Deep Learning for Lumbar Spine Classification

Arjun Ashok*
Department of Computer Science, Statistics
University of California, Davis
Davis, USA
arjun3.ashok@gmail.com

Zhian Li
Department of Biological and Agricultural Engineering
University of California, Davis
Davis, USA
lionellee0126@gmail.com

Ayush Tripathi
Department of Computer Science
University of California, Davis
Davis, USA
atripathi7783@gmail.com

[1]

***Abstract*—As deep learning becomes increasingly prevalent across industries, research into how such models can assist medical professionals in diagnosing conditions has become a popular endeavor. In this project, we explored how baseline vision models (e.g. classic CNN), state-of-the-art vision models (ResNet, Vision Transformer), and novel model architectures (Kolmogorov-Arnold Networks) stack up against each other in the context of diagnosing the severity of lumbar conditions via radiology imagery provided by RSNA. We then compare our architectures' results against prior work using ResNets to generate a more thorough conclusion on what models may fair better in future applications of deep learning in diagnosis from the perspective of performance and interpretability.**

***Index Terms*—Deep Learning, Medical Imaging, CNN, ResNet, Transformer, KAN**

I. Introduction

A. Problem

Lower back pain is the leading cause of disability in the world, impacting over 619 million people world-wide [6]. Spondylosis, in particular, encapsulates a set of degenerative spinal conditions that can be diagnosed via Magnetic Resonance Imaging (MRI) as follows;

- Left Neural Foraminal Narrowing
- Right Neural Foraminal Narrowing
- Left Subarticular Stenosis
- Right Subarticular Stenosis
- Spinal Canal Stenosis

Although widespread in its effect, Spondylosis often goes undiagnosed and untreated, leading to chronic health issues for those suffering from lower back pain. Accordingly, there exists a strong need for diagnosing these conditions early, accurately, and in an explainable way across all 5 subsets of the condition.

B. Motivation

As the application of deep learning for aiding medical diagnoses surges, the importance of introducing accurate and interpretable models becomes increasingly relevant. In fact, prior to a competition hosted by the Radiologists Society of North America (RSNA) for using deep learning on this dataset, no research or intense application of machine learning for this subset of conditions has been developed.

As such, there is a strong need for the following:

- **Accurate Diagnosis**: building upon prior research to produce a more accurate model fit for real-world deployment.
- **Explainability**: limit model complexity to retain interpretability for diagnosticians leveraging the pipeline; after all, an accurate diagnosis is worthless without sufficient explanation of why.
- **Comparison for Future Endeavors**: in the process of constructing the most capable pipeline for this task, we want to explore a variety of architectures and approaches to conclude which is the most promising for future work to explore, both within this problem and in related contexts.

C. Prior Work

As previously touched on, the majority of previous research into this problem comes from the recent competition hosted by RSNA on Kaggle. Prior work has thus far failed to achieve remarkably good performance. The large majority of proposed models average a weighted-log loss of 0.36 at best. We'll reference back to this loss as our baseline performance measurement.

Related work also includes work on the individual architectures employed [4], [2], [10], [7], and [5], although they generally touch less on downstream application and focus more on the core model design.

II. Dataset

A. Source

As mentioned previously, we leverage the dataset collected by the Radiological Society of North America (RSNA) in partnership with the American Society of Neuroradiology (ASNR). The dataset tracks 1975 patients, each with multiple scans across multiple angles.

[1]* corresponding author

Fig. 1. A quick sampling of the dataset (un-processed except for resizing) we use and their corresponding target labels.

Each imaging study in the dataset includes the target severity scores (Normal/Mild – 0, Moderate – 1, Severe – 2) and intervertebral disc level information (L1/L2, L2/L3, L3/L4, L4/L5, L5/S1). We only leverage the severity scores for the classification task, thus requiring only one classification head of 3 nodes.

B. Transforms

Prior to training, we briefly pre-process the MRI data across all models to improve performance remain consistent in our comparison. The transforms we conduct are as follows:

1) **Resize**: we resize all images to 224×224 to ensure a consistent input for all models–this allows an even comparison for the downstream computational efficiency and further standardizes the input.
2) **Normalize**: we normalize all images such that every channel (just one in this case since MRI images are grayscale) is distributed with mean pixel brightness of 0.5 and standard deviation of 0.5. Not only does this aid with consistency across each batch, but we get the additional benefit of a model less sensitive to learning rate since all inputs will adhere to a set range of values. The observed downstream impact is a faster, stabler convergence across all architectures during empirical testing.

