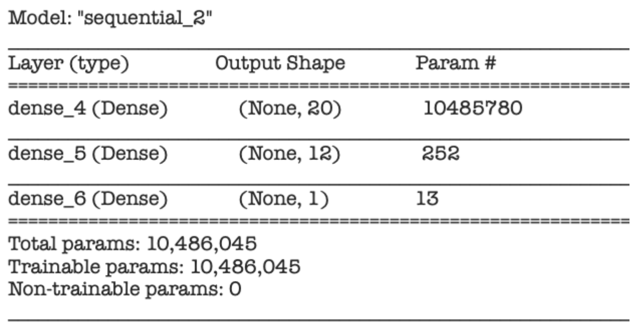
Motivation	Input	Model	Result	Inference
Segmented Images of X-ray may capture only the relevant features while discarding the rest and assisting with classification.	Segmented images along 8 different channels of input array.	Standard VGG Net, ResNet-50 and Inception Net with random initialization and data augmentation.	Approx. 50% accuracy	Learning is negligible. Either the model or the input needs to be changed.
Use complete X-ray images instead of the segmented ones to prevent any information loss.	AP and LT X-rays in two in different channels	Standard VGG Net, ResNet-50 and Inception Net with random initialization and data augmentation.	Approx. 50% accuracy	Learning is negligible. Either the model or the input needs to be changed.
Transfer learning stores knowledge gained while solving one problem and applies it to a different but related problem to reduce learning time as well as enhance performance.	AP and LT X-rays in two in different channels	ResNet-50 with initialization of weights by pre-training on imagenet data set and data augmentation.	Approx. 55% accuracy	Pre-trained models preform better. Further, format for providing input needs to be changed.
Freeze first 5 layers of ResNet-50 and train the rest	Only AP X-ray images	ResNet-50 pre-trained on imagenet data set and freezing weights of first 5 layers.	Approx. 75% accuracy	Pre-trained models preform better. AP or LAT X-ray images should not be combined in different channels of input.
Initialize the first few layers from a network that is pre-trained on imagenet. This is because first few layers capture general details like color blobs, patches, edges, etc.	Only AP X-ray images	Complete ResNet-50 except last layer and add 2 dense layers	Greater than 80% accuracy	Instead of randomly initialized weights for initial layers, it would be much better if you fine tune them to capture general features.

Table 4: Experiments on spine image classification

Choosing a data set for pre-training requires it to have similarity with the current data set in terms of nature. Imagenet is a project aimed at manually labeling and categorizing images into almost 22,000 separate categories for computer vision research. Transfer learning and feature extraction can be extremely useful to speed up the model and enhance classification accuracy [15].

3.3 Methodology

ResNet-50 model pre-trained on imagenet data set is used to extract the feature vectors of AP X-ray and LAT X-ray separately. Both the feature vectors are then flattened and appended into a single vector. Since we input feature vectors instead of images to the next model, data augmentation is not used.



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)              Output Shape          Param #
=================================================================
dense_4 (Dense)             (None, 20)            10485780
_________________________________________________________________
dense_5 (Dense)             (None, 12)            252
_________________________________________________________________
dense_6 (Dense)             (None, 1)             13
=================================================================
Total params: 10,486,045
Trainable params: 10,486,045
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Classification architecture with feature vector as input

The architecture contains 2 dense layers followed by an output layer. The activation function used is ReLu as the data requires binary classification only.

The loss function is set as binary cross entropy loss with accuracy metric. Again, the batch size is set to 1 due to resource limitation. The training data is shuffled followed by training on 600 images and validation on 78 images. Learning rate scheduler is enabled to allow learning rate to decay during the latter iterations. The model is required to run for nearly 50 iterations as shown below.

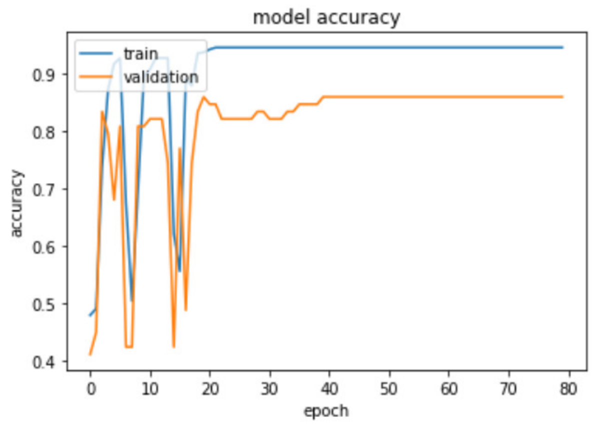


Figure 7: Training vs Validation accuracy for classification

3.4 Results

The dimension of the output from ResNet-50 is (678, 16, 8, 2048) - indicating that 16*8 size images and 2048 channels of features are generated per input image. Some sample features captured by the last layer of ResNet-50 are shown below.

