Deep Learning Models for Wireless Capsule Endoscopy Images

An essay submitted in partial fulfillment of

the requirements for graduation from the

Honors College at the College of Charleston

with a Bachelor of Science and Bachelor of Arts in

Data Science and French and Francophone Studies

Jane Shelby Porter

May 2023

Advisor: Dr. Mukesh Kumar

Deep Learning Models for Wireless Capsule Endoscopy Images

Jane Shelby Porter
Department of Mathematics
College of Charleston
Charleston, South Carolina, United States
porterjs@g.cofc.edu

Abstract

Wireless capsule endoscopy (WCE) has become an irreplaceable tool for diagnostic inspection of the Gastrointestinal (GI) tract. Accurate recognition of polyps in WCE images is a difficult task due to the complicated characteristics. Therefore, an automatic computer-aided diagnosis system is crucial to assist physicians to analyze and separate polyp images. We aim to develop an automated system for polyp detection in WCE images based on deep learning models which is an improvement to the neural network that contains more computational layers that allow for higher levels of abstraction and prediction in the data. We first present pre-trained VGG-16 model with the help of transfer learning approach for image classification problems in WCE images. We presented a comparison of four different methods to perform transfer learning using VGG-16 model. We have shown that transfer learning framework will be beneficial as it saves time of training the CNN models from scratch and can be vital in the field of medicine in the future. Also, we presented how VGG-16 could be used as a feature extraction tool to improve the performance of classical classifiers such as Random Forest and Support Vector Machine. At the end, we included a deep feature learning method, named stacked sparse autoencoder, to recognize polyps in the WCE images.

I. Introduction

Colorectal cancer is a one of the most common types of cancer in the United States. An important precursor to this cancer is colorectal polyps, which can be detected and removed early to prevent deterioration into cancer cells. Wireless capsule endoscopy (WCE) has become the key to visualizing and exploring the gastrointestinal (GI) tract to capture images of these polyps. A capsule enters through the mouth of the human body, and once it reaches the GI tract, it captures approximately 8 hours of footage, or 55,000 color images. The device transmits these images to a device on the patient's waist. Doctors then download these images and traditionally, they identify polyps through manual examination of the images. However, polyps are difficult to identify due to variations in size, texture, and other features. Also, polyps typically make up about 5% of the images collected by the WCE, so they are time-consuming to locate and analyze [1]. This project's goal is to create an automated system using deep learning that can accurately identify abnormal images produced by the WCE.

Before deep learning, neural networks were made up of one level and a classifier. This level was a feature extractor that would identify key features from the raw data. These features would then be passed to the classifier, so it could detect patterns or classify the input. Deep learning contains multiple levels of neural networks, so it has multiple levels of representation learning. These added levels increase the level of abstraction, allowing key features to be easier to identify and decreasing the nonrelevant features. These features are not chosen by the user; instead, they are learned and identified by the program. Deep learning allows us to solve more challenging problems by taking advantage of the extra layers [2].

Convolutional neural networks (CNN's) are designed to process image data. The layers are typically structed into stages of convolutional and pooling layers. The convolutional layers identify features, and the pooling layers simplify these features, merging similar features together. This architecture is based on the visualization of cells in neuroscience, and it was rarely used until the 2012 ImageNet competition.

After CNNs massively improved the error rate, they became the dominant solution for image classification and detection tasks [2]. An example of deep convolutional neural network for wireless capsule endoscopy images is shown in Figure 1.

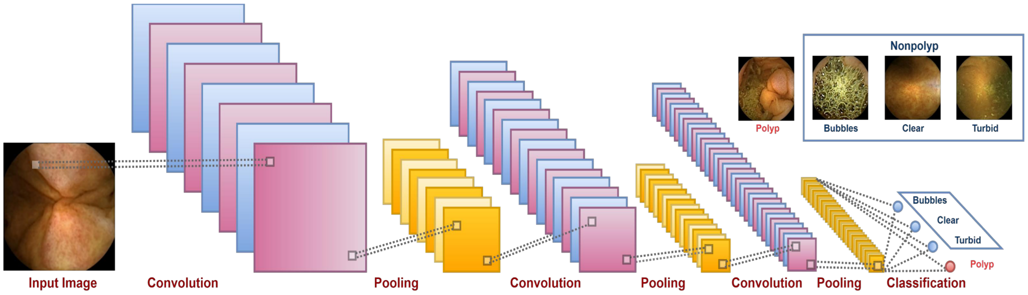


Fig. 1. Deep-CNN network for WCE image classification.

