Table of Contents

Lumbar Spine Degenerative Classification

Final Proposal

Introduction/Background

Our topic uses machine learning to detect and classify degenerative spine conditions using MRI images of lumbar spines. We will be classifying the presence of degenerative conditions into five different conditions, unless there are none present for which we will classify its absence. Our dataset consists of MRI images along with severity scores for each condition along the intervertebral disc levels. The features include the study ID (8 studies), the images from each study, and the vertebrae level. The labels are the x-y coordinates of the center of the relevant area, the condition, and the severity level. Dataset link: https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification/overview

There is prior research that's applicable to computer vision for spine imagery analysis. A common occurrence is high precision and resolution segmentation for clear viewing and analysis of the vertebrae in the spine [1]. This includes conventional approaches such as template matching in addition to deep learning approaches such as U-Net architectures [2]. This allows for follow up algorithms to perform diagnosis, such as estimating deviation from baseline healthy spines using simple distance measurements [1]. Other methods include using geometric centroids or varied reference points on vertebrae for angle and distance calculation [2]. An example of an approach that may increase performance includes using spatial coordinate estimation to derive stronger understandings of vertebrae positioning [3].

Problem Definition

Low back pain is a leading cause of disability, attacking 619 million people annually. Being able to diagnose through computers would make diagnosis more accessible, as it can be quickly incorporated into a general medical examination without the presence of an expert, and detect the conditions in its earliest stages.

Methods

Our preprocessing pipeline and machine learning methodologies have been refined through the semester to handle MRI images for classifying conditions in the lumbar spine degeneration with better performance. Preprocessing involved the segmentation of images, where specific regions of the spine, such as intervertebral discs, were isolated using cropping techniques based on annotated coordinates. The key methods to overcome overfitting and improve generalization were numerous data augmentations, including rotations, translations, resizing, scaling, implemented using either TensorFlow/Keras ImageDataGenerator or Albumentations. To further improve image clarity and reduce artifacts, GaussianBlur from OpenCV was used for noise reduction. The dataset was also restructured by reshaping and merging multiple CSV files, separating critical metadata fields like condition, severity, and study information into a cohesive format compatible with our models.

Our machine learning models went through three major iterations: the baseline model, which consisted of a 3-layer CNN followed by dense layers; this was more of a starting point and suffered from overfitting, failing to generalize well. Building on this, we further implemented a VGG-16 model, leveraging transfer learning from pre-trained weights, which significantly improved performance and reduced overfitting. The architecture was fine-tuned with additional layers tailored to our dataset. Finally, an Inception Net architecture was developed, featuring two convolutional layers followed by an Inception module, and

dense layers for classification. However, its performance on validation data was lower than that of VGG-16. All models incorporated data augmentation and training callbacks such as early stopping and learning rate reduction to optimize results.

While we initially planned to integrate UNet and DeepLab for image segmentation and KNN for clustering, the project's focus shifted toward improving classification models. Future iterations may revisit these methods to further enhance localization and segmentation capabilities.

Preprocessing/Visualizations

Before diving into creating our models, we first explored our dataset to better understand its components. Below is an image of a Pandas Dataframe consisting of our input CSV file. This CSV file contains information regarding our training data. We see that our input consists of primarily images along with metadata regarding the coordinates of the degenerative spine condition.

	study_id	condition	severity	image_path	x	y	series_description
0	4003253	Spinal Canal Stenosis	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	322.831858	227.964602	Sagittal T2/STIR
1	4003253	Spinal Canal Stenosis	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	320.571429	295.714286	Sagittal T2/STIR
2	4003253	Spinal Canal Stenosis	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	323.030303	371.818182	Sagittal T2/STIR
3	4003253	Spinal Canal Stenosis	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	335.292035	427.327434	Sagittal T2/STIR
4	4003253	Spinal Canal Stenosis	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	353.415929	483.964602	Sagittal T2/STIR
5	4003253	Left Neural Foraminal Narrowing	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	196.070671	126.021201	Sagittal T1
6	4003253	Left Neural Foraminal Narrowing	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	191.321555	170.120141	Sagittal T1
7	4003253	Left Neural Foraminal Narrowing	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	187.878354	217.245081	Sagittal T1
8	4003253	Left Neural Foraminal Narrowing	moderate	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	186.504472	251.592129	Sagittal T1
9	4003253	Left Neural Foraminal Narrowing	normal_mild	/kaggle/input/rsna-2024-lumbar-spine-degenerat...	197.100569	289.457306	Sagittal T1

Predicted Conditions and Severity Levels

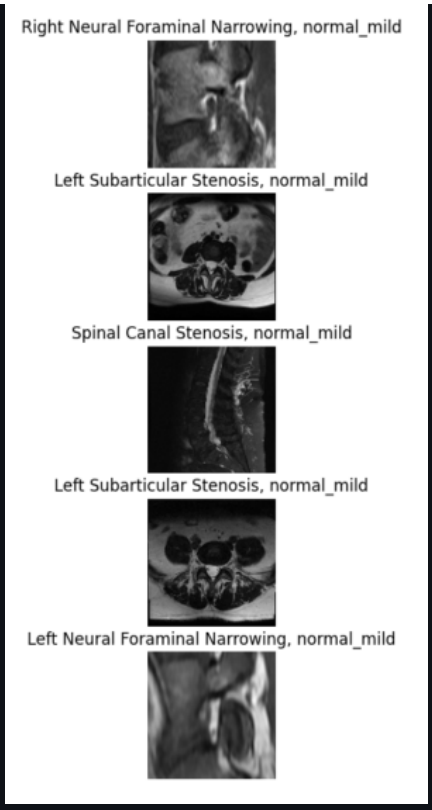
We aim to predict five distinct types of conditions, represented as categorical variables:

- **Spinal Canal Stenosis**: `0`
- **Left Neural Foraminal Narrowing**: `1`
- **Right Neural Foraminal Narrowing**: `2`
- **Left Subarticular Stenosis**: `3`
- **Right Subarticular Stenosis**: `4`

Additionally, for each predicted condition, we assign a **severity level**, categorized as follows:

