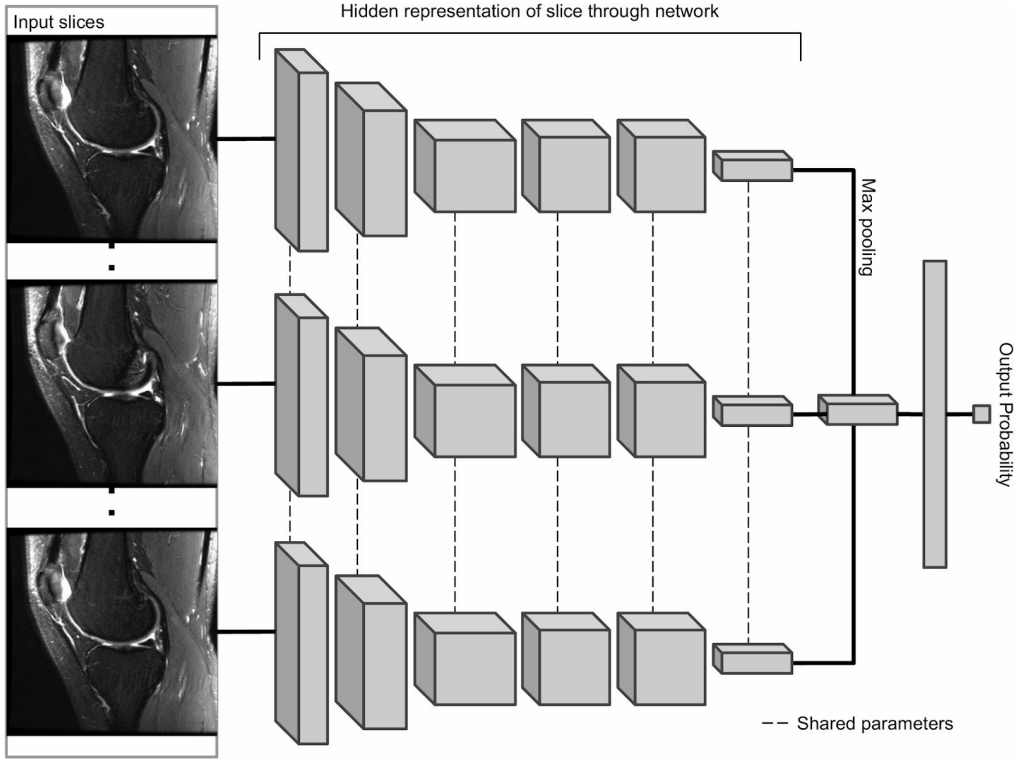
Operations Research

Final Project (Group Report)

MIRNET for knee diagnosis



Team members:

Sherif Mohamed Mostafa 22

Hazem Morsy 18

Ahmed Ali 7

Jamal eldin ahmed 17

Abdelrahman Kamal 25

Abstract

Background

Magnetic resonance imaging (MRI)

It's a type of medical scan and it's a common procedure around the world.

MRI uses a strong magnetic field and radio waves to create detailed images of the organs and tissues within the body.

(MRI) is preferred for diagnosing knee injuries. However, its interpretation is time-intensive and subject to diagnostic error and variability.

Thus the need for An automated system for interpreting knee MRI rises, it could prioritize high-risk patients and assist clinicians in making diagnoses. Deep learning ,in learning layers of features, is a good choice for modeling the complex relationships between medical images and their interpretations.

So here we are trying to develop a deep learning model for detecting general abnormalities and specific diagnoses (anterior cruciate ligament [ACL] tears and meniscal tears) on knee MRI exams.

Methods

We used the data provided in the MRnet paper for training, validation and testing. The data was first preprocessed to be used in our implementation. This preprocessing involved stacking each input slices three times to obtain an image-like format. This is because the input format provided is in the form of numpy arrays of shape sx256x256 not sx3x256x256 as stated in the paper. Data augmentation was used in training the feature extractors to obtain more input data for training.

