A Deep Learning Model for Scoliosis and Spondylolisthesis Detection

1st Omer Sabri Emeksiz
Electrical Electronics Engineering
Marmara University
Istanbul, Turkey
omeremeksiz@marmara.edu.tr

I. Introduction

The objective of this project is to develop a deep learning model that detects scoliosis and spondylolisthesis from x-ray images. This will be achieved by leveraging the pre-trained networks included in the referenced article. The evaluation of the model's performance will be conducted using various metrics, including F1 score, precision, recall, specificity, and examination of training, validation, and test curves.

II. System Model

In the project 2 model implementation was made. The first of these implementations was done with GoogLeNet and the second one without using any pre-trained network. The reason why different implementations were tried in the first place was that despite the different approaches, satisfactory results could not be achieved because the overfitting problem could not be solved in the implementations using pre-trained models.

A. GoogLeNet Model

The first model is a three-class image classification model using transfer learning with InceptionV3. The dataset is split into training, validation, and test sets, and data augmentation is applied to the training images for improved model generalization. The InceptionV3 model, originally trained on a diverse dataset, has its final classification layer replaced with a custom classifier suitable for the target classes (Normal, Scol, Spond). This involves integrating a global average pooling layer, a dense layer with ReLU activation and L2 regularization, and a dropout layer to prevent overfitting. The model is compiled with the Adam optimizer and categorical crossentropy loss. Training occurs for specific number of epochs, and the model is evaluated on the test set, providing classification metrics such as accuracy, confusion matrix, and specificity for each class. The training history is visualized to assess model performance over epochs.

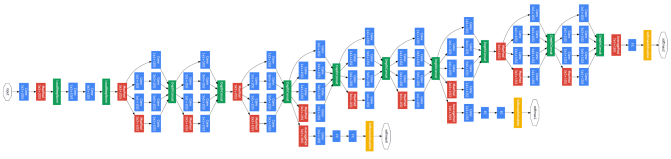

Fig. 1. The architecture of GoogLeNet

GoogleNet, also known as Inception V1, features a 22-layer deep architecture designed for image classification, with a focus on computational efficiency for deployment on a variety of devices, including those with limited resources (see Figure 1). The architecture's notable aspects include the use of inception modules, incorporating parallel convolutional and pooling operations to capture multi-scale features. Two auxiliary classifier layers are strategically connected to Inception (4a) and Inception (4d), each consisting of an average pooling layer (5x5, stride 3), followed by a 1x1 convolutional layer (128 filters, ReLU activation), a fully connected layer (1025 outputs, ReLU activation), dropout regularization (dropout ratio 0.7), and a softmax classifier with 1000 classes, mirroring the main softmax classifier. These auxiliary classifiers, as illustrated in Figure 1, play a crucial role in addressing the vanishing gradient problem during training, providing additional supervision signals while ensuring computational efficiency. Additional layer details, including the number of parameters, are presented in Figure 1, offering a comprehensive view of the network's structure and parameter distribution. GoogleNet, in total, comprises 18,278,353 parameters.

B. CustomNet Model

The third model is a Convolutional Neural Network (CNN) designed for a three-class image classification task. The architecture is built using the Sequential model from the Keras library. The dataset is organized into training and validation sets, and data augmentation techniques are applied during training to enhance the model's ability to generalize to various input variations.

The CNN architecture consists of three convolutional layers, each followed by a max-pooling layer to extract hierarchical features from the input images. The convolutional layers use Rectified Linear Unit (ReLU) activation functions, promoting non-linearity in the learned representations. The final architecture incorporates a flattening layer to transform the convolutional outputs into a one-dimensional array, followed by two dense (fully connected) layers. The first dense layer has 128 neurons with ReLU activation, facilitating feature aggregation, and the second dense layer serves as the output layer with three neurons and softmax activation corresponding to the three classes: Normal, Scoliosis (Scol), and Spondylolisthesis (Spond).

