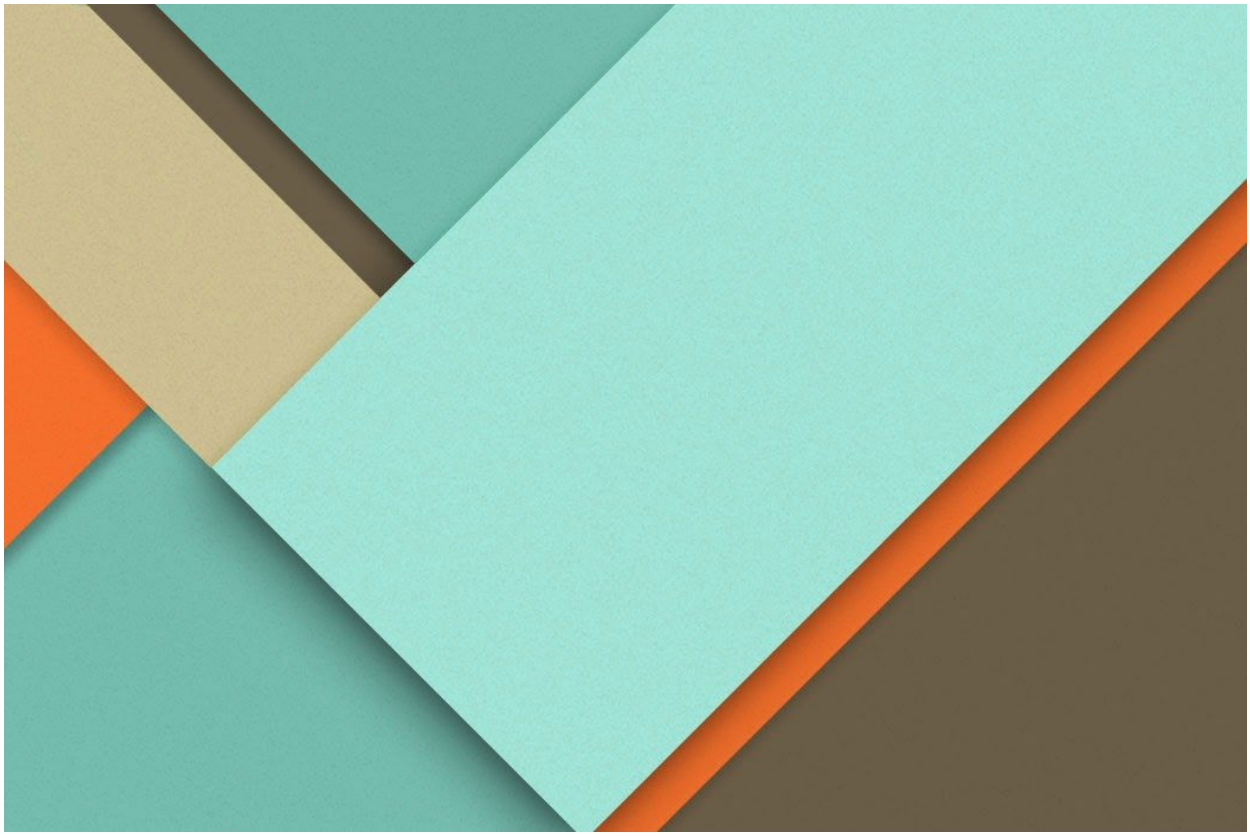


MRNet Model

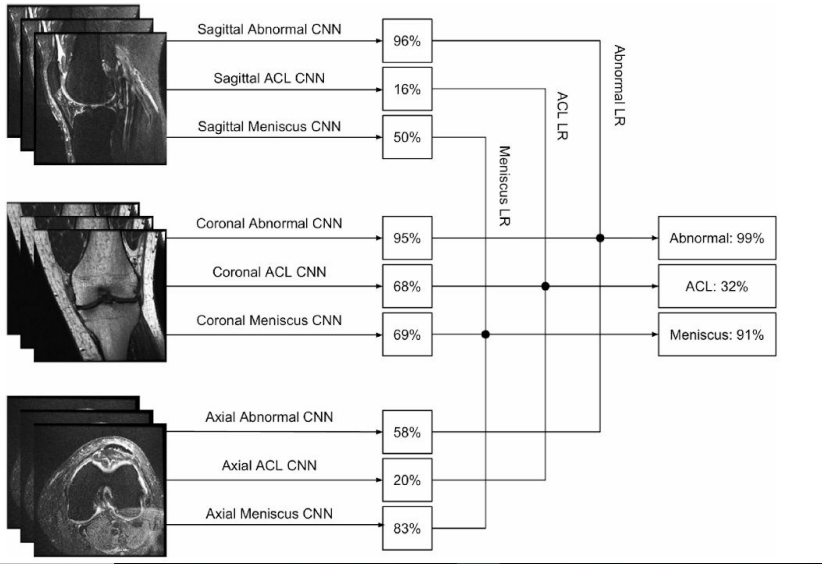
Eman Rafik Abd el-Kader ID:13

Introduction

Deep learning models have great impact in the medical field. One of the well known problems is interpreting Magnetic Resonance Imaging (MRI) using a fully automated system. Due to the special nature of MRI exams, considering the multi-plane and multidimensions, traditional image analysis methods are not effective. However, deep learning models with convolutional neural networks, in the way they extract features and the ability to interpret images with high accuracy are the most effective.
MRNet is an automated deep learning model proposed by Stanford ML Group. The model is an assistant system that interprets MRI exams to prediction. The model can detect two knee injuries: anterior cruciate ligament(ACL) tears and meniscal tears, in addition to detecting general abnormalities.

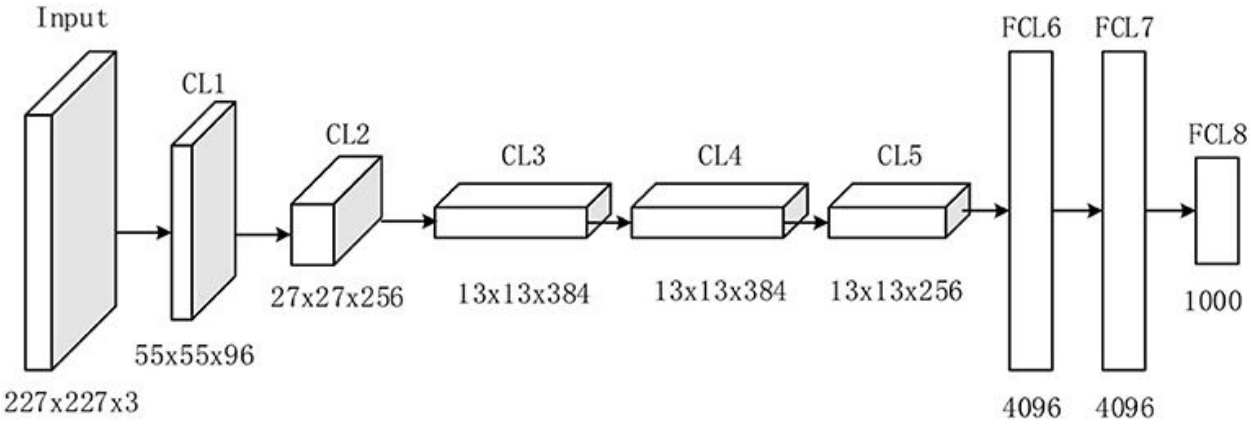
MRNet Model

The model, in its overview, is fed by an MRI exam and maps it to three predictions, one for each specified knee injury. As for details of the model, we have three similar networks, one for each injury. Each network is composed of three basic convolutional neural networks (CNN), these basic CNNs are the MRNet. Each MRNet is fed by a series of images in one plane, which are passed through a feature extractor based on AlexNet. The output is then passed through an average pooling layer, a max pooling layer and a fully connected layer with sigmoid activation function to give a prediction for the specified injury. For each injury there are three predictions one from each plane, these three predictions are combined through a fully connected layer to give one output prediction for a specific injury.



Feature Extractor

AlexNet Architecture



The AlexNet network is mainly composed of convolutional and max pooling layers that give feature maps of the input image. These feature maps are then fed to three dense layers followed by an output layer with 1000 possible outputs. The input layer of AlexNet accepts 227 x 227 x 3 images, the input images are then passed to a convolutional layer with 11x11 96 filters and stride of 4. The following layer is a 3x3 max pooling layer with stride of 2. The following convolutional layer has 256 filters with kernel size 5x5 and stride equals 1, then it is followed by a max pooling layer similar to the previous one. The network then has three consequence convolutional layers with kernel size 3x3, stride equals 1 and feature maps of sizes 384, 384 and 256, respectively. A max pooling layer with kernel size 3x3 and stride equals one follows these convolutional layers. There is a flatten layer with 9216 units. The flattened output is then passed through two dense layers of size 4096. Finally the output layer has 1000 units. All layers, except for the output layer, have Rectified Linear Unit activation function (ReLU). As for the output layer, it has Softmax activation.

The first part of AlexNet is used as a feature extractor in each MRNet block, where it is fed by an image of dimension 3x256x256 and gives an output 256x7x7.

Implementation

The proposed model was implemented in tensorflow keras.

