Checkpoint 2: Preliminary implementation and writeup

Anonymous submission

Abstract

This checkpoint submission provides an overview of the project, highlighting important literature and the proposed workflow of the implementation. Using LaTeX and following the AAAI-25 guidelines, this write-up aims to create a solid foundation for the final project. It includes literature reviews of the three papers, a clear method for the implementation of the project paper, and a discussion of the project proposal.

Literature Review

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Background (Mildenhall et al. 2021) addresses the challenge of synthesizing 3D views of a scene from a available set of 2D images. Then available methods for view synthesis relied only on geometric representations. Complex scenes and lighting variations were difficult to be captured using these methods. The NeRF method greatly impacts how we create images of 3D scenes in computer vision and graphics.

Problem Definition NeRF is a deep learning approach to view synthesis. View synthesis is the process of generating new images of a scene from different viewpoints based on existing images. It involves simulating how a scene would appear if seen from angles or positions that were not captured in the original photographs.

Summary of the Paper NeRF represents a scene as a continuous 3D field of color map and density, which allows for high-quality image generation from different viewpoints. The core idea is to use a simple, light-weight feed-forward neural network to model how light interacts with a scene and encode information about the color and density of points in a 3D space.

NeRF starts with a set of 2D images of a scene taken from different angles, along with the camera positions and camera angle of each image. For each image, rays are simulated as originating from the camera position and passing through each pixel in the image. Multiple points are sampled along each ray to gather information about the scene at different depths. For each sampled point along the rays 3D coordinates and viewing direction is simulated. The neural network takes in 3D coordinates and the viewing direction of the as inputs and trained to predict the RGB color values and density value at these points. Once the network is trained, new views of the scene are generated by sending rays from the camera position through the scene. Using a technique called volume rendering, the predicted the color and density values are converted to pixel values, representing a 2D image as though viewed from that direction. NeRF essentially simulates how light would pass through the scene

Related Works and Critique of the Paper NeRF builds upon several key concepts in 3D modeling and rendering. For example, the work by Curless and Levoy (Curless and Levoy 1996) had introduced a method for constructing complex 3D models from available images as early as 1996. Similarly, Niemeyer et al. (Niemeyer et al. 2020) proposed ”differentiable volumetric rendering” which provided a groundwork for neural networks to learn 3D representations. In addition, Isola et al. (Isola et al. 2017) introduced a technique for translating images from one domain to another using conditional adversarial networks. This idea of transforming images is similar to NeRF's idea of synthesizing new views of a scene from existing images. Lastly, the Sitzmann et al. (Sitzmann et al. 2019) paper presents a method for learning 3D features from images using a voxel-based representation. The term ”voxel” is a combination of ”volume” and ”pixel,” and it represents a value on a regular grid in three-dimensional space.

Building upon NeRF, Mip-NeRF (Barron et al. 2021) introduces a way to represent scenes at multiple scales. It captures fine details better than the original NeRF, making final output smoother and clearer. (Yu et al. 2021) presents a new data structure called Plenoctrees that allows for faster rendering of neural radiance fields. It organizes the voxel data in a way that allows quicker access and processing, which is very useful for interactive applications like video games or virtual reality. (Martin-Brualla et al. 2021) can generate 3D scenes from images taken in various conditions and perspectives without strict control over the shooting environment. This method is more versatile and applicable in real-world situations, where images may not be perfectly aligned or captured under ideal conditions.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Background Transformers, originally used for language processing, were adapted to work on images, but they had some challenges, especially with large images. (Liu et al. 2021) paper introduces a new model for computer vision tasks that improves on previous transformer-based approaches. Swin transformer architecture makes the transformers more efficient and effective for computer vision tasks like object detection, image classification, and segmentation.

Problem Definition When traditional transformers are applied to images, they treat the entire image as a sequence of pixels. This leads to a huge amount of computation, especially for high-resolution images. This makes them slow and difficult to use for large images or real-time tasks. Further, they break an image into patches and process them independently, but they don't allow enough interaction between these patches, making it harder for the model to understand relationships between distant parts of the image.