We created 3 Models to Simulate MRNet VGG , RESNET and inception v3 .Each one is a convolutional neural network for classifying MRI as well as some transfer learning based models . Then we compare the final results of these networks.

Our approach for training was to divide the MRnet model to three parts: feature extraction, average and maximum pooling and binary classification. The results corresponding to the same anomaly were used in training a logistic regression layer to obtain 3 final results as was done in the original paper implementation.

The results of using this approach without transfer learning (using the three specified models from scratch) were very close so we tried transfer learning on all the models. When using transfer learning on VGG16, inception V3, ResNet50 and DenseNet networks that are built in Keras (using image net weights), we found that ResNet50 was the best in this approach and inception V3 was the worst. This goes along with the results stated in the paper ***"Deep Learning for Musculoskeletal Image Analysis"****.*

We then tried the model used in the paper ***"Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases"*** and recorded the results.

Introduction

Problem statement

The problem required to be solved is at its root an image recognition problem (more precisely image classification). Given as input three sets (slices) of image series for a patients MRI knee examination, the required is to train a neural network model that diagnoses the examination and gives three outputs. The first is the probability of the patient having abnormalities in his knee. The second and third further classify these abnormalities showing the probability that the patient has ACL tears or meniscal tears respectively. This is done by building the MRnet model and training it by the help of the MRnet data set using deep learning techniques.

Objective

Our main objective is implementing the deep learning neural network MRnet. This is done by trying the paper's approach using different building blocks than those used in the paper. Mainly by using feature extractors based on architectures other than AlexNet that were used in the paper's original implementation and comparing the obtained results. These included other sub objectives as building the model architectures from scratch and using transfer-learning techniques. Also, a final objective is to try new solutions based on other papers results as a contribution to the problem at hand.

Dataset

The data set was the same as the data set used in the paper but with some modifications to make it more convenient . It was downloaded and uploaded to the drive to be used in the training. The data was preprocessed to be in format similar to that stated in the paper (with shape of sx256x256x3). We used channels last convention instead of channels first that was used in the paper. The test data is inaccessible, so we used the validation data set for testing and the training dataset was divided into training and validation at a ratio of 90% to 10% (1017 samples in training data set and 113 samples in validation data set). When training the feature extractors, data augmentation was used based on the approach of the paper (more details are present in the elaboration of our used model approach).

Models

Due to the problem we faced in the models resulting from that the max pool layer in the model is not performed on the parameters of the same channel but on the multiple images of the slice we had to split the MRnet model itself to 3 parts (extractor then the pooling then the classifier).

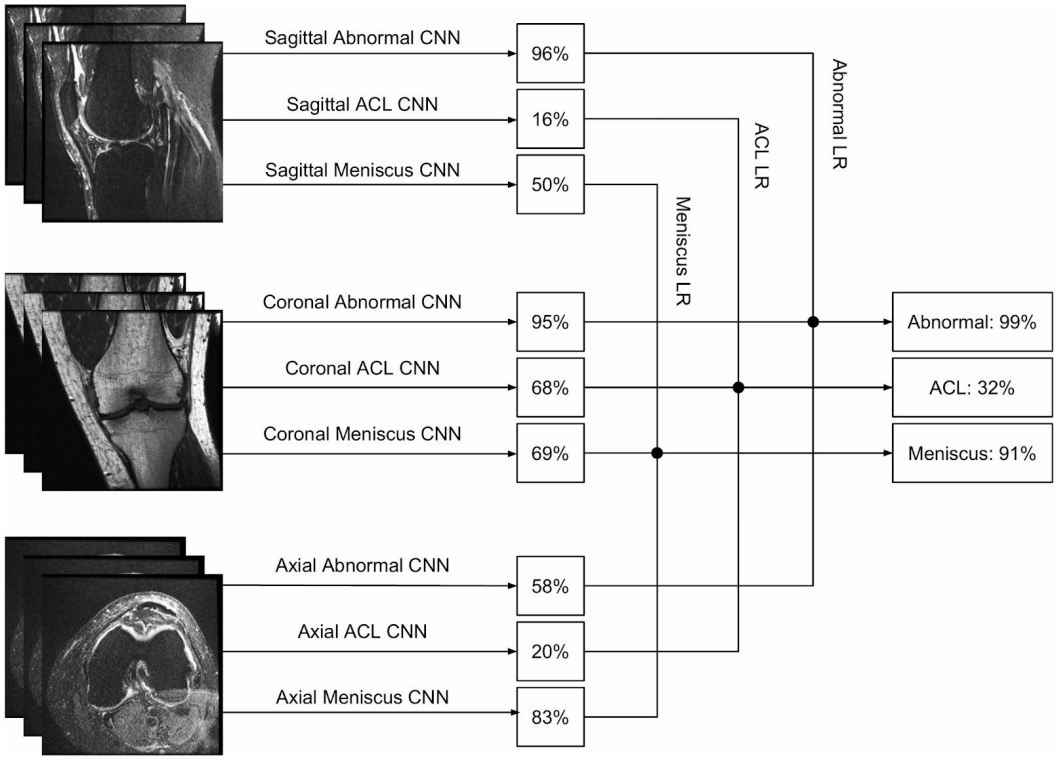
To train the extractor we added a temporary classifier to it so that we can train this full model using one representative image from the s images .

