Lung Disease Prediction from X-ray Images with Vision Transformer

Hieu Nguyen Minh[†]

Abstract—**Medical image classification has been receiving growing attention from the medical industry due to its practical application in the healthcare system. Traditional methods use CNN-based models to perform the task, and recent methods leverage the attention mechanism to create the Vision Transformer for image classification. In this project, we build a Vision Transformer that combines CNNs and the Single-head Self-attention to classify lung-diseases using X-ray images. The results on MedMNIST dataset show that, compared to other baseline models, our Vision Transformer achieve both a good performance (second best) and a high throughput, which is an excellent balance of complexity-inference speed.**

Index Terms—**CNN, Vision Transformer, Medical image classification, self-attention**

I. Introduction

Modern healthcare sector utilizes advancements in artificial intelligence to help processing medical data, thus getting more accurate results in a faster way. A typical task in healthcare is medical image classification, which analyzes medical images to indicate associated diseases, for example detecting lung cancer type based on X-ray images. Recent publications of medical dataset [1], [2], [3], [4] and advancements in computer vision models have led to increasing research in medical image classification [5], [6], [7], [8].

Convolutional Neural Networks (CNNs) are the most widely used models in image classification due to their effectiveness in exploiting both low and high-level spatial features in images [9], [10]. However, one weakness of CNNs is the limitation in capturing long-range dependencies in visual data [11]. To address this limitation, Vision Transformer (ViT) architectures [12], [13] have been developed to model long-range dependencies across visual domains by employing self-attention mechanisms [14], which was originally proposed in sequence-to-sequence models for natural language processing. Specifically, ViTs view images as sequences of smaller patches, which can be done by splitting images into patches. Each patch is then considered a word or token and projected to a feature space, and fed into the Transformer for training.

ViTs have achieved remarkable results in various computer vision tasks due to their strong modeling capabilities [15], [16]. Compared to CNNs, ViTs are particularly effective at capturing long-range dependencies and can scale efficiently with large datasets and numerous model parameters. However, standard ViTs lack inductive bias, making them heavily reliant on extensive training data. Hybrid approaches integrate ViTs with CNNs to get the advantages from both models, and develop more computationally efficient attention mechanism. In this project, we build a Vision Transformer that includes CNN layers and the single-head attention, which reduces computational resource but maintain a comparable performance. We compare our model to popular architectures, including VGG, ResNet, and the recent proposed MedViT [8] in terms of accuracy, memory taken, and throughput.

This report is structured as follows. Section II outlines widely used approaches in medical image classification, and Section III give an overview of our processing pipeline. Section IV describe the dataset while Section V presents models in details. The performance evaluation and visualizations are provided in Section VI, and Section VII is our conclusions.

II. Related Works

Throughout the last decade, CNNs have made significant contributions to various computer vision applications. VGG [17] built a deep CNN comprised of 16-19 layers for image classification and achieved the SOTA results at that time. ResNet [9] introduced residual connections to address the vanishing gradient issue, enabling deeper networks to capture high-level features for image classification. MobileNets [18] improve CNN efficiency on mobile applications through pointwise and depthwise separable convolutions. DenseNet [19] enhances feature propagation by using skip connections between layers and replacing summation with concatenation to create dense feature maps.

Recent Vision Transformer (ViT) [12] was developed based on the original Transformer architecture [14] and has demonstrated that a pure Transformer-based approach could achieve competitive results in image classification. ViT segments an image into patches (tokens) and applies Transformer layers to model global relationships among them. Research on architectural design [20] incorporates convolution and attention for processing low-resolution, high-level features, achieving strong performance without relying on complex operations. Meanwhile, efficient multi-head self-attention (MHSA) techniques lower the computational cost of attention through sparse attention mechanisms [21], [22] or low-rank approximations [23], [24]. Another remarkable study is the Single-Head Vision Transformer [25], which demonstrates that there is spatial redundancy in the early stages of standard ViT, thus employing a three-stage architecture and the single-head attention to eliminate the computational redundancy derived from multi-head mechanism.

