A REPORT ON

VISION TRANSFORMERS FOR USE IN MEDICAL IMAGING

By

NAME - GRANTH JAIN

BITS_ID - 2022A7PS0172P

AT

DEEP TEK MEDICAL IMAGING PVT. LTD.

A PRACTICE SCHOOL-1 STATION AT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to **Mr. Aalhad Patwardhan**, my project advisor, and **Mr. Tanveer Gupta** at Deep Tek, for giving me the opportunity to work in this esteemed organisation. Their invaluable guidance, unwavering support, and profound insights have significantly enriched my learning experience in this rapidly growing field. I am deeply grateful for their mentorship, which has been instrumental in shaping my understanding and skills in the realm of advanced AI and machine learning technologies.

I would also like to extend my sincere thanks to **Dr. Annapurna Gopal**, Dean of the Practice School Division at BITS Pilani, Pilani Campus. Her efficient organisation and coordination with the institution have been pivotal in the successful implementation of the Practice School-I program. Dr. Gopal's dedication to facilitating practical learning opportunities for students is truly commendable, and I am immensely thankful for her efforts in making this program a reality.

My heartfelt appreciation goes to **Mr. Bikram Paul**, my PS-I faculty in-charge. His guidance on the academic aspects of this project has been invaluable. Mr. Paul's continuous motivation and support have been a constant source of inspiration, helping me navigate through challenges and achieve my project goals. His commitment to student success and academic excellence is deeply appreciated. I would also like to acknowledge the support and encouragement of my colleagues and fellow interns at Deep Tek.

ABSTRACT

This report gives an idea of the tasks assigned by Deep Tek and explains their implementation across multiple projects. The following sections outline the main technologies involved, highlighting their advantages and drawbacks. The given project depicts the advantage of using Vision Transformers in the field of medical imaging. This project shows how Vision Transformers (ViTs) have advantages over other methods of classification, such as Convolutional Neural Networks (CNNs), for medical image analysis tasks. The report discusses the key architectural differences between ViTs and CNNs, and how the self-attention mechanism in ViTs allows them to capture long-range dependencies more effectively. This proves beneficial for medical imaging applications where subtle visual cues spread across the image can be crucial for accurate diagnosis. The report presents a comparative analysis of the performance of ViT-based models against CNN-based counterparts on a medical image classification benchmark. The results demonstrate the superior classification accuracy and robustness of the ViT approach, highlighting its potential to revolutionise computer-aided diagnosis in the healthcare domain.

TABLE OF CONTENTS

INTRODUCTION

The project report focuses on tasks assigned during the 3 weeks of PS-1 internship by DeepTek which is mainly theory of Convolutional Neural Networks(CNNs) and Vision Transformers in Computer Vision and some coding experience for the different neural networks. The internship developed a sense of understanding on why Vision Transformers (ViT) are more powerful when it comes to medical imaging (which are often complex and contain rich information). By introducing the concept of Vision Transformers DeepTek taught us how to improve accuracy and efficiency in medical imaging tasks.

The report aims to provide a comprehensive overview of the concepts, architectures, and practical implementation details covered during this internship. DeepTek helped us step by step to reach a stage where we can implement ViT for medical imaging all by ourselves. The internship was designed to equip participants with a deep understanding of the evolution of computer vision solutions, from the foundational principles of CNNs to the more recent advancements in Vision Transformers. By exploring these topics, the internship aimed to prepare participants for the challenges and opportunities in the field of medical image analysis, where the ability to effectively process and interpret complex visual data is of great significance.

Major Subparts of the report are as follows:

- How convolution works
- CNNs in Image Processing
- Popularly used CNN architectures
- Overview of solutions leading up to vision transformers
- Overview of Deep Learning NLP strategies, leading up to transformers
- Hands on with PyTorch
- Classification task on Kaggle Pneumothorax dataset using DenseNet121

PROJECT DESCRIPTION

To explain briefly, our task is to study the basics of Vision Transformers and compare performance of different architectures (eg. CNNS) and Vision Transformers in the field of medical imaging. We were given several materials by our project advisor on theories and basics of different architectures such as CNNs and also on Vision Transformers.The material included youtube videos, research papers and medium articles. Then we moved on to gain some practical experience using PyTorch using the tutorials provided by DeepTek.