Then we loaded only the extractor weights and then used them to extract the features from the images and use them as input to the pooling layer, then use the output of the pooling layer as input to the classifier to train it (As if we are using transfer learning on our own trained model).

Then the results of the three scans of the same class are used as input to the regressor that outputs the final prediction of this class.

All of this work was made in a generic approach to support any architecture for the feature extractor, binary classifier and logistic regression layer used.

To separate the model parts, we saved the results of training each part on our drive to be used later in testing and training other parts. To save the trained models, we used two callbacks: one saves the model weights at the epoch giving best validation accuracy and another one for early stopping of the training procedure if the validation loss didn’t improve for 5 epochs. The training was made with batch size of 20 and total 50 epochs which was not reached due to the early stopping call back kicking in.



This notebook contains the complete implementation of the functions.

More details about the model are in the first individual report.

Workflow

The paper

First we had to understand the problem , dataset and the model we are going to implement , so the paper itself helped with this

Link of the paper https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002699

Then to deal with the dataset , download it to the drive to use it in the train , know how to split it , also to train in the same way for the three models we created one notebook to be used as reference for all models and the train becomes generic

Link of the notebook

https://colab.research.google.com/drive/1CfZr__riY1RwLWtPx4FhIhmCHGfX1rVn?usp=sharing

And then there is a notebook for each model with the results of the training and the test

First: vgg model

VGG consists mainly of 16 sequential layers . it is a simple convolutional neural network (than other CNNs like AlexNet) and it has simple hyperparameters because all the filters are of the same size for convolutional layers (3 X 3) and same padding and of size (2 X 2) for the max pooling layers with stride of 2. We also added some dropout layers to overcome the overfitting and enhance the model

More details about the model is found in the second individual report (from the beginning until the transfer learning section)

The following links of the notebooks are for training extractors , classifiers and regressors of VGG

Link of the notebook(training extractors)

https://colab.research.google.com/drive/1WiCr7szj6qCFPv-9CLdg6M-tlOc9bVEb?usp=sharing

Link of the notebook(training classifiers and regressors)

https://colab.research.google.com/drive/1GsrltW1siTznBpY-133pu2Kry4uA9c61?usp=sharing

Second: resnet model

Resnet consists of residual block which is not as the regular (sequential) blocks , it's used to overcome the problem of overfitting in the complex networks as it allow the network to skip some layers if they make the result worse (if it's hard to learn new things in this layer) and this is implemented by adding the input of the block to the output of the block before doing the activation function , there are 2 types of residual blocks identical and conv , the main difference is that if the i/p and o/p dimensions are not the same we can't perform the addition operation so we have to do conv operation to make the o/p and the i/p of the same dimensions to perform the addition . and this addition (shortcut or skipping) is what allows the model to skip some layers and just pass the weights of the i/p to the o/p of the same block

We implemented the resnet with the 50 layer, details of which are in the fifth individual report.