Respectively, we allocate approximately 70%, 10%, and 20% of the data for training, validation, and testing.

III. Model Architectures

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks have become the de-facto architecture employed with image processing tasks in recent years. Their flexibility in learning task-specific filters rather than needing pre-built/manually specified convolutional kernels allows them to outperform traditional MLPs quite effectively.

Our deployed architecture makes use of two fundamental blocks:

1) **Convolutional Process**: each convolutional process consists of a 2D-convolutional layer (7×7 kernel), a ReLU activation, and 2D-Max-Pooling (2×2 kernel). We stack 3 convolutional processes to conduct sufficient feature extraction with 8, 16, then 32 channel outputs respectively.
2) **Multi-Layer Perceptron (MLP)**: we feed the last convolutional process' outputs into a traditional multi-layer perceptron head of 3 fully-connected, linear layers to perform the classification. Recall our output layer will have 3 nodes, one for each severity.

B. Modified Convolutional Neural Network (MCNN)

The most recent trend in deep learning for computer vision is attention-based networks, i.e. vision transformers. To continue maintaing its relevancy, one group of researchers have augmented ConvNet to produce ConvNeXt: a CNN-based architecture that lends some architectural decisions from attention-based models [4]. Due to the substantially different training approach they utilized, we opted to use a subset of their augmentations on ConvNet to apply on our own CNN (III-A):

- **Activation Functions**: rather than applying ReLU after each operation, we utilize a single GeLU operation in every convolutional process [4].
- **Normalization**: taking inspiration from vision transformers, we utilize layer normalization over batch normalization [4].

Though these changes appear small, their downstream impact has proven rewarding in prior work and may encourage similar borrowing in future work [4].

C. Residual Neural Network (ResNet)

ResNet was first developed by Microsoft Research (MSR) with the goal of deepening neural architectures [2]. Previously, models with deep hidden layers would suffer from the vanishing gradient problem–gradients would start to approach zero as backpropagation moved away from the output layers, thereby rendering some layers relatively untrainable. To remedy this, ResNet relies on learning a function $F(x)$ for the input x such that $H(x) = F(x) + x$. More broadly, the model learns to augment the input over each residual block rather than transforming the whole input as in more basic convolutional architectures. Thus, each residual block has the output of the previous residual block added to its own output [2].

With this mechanism, ResNet architectures can be much deeper while retaining trainability. In our implementation of ResNet, we leverage the ResNet-18 architecture (18 layer CNN) without pre-training.

D. Vision Transformer (VIT)

Transformers were initially developed by Google Brain for processing and generating large sequences of data, particularly in text-related tasks such as translation [10]. Recent work, however, has explored the use of transformers in vision settings–unsurprising since convolution has similarly proved useful for text-related tasks [1] [10]. The vision transformer has proved to be extremely effective at learning context within images, matching the performance of state-of-the-art models at a fraction of the compute [1].

Briefly, vision transformers work by dividing an image into a grid of smaller cells, where each cell can be thought of as a token. As with text-related tasks, attention blocks are highly effective at making associations between tokens, allowing the vision transformer to leverage broader context across all grid cells. A convolutional network, in comparison, inherently relies on the surrounding region of a pixel to extract information, thereby prone to ignoring important context that is not spatially local.

For our implementation of the vision transformer, we built a backbone of several essential components, including patch embedding, multi-head attention, positional encoding, and a sequence of transformer encoder layers.

1) **Patch Embedding**: the image is divided into non-overlapping patches of size 16×16 , which are then flattened and passed through a convolutional layer to project them into an embedding space of 768 dimensions. This process effectively transforms the image into a sequence of tokens.
2) **Positional Encoding**: since transformers lack a built-in understanding of spatial relationships, a learnable positional encoding is added to each token to retain information about the relative positions of patches in the image.
3) **Transformer Encoder Layers**: these consist of multiple encoder layers, each containing a multi-head self-attention mechanism followed by a feed-forward neural network.
4) **Classification Head**: After passing through all transformer encoder layers, the output corresponding to a special class token (CLS token) is extracted. This token's representation is then passed through a fully connected layer to output class logits for classification.

