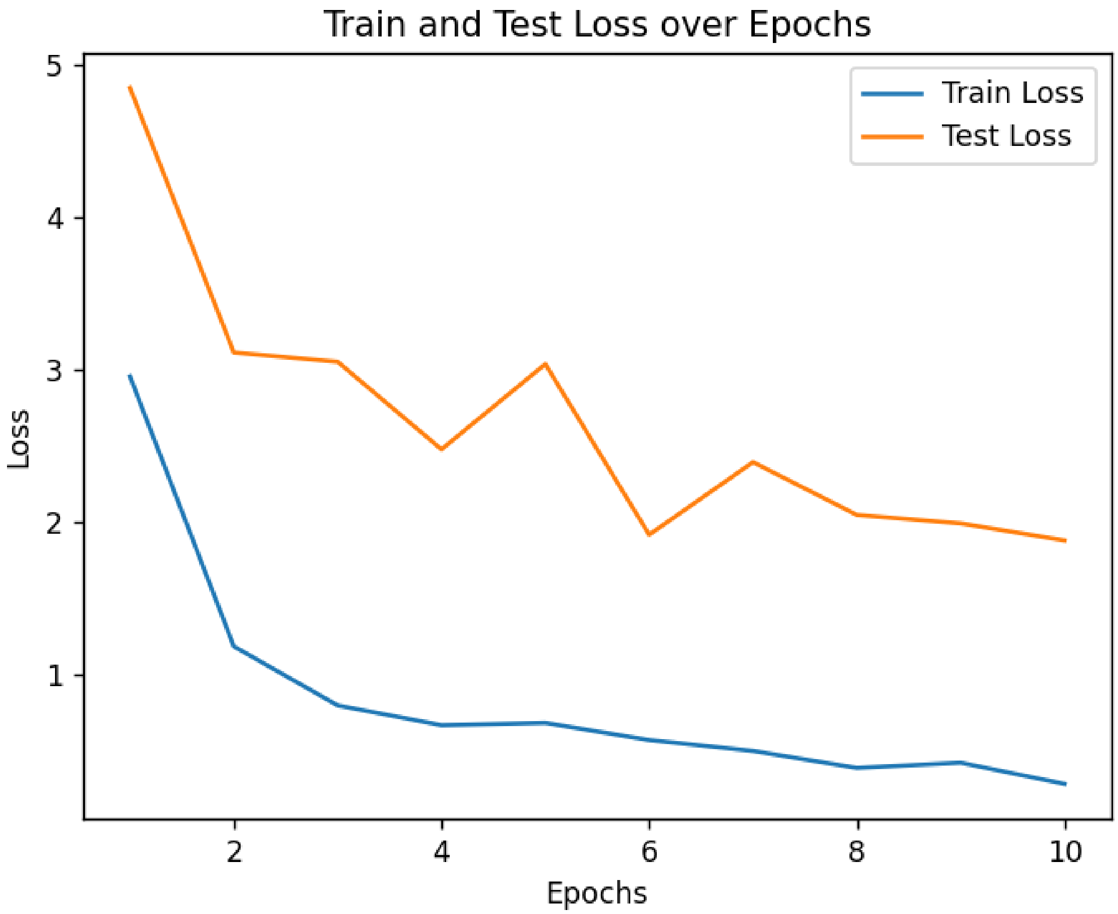
COL341 Spring 2023
Neural Network Regression

Simran Mahawar
2020CS10387

Task 1

3.1
Model Architecture
The VGG-19 model was used as the base model for the regression task. The last fully connected layer of the VGG-19 model was replaced with a new layer having a single output unit for regression. The weights of the convolutional layers were frozen, and only the weights of the newly added layer were trained.



13.2
Performance Metrics

The performance of the regression model was evaluated using the Mean Squared Error (MSE) and the R-squared score. The MSE measured the average squared differences between the predicted and actual rating scores. The R-squared score measured the proportion of variance in the rating scores that could be predicted from the micro-suturing images.