Quin et al. [3] is a survey paper of CNNs built for wireless endoscopy. Through this paper, they reviewed current research and the best models for this problem. They found that CNNs have the potential to become useful in WCE diagnosis in the future. The 23 independent studies completed studied erosion/ulcer, GI bleeding and polyps in WCE images. Each of these projects analyzed had different frameworks, but this survey shows that most of them had an accuracy of at least 90%, which is similar to an endoscopist's accuracy. So, CNNs are an effective tool for WCE classification. Our goal in this project is to determine the most effective CNN to classify WCE images.

II. Data Set

The dataset for this project contains 400 WCE images split into 200 polyp images and 200 non-polyp images. We divided the dataset into a training set of 300 images and a validation set of 100 images. The Department of Gastroenterology in the University Hospital of Coimbra (Portugal) used the PillCam SB to collect this data [4]. Each image in this dataset has a resolution of 256 x 256 pixels. Fig. 2 shows two images from the training set.

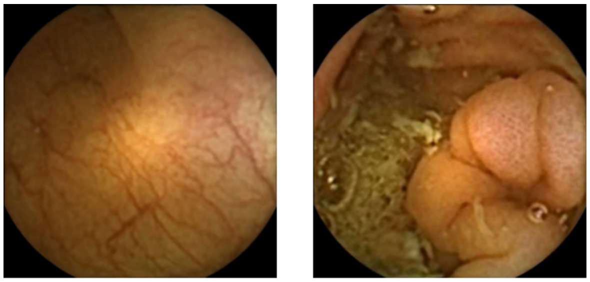


Fig. 2. WCE images: Normal (left) vs. Polyp (right).

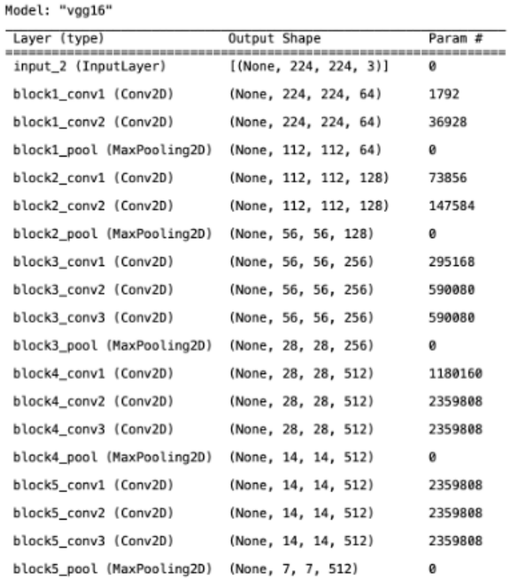
III. Methodology

A. Transfer Learning

Transfer learning is a technique where a model trained for one task is applied to a new task. Transfer learning is especially useful for applications where there are limited amounts of training samples or a large of number of parameters because it requires little retraining of a model. The pre-trained network already has established weights that can be updated accordingly through fine tuning, or the freezing certain layers of the model and retraining the rest of the model based on the new dataset [5]. The three most popular convolutional neural networks are ResNet50, VGG-16, and AlexNet.

B. VGG-16 Model

In 2014, K. Simonyan and A. Zimmerman proposed the VGG16 model as a solution to the ImageNet 2014 Challenge. This model's input is a 224 x 224 RGB image, which is then passed through five blocks of two or three convolutional layers and a max-pooling layer with a 2 x 2-pixel window and stride 2 [6]. In total, the VGG16 has 19 layers, and 14,714,688 parameters. Fig. 3 contains more information about the layers of the VGG-16 model.



```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
```

Fig. 3. Original VGG-16 architecture.

For this project, before fine tuning the model, we added four new layers at the end. The first three layers are a 2D max pooling layer, a fully-connected layer with 512 nodes and a 0.5 dropout layer. The final layer, a fully connected sigmoid layer with 2 nodes serves as the classifier. Fig. 4 shows this framework.