Link of the notebook

https://colab.research.google.com/drive/1KFx0QMeA2wHgIfDFKukuDjga0gqdyEpb?usp=sharing

Third: inception model

Inception-v3 is a convolutional neural network, used in image recognition, that is 48 layers deep and consists of 11 inception blocks of 5 different types.

The model is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers.

Details about the architecture are shown in the third and fourth individual reports.

Link of the notebook

https://drive.google.com/file/d/1b3zR59aFtxFe-6JtClIcezZnWDOCvfYT/view?usp=sharing

Then the transfer learning models

Transfer learning is one of the widely used practices in deep learning. It helps in obtaining good results even if a small data set is available for training. This is done by using pre-trained feature extractors, used in solving a similar problem similar to the one at hand, in extracting features to build the model for solving our problem.

The requirement was to try transfer learning on the model giving the best results from the 3 models implemented from scratch, however we found that the results before transfer learning were very close. So we tried to perform transfer learning using several different architectures in order to find the best architecture.

Transfer learning was made using the image net weights for the built in models in Keras.

First: transfer learning on inception V3

We found that this approach gave the worst results when using transfer learning. We tried several classifier formats to achieve better results. We then trained 3 logistic regression models based on the obtained classifiers and recorded the testing results. The final results were:

Anomaly	Accuracy	Loss
Abnormal	79.17%	0.5504
ACL	54.17%	0.7141
Meniscal	56.67%	0.7

For more details about the implementation, results, failed trials and the successful ones, refer to the first individual report (transfer learning section) and notebook links.

Link of the notebook

https://colab.research.google.com/drive/1i_j9tqZOMTrgN1C6lBixZWqfQj5PwTg6?usp=sharing

Second: transfer learning on VGG16

We found that this approach gave intermediate and somewhat satisfactory results when using transfer learning. Same as with transfer learning using inception V3, we tried several classifier formats to achieve better results. We then trained 3 logistic regression models based on the obtained classifiers and recorded the testing results. The final results were:

The results were found to be better than using inception V3 with the same binary classifier architecture.

Anomaly	Accuracy	Loss
Abnormal	80.83%	0.5338
ACL	62.5%	0.6529
Meniscal	59.17%	0.6374

For more details about the implementation, results, failed trials and the successful ones, refer to the first individual report (transfer learning section) and notebook links.

Link of the notebook

https://colab.research.google.com/drive/1lXXRQT74w4aiwJVIOH7Pq-CvNsTxnetK?usp=sharing

Third: transfer learning on RESNet50

We found that this model gives the best results among the three models used for transfer learning (VGG , RESNET and Inception V3) . We made some trials to try to get the best classifier that is suitable for the model.

In the following table we will show the results for test data that we got after training the regressors of the three anomalies.

	Accuracy	Loss
Abnormal	0.8333	0.3940
ACL	0.6583	0.6274
Meniscal	0.7083	0.5947

For more details about the implementation, results, failed trials and the successful ones, refer to the second individual report (transfer learning section) and notebook links.

Link of the notebook

https://colab.research.google.com/drive/1Q8nBwQi1rF4S0btOmzYwJfRVxl-XHGOg?usp=sharing

Contributions

First contribution

Conclusion

The paper used the same model as the MRNet paper by transfer learning (training extractors on Imagenet) but in addition to AlexNet used as a feature extractor , they also used GoogleNet and ResNet-18 convolutional neural networks.

It also suggests using advanced network architecture such as DenseNet(the model we used) or CapsNet model.

For more details about DenseNet , implementation, results, failed trials and the successful ones, refer to the second individual report (Contribution section) and notebook links.

Link of the notebook

https://colab.research.google.com/drive/1i5K2d9tIarfOoYeMyjJZ64n2rb0nnHBT?usp=sharing

second contribution

Conclusion

We used the CNN architecture instructed in the paper with a few changes to fit our model.

The purpose of the model used in the paper was to automatically identify the contrasts of MRI scans based on image intensity which helped them to overcome unlabeled contrasts within the received data.