1. The basic MRNet block

The input of MRNet is of dimension sx256x256x3, where s represents the number of slices in a series, and 256x256x3 is the RGB image fed to the feature extractor. The input image is cropped from the middle to match the AlexNet input. The following layers follow the AlexNet architecture. The AlexNet outputs a feature map of dimension 7x7x256 which is passed through an average pooling layer that gives a 256-vector for each slice (sx256), then max pooling is applied to the whole series to give a 256-vector that is passed through a dense layer with one output unit to give a prediction.

```
def mrnet_model():
  alexnet_input = keras.Input(shape=(None,256, 256, 3))
  x = layers.Cropping3D(cropping=((0,0),(15,14),(15,14)))(alexnet_input)
  x = layers.Conv3D(filters=96, kernel_size=(1,11,11), strides=(1,4,4), padding='valid', activation='relu')(x)
  x = layers.MaxPooling3D(pool_size=(1,3,3), strides=(1,2,2), padding='valid')(x)
  x = layers.Conv3D(filters=256, kernel_size=(1,5,5), padding='valid', activation='relu')(x)
  x = layers.MaxPooling3D(pool_size=(1,3,3), strides=(1,2,2), padding='valid')(x)
  x = layers.Conv3D(filters=384, kernel_size=(1,3,3), padding='valid', activation='relu')(x)
  alexnet_output = layers.Conv3D(filters=256, kernel_size=(1,3,3), padding='valid', activation='relu')(x)
  alexnet = keras.Model(inputs=alexnet_input, outputs=alexnet_output, name='alexnet')
  x = layers.AveragePooling3D(strides=(1,7,7))(alexnet_output)
  x = layers.GlobalMaxPooling3D()(x)
  pooling_output = layers.Dense(1, activation='sigmoid')(x)
  mrnet = keras.Model(inputs=alexnet_input, outputs=pooling_output)
  mrnet.compile(optimizer='sgd', loss='binary_crossentropy', metrics=["accuracy"])
  return mrnet
```

2. Predictor

The predictor is a simple logistic regression model with three inputs of dimension 1x1 and one output. It combines the three predictions of the three views.

```
def predictor():
  logistic_input = keras.Input(shape=(3,))
  logistic_output = layers.Dense(1, activation='sigmoid')(logistic_input)

  predictor_model = keras.Model(inputs=logistic_input, outputs=logistic_output)
  predictor_model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
  return predictor_model
```

3. Training

Nine MRNets are trained, three for each injury. The models were trained separately and independent of each other.

4. Combining

Considering one injury, the three pre-trained models for: sagittal, axial and coronal views predict the whole training set. The predictions are concatenated in nx3 tensor to be fed to the logistic regression model as training input, where n is the training data size. Same was done with the validation data.

```
# combining models
training_abnormal_sagittal_generator = data_generator('abnormal', 'sagittal','train')
validation_abnormal_sagittal_generator = data_generator('abnormal', 'sagittal','valid')

training_abnormal_coronal_generator = data_generator('abnormal', 'coronal','train')
validation_abnormal_coronal_generator = data_generator('abnormal', 'coronal','valid')

training_abnormal_axial_generator = data_generator('abnormal', 'axial','train')
validation_abnormal_axial_generator = data_generator('abnormal', 'axial','valid')

training_sagittal_prediction = abnormal_sagittal.predict_generator(training_abnormal_sagittal_generator)
training_coronal_prediction = abnormal_coronal.predict_generator(training_abnormal_coronal_generator)
training_axial_prediction = abnormal_axial.predict_generator(training_abnormal_axial_generator)

validation_sagittal_prediction = abnormal_sagittal.predict_generator(validation_abnormal_sagittal_generator)
validation_coronal_prediction = abnormal_coronal.predict_generator(validation_abnormal_coronal_generator)
validation_axial_prediction = abnormal_axial.predict_generator(validation_abnormal_axial_generator)

training_input = tf.concat([training_sagittal_prediction, training_coronal_prediction, training_axial_prediction],1)
training_output = tf.convert_to_tensor(get_output('abnormal','train'))
validation_input = tf.concat([validation_sagittal_prediction, validation_coronal_prediction, validation_axial_prediction],1)
validation_output = tf.convert_to_tensor(get_output('abnormal','valid'))

abnormal_model = predictor()
abnormal_model_hisory = abnormal_model.fit(training_input, training_output,epochs=30,validation_data=(validation_input,validation_output),

abnormal_model.save('combine_mrnet_abnormal.h5')
```

Results

The built model was tested against 120 exams of MRNet data and achieved accuracies of: 0.792, 0.6917 and 0.617 for abnormalities, ACL tears and meniscus tears, respectively.

Conclusion

We re-built the MRNet model proposed by Stanford ML Group to interpret MRI exams to predictions. We used keras models to build the basic blocks, build a full network for predicting each injury. The networks were built and trained from scratch, however in the proposed model a transfer learning approach was applied, where the AlexNet weights were initialized to the ImageNet database weights. Transfer learning approach would improve the performance to reach higher accuracies.

References

- https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002699
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- https://engmrk.com/alexnet-implementation-using-keras/
- https://www.learnopencv.com/understanding-alexnet/