E. Convolutional Kolmogorov-Arnold Network (CKAN)

Kolmogorov-Arnold Networks are an alternative basis to the perceptron-learning networks that have thus far powered deep learning architectures.

Briefly, deep learning is based on approximations–the objective of neural networks is to learn a manifold of some kind, i.e. approximate a function [3]. The Universal Approximation Theorem enabled the Multi-Layer Perceptron architecture by roughly proving that feed-forward networks with non-polynomial activations can map between two Euclidean spaces (i.e. approximate any function) [3]. Note here that this doesn't explicitly prove a method of convergence (global minimum), although backpropagation tends to estimate parameters well enough for practical use (local minimum) [3].

However, an alternative basis has recently been explored with the Kolmogorov-Arnold Representation Theorem [5]. The theorem roughly states that any multivariate, continuous function can be decomposed into a *finite* sum of single-variable, continuous functions [8]. The most promising application of this basis has been in learning activation functions as splines rather than a set function (e.g. ReLU) [5]. In addition to enabling a smaller network to emulate the performance of a larger MLP, this approach allows for increased interpretability by way of these learned splines [5]. This is analagous to a learned kernel in CNNs for better performance and interpretation.

Due to its relatively new deployment, we opted to simply replace some of the linear layers from our MCNN model classification head to these spline linear layers, retaining the convolutional backbone as before. Thus, our best point of comparison for KAN spline layers versus traditional MLP layers will be the MCNN.

F. Model Interpretation

For convolutional architectures (e.g. CNN, MCNN, CKAN, ResNet), we can leverage a popular interpretation technique Grad-Cam to interpret the model. Briefly, Grad-Cam works by taking the gradients (i.e. back-propagating) from the target class' node in the output layer back to the final convolutional layer, highlighting spatially relevant pixels of the image for triggering the output activation [9].

G. Future Explorations

To ensure the cleanest comparison across architectures (and deal with time and computation constraints), we opted against integrating some potentially beneficial model-agnostic techniques. We highlight them here as future paths of exploration:

- **Data Augmentation**: given the controlled environment of an MRI machine, most augmentations would likely not apply to our context without causing more harm than good for model training. However, some techniques such as jitter, horizontal flips, and slight blurring may help the model in deployment since patients may get restless during the long imaging process of an MRI and/or be place in either orientation.
- **Pre-Training**: across all model, pre-training via a related task (e.g. X-ray imaging diagnosis) or even a generic task (e.g. ImageNet) may benefit the model's downstream performance. Exploring certain pre-text (pseudo) tasks may also be a worthwhile branch for future work.
- **Region Proposal**: although RPNs have proven highly effective in reducing overhead computation and improving bounding box regression, their inclusion in our models seem to distract from the simpler classification task. That being said, if future work aimed to provide doctor's with a region of interest to inspect (i.e. a bounding box output), an RPN would certainly warrant further consideration.

- **U-Net (UNET)**: the U-Net is a convolutional architecture primarily developed for biological segmentation with limited annotated data points [7]. Although the underlying backbone could potentially be leveraged for classification, computational constraints became apparent during experimentation. For future work, we recommend utilizing this model for segmenting the images for further interpretability for the diagnosticians.

IV. Results

A. Computational Efficiency

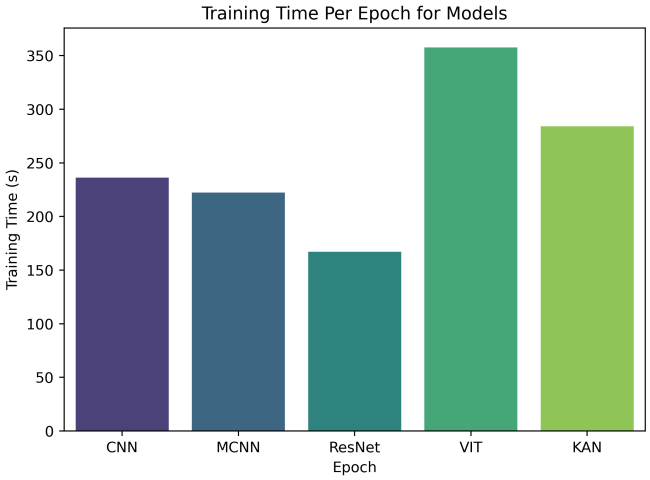

Fig. 2. A Comparison of training times per epoch across studied model architectures. Note that for all trials we use 1 (RTX 4080 Super), except for the CKAN which can only run on CPU.