- **Normal to Mild**: `0`
- **Moderate**: `1`
- **Severe**: `2`

To better understand our input data, we also plot a sample of MRI images from our dataset:



Here we notice that our images are black/white, meaning that they only have 1 channel. In addition, we also notice that the relevant features of the image appear to have small rotational variances and translations. In addition, the resolution between images appears to vary dramatically. As a result, we infer that it is likely fine to use common data augmentations such as rotations, translations, Gaussian blur, etc. Accordingly, through our three models, we implement preprocessing of the data through the following different methods:

- Row-wise reshaping: we needed to structure the study id, condition, level, and severity data for each entry in the dataset, in a way that is intuitive for access and cleaned of unnecessary information. This includes separating disease instances and condition/severity data (which are partially merged in the raw dataset). The reshaped dataset (when this operation is applied to all raw rows) has the 4 aforementioned columns.
- Dataframe merging: we add more necessary information to the dataframe by conducting a join operation with the 'label' dataframe using common columns (study id, condition, level) to add columns about series id and x and y coordinates of the center of the area of disease interest. Another join is done to add a column on the series id's corresponding description.
- Augmentation: we perform image augmentations like rotation and resizing to increase generalization capability.
- Cropping: we crop the images to a region around the center of the area of interest as a way of normalizing; by cropping to center all of them, that the model does not need to learn how to find the relevant features in the image in addition to classifying.

Model 1: Row-wise reshaping, Dataframe merging, Augmentation, Cropping

Model 2: Row-wise reshaping, Dataframe merging

Model 3: Row-wise reshaping, Dataframe merging, Augmentation (HorizontalFlip(p=0.5), RandomBrightnessContrast(p=0.2), ShiftScaleRotate(shift_limit=0.05, scale_limit=0.05, rotate_limit=15, p=0.5), GaussianBlur(p=0.2))

Model Selection and Details

Model 1: Simple CNN

We chose to implement a Convolutional Neural Network (CNN) for our machine learning model. We chose this model as our first solution implementation for multiple reasons. Primarily, CNNs form the core operation of computer vision methods; in our prior work and literature review, all models used complex forms and combinations of CNNS. As a first model, a more streamlined CNN would allow us to understand data complexity and the capability of a limited number of parameters. There are 3 convolution layers in our architecture, followed by 2 fully connected layers to aid final transformation for the classifier. Two separate ultimate output layers were used - one for condition classification, and one for severity classification. Each of these layers outputs a probability distribution for its respective task. We process these layers to achieve a final output, for example, as Foraminal Narrowing + Moderate.

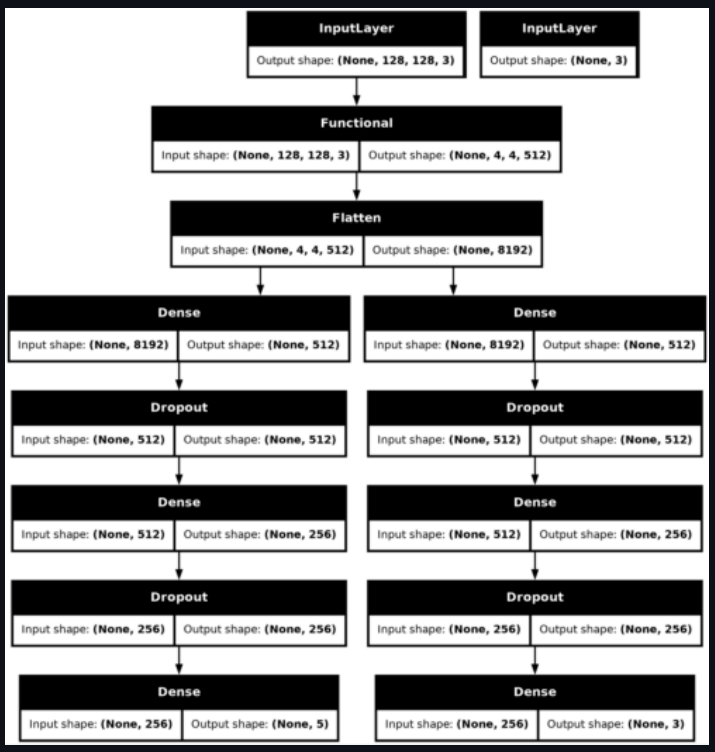
Model 2: Inception CNN

Because of the nature of the problem we are trying to solve (with images), we chose to keep the general ML algorithm of CNN and work on improving it. Inception CNN, from GoogLeNet is an improvement from the traditional simple model we created for Model 1 with the main benefit of being able to use multiple varying filter sizes in parallel within the same layer. This will give us the flexibility of learning new and important feature, especially smaller patterns (which is neccessary as we are looking through spine disks). This model consists of simple inception architecture with an initial convolutional layer followed by an inception module. This is then followed by dense layers for classification for both condition and severity. The inception module has 4 branches computed in parallel and then concatenated: 1x1 convolution; a 1x1 convolution followed by a 3x3 convolution; 1x1 convolution followed by 5x5; and 3x3 max pooling followed by a 1x1 convolution. This allows capture of features at 3 levels of granularity (kernels of size 1, 3, and 5) plus local invariant features (inference after pooling).

Model 3: VGG-16 CNN

With the model 2 performing a little better than model 1, we wanted to further explore other architectures that will significantly boost our accuracy with classification. Thus, we decided on using the VGG-16 backbone for our CNN. VGG-16 is a very popular model used for transfer learning with strengths in capturing spatial hierarchies of features as well as doing well in object classification. What we did was use

VGG-16 first, followed with two identical neural nets in parallel. One network is used to predict condition while the other is used to predict severity.