Summary of the Paper Unlike regular transformers that process images as a whole, the Swin Transformer breaks an image into small patches or windows, which are then processed in a hierarchical manner. Low-level layers handle small, detailed parts of the image, while higher-level layers combine these small parts to understand the bigger picture. Further, the model splits the image into small windows and processes them individually. To make sure that windows aren't isolated and can still interact, the model shifts these windows by a small amount at different stages. This allows the model to learn relationships between neighboring windows, ensuring that important information is shared across different parts of the image. This approach reduces the computation required, making the Swin Transformer faster and able to handle larger images while maintaining good performance.

Related Works and Critique of the Paper (Krizhevsky, Sutskever, and Hinton 2012) paper was a revolutionary work that introduced Convolutional Neural Networks (CNNs) and demonstrated how CNNs could perform extremely well on large-scale image classification tasks. The Swin Transformer builds on the success of CNNs by adopting a hierarchical structure similar to CNNs. Like in CNNs, Swin uses the idea of gradually reducing spatial dimensions, although it uses self-attention instead of convolution. The (Vaswani 2017) paper introduced the Transformer architecture, based on the self-attention mechanism. It was originally designed for natural language processing (NLP) and it demonstrated the power of self-attention for sequence modeling without using time series learning (like in Recurrent Neural Networks) or convolution (like in CNNs). The Swin Transformer is built on the fundamental concept of self-attention from this paper, but adapts it for vision tasks. The (Dosovitskiy 2020) paper introduced Vision Transformers (ViT), which applied the Transformer architecture directly to image patches (16x16 pixels) and showed that self-attention could outperform traditional CNNs on image recognition tasks. This is the main paper which the Swin Transformer builds on and solves some of its limitations, particularly in handling images of varying sizes and achieving high efficiency for dense prediction tasks (like object detection and segmentation). The hierarchical feature extraction in Swin allows for better scalability compared to ViT, which processes the entire image at a fixed resolution. The (Hu et al. 2019; Ramachandran et al. 2019) papers focused on capturing local relationships between pixels by introducing local relation layers. This enabled the model to focus on specific regions in an image. Swin's shifted window mechanism was likely inspired from these papers. This allowed Swin to maintain computational efficiency and still capture important local patterns, like a CNN.

The CSWin Transformer (Dong et al. 2022) introduces cross-shaped windows for self-attention. This allows the model to capture both horizontal and vertical dependencies across the image. Swin Transformer uses shifted square windows, which focus on local interactions, but CSWin enhances this by expanding the receptive field in a cross pattern. This allows for better handling of long-range dependencies, leading to improved accuracy in vision tasks. Swin-Unet (Cao et al. 2022) adapts the Swin Transformer for medical image segmentation. It combines the Swin architecture with the U-Net structure, which is commonly used in medical imaging. This makes Swin more suitable for dense pixel-level predictions, such as segmenting tumors or organs in medical images, where precise localization is critical. Video Swin Transformer (Liu et al. 2022) extends the Swin model to handle video data. It incorporates time related information alongside spatial data, allowing for efficient video processing. This also builds on Swin's hierarchical design but adapts it for the challenges of video sequences. SwinIR (Liang et al. 2021) uses the Swin Transformer architecture for image restoration tasks, like super-resolution, denoising, and artifact removal. By applying Swin's local attention mechanism, SwinIR achieves better performance in restoring high-quality images from low-quality inputs, particularly for tasks that require recovering fine details. Focal Self-Attention (Yang et al. 2021) introduces a way for the model to focus on both local and global interactions more efficiently. While Swin focuses on local self-attention using small windows, Focal Self-Attention dynamically adjusts its focus between local and global contexts. This makes it even more powerful in capturing large-scale relationships.