Across all architectures, we find that training times are roughly similar, with the notable exception being the vision transformer model roughly 50% longer per epoch 2. This discrepancy can be attributed to the steeper computational cost required for the model compared to the others in the study given its larger architecture.

B. Performance

Across all model architectures implemented with our training regime, we see a similar reported accuracy I. The differentiator comes when inspecting the loss, i.e. how well does each model predict the likelihood of each severity between each case. The CKAN and CNN stand out as the best models by a margin of $\geq 0.7\%+$ accuracy and ≥ 0.017 weighted log loss, with MCNN close behind.

Surprisingly, the VIT failed to outperform even the more basic ResNet; comparing the $\mathcal{WLL}$ of each, we can conclude that the ResNet tended to be confidently wrong while the VIT was more uncertain during incorrect predictions–thus, a lower accuracy but also lower loss for the VIT compared to the ResNet.

TABLE I
Performance across Model Architectures

Architecture	Accuracy	Weighted Log Loss	Inference Time (ms)
CNN	**90.2**%	0.348	5.692
MCNN	89.5%	0.359	5.105
ResNet	89.5%	0.442	**4.179**
CKAN	**90.2**%	**0.342**	5.034
VIT	88.4%	0.395	8.596

We conjecture that the VIT and ResNet's lower than expected performance can be attributed to their larger architectures with a relatively smaller dataset. In future work, pretraining either model (especially ResNet) may yield much better convergence and overall performance.

Although not as essential in the context of medical diagnosis, the inference time of each model speaks to its potential for real-time deployment. A system which can make on-the-fly predictions during the imaging process presents great value for radiologists during time-critical work. All models meet the criteria for real-time deployment with most Magnetic Resonance Imaging machines taking at most 1 frame every $\approx$ 40ms–a framerate all architectures we've employed can exceed.

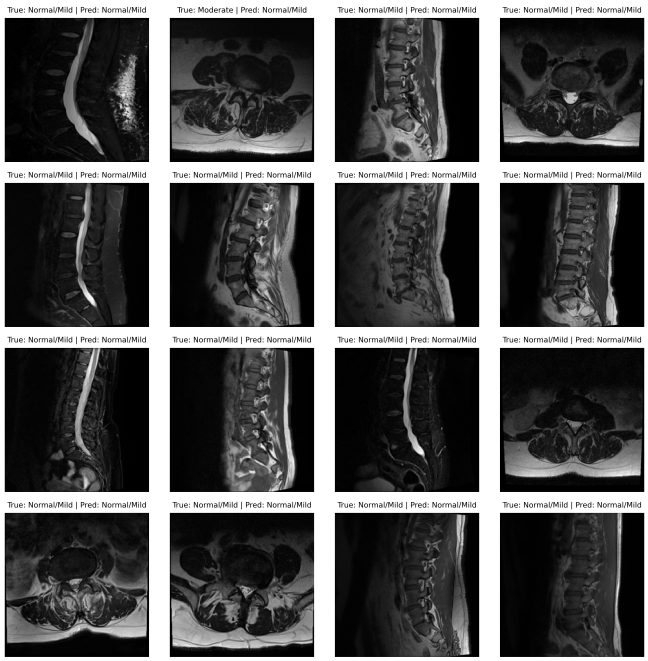

Fig. 3. Dataset Sample with CKAN Predictions

C. Model Interpretation

With our CNN-backbone models (CNN, MCNN, CKAN, ResNet), we applied Grad-Cam to visualize what powered the model's ultimate decision by projecting areas of high activation onto the original image. Such areas of high acti-

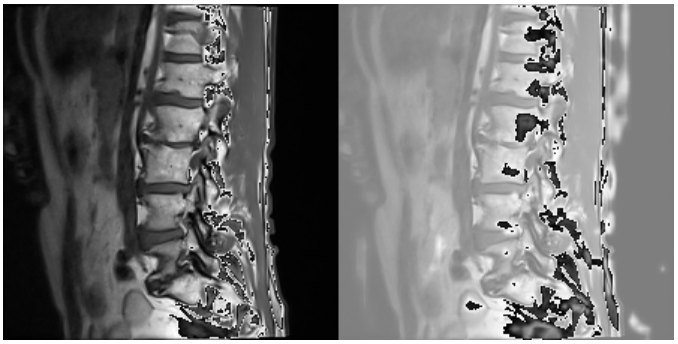

Fig. 4. CKAN Grad-Cam visualization of the last convolutional layer as heatmap of importance. The darker regions indicate elevated importance.