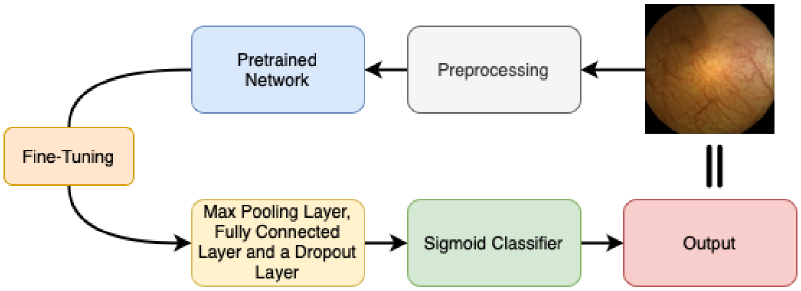


Fig. 4. Transfer learning framework.

We then used four methods to evaluate the performance of this model.

- Method 1: The first method uses the pre-trained weights from the VGG-16 and only trains the added layers and the classifier.
- Method 2: The second method uses half of the pre-trained weights in the VGG-16 model by retraining its last four pre-trained layers.
- Method 3: The third method uses only the weights from the first convolutional block of each network and retrains the remaining layers.

- Method 4: The final method uses only the weights from the input layer and retrains the rest of the network.

C. *Support Vector Machines (SVMs)*

We also studied if the VGG-16 model could improve the performance of classical methods such as SVM and Random-Forest Classifiers. Support vector machines (SVMs) are supervised learning methods used for regression, classification, and anomaly detection and they are highly effective in high-dimensional spaces. The objective of SVM is to classify the data by finding a hyperplane in an N-dimensional space where N is the number of features. This hyperplane separates the data into distinct classes by maximizing the distance between the classes and minimizing mislabeled instances [7].

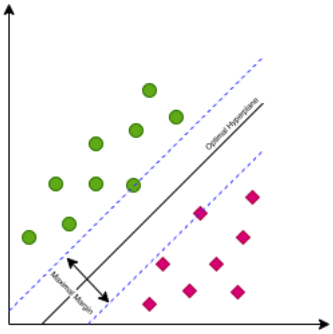


Fig. 5. Support vector machine diagram.

D. *Random Forest Classifiers*

Random Forest is an ensemble classification algorithm that creates many different decision trees. Each decision tree produces a class prediction, and the class is chosen by majority vote. Random Forest Classifiers have similar error to other ensemble techniques like bagging and boosting [8]. Fig. 6 is a diagram of how Random Forest decision trees work.

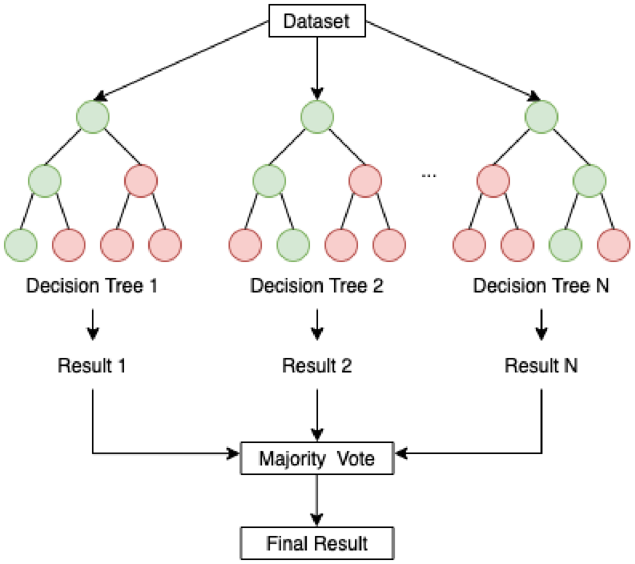


Fig. 6. Random forest diagram.

E. Autoencoders

Autoencoders use an encoder-decoder architecture. The encoder represents the inputs at lower dimensions, and the decoder reconstructs the images using the features extracted by the encoder. The encoder can be used for classification. A simple autoencoder contains one encoder and one decoder while a stacked autoencoder contains multiple encoders and decoders like in Fig. 7. In a stacked autoencoder, the outputs of each encoder or decoder layer are passed as the inputs to the next layer [9]. A sparse autoencoder is an autoencoder trained with additional weight decay and sparsity terms in the loss function.