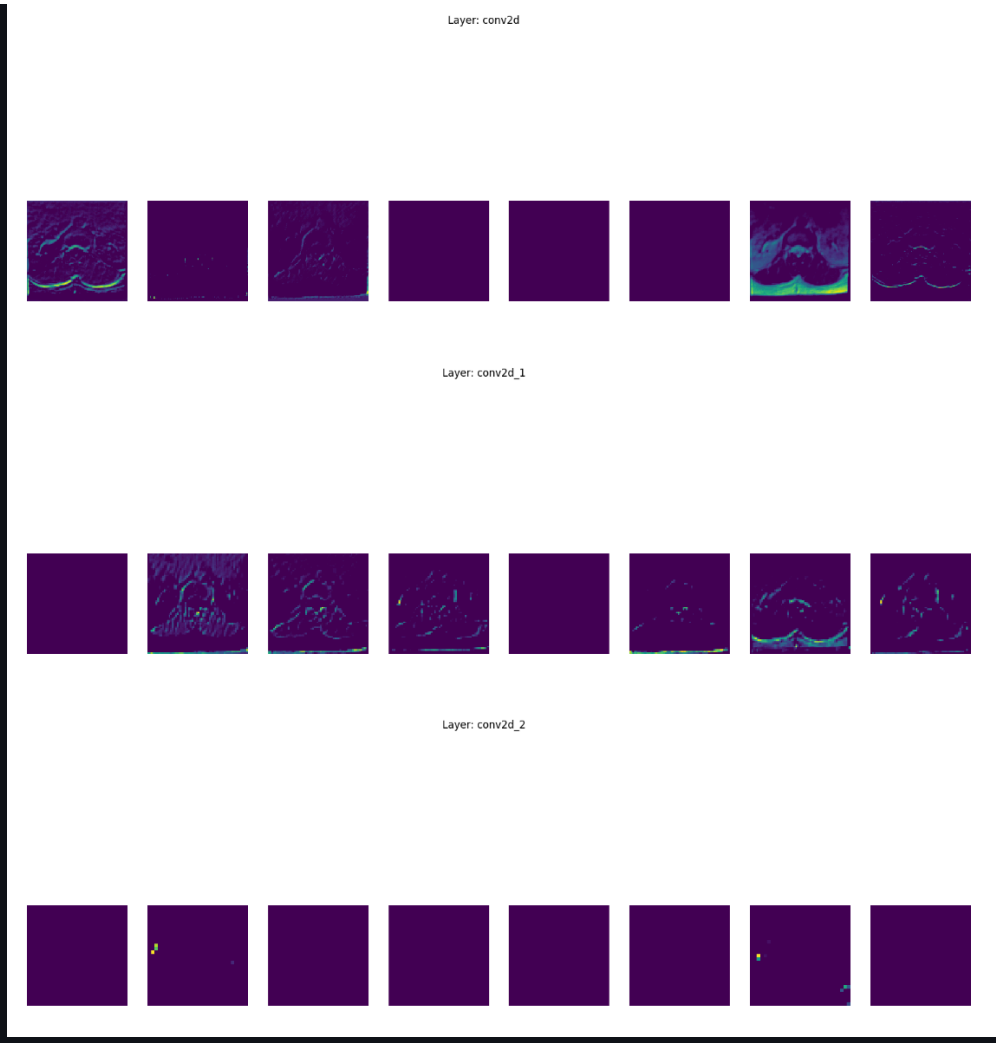


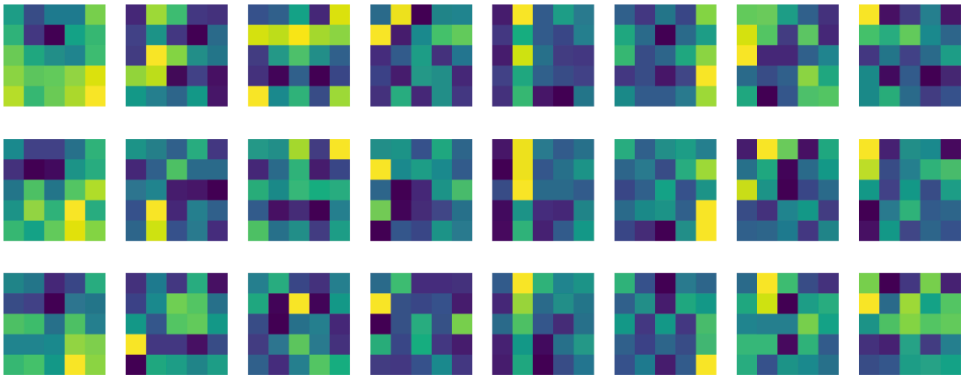
Results and Discussion

To visualize our model in action, we created feature maps and filter plots to examine intermediate activations within convolutional layers, allowing for a visual assessment of model focus across layers. We notice that the first layer in the CNN appears to capture high level features regarding the image while the second layer captures finer details in the image.

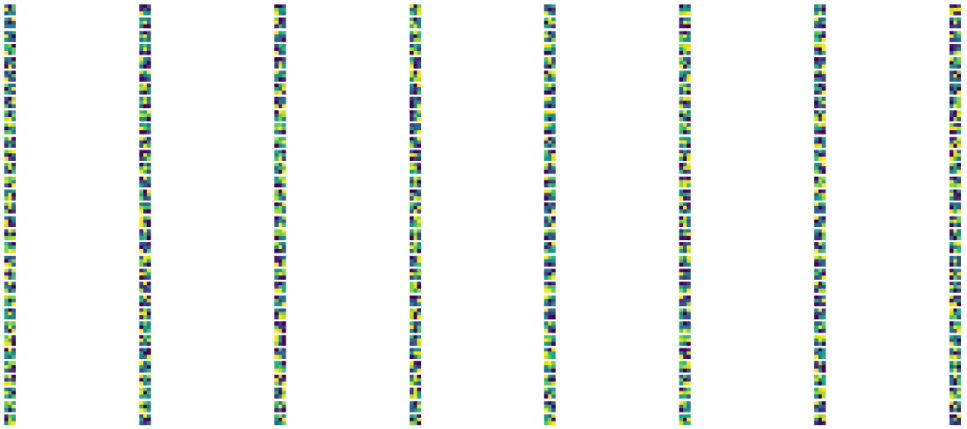
Below are images of our convolutional kernels. We provide these as a way to identify potential kernels that the model may have learned to specialize in specific tasks. While there is not much interpretable information present, we do notice that each kernel appears to have certain “activated” regions, likely indicating that the model is performing some sort of feature extraction from the input images.