Graph Neural Networks for Channel Decoding - Project Paper

Background Wireless communication systems use codes (like LDPC and BCH codes) to protect data from errors during transmission. These codes are represented as graphs, and current methods (like belief propagation) are good but still have room for improvement, especially for shorter data blocks. The paper (Cammerer et al. 2022) discusses a method for improving error correction in wireless communication systems using Graph Neural Networks (GNNs). Error correction helps fix mistakes that occur when information is transmitted, and this method focuses on decoding the transmitted data efficiently.

Problem Definition The problem this paper addresses is how to improve the decoding of error-correcting codes in wireless communication systems. when information is sent over a network, errors can happen because of noise or interference. To fix these errors, special codes are added to the data, and the receiver has to decode these codes to correct any mistakes. Current decoding methods, such as belief propagation (BP) and Maximum Likelihood (ML), work well but still have limitations, especially when the message size is small or the code structure is complex. Existing decoding methods are not always efficient or accurate, especially for certain types of codes like BCH or LDPC codes. The authors propose using Graph Neural Networks (GNNs) to learn a better way to decode messages, aiming to make the decoding process more accurate and faster, even for different types of codes and message lengths.

Summary of the Paper The authors propose a new method that uses Graph Neural Networks (GNNs) to learn how to decode messages. GNNs are a type of neural network designed to work with data structured as graphs. Instead of manually coding the decoding process, the GNN learns the best way to pass messages across the graph to decode the data. The GNN allows information to flow between the nodes in the graph. Each node sends and receives messages based on its current state and the states of its connected nodes. After exchanging messages, the nodes update their states based on the received information. This process is repeated for a number of iterations, allowing the GNN to refine its understanding of the data. The GNN is trained using examples of transmitted and received data, learning how to correct errors. When decoding a message, the GNN takes in noisy received data and uses its learned message-passing rules to estimate the most likely original message. It can perform well even with fewer iterations compared to traditional methods, making it faster, more efficient and scalable.

The GNN-based method can scale well to larger problems and can be trained to perform better than current decoding methods, like belief propagation, with fewer iterations. It also works across different types of codes (e.g., BCH and LDPC) and is flexible for various communication scenarios. The paper compares the GNN decoder with traditional methods, showing that it performs better in certain cases, especially with fewer decoding steps.

Related Works and Critique of the Paper The (Wu et al. 2020) paper provides a broad overview of GNNs, discussing their architecture, applications, and advantages. It is a reference that can help understand the principles behind using GNNs for decoding tasks, as explored in the main paper. It shows how GNNs can effectively manage graph-based data, which is crucial for improving channel decoding technique. The (Forney 2001) is the foundational paper that introduces the concept of representing error-correcting codes using graphs. It explains how graphs can be utilized to understand coding structures, which is a core idea in the proposed paper. While the (Gilmer et al. 2017) paper focuses on quantum chemistry, it introduces the concept of message passing in GNNs, which is also utilized in the proposed paper. The idea of nodes exchanging information to improve predictions is directly applicable to the decoding process. The (Satorras and Welling 2021) tells how integrating neural networks can lead to better message-passing strategies and (Liao et al. 2023) uses GNNs for constructing polar codes and focuses on improving the decoding process.

The (Wu et al. 2023) paper proposes a method that uses shared parameters in the GNN for decoding. This means that instead of each node and edge in the graph needing its own set of parameters, the same parameters can be used across different parts of the network. This approach reduces the number of parameters needed, which makes the model lighter and easier to handle while still maintaining good performance. The authors show that this method achieves similar decoding accuracy with fewer resources compared to the proposed paper's approach. The (Varadarajulu et al. 2024) introduces pooling techniques within GNNs to enhance decoding performance. Pooling helps to simplify the graph by aggregating information from multiple nodes, which can lead to better performance and efficiency. The application of pooling techniques helps the model focus on the most relevant information, improving accuracy and decodinhg speed. The (Clausius et al. 2024) paper combines equalization and decoding into a single GNN framework. Equalization is the process of correcting signal distortions before decoding. By integrating these two processes, the proposed method can improve overall system performance and reduce complexity.