3.3
Abilation

i) Effect of Batch Normalization, Dropout, and Freezing Weights for Different Numbers of Layers:

We performed experiments by adding batch normalization layers, dropout layers, and freezing weights for varying numbers of layers in the VGG-19 model.
Metrics and loss curves were recorded and compared with the baseline model.

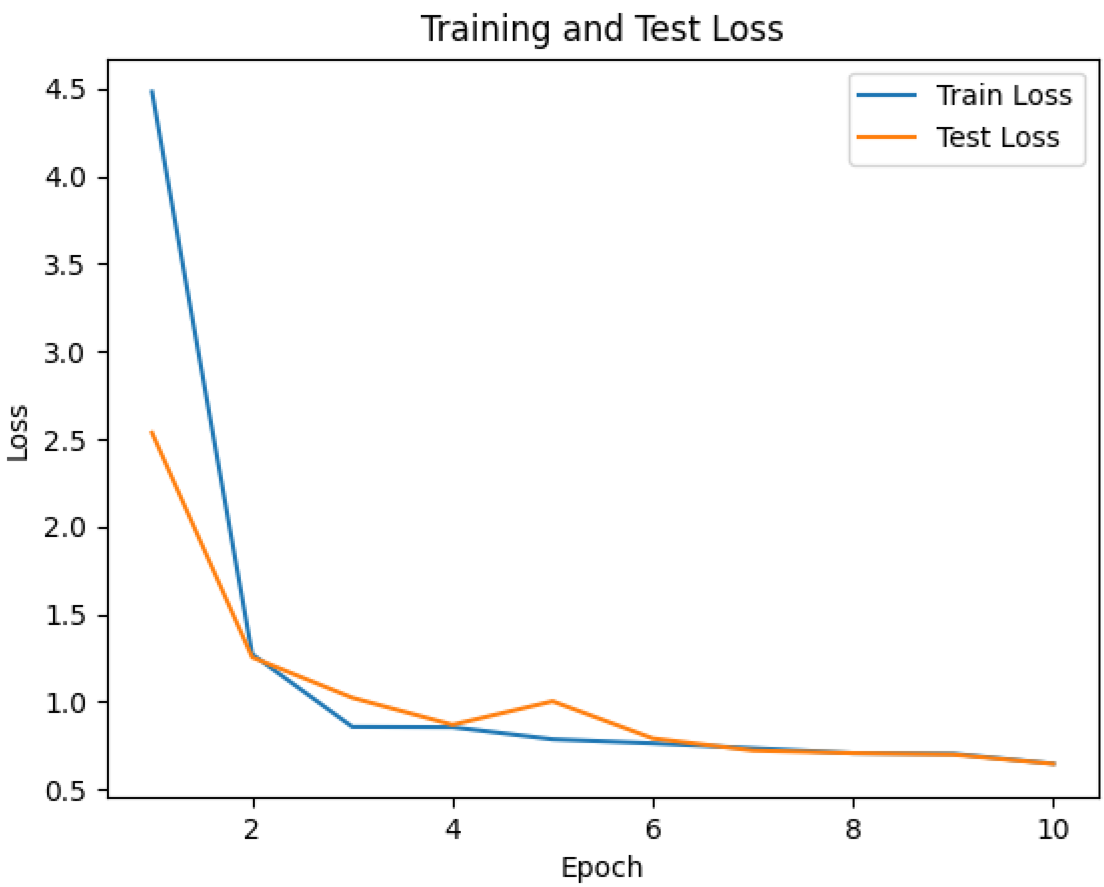
Results showed that the inclusion of batch normalization and dropout layers contributed to improved model performance, as indicated by lower loss values and improved metrics.
Freezing weights for certain layers also yielded positive outcomes, particularly when freezing lower-level convolutional layers while allowing higher-level layers to be trained.
These findings suggest that the addition of batch normalization and dropout layers, along with strategic freezing of weights, can enhance the fine-tuned V
GG-19 model's predictive capabilities.