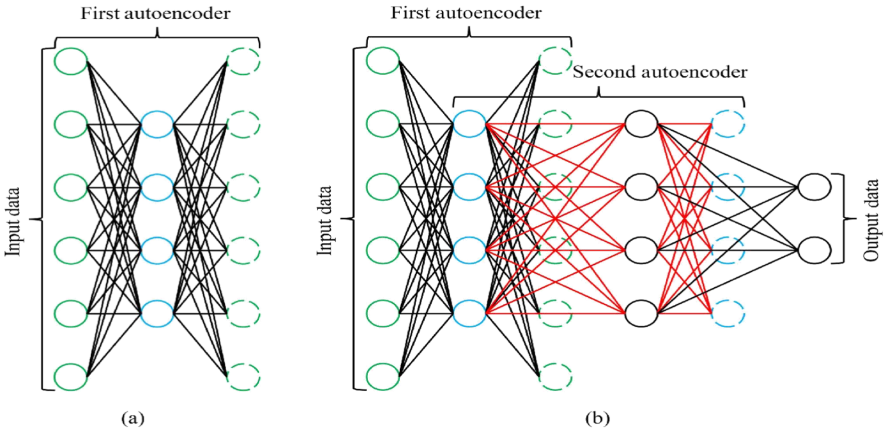


Fig. 7. (a) Autoencoder and (b) Stacked autoencoder..

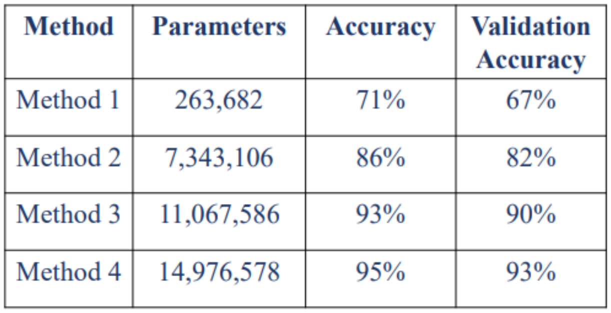
F. Data Augmentation

CNN models perform very well for image classification, but they require a large amount of data for training to reduce overfitting. Overfitting happens when a deep learning model learns a function with high variance that predicts the training data well, but it does not generalize to other data sets. For many image classification tasks, especially medical analysis, it is difficult to obtain a large enough data set. Therefore, researchers use data augmentation techniques to supplement the initial data set. These techniques use the training data set to create new dummy data through translations and shifts. The images are resized 1/255, the rotation range is set at 45, the width and height shifts are set at 0.3, horizontal flip is set to true, and fill mode is set to nearest [5]. The model is then trained using the generated data, so each epoch it receives 300 new training images and 100 new validation images based on the original images.

IV. Results

A. VGG-16 Model Fine Tuning

In this work, we used the Keras implementation of the VGG-16 model pre-trained on the ImageNet data set. When we fine tuned the model, we used an image generator to pass in the augmented data, and we used binary cross entropy as the loss. We also used the stochastic gradient descent (SGD) optimizer with a learning rate of 0.001. For each fine tuning method, we retrained the VGG-16 model for 70 epochs. After retraining the VGG-16 model, we compared the results as shown in Figure 8. Method 1 had 263,682 parameters, and after fine tuning, it had a training accuracy of 71% and a validation accuracy of 67%. Method 2 had 7,343,106 parameters with a training accuracy of 86% and a validation accuracy of 82%. Method 3 had 11,067,586 parameters with a training accuracy of 93% and a validation accuracy of 90%. The final method, where all the parameters were retrained, had 14,976,578 parameters with a training accuracy of 95% and a validation accuracy of 93%. Figs. 9-11 show graphs of the training accuracy, validation accuracy, and the loss for these methods as the model was trained.



Method	Parameters	Accuracy	Validation Accuracy
Method 1	263,682	71%	67%
Method 2	7,343,106	86%	82%
Method 3	11,067,586	93%	90%
Method 4	14,976,578	95%	93%

Fig. 8. Comparison of transfer learning models.