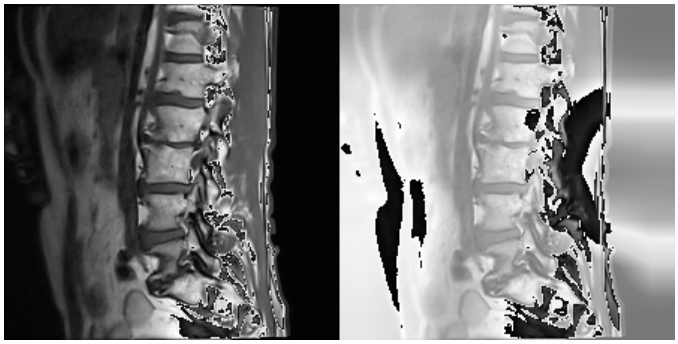

Fig. 6. ResNet Grad-Cam visualization of the last convolutional layer as heatmap of importance. The darker regions indicate more importance.

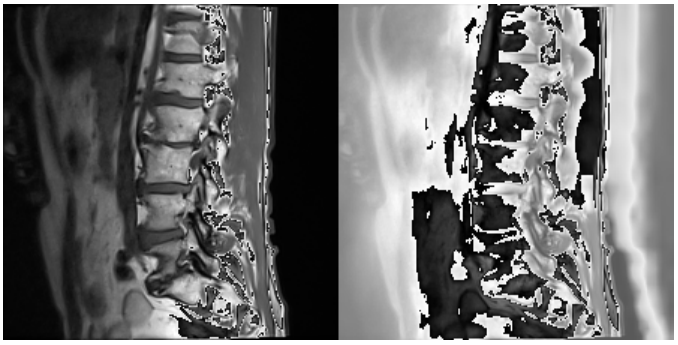

Fig. 5. CNN Grad-Cam visualization of the last convolutional layer as heatmap of importance. The darker regions indicate elevated importance.

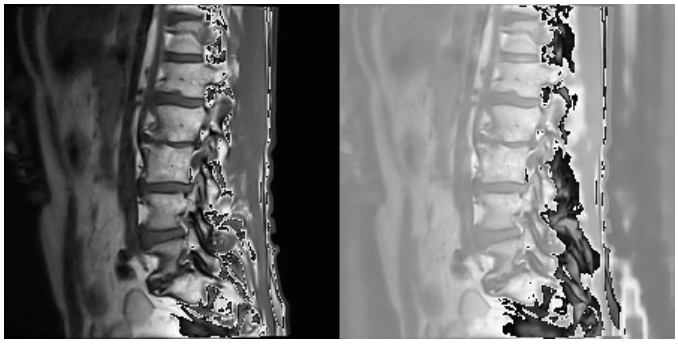

Fig. 7. MCNN Grad-Cam visualization of the last convolutional layer as heatmap of importance. The darker regions indicate more importance.

vation can be though of as a heatmap of importance, thus enabling a diagnostician to look for regions of interest. In the CKAN Grad-Cam 4, for example, notice how the activation is concentrated around the fluid build-up in the spine, thus indicating a more severe lumbar diagnosis.

In comparison, the CNN Grad-Cam 5 shows a much broader range of activations across the image, indicating a larger area of focus. Given our lack of expertise in diagnosing lumbar conditions, we can either interpret this larger region of interest as a more weighted, thorough decision. Equally likely, we can interpret the larger region as a model that hasn't learned the underlying patterns well enough to be specific. Given the downstream performance, we lean closer to the latter conclusion, although the difference is too marginal to conclude anything definitively.

ResNet's Grad-Cam 6 showcases a reason for its disappointing performance–mistakenly, it highlights the front of the abdomen as important in its final judgment of the output. We conjecture that it (a) hasn't yet learned to ignore it, i.e. more data and more epochs could help in future work, or (b) it has learned that posture of the abdomen could indicate severity. Again, lack of background limits our ability to conclude anything definitively.

We briefly mention MCNN's Grad-Cam 7 to point out its similarity to the CKAN's interpretation. It seems, however, that the spline linear layers do a better job of selecting out the regions of interest to care about.

