Image Classification on Chest X-Ray Images

Vernon Toh[*], **Mohammad Alif Daffa**[*], **Hung Chia-Yu**[*]
Information Systems Technology and Design
Singapore University of Technology and Design
`{vernon_toh, mohammad_alif, chiayu_hung}@mymail.sutd.edu.sg`

Abstract

Accurate and timely detection of diseases is essential for effective medical treatment. X-ray imaging is typically used for detecting various diseases, but interpreting these images can be challenging for healthcare professionals. In this project, we experimented with various models in predicting which of 14 diseases (Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening and Hernia) are detected in a patient, based on x-ray images from the ChestX-ray14 Dataset. We evaluated our models against ChexNet, which is a 121-layer convolutional neural network trained on ChestX-ray14 using AUROC performance measure.

1 Introduction

The chest X-ray is one of the most commonly accessible radiological examinations for screening and diagnosis of many lung diseases. In this project we use the ChestX-ray14 medical imaging dataset (Wang et al., 2017) which comprises 112,120 frontal-view X-ray images of 30,805 (collected from the year of 1992 to 2015) unique patients with the text-mined fourteen common disease labels, mined from the text radiological reports via NLP techniques. Figure 1. shows some examples of X-ray images that can be found in the dataset.

We explored various popular CNN architectures including ResNet, DenseNet, Unet and Vision Transformer (ViT) (Dosovitskiy et al., 2020) which is the first computer vision model to rely exclusively on the Transformer architecture that is comparable to other state-of-the-art CNN architectures.

[*]Equal contribution towards report writing, implementation of models, and coming up with ideas for the project.

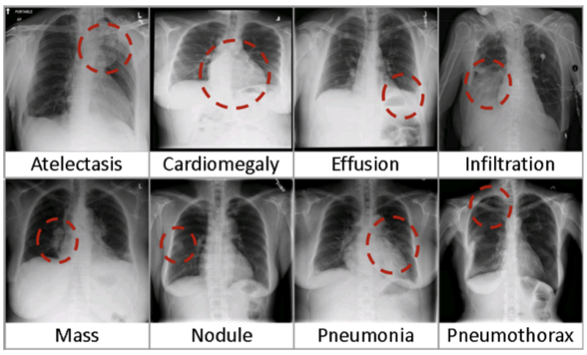

Figure 1: Eight common thoracic diseases observed in chest X-rays. Taken from (Wang et al., 2017).

2 Models

2.1 CNN with residual connections

Our first implementation was a standard convolution neural network comprising 6 identical blocks. Each block consist of a 3x3 convolution with same padding, 2D batch normalization layer, ReLU activation function and 2x2 max pooling.A skip connection adds the activation from the start of the block to the output activation after the ReLu layer. This block is repeated 6 times, followed by 2 fully connected layers with ReLu activation again.

2.2 ResNet

ResNet, short for Residual Network, introduces the use of residual or skip connections with the aim of alleviating the vanishing gradient problem (He et al., 2015). These skip connections work by allowing the output of a layer to skip one or more layers and be added to the output of a following layer. These connections provide a shortcut for the gradient to flow during backpropagation, effectively addressing the vanishing gradient problem and allowing the training of much deeper networks than previously possible. We implemented a smaller version of the ResNet, comprising of an initial convolutional layer with max pooling, batchnorm and ReLU, followed by 4 layers containing 2 residual blocks each, where each residual block comprises of 2 convolutional layers and a skip connection.

2.3 DenseNet

Inspired by the success of the DenseNet architecture in the ChexNet paper, we implemented and validated the performance of our own DenseNet-based model. Similarly to the ResNet, the DenseNet architecture also leverages the benefits of skip-connections for mitigating vanishing gradients (Huang et al., 2018). However, in the case of a DenseNet architecture, each convolutional layer receives feature maps from every preceding layer in the same 'block'. This dense connectivity provides more direct paths for gradients, resulting in better mitigation of vanishing gradients and more effective feature reuse, allowing the model to perform better with fewer parameters. With some tuning, we found our best performing DenseNet model to comprise of 4 dense blocks, each containing 4, 8, 16 and 12 dense layers respectively, where each dense layer contains 2 convolutional layers each.

2.4 Unet

Unet was shown to be one of the state in the art models in medical image segmentation, utilizing an encoder-decoder architecture (Ronneberger et al., 2015).The encoder is responsible to extracting relevant features from the input image and the decoder takes the extracted features and reconstructs a segmentation mask. Since we are only interested in the extracted features, we build a UnetEncoderClassifier that utilizes Unet's encoder as feature extractor and perform classification task on it.