TASKS AND PROJECT

1. How Convolution Works

The internship started with introducing us to the basics which were how convolution works using the Brandom Rohrer videos on youtube. Convolution is a fundamental operation in convolutional neural networks (CNNs) that detects specific features in images by sliding a kernel over the input data. The kernel, also known as a filter, is a small matrix of weights that slides over the input image, performing element-wise multiplication and summing the results. This process is repeated for every location the kernel slides over the input, creating a new output pixel. The output of the convolution operation is a new 2D array of pixel values, known as a feature map, which represents the presence and strength of the detected features. This process is repeated multiple times with different kernels, resulting in a hierarchical representation of the input image.

2. CNNs in Image Processing

Then we explored the working of CNNs, including their application in image processing. We also learnt about maxpool and all other concepts of CNNs. We understood how Pooling layers reduce spatial dimensions and increase robustness to small translations and also about Fully connected layers for classification and regression tasks. Convolutional Neural Networks (CNNs) are widely used in image processing for tasks such as image classification, object detection, and segmentation.

3. Popularly used CNN Architectures

DeepTek introduced us with a number of different and a variety of architectures used for image processing. Some of them are as follows:

- **ResNet**: Developed by Kaiming He et al. in 2016, this model introduced the concept of residual connections to improve the performance of deep neural networks. ResNets have achieved state-of-the-art performance on various computer vision tasks and have become one of the most widely used neural network architectures in deep learning.
- **LeNet**: LeNet is a convolutional neural network (CNN) architecture developed by Yann LeCun in the late 1990s, designed to recognize handwritten digits and later extended to various applications such as object recognition and natural language processing.
- **AlexNet**: AlexNet is a pioneering convolutional neural network (CNN) architecture designed by Alex Krizhevsky and his colleagues in 2012. It won the ImageNet Large Scale Visual Recognition Challenge in 2012, marking a significant breakthrough in the field of deep learning and computer vision.
- **Unet**: U-Net is a convolutional neural network architecture primarily used for image segmentation tasks, particularly in the field of medical imaging. It was introduced in 2015 and is characterised by its symmetric "U-shaped" architecture, with an encoder (convolutional) path and a decoder (transposed convolutional) path that allows for precise localization of features.

4. Overview of solutions leading up to Vision Transformers

The internship covered the evolution of computer vision solutions, leading up to the emergence of Vision Transformers (ViTs). Key concepts included:

- **Convolutional Neural Networks (CNNs)**: These models are particularly well-suited for image recognition and processing tasks, leveraging convolutional and pooling layers to extract features from images.
- **Transformers**: These models are based on the attention mechanism and have achieved state-of-the-art performance in various NLP tasks. The Vision Transformer (ViT) adapted this architecture for computer vision tasks. It breaks down an input image into a sequence of patches, flattens them into vectors, and processes them using a transformer encoder to capture long-range dependencies and global context within images