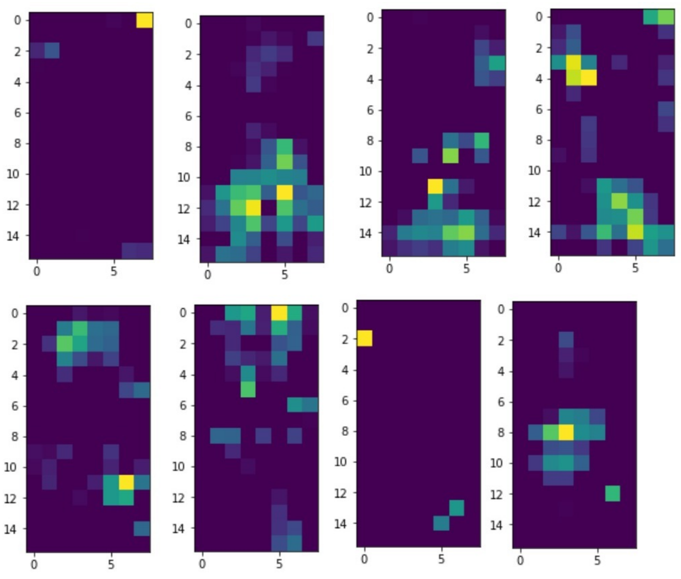
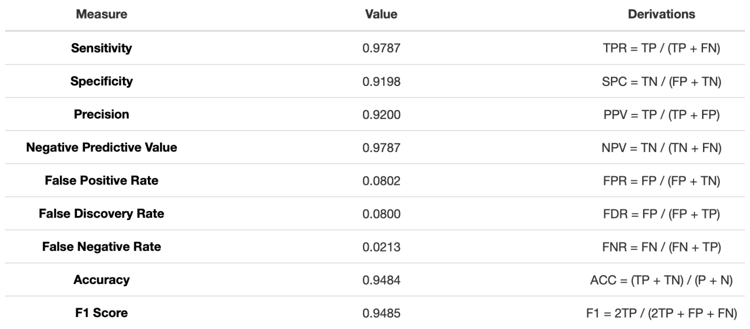


Figure 8: Feature vectors

The data below documents the classification accuracies obtained during validation.

	True Positive	True Negative
Predicted Positive	322	28
Predicted Negative	7	321



Measure	Value	Derivations
Sensitivity	0.9787	TPR = TP / (TP + FN)
Specificity	0.9198	SPC = TN / (FP + TN)
Precision	0.9200	PPV = TP / (TP + FP)
Negative Predictive Value	0.9787	NPV = TN / (TN + FN)
False Positive Rate	0.0802	FPR = FP / (FP + TN)
False Discovery Rate	0.0800	FDR = FP / (FP + TP)
False Negative Rate	0.0213	FNR = FN / (FN + TP)
Accuracy	0.9484	ACC = (TP + TN) / (P + N)
F1 Score	0.9485	F1 = 2TP / (2TP + FP + FN)

Figure 9: Confusion matrix for classification

3.5 Discussion

ResNet has several compelling advantages - it alleviates the vanishing-gradient problem, strengthens feature propagation, encourage feature reuse and substantially reduces the number of parameters. ResNet architecture proposes residual connection, from previous layers to the current one making it one of the best performing architectures.

Pre-training on imagenet data set can be used as a method to extract relevant features. Typical imagenet pre-training involves over one million images iterated for one hundred epochs. This helps to speed up convergence on the target task early on in training. In addition to any semantic information learned from this large-scale data, the pre-training model has also learned low-level features that do not need to be re-learned during fine-tuning.

Finally, the feature vectors of AP and LT are appended to capture the information from both cases. Few dense layers with reasonable amount of neurons are added to the output of ResNet-50. Training the network with a lower learning rate along with decay ensures that the model takes "small steps" while learning and does not miss the optimal point.

References

[1] https://medium.com/neuronio/how-to-deal-with-image-resizing-in-deep-learning-e5177fad7d89

[2] https://www.fritz.ai/image-segmentation/

[3] https://www.ripublication.com/irph/ijict_spl/ijictv4n14spl_13.pdf

[4] https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc

[5] https://heartbeat.fritz.ai/deep-learning-for-image-segmentation-u-net-architecture-ff17f6e4c1cf

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Computer Science Department and BIOSS Centre for Biological Signalling Studies*. University of Freiburg, Germany.

[7] https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/

[8] https://www.kaggle.com/micajoumathematics/my-first-semantic-segmentation-keras-u-net

[9] https://www.jeremyjordan.me/semantic-segmentation/#advanced_unet

[10] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Computer Vision and Pattern Recognition*.

[11] https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/