2.5 Vision Transformer with Convolutions

The self-attention mechanism has been the primary focus of Transformers for capturing global dependencies in natural language modeling, leading to their widespread adoption. Additionally, the Transformer architecture has recently gained attention as a promising contender to convolutional neural networks (CNNs) for visual recognition tasks such as classification. The Vision Transformer (ViT) (Dosovitskiy et al., 2020) was the first to demonstrate that a purely Transformer-based architecture can achieve state-of-the-art results in image classification.

2.5.1 CvT: Introducing Convolutions to Vision Transformers

Our implementation is inspired from CvT (Wu et al., 2021) which introduces convolutions to vision transformers. They introduced convolutions to two primary parts of the vision Transformer: first, Convolutional Projection for the attention operation, and second, a multi-stage structure to enable varied resolution of 2D re-shaped token maps, similar to CNNs. CvT model architecture can be seen in Figure 1.

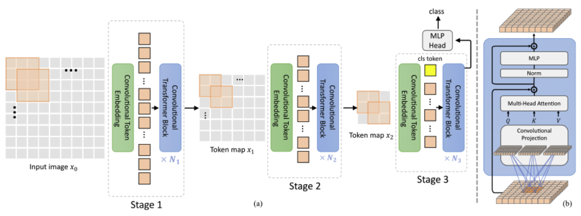

Figure 2: The pipeline of the proposed CvT architecture. (a) Overall architecture, showing the hierarchical multi-stage structure facilitated by the Convolutional Token Embedding layer. (b) Details of the Convolutional Transformer Block, which contains the convolution projection as the first layer. Taken from (Wu et al., 2021).

2.5.2 Our Implementation: XvT

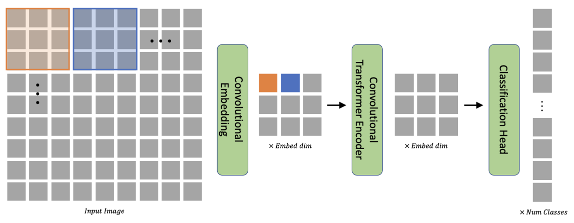

Figure 3: Our implemented model architecture. Convolutional Transformer blocks follows roughly the same as CvT.

The main difference between our implementation and CvT is the convolution embedding where our kernel size matches the kernel stride and we do not have the hierarchical multi-stage structure. The idea is to create a lightweight model that is capable of running on our laptops.

Convolutional Embedding

The convolution operation aims to model local spatial contexts into each token. Given a 2D image $x_0 \in \mathbb{R}^{H_0 \times W_0 \times C_0}$, we do a 2D convolution operation of kernel size $s \times s$, stride s. The new token map has height and width

$$H_t = \lfloor \frac{H_0 - s}{s} + 1 \rfloor, W_t = \lfloor \frac{W_0 - s}{s} + 1 \rfloor \quad (1)$$

and channel C_t. It is then flattened into size $H_t W_t \times C_t$ and normalized by layer normalization in the convolutional transformer encoder. By having our kernel size to match the kernel stride, we are having a non-overlapping token embedding as compared to the original CvT where their token embedding is overlapping. The convolutional embedding layer allows us to adjust the token feature dimension and the number of tokens before feeding the embeddings into the convolutional transformer encoder.

Convolutional Transformer Encoder

Our convolutional transformer encoder is similar to the works of CvT where they implemented convolutional projection layer to achieve additional modeling of spatial context. The convolutional projection layer is implemented using depth-wise 2D convolutional layer with kernel size s. Figure 3 shows a good visualization of what convolutional projection is doing. After the convolutional projection, the projected query, key, value tokens are then flattened into 1D and feed into the multi-head attention to calculate the attention scores.

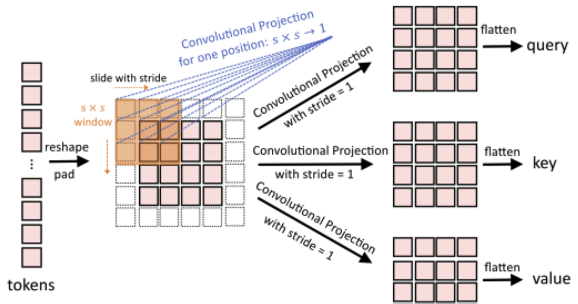

Figure 4: Convolution Projection layer. Taken from (Wu et al., 2021).