Model 1 Visual: Simple CNN

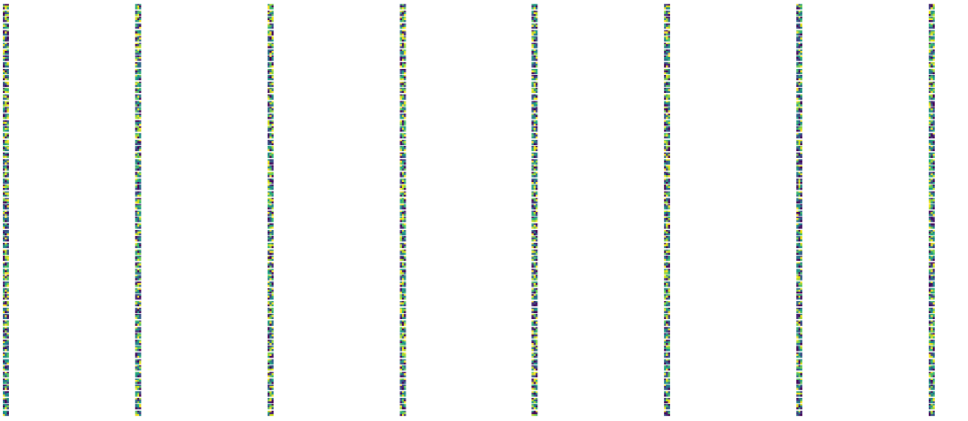

Layer: conv2d
Layer: conv2d_1
Layer: conv2d_2



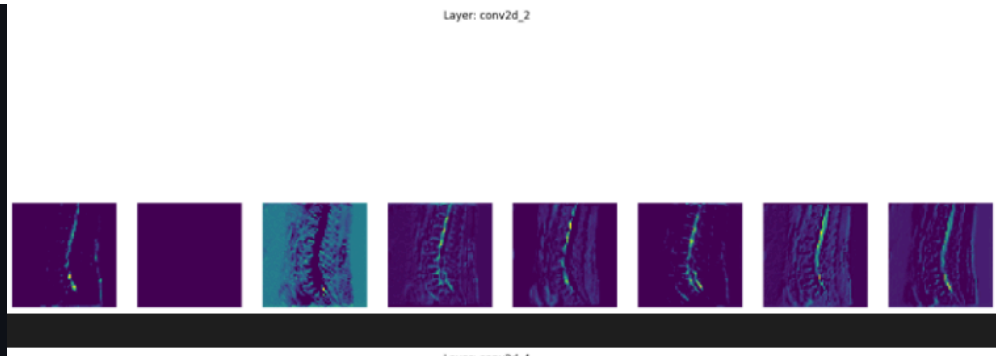
Filters of layer: conv2d_1

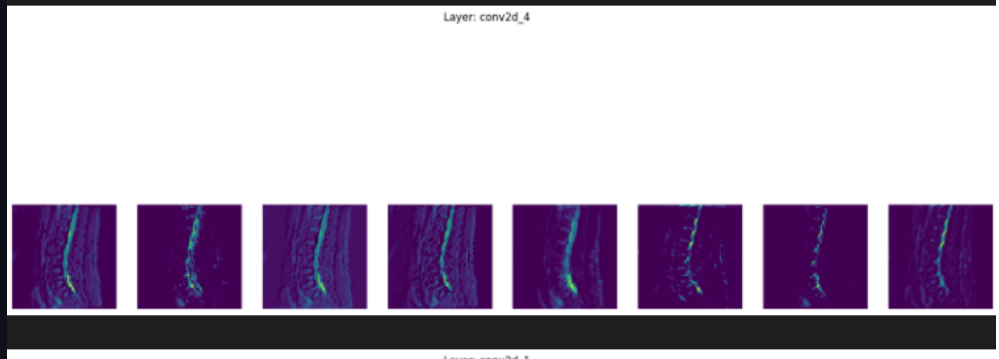


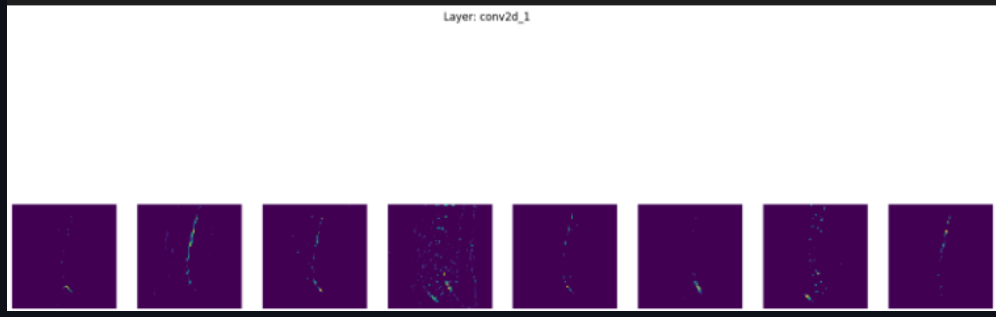
Filters of layer: conv2d_2

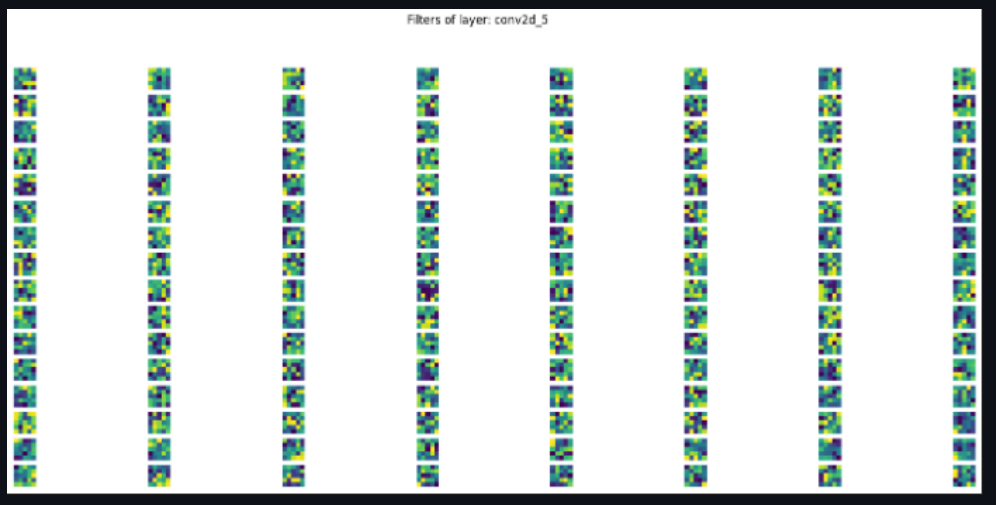


Model 2 Visual: Inception CNN

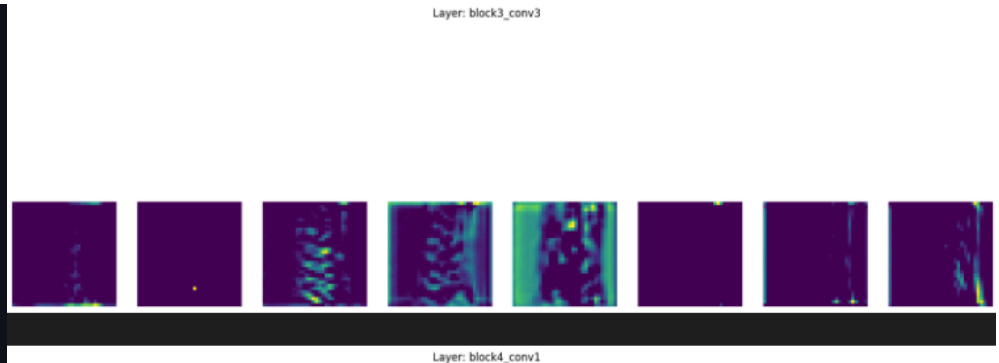


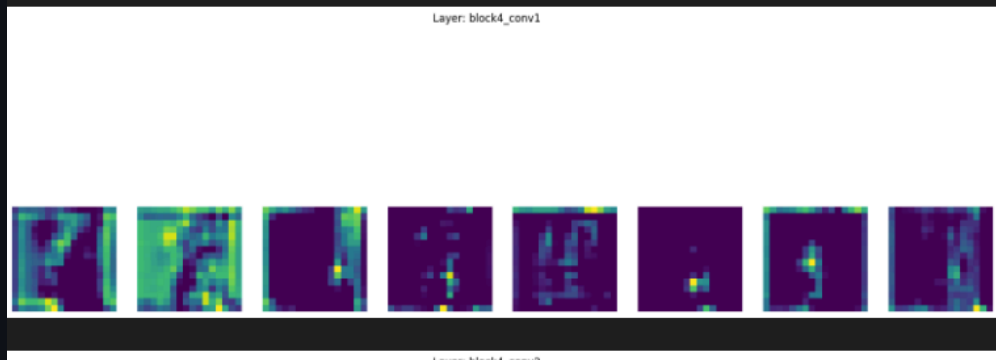