More about the model can be found in the third and fourth individual reports.

Link of the notebook

https://drive.google.com/file/d/1gEQtU6j6KTOoDyRgIs1SjBKgEA-W2GVl/view?usp=sharing

Final Results

The metric mostly used in the MRnet paper was **AUC**, however there was only a reference to the accuracy achieved in the abnormal case

Model	regressor	acc	loss	val_acc	val_loss	test_acc	test_loss
MRnet paper model	Abnormal	/	/	/	/	0.85	/
	ACL	/	/	/	/	/	/
	Meniscal	/	/	/	/	/	/
VGG	Abnormal	0.8083	0.4796	0.8053	0.4261	0.7912	0.5356
	ACL	0.8122	0.4837	0.8460	0.4288	0.5500	0.6903
	Meniscal	0.6559	0.6199	0.6637	0.6091	0.5667	0.6982
RESNET	Abnormal	0.8083	0.4898	0.8053	0.4936	0.7917	0.5182
	ACL	0.8122	0.4836	0.8496	0.4212	0.5500	0.7051
	Meniscal	0.6470	0.6492	0.6637	0.6402	0.5667	0.7024
Inception V3	Abnormal	0.8083	0.4779	0.8053	0.4862	0.7917	0.5167
	ACL	0.8122	0.4837	0.8496	0.4295	0.5500	0.7071
	Meniscal	0.6470	0.6357	0.6637	0.6313	0.5667	0.6912
Transfer learning InceptionV3	Abnormal	0.8083	0.4608	0.8053	0.4618	0.7917	0.5504
	ACL	0.8151	0.4950	0.8584	0.4291	0.5417	0.7141
	Meniscal	0.6470	0.6293	0.6637	0.6270	0.5667	0.7
Transfer learning VGG16	Abnormal	0.8092	0.3756	0.8053	0.3952	0.8083	0.5338
	ACL	0.8122	0.4364	0.8496	0.4151	0.625	0.6529
	Meniscal	0.7424	0.5645	0.7434	0.5781	0.5917	0.6374
Transfer learning ResNet50	Abnormal	0.9086	0.2393	0.8496	0.3378	0.8333	0.3940
	ACL	0.8741	0.3093	0.8407	0.3776	0.6583	0.6274
	Meniscal	0.7974	0.4415	0.7522	0.5539	0.7083	0.5947

Model	regressor	acc	loss	val_acc	val_loss	test_acc	test_loss
First contribution (transfer learning DenseNet)	**Abnormal**	0.8751	0.2908	0.8319	0.3986	**0.8167**	**0.4418**
	ACL	0.8122	0.4707	0.8496	0.4051	**0.5500**	**0.6972**
	Meniscal	0.7965	0.4516	0.6991	0.5771	**0.6417**	**0.6453**
Second contribution	**Abnormal**	0.8083	0.4695	0.8053	0.4833	**0.7917**	**0.5096**
	ACL	0.8122	0.4858	0.8496	0.4306	**0.5500**	**0.8062**
	Meniscal	0.6450	0.6322	0.6637	0.6286	**0.5667**	**0.6925**

Ranking models according to test results

This is a ranking according to our approach for all the models that we tried out to solve this problem for worst to best according to final test accuracy and loss:

1. Using VGG16, Inception V3 and ResNet50 made from scratch as well as using Inception V3 with transfer learning and using the model of the second contribution gave more or less the same results. They had the same accuracies except for Inception V3 with transfer learning and VGG16 giving lower accuracies for testing of ACL and abnormality respectively. In terms of loss, they had near results with inception V3 giving slightly better loss values.
2. Using transfer learning with VGG16.
3. Using transfer learning with DensNet (although it gave worse results in the ACL identification that VGG16 with transfer learning, it was better in both the cases of abnormality and Meniscal tear identifications).
4. The best case was using ResNet50 with transfer learning, it achieved highest accuracies and lowest loss in all the classification cases.