3 Experimental Procedure

3.1 Validation Split

For training and evaluating our models, we use the train-valid-test split published by (Rajpurkar et al., 2017). The training set is used for training the model. The validation set is used to compute the validation loss of the model after each epoch, and a plot of the validation loss curve is used to select the model checkpoint which was best able to generalise. The performance of this instance of the model is then computed on the test set, and is used for comparisons against the baseline developed by (Rajpurkar et al., 2017) as well as other model architectures we experimented with.

3.2 Data Preprocessing and Augmentation

Similarly to (Rajpurkar et al., 2017), our preprocessing steps include normalizing images based on the mean and standard deviation of the ImageNet dataset. However, we resize the images to a resolution of 224x224 using the OpenCV library prior to training, as resizing the images while iterating through the dataloader proved to be computationally costly. Additionally, we also omitted the use of random horizontal flipping as a data augmentation technique due to the asymmetrical nature of human thoracic anatomy. Instead, we experimented with other techniques such as random cropping and slight random rotation, but this resulted in poorer ability of the model to converge, and we therefore chose to omit the data augmentation step altogether.

3.3 Training

Due to the significant class imbalance found in our dataset, our models are trained to minimize the weighted binary cross entropy loss across the 14 classes. The weight assigned to each class is computed as the number of negative samples of that class divided by the number of positive samples.

3.3.1 General Hyperparameters

Adam optimizer is used, with a learning rate of 0.0001, and the default value for other hyperparameters. We also added a scheduler which reduces the learning rate upon hitting a plateau on validation loss, signified by a stagnating validation loss for 3 epochs in a row.

3.3.2 XvT Hyperparameters and Setup

AdamW optimizer is used with the weight decay of 0.05 for XvT. We train XvT with an initial learning rate of 0.00025 and a batch size of 128 for 40 epochs, with a cosine learning rate decay scheduler.

For our XvT model, we set the number of attention heads to be 3, number of blocks to be 6, patch size and stride size to be 14, embedding dimension to be 192, and convolution projection layer to have kernel size of 3 and finally a dropout rate of 0.3.

Pathology	CheXNet	CNN	ResNet	DenseNet	UnetEncoder	XvT
Atelectasis	0.8217	0.6790	**0.7330**	0.7139	0.7193	0.6866
Cardiomegaly	0.9054	0.7221	**0.8450**	0.8412	0.8018	0.7727
Effusion	0.8821	0.7478	**0.8176**	0.7994	0.7758	0.7439
Infiltration	0.7082	0.6376	**0.6689**	0.6600	0.6407	0.6336
Mass	0.8491	0.5716	**0.6874**	0.6766	0.6240	0.6132
Nodule	0.7740	0.5431	**0.6407**	0.6392	0.5759	0.5517
Pneumonia	0.7730	0.6474	**0.7111**	0.6907	0.6281	0.6568
Pneumothorax	0.8661	0.6238	**0.7782**	0.7676	0.7110	0.6753
Consolidation	0.8026	0.7348	**0.7748**	0.7644	0.7318	0.7382
Edema	0.8897	0.8279	**0.8644**	0.8539	0.8398	0.8130
Emphysema	0.9113	0.6120	**0.7783**	0.7400	0.6929	0.6443
Fibrosis	0.8220	0.6831	**0.7502**	0.7263	0.6974	0.6506
Pleural Thickening	0.7742	0.6072	**0.6961**	0.6838	0.6412	0.6073
Hernia	0.8867	0.8372	**0.8761**	0.8547	0.7641	0.6970
Average AUROC	0.833	0.677	**0.759**	0.744	0.703	0.677
Trainable params	7,051,854	19,026	6,299,342	2,136,302	21,129,038	2,823,758

Table 1: AUROC scores. Bolded values are the highest AUROC score for each pathology in our experiments.

3.4 Hyperparameter Tuning

Taking into account the long duration of training a model on this dataset, automated hyperparameter tuning techniques such as Grid Search were infeasible. Instead, we adopted a heuristic-based approach for deciding whether to increase or decrease the size and complexity of our models. This method revolved around detecting signs of overfitting or underfitting, and modifying the size and complexity of our model accordingly.

While an overly complex can be determined by briefly observing the validation loss curve for signs of overfitting, it is not as straightforward to distinguish whether a model is underfitting due to a lack of complexity. One observation we made was that in some experiments, the performance of the model would begin to compromise its performance on one of the classes in order to optimize its total loss. We interpreted this as a sign that a model lacked the capacity to capture relevant features for all the classes, and would require more trainable parameters.