5. Overview of Deep Learning NLP strategies, leading up to transformers

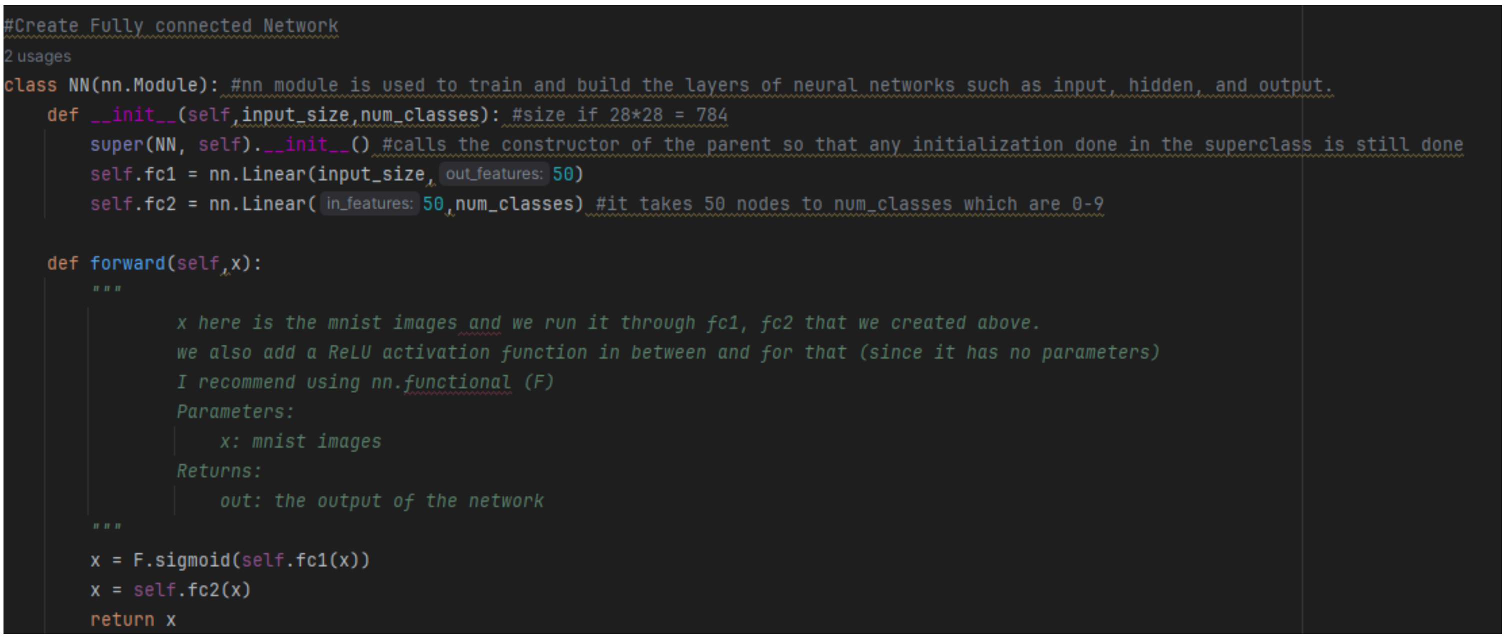
The internship covered recent developments in deep learning techniques for NLP, including:

- **Transformer-based models**: These models, such as BERT and GPT, have achieved remarkable performance across a range of NLP tasks.
- **Pretraining methods**: These methods, such as masked language modelling and next sentence prediction, are used to pretrain transformer-based models before fine-tuning for specific tasks

6. Hands on with PyTorch

To ensure all the students are on the same page in practical experience on PyTorch , our project advisor made a step by step plan to get confident with the coding part of the project. In this the first task was going through PyTorch basics using youtube playlist on PyTorch tutorials and the PyTorch documentation. These videos helped us understand how the theory we just learned is coded out. Key concepts covered in those videos were:

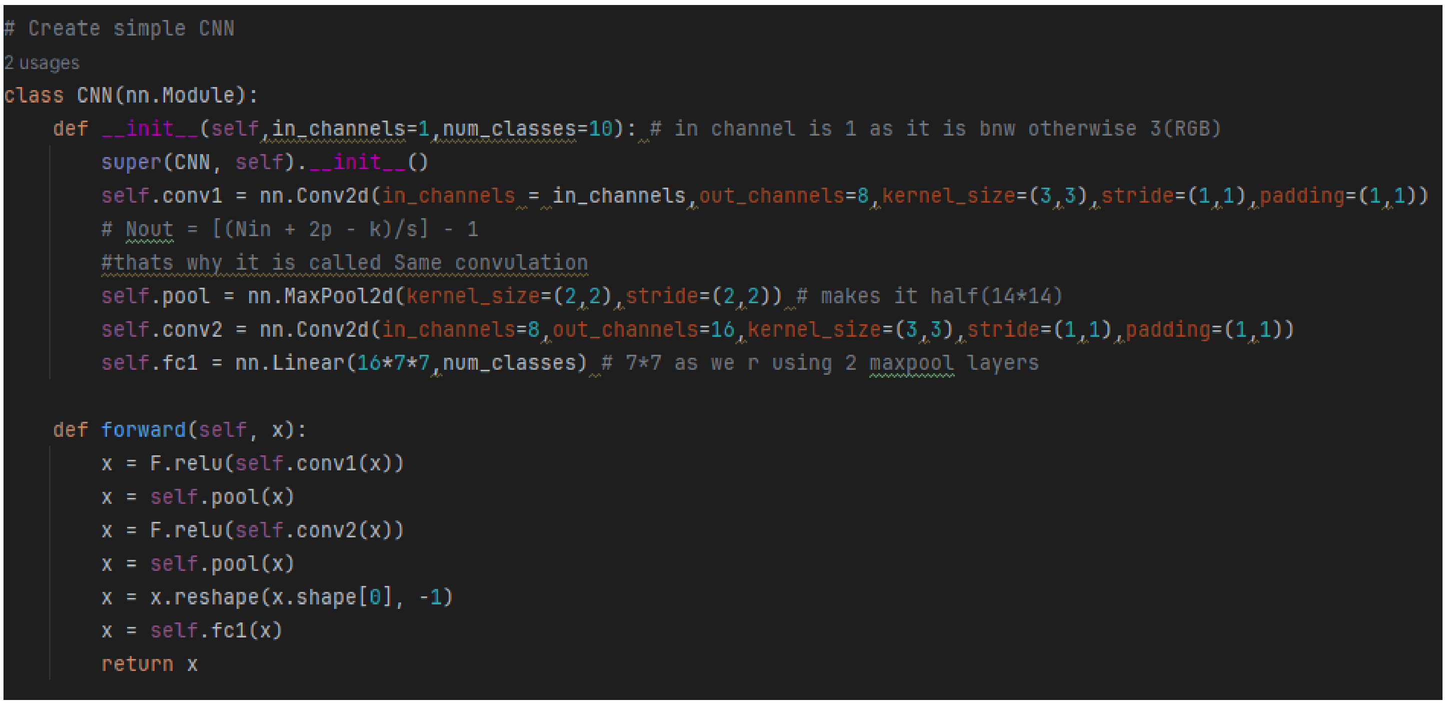
- **Introduction to PyTorch Tensors**: It helped us to get familiar with PyTorch syntax for mathematical operation, Indexing and Reshaping of Tensors.
- **PyTorch Neural Network example**: This example demonstrated the basic structure of a neural network in PyTorch. It made us familiar with how to load datasets, initialise a network, loss optimizers, Training loop and outputting the accuracy.



```python
#Create Fully connected Network
2 usages
class NN(nn.Module): #nn module is used to train and build the layers of neural networks such as input, hidden, and output.
    def __init__(self,input_size,num_classes): #size if 28*28 = 784
        super(NN, self).__init__() #calls the constructor of the parent so that any initialization done in the superclass is still done
        self.fc1 = nn.Linear(input_size, out_features: 50)
        self.fc2 = nn.Linear( in_features: 50,num_classes) #it takes 50 nodes to num_classes which are 0-9

    def forward(self,x):
        """
                x here is the mnist images and we run it through fc1, fc2 that we created above.
                we also add a ReLU activation function in between and for that (since it has no parameters)
                I recommend using nn.functional (F)
                Parameters:
                    x: mnist images
                Returns:
                    out: the output of the network
        """
        x = F.sigmoid(self.fc1(x))
        x = self.fc2(x)
        return x
```