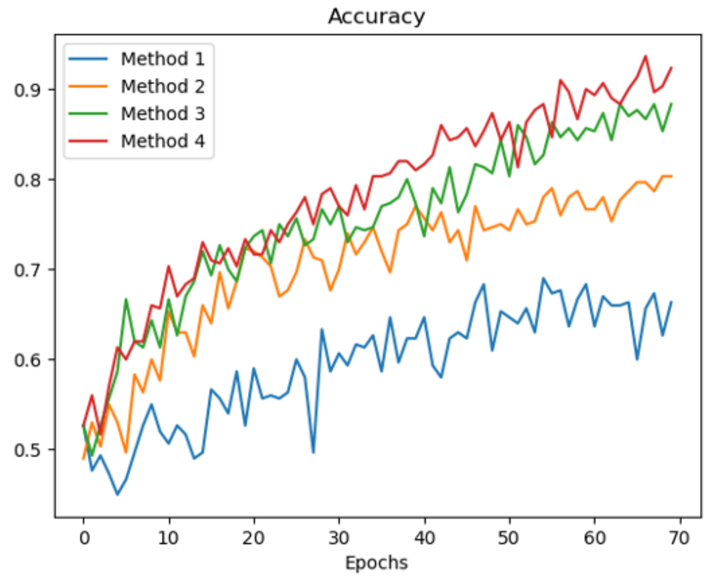


Fig. 9. Graph of training accuracy.

Based on these metrics, Method 4, where the entire model was retrained, consistently outperformed the other methods. Figs. 9and 10 show that Methods 3 and 4 have similar training rates. Method 2's accuracy began close to Methods 3 and 4, but it plateaued overtime while Method 1 consistently under-performed when compared to the other methods.

Therefore, Method 4 has the best performance, but it also requires the most computational power because the entire model must be retrained. Method 3 appears to be the best choice for fine tuning because the performance decreases minimally with the decrease in parameters. It has similar results to Method 4, but it has 3,908,992 less parameters.

B. SVM and Random Forest Classifiers

We also evaluated the performance of SVM and Random Forest Classifiers with the addition of the VGG-16 model. To do this, we trained the VGG-16 model using Method 3 for 70 epochs, and then removed the classifier and the added layers. Instead of adding a new classifier, we passed the data in the scikit-learn versions of SVM and Random Forest. Our Random Forest classifier had 100 random trees.

To evaluate this model, we used the image generator to generate 100 images. We passed these images into the simple SVM and Random Forest algorithms and the VGG-16 models to see which would perform

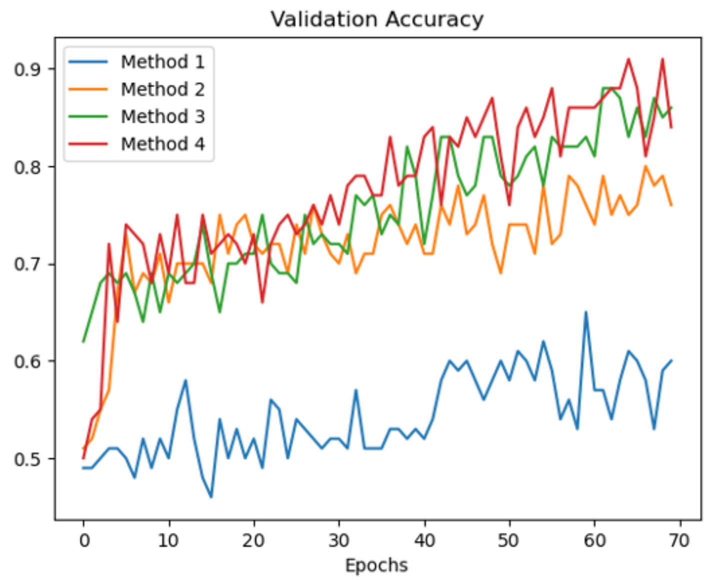


Fig. 10. Graph of validation accuracy.