V. Conclusion

In this paper, we've explored how baseline vision models (e.g. classic CNN), state-of-the-art vision models (ResNet, Vision Transformer), and novel neural network formulations (Kolmogorov-Arnold Networks) compare in the context of diagnosing the severity of lumbar conditions via MRI scans. In comparison to prior work, we exceeded the best result of $\mathcal{WLL} = 0.36$ with three alternative architectures (CKAN, CNN, MCNN) coupled with an optimized training, inferencing, and interpretation pipeline for real-time deployment.

Our comparison of across popular approaches of deep learning for computer vision reveals a few key lessons:

1) **KAN**: our results help establish the real-world efficacy of Kolmogorov-Arnold splines for replacing the role of activation functions in traditional multi-layer perceptrons.
2) **Data Intensity**: larger models like ResNet and VIT rely on large amounts of data to outperform competing architectures. In this application, the low amount of labeled data indicates (a) a need to pre-train models on pseudo tasks and (b) that smaller, more basic architectures are better suited.

3) **Interpretability & Inference**: however accurate a model is, in the context of medical diagnosis it is essential to remain interpretable to the diagnosticians. By creating a heatmap mask of importance, the smaller CNN-backbone architectures appear far more practical to real-world applications. A side benefit of high inferencing speeds also opens up more possibilities for deployment contexts (i.e. real-time inferencing).

For future work, we encourage researchers to focus on how pre-training each backbone on related (e.g. X-Ray imaging) or unrelated (e.g. ImageNet) datasets might lead to downstream performance boosts, especially for the vision transformer (VIT) and ResNet. Similarly, we encourage pseudo-tasks on this dataset or related (e.g. X-Ray imaging) for pre-training without labels.

Threats to validity primarily consist of the imbalance in labels contributing to a susceptibility of the models in real-world deployment. Although we leverage weighted log loss during training, we highlight this detail here for future work.

VI. Authors & Acknowledgment

- **Arjun Ashok**: Project lead. Built core model architecture & methods & pipeline, implemented CNN/MCNN/KAN/ResNet, researched architectures & underlying papers, and built model interpretation via GradCam. Wrote abstract, introduction, dataset, model architecture, parts of results, conclusion, and appendix.
- **Zhian Li**: Establish model architectures, designed and developed data loading pipeline, Implemented Vision Transformer Architecture, optimized pipeline for testing framework, trained model locally. Wrote part of dataset section, produced slides.
- **Ayush Tripathi**: Established model architectures, refined interface, implemented U-Net, designed and optimized pipeline for testing framework, built visualization and metrics capabilities, procured visuals within report, trained models locally. Wrote part of results, produced slides.

We'd also like to extend our appreciation to Professor Hamed Pirsiavash for an informative course and the TA's of ECS 174, Raymond Kang and Kossar Pourahmadi-Meibodi for their efforts in teaching.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[4] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[5] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.

[6] World Health Organization. Low back pain. *World Health Organization*, Jun 2023.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[8] Johannes Schmidt-Hieber. The kolmogorov-arnold representation theorem revisited, 2021.

[9] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Appendix

A. Computation

All compute was done on one of two machines:

1) Nvidia RTX 4080 Super, AMD 9900X
2) Nvidia RTX 4070 Super, AMD 9800X3D

We leveraged a CUDA-enabled PyTorch compilation for all models except CKAN–the CKAN library we've used is unable to run on GPUs yet, so we trained purely on CPU.

B. Hyperparameters

All models used the same set of initial hyperparameters:

- **Epochs for Early Stopping**: 5. All models employed early stopping for training to limit overfitting. This parameter controls how many epochs we would wait with no improvement in validation performance before making the decision to early stop.
- **Number of Epochs**: 30. We trained all models up to 30 epochs, although in most cases we needed less to converge before early stopping kicked in.
- **Batch Size**: 32
- **Learning Rate**: 1×10^{-4}
- **Momentum**: 0.7. We don't pass this parameter in since we leverage the Adam optimizer
- **Batch Size**: 32
- **Dropout Rate**: 0.3. Some models don't implement dropout.
- **Optimizer**: Adam
- **Loss Criterion**: Cross Entropy Loss
- **Classification Function**: Softmax. The alternative is to use sigmoid.

The vision transformer (VIT) has additional default parameters not mentioned here.