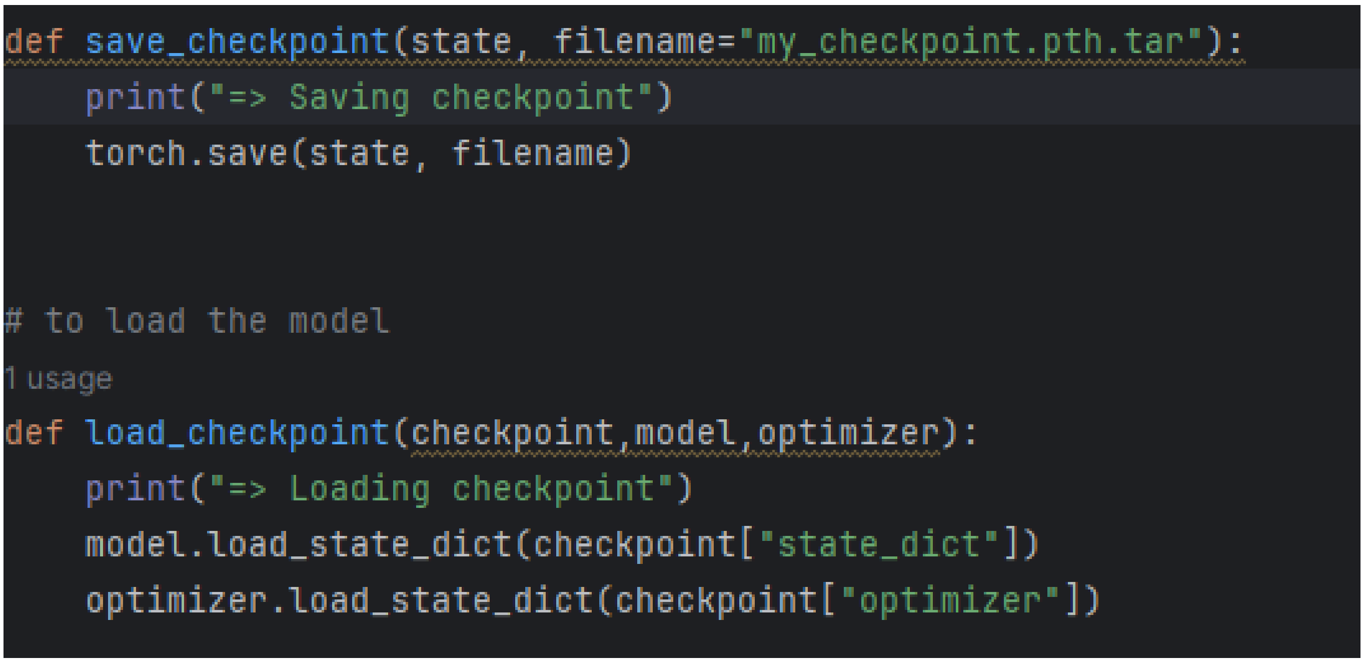
- **PyTorch CNN example**: This example showed how to implement a CNN in PyTorch, including convolutional and pooling layers. We used MaxPool2d, Conv2d and Linear.



```python
# Create simple CNN
2 usages
class CNN(nn.Module):
    def __init__(self,in_channels=1,num_classes=10): # in channel is 1 as it is bnw otherwise 3(RGB)
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels = in_channels,out_channels=8,kernel_size=(3,3),stride=(1,1),padding=(1,1))
        # Nout = [(Nin + 2p - k)/s] - 1
        #thats why it is called Same convulation
        self.pool = nn.MaxPool2d(kernel_size=(2,2),stride=(2,2)) # makes it half(14*14)
        self.conv2 = nn.Conv2d(in_channels=8,out_channels=16,kernel_size=(3,3),stride=(1,1),padding=(1,1))
        self.fc1 = nn.Linear(16*7*7,num_classes) # 7*7 as we r using 2 maxpool layers

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = x.reshape(x.shape[0], -1)
        x = self.fc1(x)
        return x
```

- **Saving and Loading Models in PyTorch**: This topic covered how to save and load trained models in PyTorch using checkpoints.



```
def save_checkpoint(state, filename="my_checkpoint.pth.tar"):
    print("=> Saving checkpoint")
    torch.save(state, filename)


# to load the model
1 usage
def load_checkpoint(checkpoint,model,optimizer):
    print("=> Loading checkpoint")
    model.load_state_dict(checkpoint["state_dict"])
    optimizer.load_state_dict(checkpoint["optimizer"])
```