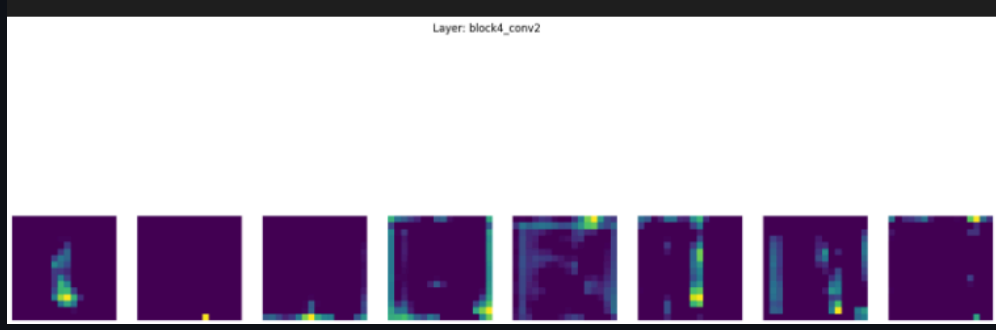


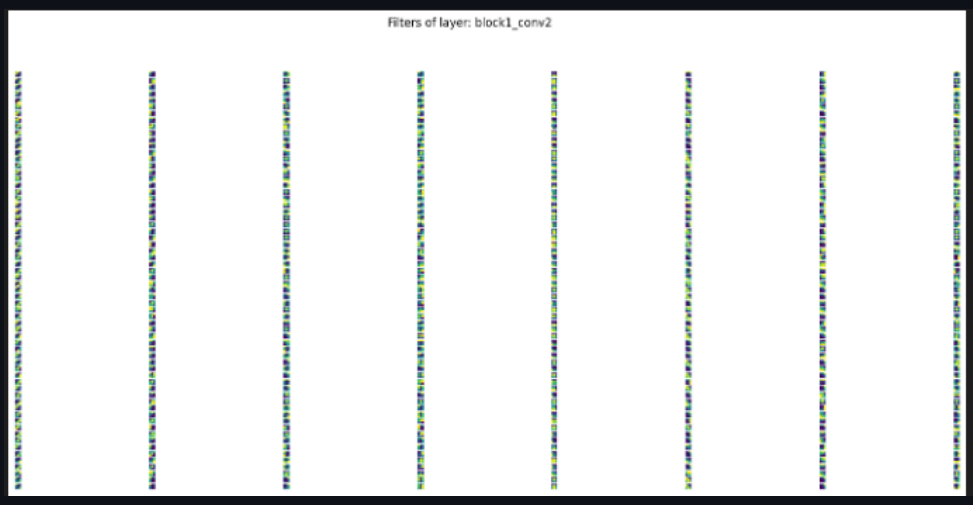


Model 3 Visual: VGG-16 CNN









Metrics

To evaluate our models, we used a few different types of metrics. First, we used accuracy to identify how well our models were able to predict the type of condition and severity of the condition present in a given image. This metric simply calculates the number of correct outputs divided by the number of total outputs.

