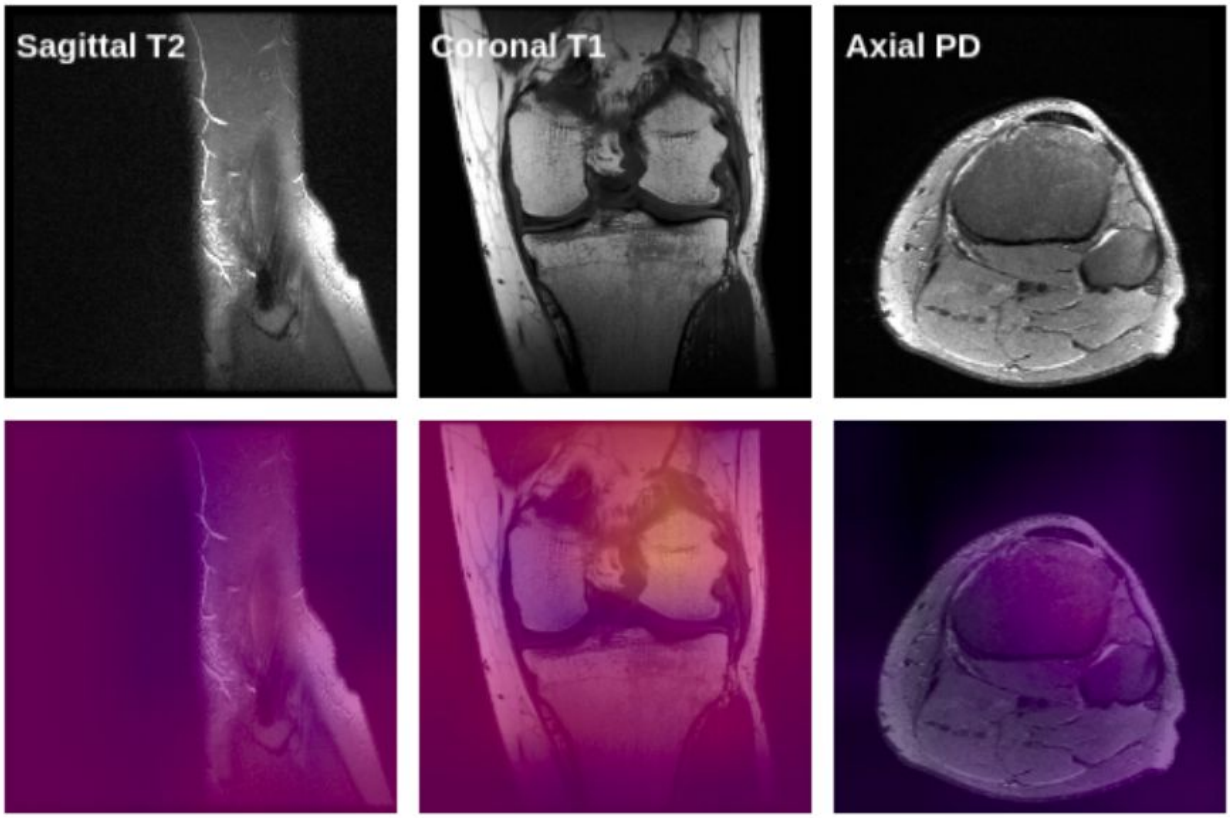
Operation Research

MRNet: Deep-learning-assisted diagnosis for knee magnetic resonance imaging

The individual report

Name: Ahmed Mohamed Abdel-Hameed Elzeny ID: 08



Introduction

This is an individual report for the Operation research course 3rd-year CSED, to record the making of a Resnet CNN using Keras from scratch and the training and parameter Tunning to get the most out of the network.

Methods

First of all, we took some time to analyze and understand the dataset sense we have no medical background, then we needed a Keras crash course to figure out how we'll but the theoretical knowledge we have into action, learning about Keras APIs it was clear that we'll be using the functional API to implement the Resnet model to simulate its links between different node.

Data loading and preprocessing

The model loads the data from a data generator, I tried data augmentation on the fly but the training and validation results were bad, so I trained the models without data augmentation

.Model structure

The model has two main blocks, the Identity block, and the convolution block figure 1,2.

The ID block consists of three sub-block of 3 layers a convolutional then a BatchNorm layer then a ReLu activation function, and a shortcut between the start of the block and before the last activation function.

the Convolution block is similar to the ID block except that the shortcut contains a convolutional layer and a BatchNorm layer.

The model is laid out like figure 3 we first scale the dataset images to 3*224*224, and its full description is figure 4