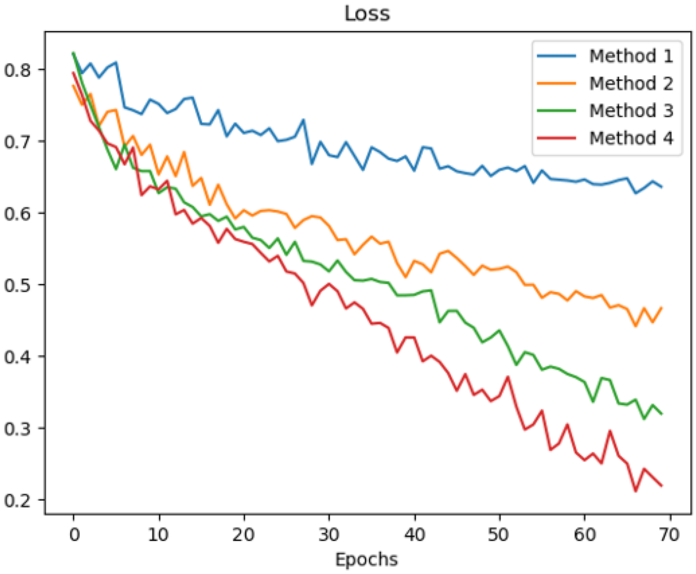
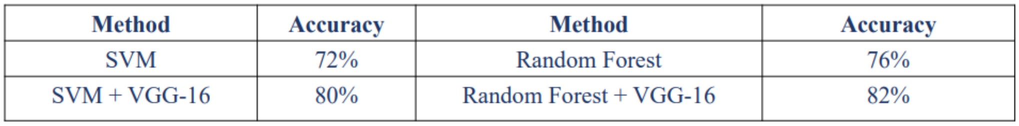


Fig. 11. Graph of loss.



Method	Accuracy	Method	Accuracy
SVM	72%	Random Forest	76%
SVM + VGG-16	80%	Random Forest + VGG-16	82%

Fig. 12. Comparison of SVM vs SVM+VGG-16 and RF vs RF+VGG-16 models.

the best. The simple SVM had an accuracy of 72%, and the VGG-16 with SVM had 80% accuracy. For Random Forest, the simple version had an accuracy of 76%, and the accuracy of the VGG-16 version was 82%. Both classifiers performed better with the VGG-16 model, so the classifiers predict better with the extracted features than the original images. Also, the Random Forest classifier had a higher accuracy than the SVM classifier. However, neither of these models performed as well as the sigmoid classifier used in Method 3.

C. *Autoencoder*

Finally, we also created a stacked sparse autoencoder to compare its performance to the VGG-16 model. Our model contains stacked autoencoders of size 3000, 600, and 64. For this project, we used mean squared error with L_2 and sparsity regularization for the loss function and set the weight decay control parameter α= 0.002, the sparsity penalty control β= 3, and the sparsity parameter ρ= 0.05 in the KL divergence term:

$$\mathcal{L}_{SAE} = \frac{1}{2}\sum_{i=1}^{N} ||x_i + \widehat{x_i}||^2 + \frac{\alpha}{2}(||W_1||^2 + ||W_2||^2) + \beta \sum_{j=1}^{K} KL\,(\rho||\widehat{\rho_j})$$

$$KL(\rho||\hat{\rho}) = \rho \log\left(\frac{\rho}{\widehat{\rho_j}}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\widehat{\rho_j}}\right)$$

Figure 13 shows the decay of above loss function during the training phase. We obtained an accuracy of 94.3% using the stacked SAE model.