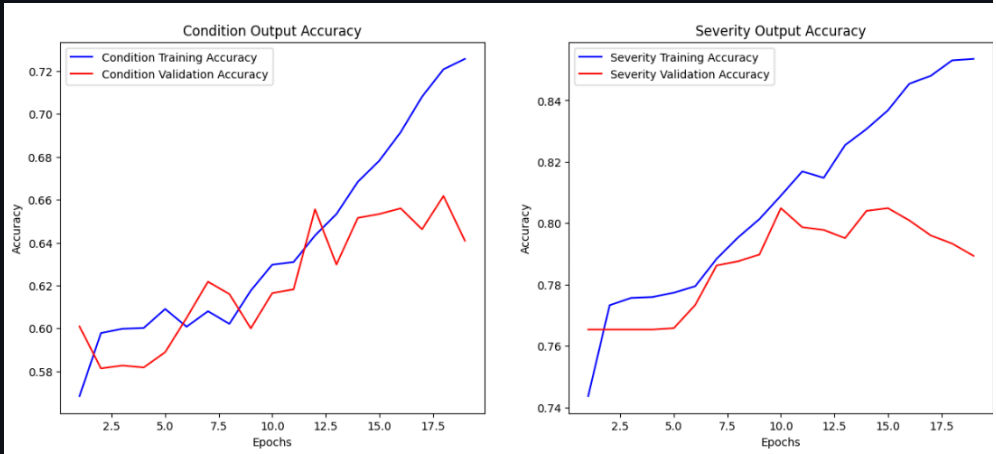
We also use precision, recall, and the F1-Score of our model as another way to evaluate the model performance during test time. Precision measures how well our model performs in its positive predictions. Specifically, given a set of predictions, precision measures how many of those predictions are truly correct. On the other hand, recall measures how well a model is able to predict all true instances of a given class. Overall, precision measures how accurate a model's positive predictions are while recall

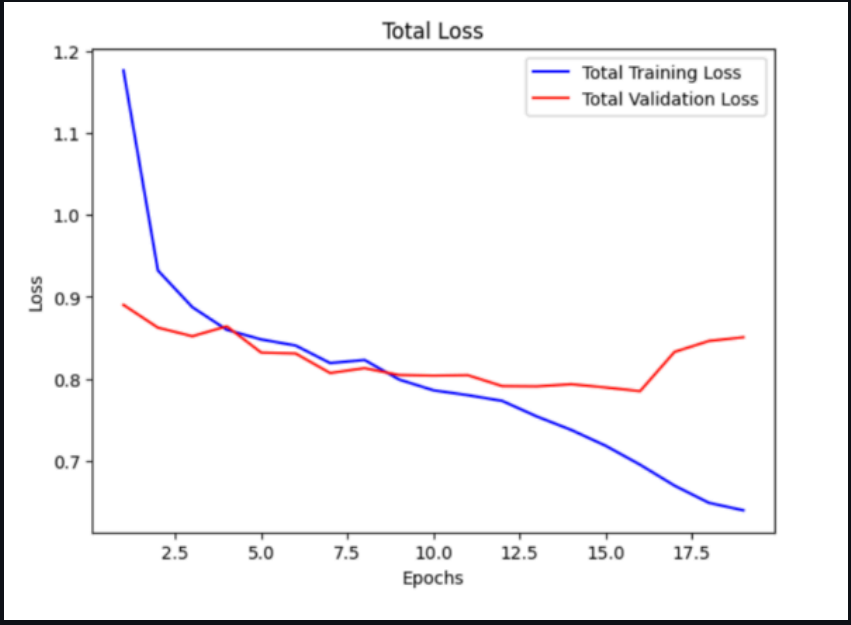
measures how well a model is able to identify all positive instances. The F1-Score is the harmonic mean of the precision and recall and hence provides a metric that provides insight into how well the model is performing in terms of both precision and recall. A higher F1-Score indicates that a model usually performs better in terms of precision and recall, however, this cannot be said with certainty.

Model Analysis

Model 1 Analysis: Simple CNN

To test and evaluate our model, we divided the data into training, validation, and test sets using a 70-15-15 split, and utilized various training callbacks such as early stopping and learning rate reduction during our model's training process. When doing initial tuning, we would utilize plots to visualize the training and validation set accuracy differences in an epochs vs accuracy graph, along with plotting losses, to help identify evident cases of underfitting and overfitting.





As seen in the plots above, the validation accuracy happens to plateau around 0.65 for the condition output accuracy and 0.79 for the severity output accuracy, whereas the training accuracies seem to continuously rise as a steady rate with accuracy values much higher. This implies that there's some overfitting within our baseline model, as it's not performing as well on this validation set of new data. The loss plot supports this; as the number of epochs continues, the training loss has a rather steady decline, yet the validation loss plateaus doesn't have that steep of a slope as the graph progresses. For the 2 classes we were predicting, condition and severity, we used the evaluation metrics of accuracy, precision, recall and F1-score to evaluate the model on the testing set.

Condition Metrics:

- **Accuracy**: `0.6720`
- **Precision**: `0.6728`
- **Recall**: `0.6720`
- **F1 Score**: `0.6694`

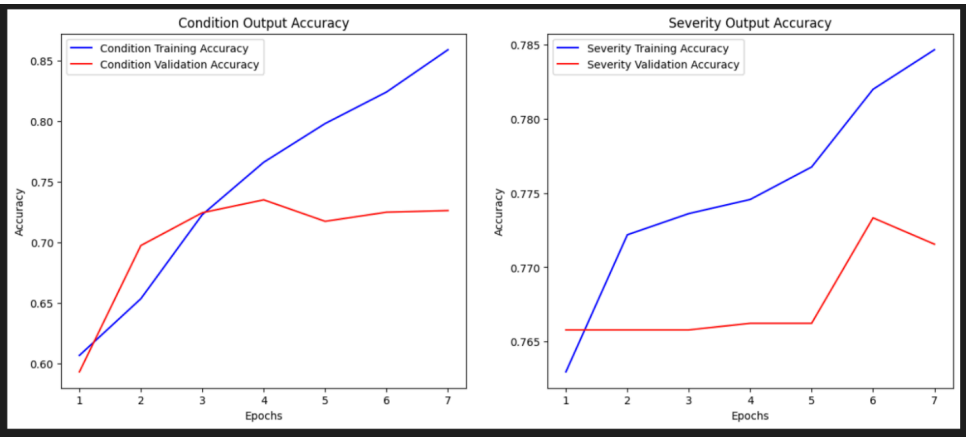
Severity Metrics:

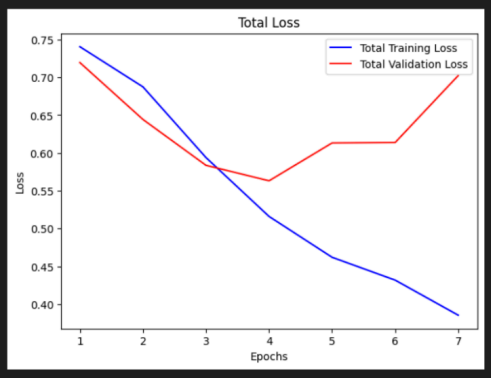
- **Accuracy**: `0.8156`
- **Precision**: `0.7981`
- **Recall**: `0.8156`
- **F1 Score**: `0.7946`

Since the metrics vary significantly for each of the two predictive classes, we will analyze them separately. For the condition metrics, the precision and recall are fairly similar values which indicates a balanced performance with no significant or extreme trade-offs. Given that they're also very close to the classification accuracy, this indicates there's both a balanced rate of correct positive predictions and is consistent in identifying true positive cases. The accuracy is rather low, though, which is probably a byproduct of the model's overfitting problem, and is to be improved in the next iteration of the model. As per severity metrics, the metric values are also centered around each other with very little deviation, indicating that there's also a balanced rate of correct positive predictions and consistency in identifying true positive cases. The major difference is the accuracy; the accuracy is much greater for predicting the severity class, although will be improved in a later iteration of the model once we improve the image processing and model architecture.

Model 2 Analysis: Inception CNN

We had the same data split as Model 1, but augmentation is unused. Training utilized a learning rate scheduler with a patience of 2 and a multiplier of 0.75, enabling more refined optimization. The only difference was that the batch size was changed to 32 for this model instead of 16 as it was in the VGG-16 based model.





+ Code + Markdown

While the inception-based model shows an iterative improvement in training and validation performance through epochs, it struggles to generalize, as reflected in its testing metrics. The plots below depict a large difference in performance trend for training versus validation across condition and severity prediction tasks. During training, the model achieves a final condition classification accuracy of 85.13% and a severity classification accuracy of 78.40%, with a corresponding validation accuracy of 72.62% for condition and 77.33% for severity at the end of training. The loss curves reflect a gradual decrease in the condition classification loss, whereas the severity classification loss stabilizes much earlier, probably reflecting the fact that the task of severity may be intrinsically less complex than condition classification for the Inception architecture. Therefore, while the Inception-based model is effective in capturing the spatial hierarchies through its multi-branch structure, it is inherently sensitive to the preprocessing variations, which makes it prone to inconsistencies in the input data and likely resulted in the observed overfitting for both tasks.

Condition Metrics:

- **Accuracy**: `0.7124`
- **Precision**: `0.7125`
- **Recall**: `0.7124`
- **F1 Score**: `0.7120`

Severity Metrics:

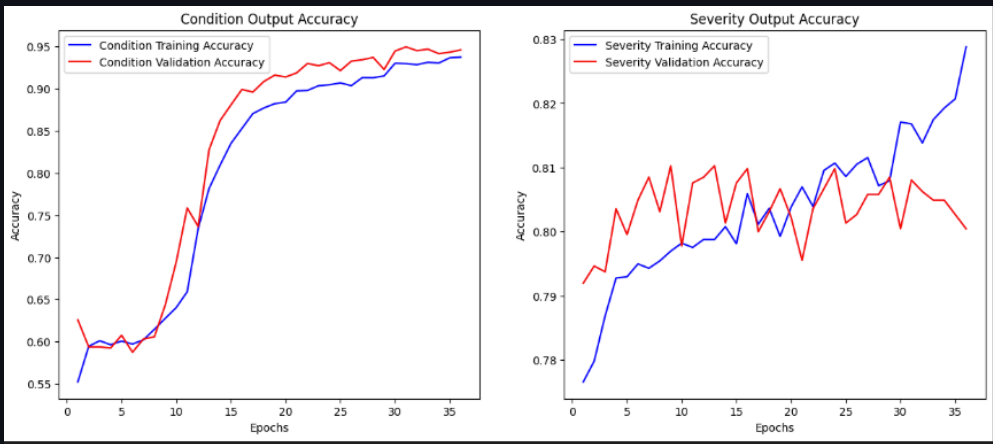
- **Accuracy**: `0.7911`
- **Precision**: `0.7473`
- **Recall**: `0.7911`
- **F1 Score**: `0.7253`

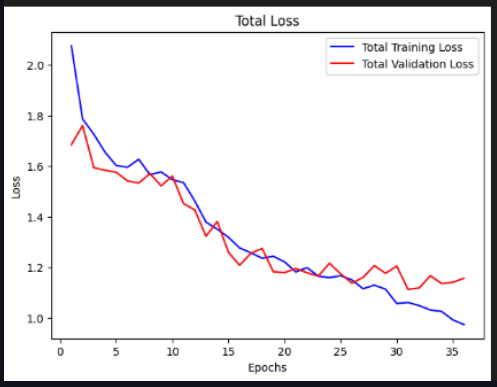
In terms of testing performance, the model achieves a Condition Accuracy of 71.24%, with balanced metrics across precision (71.25%), recall (71.24%), and an F1 score (71.20%). These are in line with the model's validation performance, showing that it generalizes comparatively well for the severity output (as compared to the condition output). The accuracy is higher for the severity classification, reaching 79.11%,

strongly supported by its recall (79.11%) but slightly lower in precision (74.73%), giving an F1 score of 72.53%. This discrepancy in precision and recall implies that the model slightly overpredicts some levels of severity, which is a trade-off that may arise from the class imbalance or distribution inherent in the dataset.