In all, the authors conduct thorough experiments and benchmarks against state-of-the-art decoding methods. They provide clear results that demonstrate the effectiveness of their GNN approach, particularly in terms of bit error rates (BER). In all the main ideas and the framework of GNNs are presented in a way that can be grasped by readers familiar with the field. The paper provides sufficient background information to help understand how GNNs can be applied to channel decoding. Importantly by making the code available on GitHub, the authors contribute to the research community, allowing others to replicate their work and build upon it.

However, the actual implementation of the GNN decoder is quite complex. This may deter some researchers or practitioners who want to apply the methodology but struggle with the intricacies of the code. The paper primarily focuses on theoretical performance and simulations. From a purely wireless communication perspective, there are gaps in discussing how well this method performs in real-world scenarios, such as varying noise conditions or different communication environments. Finally since the performance of GNNs can be highly sensitive to the choice of parameters (like learning rates and architecture), the paper could have provided more guidance on how to effectively tune these parameters for different applications.

Project Proposal and Implementation Details

The main goal is to model the Graph Neural Network (GNN) and recreate the graphs discussed in the paper. This will help us understand a new, unexplored neural network architecture and apply it to my main areas of study: wireless communications and signal processing. I can also as well explore the intersection of the fields with deep learning. I am currently exploring and learning about graph neural networks and channel decoding. While I haven't written any code yet, I have a good understanding of the concepts and the challenges involved. I feel prepared to reimplement the ideas from the paper.

The general high-level overview of the implementation idea is as follows :

1. Use essential libraries such as TensorFlow, NumPy, and Matplotlib for numerical computations, deep learning, and plotting. Using the necessary components from the Sionna library, which is specialized for forward error correction (FEC) codes. Sionna is a Python library designed by Nvidia for simulating and wireless communication systems. It focuses on deep learning applications in communications. It provides tools for tasks like modeling wireless channels, encoding, decoding, and evaluating the performance. Sionna only works on TensorFlow, while I am comfortable with PyTorch. So, this needs a little attention as well.
2. Configure TensorFlow to use a specific GPU on Google colab.
3. Set parameters for the LDPC (Low-Density Parity-Check) code, GNN architecture, training, and simulation. This includes defining the code length, number of iterations, hidden units, batch sizes, and learning rates.
4. Depending on the chosen code type, load the appropriate encoder and decoder. For 5G-LDPC, the code and its parameters are initialized using Sionna's components. Create a parity-check matrix, which is important for the GNN decoder.
5. Set up the GNN decoder with specific parameters (e.g., number of hidden units, layers, activation functions). This part is essential as it defines how the GNN will process messages during decoding.
6. Create an end-to-end model that combines the encoder and the GNN decoder. This model will be used to simulate the encoding and decoding processes during training and evaluation.
7. Include the option to train the GNN model with the defined parameters. If training is complex and time consuming, I will load precomputed weights made available by the authors for evaluation. The training process, like every neural network learning, involves adjusting weights to minimize errors in decoding.
8. Run simulations over a range of signal-to-noise ratios (SNR) to evaluate the performance of the GNN decoder compared to traditional methods (like belief propagation). Generate plots to visualize the bit error rates (BER) against different parameters.
9. Use Matplotlib to create plots of the results, such as comparing the performance of the GNN-based decoder against conventional decoding methods.

The expected challenges while implementing the paper are :

1. The algorithms, especially GNNs and error-correcting codes, can be complex to understand fully.
2. Translating the theoretical concepts into code can be challenging. Understanding how to structure the code, especially when working with GNNs will require additional effort. Also, I need to ensure that the data flows correctly through the model.
3. The method needs careful attention to the parameter settings for both the LDPC code and the GNN architecture.
4. Implementing error handling for GNN configuration, loading and training components.
5. Ensuring the models (encoder and decoder) are correctly initialized before use. This includes checking that the shapes of inputs and outputs match the expected dimensions.
6. Fine tuning of the hyperparameters and selecting right optimizer is a challenging task.
7. Optimizing simulation runs by managing memory and processing time, especially when evaluating multiple configurations or parameters. Using batch processing is one workaround.