[†]Department of Mathematics University of Padova, email: hieu.nguyenminh@studenti.unipd.it

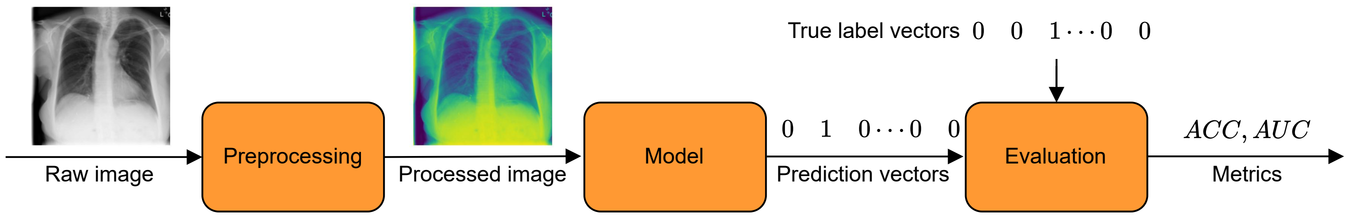


Fig. 1: Processing Pipeline for Medical Image Classification.

III. Processing Pipeline

In this section, we introduce our processing pipeline for medical image classification at a high level. Figure 1 shows the general pipeline of our project. Firstly, the raw image is preprocessed to normalize and enhance the features. Next, the processed image is fed into a model, which analyzes the image features and generate a prediction vector. Finally, we compare this vector with the true label vectors to measure the model performance by calculating evaluation metrics.

1) Preprocessing: We normalize the pixel values to the range $[0, 1]$ and augment images by creating different images variations. The purpose of image augmentation is to enhance model robustness to various views of input images. We follow the techniques in [26], including zoom, translation, rotation, and flip.
2) Model: This the main part of our pipeline. A model is a learning framework that learns to predict the diseases that the input image may have. Our goal is to build a model whose input is an X-ray image and it will yield a predicted vector having the same format as the label vector. Mathematically, a model is a function $f(\cdot)$ that takes input matrix $\mathbf{X}$ as its argument and return a binary vector $\widehat{\mathbf{y}}$ representing predicted classes.
$$\widehat{\mathbf{y}} = f(\mathbf{X}). \tag{1}$$
3) Evaluation: The prediction results will be evaluated to see how well the model make the predictions. The predictions are compared to the true labels and evaluation metrics are calculated. The metrics will be presented in more details in Section VI.

IV. Signals and Features

In this project, we use the MedMNIST dataset [4] for conducting experiments and measuring model performance. This dataset provides medical images of various diseases and parts of human body, including path, chest, dermal, optical coherence tomography (OCT), pneumonia, retina, breast, blood, tissue, organ, nodule, fracture, adrenal, vessel, and synapse. However, as the title of this report suggests, we only use the ChestMNIST dataset of X-ray images, and each image is associated with one or more diseases. There are 14 diseases in total, namely: (1) atelectasis, (2) cardiomegaly, (3) effusion, (4) infiltration, (5) mass, (6) nodule, (7) pneumonia, (8) pneumothorax, (9) consolidation, (10) edema, (11) emphysema, (12) fibrosis, (13) pleural, and (14) hernia. Figure 2 shows 14 examples of X-ray images and the associated diseases.

X-ray images in the ChestMNIST dataset are in three different sizes: 64×64, 128×128, and 224×224; and they have only one color channel. There are a total of $112,120$ images taken from $30,805$ unique patients in the dataset, and the task is multi-label classification, i.e. given an X-ray image, we need to predict if one or more diseases are in that image. We make a train-validation-test split with the ratio of 80:10:20, so we split the dataset into a training set ($78,468$ images), a validation set ($11,219$ images), and a test set ($22,433$ images). The proportions of classes in each set are summarized in Figure 3. It is clear that three diseases infiltration, effusion, and atelectasis make up over 50% of the dataset, and the distribution of classes among training set, validation set, and test set are similar, ensuring consistent performance measurement.

Each X-ray image in the dataset is a $H \times H$ matrix, where $H \in \{64, 128, 224\}$. Each matrix element is an integer from 0 to 255, representing the brightness of that pixel. To normalize the image, we divide every pixel values to 255, so that normalized pixels will be in the range $[0, 1]$. The label for an image is a 14-dimensional binary vector, where a value at position i equal to 1 indicates that the image has a disease of class i. Notice that an image may have more than one diseases, so the label vectors may have more than one position having value 1. For example, the first upper left image in Figure 2 has three diseases, and its correspending label vectors is 10000010100000. An image may also have no associated disease, in that case the label vector is a zero vector.