Model 3 Analysis: VGG-16 CNN

To test and evaluate this model, we divided the data into training, validation, and test sets using a 70-15-15 split like last time, and utilized the same training callbacks as earlier, including early stopping and learning rate reduction. We eliminated cropping, however, and added marker metadata instead, and employed data augmentation techniques as specified earlier.





As observed in the plots, the validation accuracies for the two outputs follow different trends. The condition output accuracy peaks at ~0.94, which closely follows the training accuracy (~0.9376) after approximately 35 epochs. This is a big improvement from the previous model, since the validation and training accuracies are closely aligned, suggesting that this task has minimal overfitting.

On the contrary, the severity output validation accuracy plateaus around 0.80, whereas the training accuracy steadily increases to ~0.8356. The difference between training and validation accuracy for the estimation of severity indicates a moderate overfitting within this branch; however, it is not as substantial as that within the previously presented model. Besides, the dynamics of the training and validation losses confirm the above-said: while the training loss steadily decreases, the validation loss flattens and even slightly increases in the later epochs, especially for severity prediction.

The use of a backbone of VGG-16 and image augmentations like random flips, brightness contrast, and Gaussian blur would have contributed toward generalization improvement in overall performance. A branched architecture allows the model to have separate networks for predicting condition and severity, which makes the model non-interfering across tasks and hence enables more specialized learning in each of the branches.

Condition Metrics:

- **Accuracy**: `0.9418`
- **Precision**: `0.9419`
- **Recall**: `0.9418`
- **F1 Score**: `0.9417`

Severity Metrics:

- **Accuracy**: `0.7844`
- **Precision**: `0.7496`
- **Recall**: `0.7844`
- **F1 Score**: `0.7587`

The condition metrics demonstrate exceptional performance. The close values of accuracy, precision, and recall indicate that the performance is well-balanced, with very small trade-offs between true positive predictions and false positives. The F1 score further confirms the balance between precision and recall. These improvements can be attributed to the updated architecture and augmentations, which seem to have addressed the overfitting issues observed in the previous model.

While the condition metrics improved, the severity metrics are less impressive as compared to the previous model. The gap between precision and recall indicates that the model is slightly more inclined to correctly identify positive cases (higher recall), at the cost of increased false positives. This could be a result of either not enough regularization in the severity branch or the complexity inherent in the severity prediction task.

Model Comparison

Overall, the models we trained demonstrated a clear progression in terms of both accuracy and complexity, starting with a traditional baseline CNN. This initial model allowed us to quickly and efficiently train a simple architecture while we learned more about the dataset and other aspects of this problem, but due to its shallow and simple design, it struggled to capture the complex patterns necessary for accurate classification of lumbar spine conditions. This limitation was shown in the feature maps and filter plots, as the features learned were simple and not of the more abstract and hierarchical features required for distinguishing different conditions. With the Inception model, an improvement over the basic CNN, it offered efficient feature extraction through parallel convolutions with different filter sizes; however, it was hard to tune and susceptible to irrelevant patterns and how data was preprocessed. With the lack of data augmentation and variety, the model will simply memorize training images and not actually learn features. This caused overfitting when training which is why the test accuracy is relatively low. Lastly, the VGG-16 model outperformed both the basic CNN and Inception models, delivering the best results in terms of both condition classification and severity prediction, as it incorporated transfer learning through the use of a pretrained VGG network. This can be attributed to VGG-16's deep architecture and powerful image classification capabilities, which allowed it to better capture the intricate features necessary for accurate diagnosis. It is relatively simpler in design compared to Inception, but its main drawback is its computational cost, as VGG models have a lot of parameters, which leads to a long training time and high memory requirements.

Next Steps:

With our model 3 performing the best, we would like to continue to improve the VGG model through additional preprocessing and hyperparameter tuning to give us the best result. Moreover, we will need to consider separate models for axial and sagittal planes as these two types of images do negatively affect the CNN's learning. Furthermore, we can also explore advanced model architectures, such as evaluating specialized models like U-Net and Mask R-CNN for precise segmentation to improve region-based localization of spine abnormalities. We also want to consider 3D CNNs for volumetric analysis across multiple planes, which could provide richer context in classifying conditions across multiple spine

sections, and experiment with EfficientNet or DenseNet as backbones to leverage their high performance and efficient computation, especially for handling complex image details in spine data.

References

[1] S. Ruiz-España, E. Arana, and D. Moratal, “Semiautomatic computer-aided classification of degenerative lumbar spine disease in magnetic resonance imaging,” Computers in Biology and Medicine, vol. 62, pp. 196–205, Jul. 2015. doi:10.1016/j.compbiomed.2015.04.028

[2] R. F. Masood, I. A. Taj, M. B. Khan, M. A. Qureshi, and T. Hassan, “Deep learning based vertebral body segmentation with extraction of spinal measurements and disorder disease classification,” Biomedical Signal Processing and Control, vol. 71, p. 103230, Jan. 2022. doi:10.1016/j.bspc.2021.103230

[3] Nibali, A., He, Z., Morgan, S., and Prendergast, L., “Numerical Coordinate Regression with Convolutional Neural Networks”, arXiv e-prints, Art. no. arXiv:1801.07372, 2018. doi:10.48550/arXiv.1801.07372

Contributions

Proposal

	Research and Project Proposal	Website Deployment	Presentation and Recording
0	Pranav Somu	Joshua Diao	Adrish Kar
1	Rishita Dhalbisoi		Ayush Agarwal
2	Joshua Diao		

Midterm

	Data Sourcing, Cleaning, and Model Building	Model Selection and Methods	Results, Evaluation, and Analysis
0	Adrish Kar	Pranav Somu	Ayush Agarwal
1	Joshua Diao		Rishita Dhalbisoi
2			

Final

	Data Sourcing, Cleaning, and Model Building	Model Selection and Methods	Results, Evaluation, and Analysis	Presentation and Recording
0	Pranav Somu	Pranav Somu	Joshua Diao	All
1	Ayush Agarwal	Ayush Agarwal	Adrish Kar	
2	Rishita Dhalbisoi	Rishita Dhalbisoi		

Ganett Chart