The authors have shared their code on GitHub, and I'm using it to understand the key ideas of the paper. However, their implementation is quite complex, so I plan to focus on just the main ideas while re-implementing. I'm learning a lot, but it is pretty challenging.

References

Barron, J. T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; and Srinivasan, P. P. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5855–5864.

Cammerer, S.; Hoydis, J.; Aoudia, F. A.; and Keller, A. 2022. Graph neural networks for channel decoding. In *2022 IEEE Globecom Workshops (GC Wkshps)*, 486–491. IEEE.

Cao, H.; Wang, Y.; Chen, J.; Jiang, D.; Zhang, X.; Tian, Q.; and Wang, M. 2022. Swin-unet: Unet-like pure transformer for medical image segmentation. In *European conference on computer vision*, 205–218. Springer.

Clausius, J.; Geiselhart, M.; Tandler, D.; and ten Brink, S. 2024. Graph Neural Network-Based Joint Equalization and Decoding. In *2024 IEEE International Symposium on Information Theory (ISIT)*, 1203–1208. IEEE.

Curless, B.; and Levoy, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 303–312.

Dong, X.; Bao, J.; Chen, D.; Zhang, W.; Yu, N.; Yuan, L.; Chen, D.; and Guo, B. 2022. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12124–12134.

Dosovitskiy, A. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Forney, G. D. 2001. Codes on graphs: Normal realizations. *IEEE Transactions on Information Theory*, 47(2): 520–548.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Hu, H.; Zhang, Z.; Xie, Z.; and Lin, S. 2019. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, 3464–3473.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Liang, J.; Cao, J.; Sun, G.; Zhang, K.; Van Gool, L.; and Timofte, R. 2021. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1833–1844.

Liao, Y.; Hashemi, S. A.; Yang, H.; and Cioffi, J. M. 2023. Scalable polar code construction for successive cancellation list decoding: A graph neural network-based approach. *IEEE Transactions on Communications*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Liu, Z.; Ning, J.; Cao, Y.; Wei, Y.; Zhang, Z.; Lin, S.; and Hu, H. 2022. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3202–3211.

Martin-Brualla, R.; Radwan, N.; Sajjadi, M. S. M.; Barron, J. T.; Dosovitskiy, A.; and Duckworth, D. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7210–7219.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Niemeyer, M.; Mescheder, L.; Oechsle, M.; and Geiger, A. 2020. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3504–3515.

Ramachandran, P.; Parmar, N.; Vaswani, A.; Bello, I.; Levskaya, A.; and Shlens, J. 2019. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, 32.

Satorras, V. G.; and Welling, M. 2021. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, 685–693. PMLR.

Sitzmann, V.; Thies, J.; Heide, F.; Nießner, M.; Wetzstein, G.; and Zollhofer, M. 2019. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2437–2446.

Varadarajulu, S.; Baeza, V. M.; Querol, J.; Mendonca, M. O.; and Chatzinotas, S. 2024. Graph Neural Network Pooling for BCH Channel Decoding in Software Defined Satellites. In *2024 IEEE International Conference on Communications Workshops (ICC Workshops)*, 1328–1333. IEEE.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Wu, Q.; Ng, B. K.; Lam, C.-T.; Cen, X.; Liang, Y.; and Ma, Y. 2023. Shared Graph Neural Network for Channel Decoding. *Applied Sciences*, 13(23): 12657.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Yang, J.; Li, C.; Zhang, P.; Dai, X.; Xiao, B.; Yuan, L.; and Gao, J. 2021. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*.

Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; and Kanazawa, A. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5752–5761.