V. Models

A. Loss Functions

Regardless of the model architecture, the final layer is always a Dense layer which has 14 units (as there are 14 label classes) with Sigmoid activation function, so that each of 14 output values represent the probability of the input image having the corresponding disease. The model output is a prediction vector $\widehat{\mathbf{y}} \in \mathbb{R}^{14}$. To train the model, we use the Binary Cross-entropy loss function:

$$L = -\frac{1}{N_{tr}C} \sum_{i=1}^{N_{tr}} \sum_{j=1}^{C} \left(y_{ij} \log(\widehat{y_{ij}}) + (1 - y_{ij}) \log(1 - \widehat{y_{ij}}) \right), \tag{2}$$

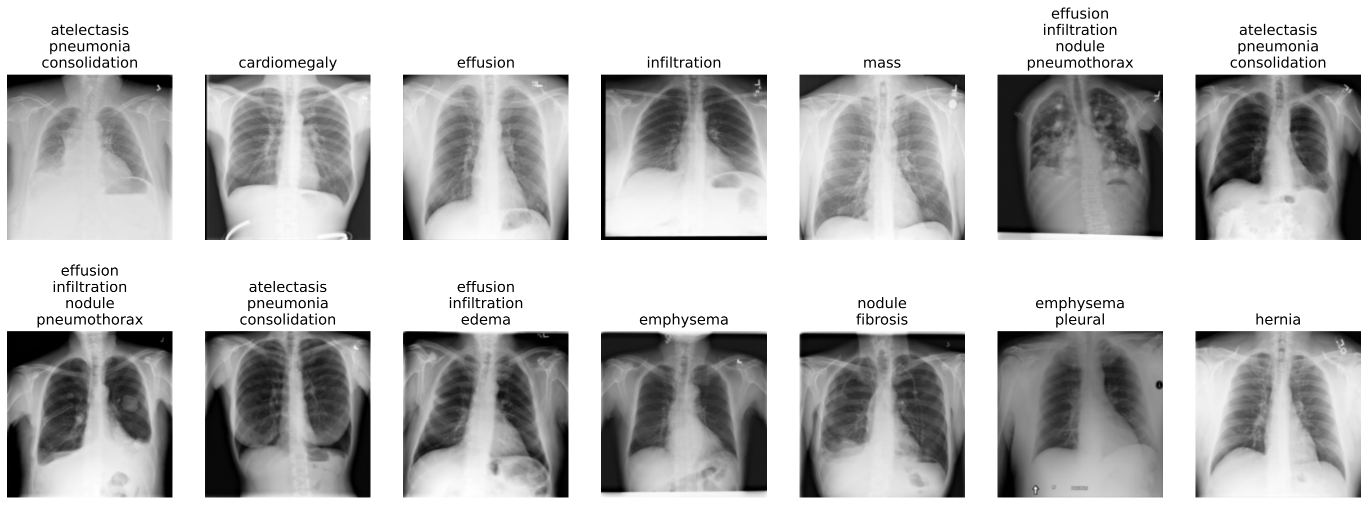
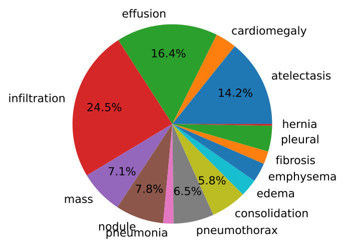
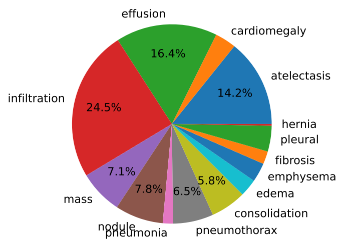


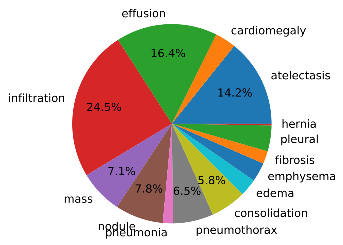
Fig. 2: Examples of images from ChestMNIST dataset.



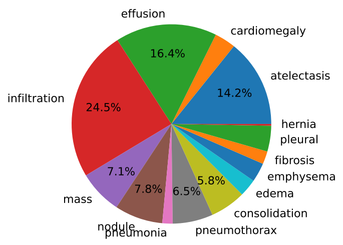
(a) Whole dataset.