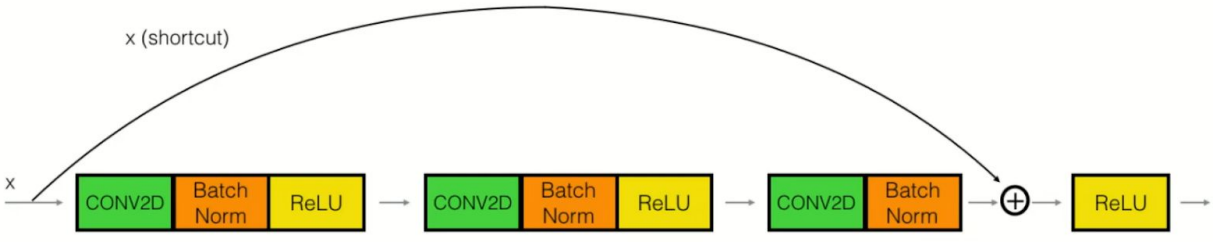


Figure 1: An Identity block skipping 3 convolutions

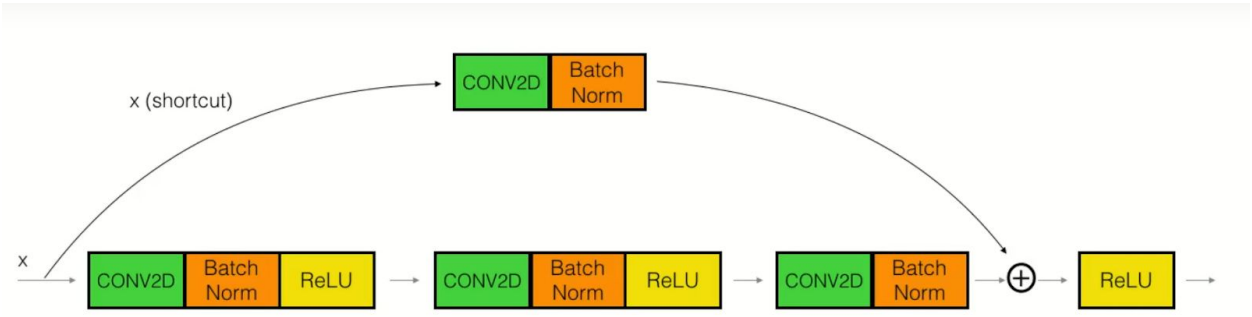


Figure 2: A convolutional block that skips three convolutions and has a convolution on the shortcut

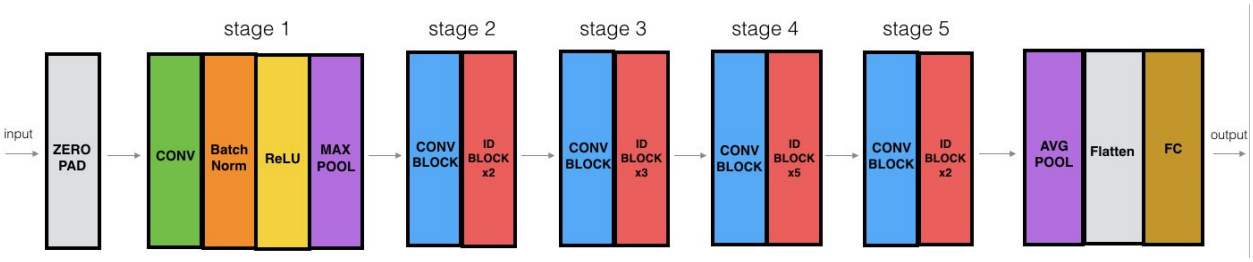


Figure 3: The implementation pipeline

$$
\begin{aligned}
&7 \times 7, 64, \text{stride } 2 \\
&3 \times 3, \text{max pool} \\
&\text{stride } 2 \\
&\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3 \\
&\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4 \\
&\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6 \\
&\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 \\
&9 \times 1, 2048, \text{stride } 1 \\
&1 \times N, \text{avg pool} \\
&\text{stride } 1
\end{aligned}
$$

Figure 4: pipeline parameters

We have added a layer that adds the model into a sequential model and compiles it with the hyperparameters we tune (will be mentioned in the trials section) changing these parameters until finding the chosen optimizer and learning rate.

Trials and findings

After implementing the model we tried training using the sagittal abnormal data as a test with data augmentation using different of optimizers with different learning rates and epochs like:

- optimizer: Adam, lr= 1e-2 : 1e-5, epochs=14 settles on Accuracy 73%
- optimizer: SGD, lr= 1e-2 : 1e-5, epochs=14 settles on Accuracy 61%
- optimizer: Adadelta, lr= 1e-2 : 1e-5, epochs=14 settles on Accuracy 66%
- optimizer: RMSprop, lr= 1e-2 : 1e-5, epochs=14 settles on Accuracy 59%

The training took an average of 20 min/epoch, going to the validation step, it dropped to arround 50% of accuracy, trying other network like resnet152 failed with an exhausted resources error but it was clear that we are continuing with the adam optimizer.

We tried to get some better training accuracy results by disabling the data augmentation, after that the accuracy of the training went up to 93% and the validation accuracy to around 78% in the chosen model, after trying different learning rates manually and different epoch numbers to avoid overfitting.

We trained nine models for all the dataset using adam optimizer, leaning rate of 1e-4, and obtained the following results:

		training		Validation	
		Accuracy %	AUC %	Accuracy %	AUC %
Abnormal	axial	93.9	98.6	78	48
	coronal	92.4	97.21	79.1	58
	sagittal	92.6	97.5	79.1	53
ACL	axial	90.1	97.37	53	48
	coronal	93.1	97.25	55.8	50.8
	sagittal	93.2	97.0	55	45
Meniscus	axial	90.82	96.72	52	54
	coronal	92.4	98.1	52.5	52.1
	sagittal	94.3	98.8	63	60

Training and validating we discovered that the models always choose the bigger class, having that in mind we tried to load the resulting weights and train them with data augmentation on the Meniscus sagittal model and that resulted in the last row of the above table, neutralizing the negative and positive classes, with the following result on the validation data:

True positive = 25, false positive = 17, true negative = 51, false negative = 27

The very time expensive training of the 22 epochs took around 7 hours concluding that this model (Resnet 50) won't be the premium model to apply transfer learning to.