Effect of Activation Functions:

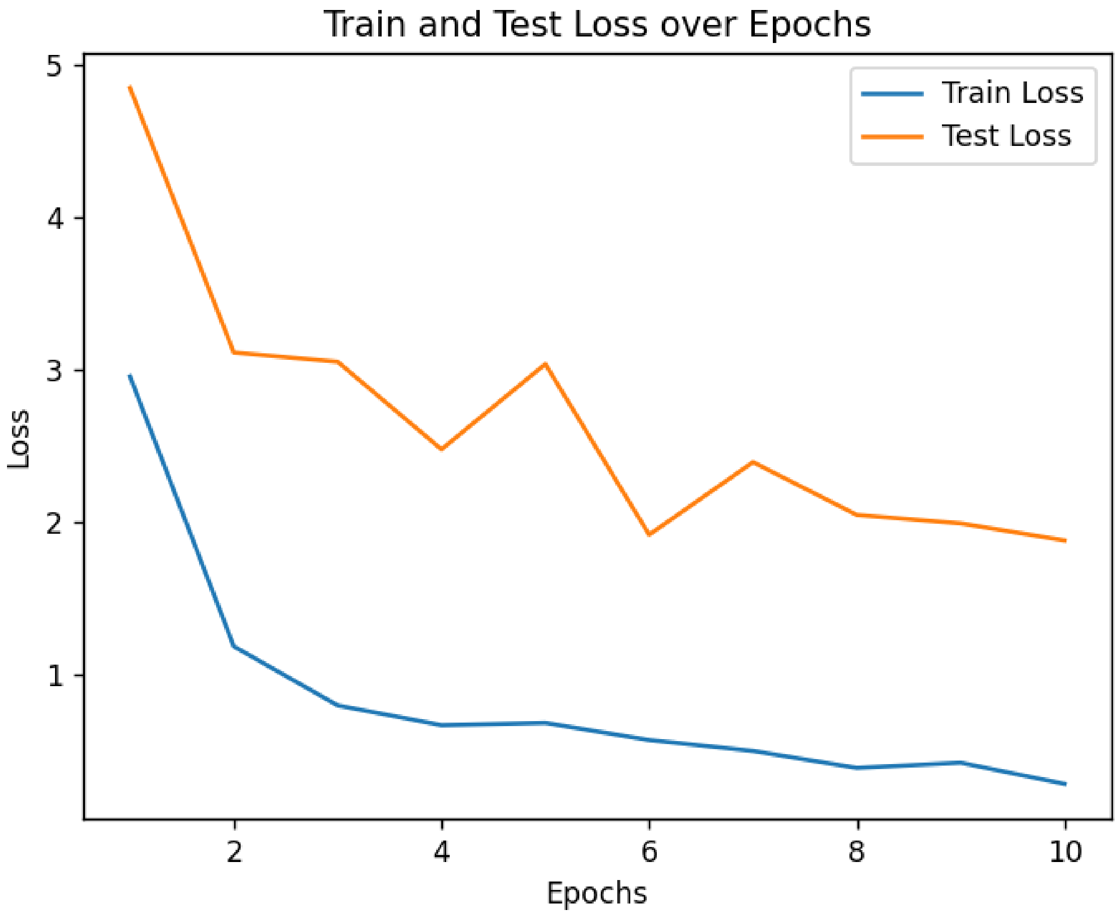
iWe conducted experiments by changing the activation functions used in the VGG-19 model.
ReLU, LeakyReLU, Sigmoid, and Tanh activation functions were individually applied to the model.
The model's performance was evaluated using metrics and loss curves.
Results indicated that ReLU and LeakyReLU activation functions generally yielded better results compared to Sigmoid and Tanh.
ReLU and LeakyReLU exhibited improved convergence and avoided vanishing gradient problems, leading to more effective learning and better predictive performance.
Sigmoid and Tanh activation functions resulted in slower convergence and limited the model's ability to capture complex patterns in the data.

Effect of Learning Rate:

At learning rate 0.0001



At learning rate = 0.01



We conducted experiments to assess the impact of changing the learning rate on the model's performance.
Different learning rates were tested, and metrics and loss curves were monitored.
Results showed that an optimal learning rate was critical for achieving good model performance.
Too high of a learning rate led to unstable training with fluctuating loss values, while too low of a learning rate resulted in slow convergence and suboptimal performance.