(b) Training set.



(c) Validation set.



(d) Test set.

Fig. 3: Distribution of classes in ChestMNIST dataset.

where N_{tr} is the number of training samples, $C = 14$ is the number of label classes, $y_{ij} \in \{0, 1\}$ and $\widehat{y_{ij}} \in (0, 1)$ is j^{th} value of label vector and prediction vector for the i^{th} training sample. It should be noted that y_{ij} is a binary value that can be 0 or 1, while $\widehat{y_{ij}}$ is a positive real value in the range $(0, 1)$. By minimizing this loss function, we can make the model update its parameters to pull the predictions value $\widehat{y_{ij}}$ towards the true value y_{ij}.

In the next subsections, we present baseline models, including Conventional CNN, VGG-16 & VGG-19, ResNet-18 & ResNet-50, and MedViT. We also describe our Vision Transformer model in details.

B. Conventional CNN

We build a conventional CNN having three types of layers: Convolutional layer with ReLU activation function, Max Pooling layer, and Fully-connected layer with Sigmoid activation function. As illustrated in Figure 4, a CNN block consists of two Convolutional layers followed by a max-pooling layer and this structure is repeated three times. For three group of convolutional layers, we increasingly employ 32, 64, and 128 filters. We set the kernel size to 3×3, the stride to 1 and apply zero padding.

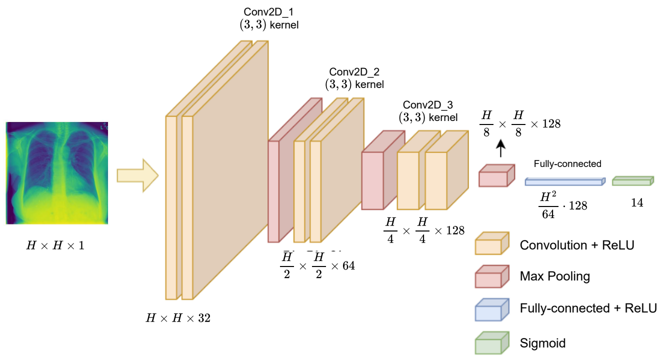


Fig. 4: Conventional CNN model.

C. VGG-16 & VGG-19

VGG [17] is basically a CNN model composed of many convolutional layers. Figure 5 visualizes the architecture of VGG-16. It is composed of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. The number 16 in model name is the number of layers having trainable parameters, which is 13 convolutional layers plus 3 fully connected layers. VGG-19 (Figure 6) is another variant of VGG with 16 convolutional layers, 3 fully connected layer, 5 max-pooling layers, and 1 softmax layer.

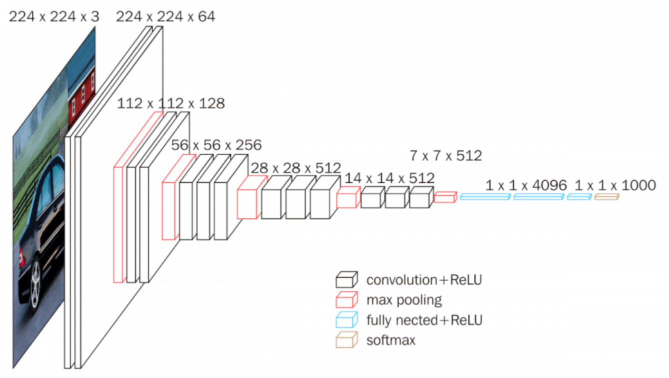


Fig. 5: VGG-16 model.

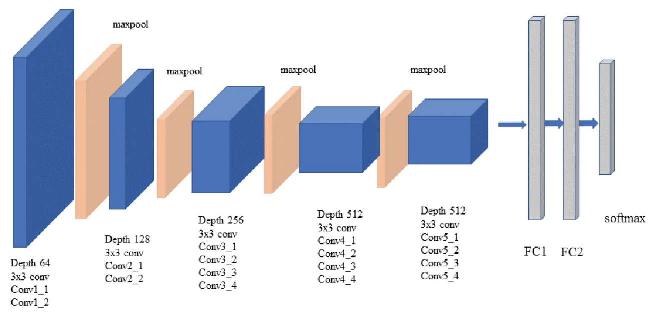


Fig. 6: VGG-19 model.