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fig. 2. The architecture of GoogLeNet

The model is compiled using the Adam optimizer, categorical crossentropy loss function. Key callbacks include a dynamic learning rate scheduler and early stopping to prevent overfitting by halting training on validation loss plateaus. Classification metrics such as accuracy, confusion matrix, and specificity are calculated and displayed. Training and validation curves are plotted to visualize the model's learning progress over epochs.

III. Numerical Results

CustomNet was developed because the overfitting problem could not be solved in the model, which was initially developed using GoogLeNet. However, after normalization was applied to the data set, the overfitting problem was solved. As can be seen in I, GoogLeNet managed to surpass CustomNet in almost all metrics.

TABLE I
Performance Metrics of Deep Learning Models

Model	F1 Score	Precision	Recall	Specificity	Accuracy
GoogLeNet	%98	%97	%96	%92	%97
CustomNet	%89	%88	%86	%81	%87

Simultaneously, a comparison between Figure 3 and 4 reveals that the model constructed with GoogLeNet attains superior accuracy and exhibits a more robust graphical curve. Upon reviewing the graphs, it becomes apparent that the curves converge, indicating the absence of overfitting.

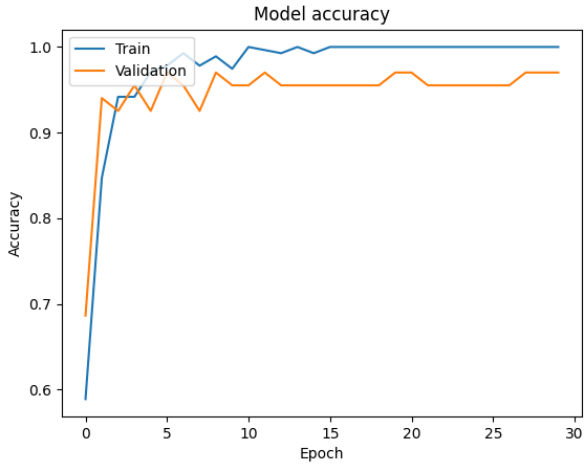

Fig. 3. The accuracy plot of GoogLeNet

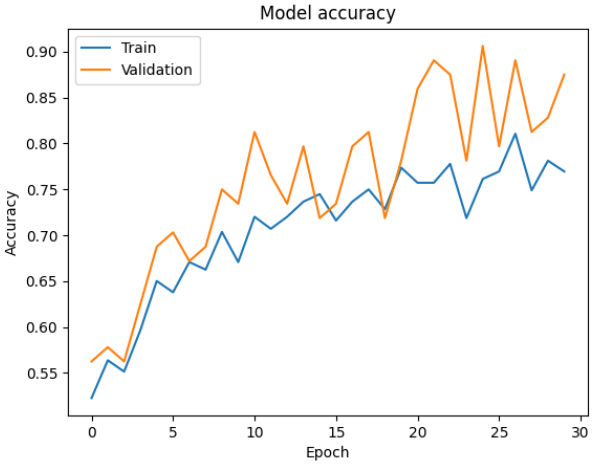

Fig. 4. The accuracy plot of CustomNet

When analyzing the loss graphs depicted in Figure 5 and 6, a notable observation is the convergence of both curves. The fact that the training and validation loss curves exhibit a decreasing trend and eventually stabilize suggests that the model is learning effectively. This convergence is a positive

indication that overfitting has been mitigated.Simultaneously, within these plots, the graph for GoogLeNet demonstrates greater resilience and exhibits superiority compared to CustomNet.