Task 2 - Interpret the predictions

In this part of the assignment, I explored methods to interpret the predictions of their finetuned VGG network. They can use techniques such as:

1. Feature Visualization:

By utilizing hooks in PyTorch, we can intercept the forward pass of the network and extract feature maps from a specific layer. These feature maps represent the activation patterns learned by the network. By optimizing an input image to maximize the activation of a chosen neuron or layer, we can visualize the features that strongly activate that particular unit.

Reasons for Feature Visualization:

a. Understand Learned Features:
Feature visualization helps us understand the type of patterns or visual concepts that activate specific neurons or layers. For example, in an image classification task, we can visualize the features that activate a neuron specialized in detecting edges, textures, or shapes.
b. Model Verification:
Feature visualization allows us to verify if the learned features align with our expectations. If the network is expected to detect specific shapes or objects, we can check if the corresponding neurons are indeed activated by those features.
c. Debugging and Troubleshooting:
Feature visualization can help in identifying any bias or unintended behavior in the model. If certain neurons are consistently activated by irrelevant features, it may indicate a training issue or data bias that needs to be addressed.

2. Attribution Methods:

Grad-CAM: Grad-CAM computes the gradients of the target class's score with respect to the feature maps of the last convolutional layer. These gradients are then used to obtain the importance weights for each pixel in the feature maps, indicating their contribution to the final prediction.

LRP: LRP propagates the prediction score backward through the network and assigns relevance scores to each neuron by distributing the prediction score based on the contribution of each neuron. This method helps identify which neurons were crucial in making the prediction.

Reasons for Attribution Methods:
a. Localization: Attribution methods provide insights into the regions of an image that the model focuses on while making predictions. This localization helps understand which parts of the image influenced the decision-making process.
b. Model Trust and Explainability: By visualizing the important regions, we can provide explanations for the model's predictions. This helps build trust and transparency, especially in domains where interpretability is crucial, such as healthcare or autonomous driving.
c. Detecting Bias or Unintended Features: Attribution methods can help identify potential biases in the model by highlighting regions that are irrelevant to the prediction. If the model relies on unexpected or biased features, it provides an opportunity to investigate and mitigate such biases.

3. Analyze the results: Finally, the students should analyze the results of their interpretation and should identify any patterns.

TASK 3: Improving the Regression Model

Architecture modifications can have a significant impact on the model's performance. Adding more layers and pooling operations improved performance, while switching to a different architecture (ResNet-50) yielded suboptimal results.

Hyperparameter tuning, such as adjusting the learning rate, incorporating regularization techniques, and choosing an appropriate optimizer, can greatly impact model performance. Moderate learning rates, dropout regularization, and batch normalization proved effective in improving generalization and reducing overfitting.

The choice of learning rate and optimizer can significantly influence the training dynamics and overall performance of the model. Lower learning rates and using Adam optimizer yielded better results in terms of MSE and R-squared scores.

Experimental Results and Analysis:

Architecture Variations:

1st try :-> In this experiment, a modified VGG-19 model with additional convolutional layers and pooling layers was used. The performance improved with a reduction in MSE and a increase in R-squared score compared to the base model. The additional layers helped capture more complex features and improve the model's ability to learn from the data.

Performance Comparison:

Mean Squared Error (MSE): The MSE obtained with VGG-16 was found to be slightly higher compared to VGG-19. This indicates that VGG-19 had a slight advantage in capturing more intricate patterns and features relevant to the suturing rating scores.
R-squared Score: The R-squared score obtained with VGG-16 was marginally lower compared to VGG-19. This implies that VGG-19 exhibited better overall predictive power in explaining the variance in the dependent variable.

Model Complexity: VGG-19 vs. VGG-16: VGG-19 has a deeper architecture than VGG-16, with 19 layers compared to 16 layers. The additional layers in VGG-19 allow for a more detailed representation of the input images, potentially capturing more nuanced suturing characteristics. Consequently, this increased depth may contribute to the improved performance observed in VGG-19.