The ResNet implementation from scratch was done by following the architecture and intuition behind how ResNet works. The key concept is the use of residual connections to solve the vanishing gradient problem, which arises when deep neural networks start to lose their generalisation capability due to the shrinking of gradients from the loss function. The architecture consists of a series of residual blocks, each containing multiple layers with skip connections that allow gradients to flow through the network more efficiently. The implementation was done using PyTorch, with the _make_layer function creating the layers one by one along with the Residual Block. The final model was trained on the CIFAR10 dataset, and the trained model performed well on the test dataset with 75% accuracy

8. Classification task on Kaggle Pneumothorax dataset using DenseNet121

This was the major task given to us during the third week of the internship program. The task was to classify the lung X-Ray images from the Pneumothorax dataset publicly available in Kaggle. The task required us to use the DenseNet121 model to train on the dataset and by adding a fully connected layer at the end to classify the output in two categories, calculating the loss and then updating the weights and biases of the model correspondingly. This taught us how to implement the models in

PyTorch and how exactly Adam optimizer and CrossEntropyLoss works. Along with that we learnt how to load the dataset by using the DataLoader which is already downloaded in your system. We were told to use DenseNet121 as it is a light weighted model. The output was a CSV file with 3 columns: image_path, probability, Label.

EXPECTED TIMELINE

The timeline is provided by our project advisor **Mr. Aalhad Patwardhan.**

- **For the first 2.5 weeks**: To gain an understanding of how CNNs work and a historical perspective on attention and specifically ViT, after which we will train CNN models for the classification task. Implementing training loop, along with a checkpointing mechanism.
- **Next 2 weeks**: To gain an understanding of classification metrics mentioned by sir and also implement and report the same.
- **Next 1.5 - 2 weeks**: Proceed with the segmentation task.
- **Remaining weeks**: Working with Transformers.

CONCLUSION

This project report provides a comprehensive overview of the key topics and tasks covered during the 2 month long learning internship. The report aims to provide a thorough understanding of the concepts, architectures, and practical implementation details in CNNs and Vision Transformers for computer vision tasks. The hands-on experience with PyTorch and the final task involving lung X-ray classification using DenseNet121 demonstrate the practical application of these concepts and architectures. Throughout the internship, participants gained a deep understanding of the evolution of computer vision solutions, from the foundational principles of CNNs to the more recent advancements in Vision Transformers. The report highlights the key takeaways from each section, including the mathematical and theoretical underpinnings of the various architectures, as well as the practical applications of these techniques. The hands-on experience with PyTorch allowed participants to implement and fine-tune their own models, gaining practical experience in the development and deployment of deep learning models. The final task involving lung X-ray classification using DenseNet121 served as a culmination of the internship, demonstrating the ability to effectively process and interpret complex visual data, a crucial skill in the field of medical image analysis.

TOOLS USED

- **PyCharm**: An environment for using PyTorch to train models.
- **YouTube**: For all the theory related videos provided by project advisor. It also helped in PyTorch tutorials and clearing doubts if any.
- **Kaggle**: This consisted of all the datasets which we downloaded for training our model which could be downloaded for free.

REFERENCES

1. Goldblum, M., Souri, H., Ni, R., Shu, M., Prabhu, V. U., Somepalli, G., ... & Goldstein, T. (2023). Battle of the Backbones: A Large-Scale Comparison of Pretrained Models across Computer Vision Tasks. arXiv preprint arXiv:2310.19909.
2. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Deep Learning for Biological Image Segmentation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2452-2460).
3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
5. https://github.com/MorvanZhou/PyTorch-Tutorial
6. https://www.kaggle.com/datasets/vbookshelf/pneumothorax-chest-xray-images-and-masks/data
7. https://jalammar.github.io/illustrated-transformer/
8. https://hackernoon.com/neural-network-layers-all-you-need-is-an-inside-comprehensive-overview
9. https://www.youtube.com/playlist?list=PLhhyoLH6IjfxeoooqP9rhU3HJIAVAJ3Vz
10. https://www.youtube.com/watch?v=B-M5q51U8SM&list=PLVZqlMpoM6kanmbatydXVhZpu3fkaTcbJ
11. https://www.youtube.com/watch?v=vsqKGZT8Qn8