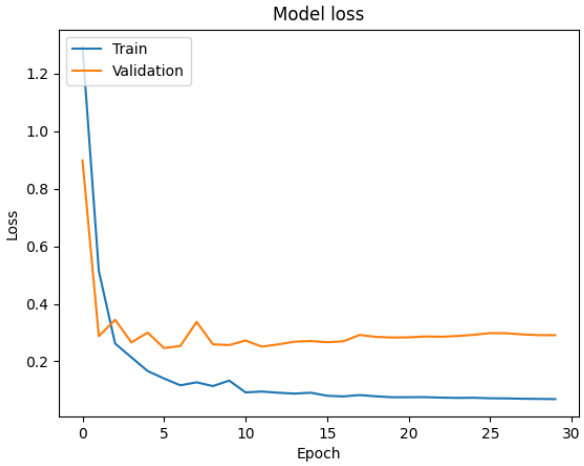

Fig. 5. The loss plot of GoogLeNet

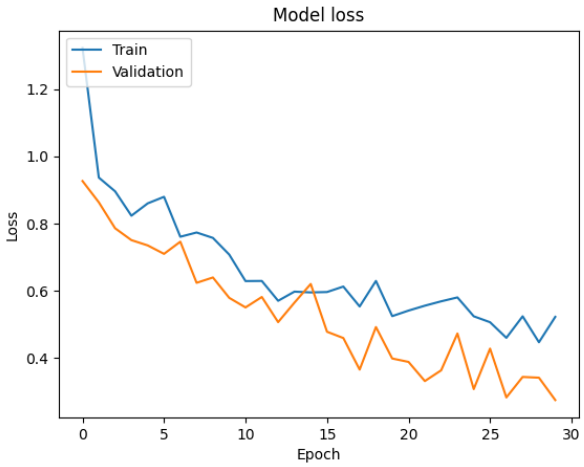

Fig. 6. The loss plot of CustomNet

Comparison of the confusion matrices in Figure 7 and 8 reveals that GoogLeNet exhibits only 2 incorrect predictions, whereas CustomNet has 15. This observation highlights the superior performance of GoogLeNet in this context.

From the analyses presented above, it is evident that the GoogleNet model, developed with fixed parameters, demonstrates superiority and greater success across various domains compared to the CustomNet model. Furthermore, GoogLeNet ensures the objectivity of the study in its entirety. In addition, although the applied data augmentation techniques made CustomNet more successful, they did not raise it to the level of GoogLeNet. Enhancing CustomNet could involve increasing the complexity level, implementing fine-tuning, and applying diverse normalization techniques to the dataset.

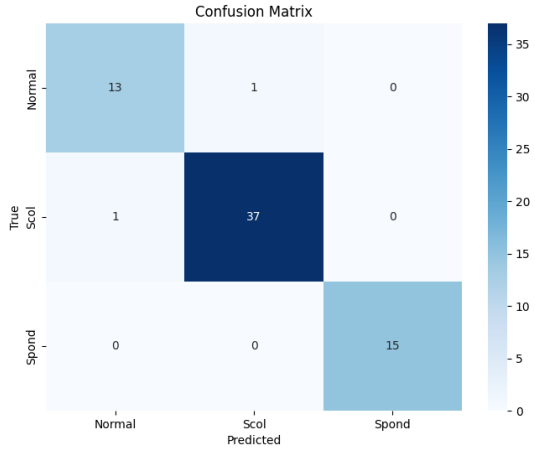

Fig. 7. The confusion matrix of GoogLeNet

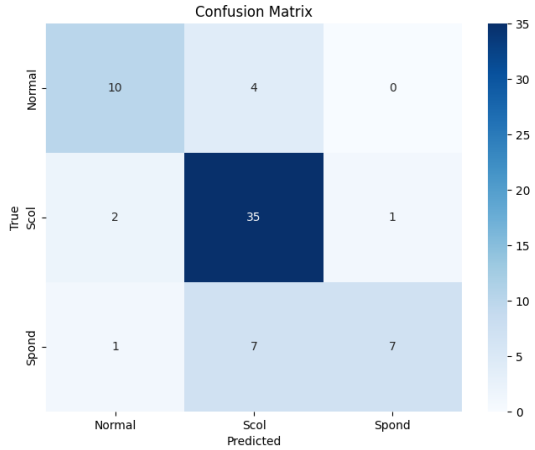

Fig. 8. The confusion matrix of CustomNet

IV. Conclusion

In the project, two different models were developed: using pretrained GoogLeNet and CustomNet without using the pretrained model. Different architectures were examined and implemented in the project. The results of the models were evaluated through different metrics. Different data processing was used and benefits were obtained from these processes. Although both models were successful to a certain extent, GoogLeNet showed better performance in all metrics and was by far the better model.