3.5 Evaluation

The evaluation metric we chose is average AUROC which is the same metric as CheXNet. Average AUROC is computed by taking the mean of AUROC score of each class of disease. Other metrics such as accuracy or f1 score is not an appropriate metric due to the extreme imbalance in the dataset, where certain classes only appear$<$ 0.1% of the entire training set. It is also difficult to perform sampling techniques to address the data imbalance issues due to the classification problem being multi-label in nature.Common techniques like over-sampling might deteriorate the data imbalance issue. Hence, we decided to follow the experiment setting as described in CheXNet.

4 Results

We assess the performance of all of our implemented models on the test set that was used in CheXNet for their multi-label classification evaluation. Table 1 summarizes the AUROC scores of each of our individual. We observed that our of all the models, our ResNet implementation gave us the best AUROC scores for all 14 classes and has an average AUROC score of 0.759. If we compare our ResNet with our benchmark CheXNet, the difference in average AUROC score is 0.074. Our second best performing model is the DenseNet with a AUROC score of 0.744 which is only 0.015 away from the ResNet. And DenseNet has 3 times lesser trainable parameters compared to ResNet

If we take a look at the other models we have implemented with such as the UnetEncoder, we realized that having that big of a model with 21 million trainable parameters does not necessary provide us with a better AUROC score. We had high hopes for the results from our XvT, however the results that we got were not what we expected. Our XvT with 2.8 million trainable parameters produced the same average AUROC score as the CNN architecture with 19k trainable parameters.

One potential reason why our XvT might not have done as well as we expected since we took some of the ideas from CvT is because maybe the hierarchical multi-stage pipeline is what drives CvT to perform very well. Furthermore, we scaled down the model that was implemented in CvT by quite a fair bit for it to be able to fit inside our laptop for training. By doing so, it might have resulted in a model architecture that does not perform very well in our use case.

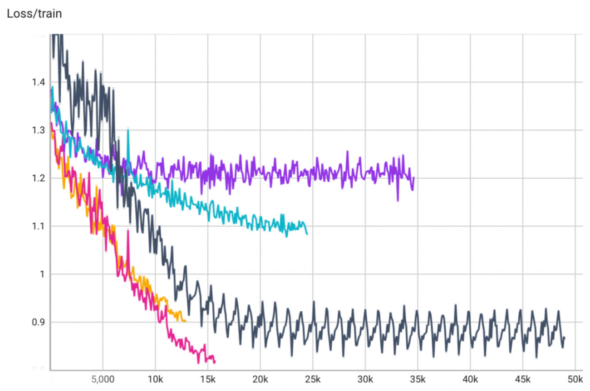

Figure 5: All training losses from our experiments. x-axis: Number of training steps Gray-Unet, Blue-XVT, Pink-Resnet, Yellow-DenseNet, Purple-CNN with residual.

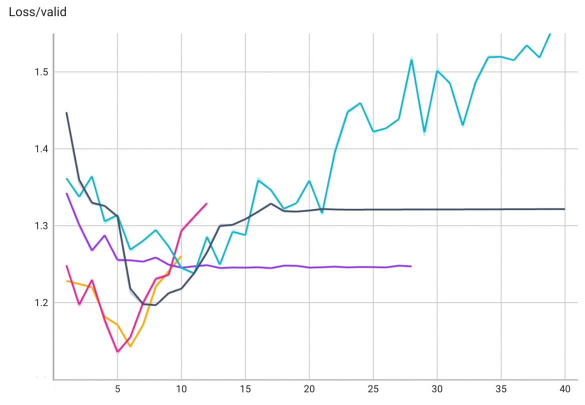

Figure 6: All validation losses from our experiments. x-axis: Number of epochs. Gray-Unet, Blue-XVT, Pink-Resnet, Yellow-DenseNet, Purple-CNN with residual.

We can see that most models show signs of overfitting after the 5th epoch. However we can see that our CNN with residual connection validation loss plateau at around loss value of 1.25. The results shown in table 1 are AUROC scores of the checkpoint with the lowest validation loss in figure 6.

5 Conclusion

We explored 5 different models of different sizes and architecture on the ChestX-ray14 dataset and achieved a 0.759 average AUROC with a ResNet architecture and uses slightly lesser amount of parameters than CheXNet. Furthermore, we did not perform any data augmentation as well as pre-training whereas ChexNet is pretrained with ImageNet dataset. We believe that it is possible to improve our result with pre-training. We also learned that using weighted binary crossentropy loss is an effective technique for improving the performance multi-label classification on an imbalanced dataset.

References

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Densely connected convolutional networks.

Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. 2017. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. 2017. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31.