Conclusion and further work

We presented our approach and results for implementing the MRnet paper and provided our results. First we tried using well known models made from scratch and tuned to work for our problem. Then we tried using transfer learning with these models. Finally, we implemented and tested models used in other similar medical classification problems. We found that the best results were obtained when using transfer learning on the ResNet50 model. This may imply that the use of residual blocks and skip connections used in the ResNet architecture helps in getting better outcomes in medical classification problems. This goes with the results of the "***Deep Learning for Musculoskeletal Image Analysis***" paper. Further work may be needed to see if further training epochs of the model may give better results or not (we used early stopping and so in some cases the model stopped training so early). Also, other training approaches that we didn't try out may still give better results than our approach as using custom layers or other techniques to train the feature extractor and classifier at the same time instead of splitting them as we did in our approach.

Problems faced:

- The first problem was in combining the MRnet model. The problem was in the maximum pool part that made maximum pooling on the s dimension which is not the channel size. To solve this problem we followed the approach stated in the report.
- The second problem was in the normalization of the data. We implemented a function that normalizes the input data for training by dividing each pixel by 255. Unfortunately, we discovered after completing the training that this function had no effect due to a mistake where we didn't store the results of the normalization in the training input. When trying to solve the problem, memory issues occurred due to increasing in the pixel size after normalization. Therefore, we removed the function calls as it is meaningless (however, the function implementation is still present), and our training is considered to be without data normalization. However, we tried to train extractors with the correct normalization added and recorded the results.

Link for notebook that we used for trying to train the extractors after adding the normalization to our dataset :

https://colab.research.google.com/drive/1rNkodCL4h0Gef32bCGGOl9oNwFNjxYrO?usp=sharing

Roles:

Sherif:

- Summarizing the MRnet paper.
- Dealing with the data set.
- Building the functions used in training and testing the model.
- Performing transfer learning using architectures based on both inceptionV3 and VGG16.
- Making the feature extractor code for one of the two models used in the contribution part (the one based on the ***"Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases"*** paper).

Abdelrahman:

- Searching for RESNET50 resources.
- Implementing the RESNET50 Model using identical and con residual blocks.
- Training and tuning the extractor , classifier and regressor of the 3 RESNET models.

Ahmed:

- Implementing the Inception-v3 Network.
- Helping in training and tuning the extractor and classifier for the Inception.
- Implementing the Model used in the paper mentioned in the contribution part (*"Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases"* paper).

Hazem:

- Implementing the VGG16 model from scratch.
- Training of extractors , classifiers and regressors for the VGG.
- Train ResNet model using Transfer learning (train classifiers and regressors).
- Implementing the suggestion proposed by the first paper in contribution "***Deep Learning for Musculoskeletal Image Analysis***" by using the DenseNet model in transfer learning (training classifiers and regressors).

Jamal:

- Modifying the implementation of the Inception-v3 Network.
- Searching for Inception-v3 resources.
- Detecting bugs and solving training issues in the Inception-v3 Network.
- Training and tuning the extractor, classifier and regressor for the Inception.
- Training and solving overfitting of the Model used in the paper mentioned in the contribution part (*"Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases"* paper).

References

- https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002699

  Link for the MRnet paper.

- https://www.ahmedbesbes.com/blog/acl-tear-detection-part-1

  This blog was helpful in understanding the data set more.

- http://www.image-net.org/

  Site of the image net dataset

- "Deep Learning for Musculoskeletal Image Analysis " and "Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases".

  These two papers were used in our contribution part.

- https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF

  Playlist on youtube for CNN concepts and different types

- https://www.udemy.com/course/cnn-for-computer-vision-with-keras-and-tensorflow-in-python/

  Course for CNN implementations by keras using google collabs

- https://youtu.be/wqkc-sj5H94

  Video on youtube to illustrate how to implement the RESNET

- https://neurohive.io/en/popular-networks/resnet/

  Link for illustrating RESNETs

- TF and KERAS documentations
- Deep Learning in the Trenches: Understanding Inception Network from Scratch
  Link about Inceptio-v1 paper helped to get familiar with the code and the inception module.

- [Rethinking the Inception Architecture for Computer Vision]
  The paper that illustrates Inception-v2 and Inception-v3.