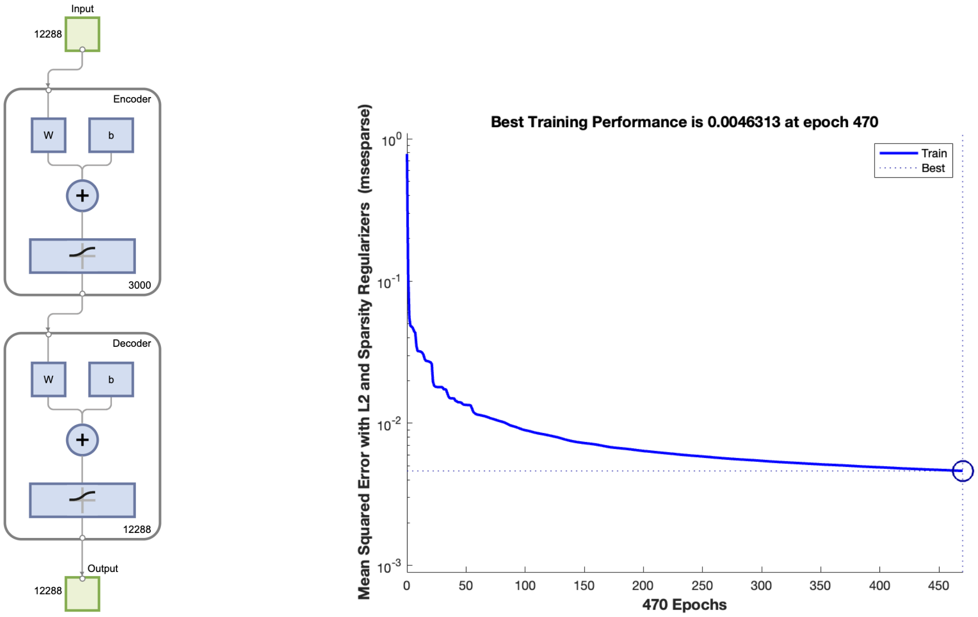


Fig. 13. Stacked auto-encoder and decay of loss function

V. Conclusions

We concluded the following findings in this project:

- In transfer learning through VGG-16, we found that Method 4, where the entire model was retrained, consistently outperformed the other methods.
- Method 4 has the best performance, but it also requires the most computational power because the entire model must be retrained. Method 3 appears to be the best choice for fine tuning because the performance decreases minimally with the decrease in parameters. It has similar results to Method 4, but it has 3,908,992 less parameters.
- Support vector machine and Random forest classifiers performed better with the VGG-16 model, so the classifiers predict better with the extracted features than the original images.
- Also, the Random forest classifier had a higher accuracy than the SVM classifier. However, neither of these models performed as well as the sigmoid classifier used in Method 3.
- We also observed an accuracy of 94.3% using the stacked SAE. This shows that features learned through transfer learning and stacked sparse autoencoders are similarly representative of the dataset.

In future, we would like to study the effects of different classifiers with deep CNNs in solving various other classification problems in medical image analysis. We also aim to consider other abnormalities presented in WCE images such as segmentation and bleeding detection problems.

References

[1] Y. LeCun, Y. Bengio, and G Hinton. ”Deep learning.” nature, 521.7553 pp. 436-444, 2015.

[2] Y. Yuan, and M. Q-H. Meng. ”Deep learning for polyp recognition in wireless capsule endoscopy images.” Medical physics, 44.4, 1379-1389, 2017.

[3] Qin, K., Li, J., Fang, Y. et al. Convolution neural network for the diagnosis of wireless capsule endoscopy: a systematic review and meta-analysis. Surg Endosc 36, 16–31 (2022). https://doi.org/10.1007/s00464-021-08689-3.

[4] M. Lanier and M. Kumar. “An Automated System for Polyp Detection in Wireless Capsule Endoscopy Images Based on Deep Learning.”

[5] J. Yogapriya, Venkatesan Chandran, M. G. Sumithra, P. Anitha, P. Jenopaul, C. Suresh Gnana Dhas, ”Gastrointestinal Tract Disease Classification from Wireless Endoscopy Images Using Pretrained Deep Learning Model”, Computational and Mathematical Methods in Medicine, vol. 2021, Article ID 5940433, 12 pages, 2021. https://doi.org/10.1155/2021/5940433.

[6] K. Simonyan and A. Zisserman. ”Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556, 2014.

[7] M. E. Mavroforakis and S. Theodoridis. ”A geometric approach to support vector machine (SVM) classification.” IEEE transactions on neural networks, 17.3, 671-682, 2006.

[8] V. Y. Kulkarni and P. K. Sinha. ”Pruning of random forest classifiers: A survey and future directions.” 2012 International Conference on Data Science ’I&’ Engineering (ICDSE). IEEE, 2012.

[9] J. Xu et al., ”Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images,” in IEEE Transactions on Medical Imaging, vol. 35, no. 1, pp. 119-130, Jan. 2016, doi: 10.1109/TMI.2015.2458702.