D. ResNet-18 & ResNet-50

ResNet [9] is also a stack of multiple CNN layers, but in addition to normal convolutional forward layers, it adds residual connections, i.e. adding the output of layer(s) to its input. Figure 7 visualizes a residual block with an input x. Let's say the block has two arbitrary layers, and the output after passing x to these layers is $F(x)$. The output of the residual block is the sum $x+F(x)$. The purpose of the residual connection is to stabilize the training and convergence of deep neural networks.

Figure 8 is the architecture of ResNet-18, where the curve arrows indicate residual connections. The model has 17 convolutional layers plus one fully connected layers, hence the number 18 in the model name. Similarly, ResNet-50 (Figure 9) has 50 layers in total, including 49 convolutional layers and one fully connected layers.

E. MedViT

MedViT [8], standing for Medical Vision Transformer, is a hybrid model for medical image classification that combines convolutional blocks and transformer blocks. As shown in Figure 10, MedViT has four stages and each stage is a stack of Efficient Convolution Blocks (ECB) and Local Transformer Block (LTBs), preceded by a Patch Embedding layer. The spatial size is gradually reduced after each stage, so that starting from an input image of size $H \times H$, after the fourth stage the spatial size is $\frac{H}{32} \times \frac{H}{32}$, while the channel dimension is doubled after each stage.

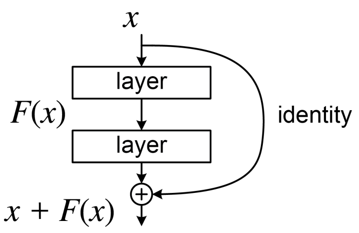


Fig. 7: Residual block.

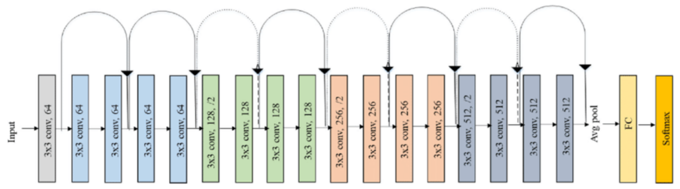


Fig. 8: ResNet-18 model.

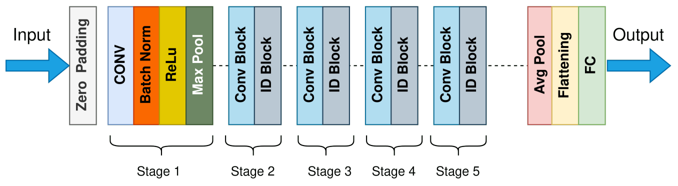


Fig. 9: ResNet-50 model.

The ECB employs a Multi-Head Convolutional Attention (MHCA) with residual connections as an token mixer, followed by a Locally Feed Forward Network (LFFN) with residual connections as a depth-wise convolution. The MHCA is a concatenation of convolutional attention (CA), which jointly attends to information from different representation subspaces.

$$\begin{aligned} \operatorname{MHCA}(x) &= \operatorname{Concat}(\operatorname{CA}_1(x), \ldots, \operatorname{CA}_h(x_h))\mathbf{W}^O, \\ \operatorname{CA}(X) &= (W \cdot T_{i,j}), \end{aligned} \tag{3}$$

where h is the number of heads, $\mathbf{W}^O$ is the output parameter matrix, and $T_{i,j} \in X$ are adjacent tokens in input feature X.

The LFFN first rearrange the sequence of tokens into a 2D lattice, apply a convolutional layer, then convert the image feature map back into a sequence.

$$\begin{aligned} \mathbf{Z}^r &= \operatorname{Seq2Img}(\mathbf{Z}), \\ \mathbf{Y}^r &= f\left(\mathbf{Z}^r \circledast \mathbf{W}_1^r\right) \circledast \mathbf{W}_2^r, \\ \mathbf{Y} &= \operatorname{Img2Seq}\left(\mathbf{Y}^r\right), \end{aligned} \tag{4}$$

where $\mathbf{Z}$ annd $\mathbf{Y}$ are the input and output sequence, $\mathbf{W}^r$ is the kernal matrix of convolution layers, and $\circledast$ denotes the covolution operation.

Overall, the ECB can be formulated as follows:

$$\begin{aligned} \tilde{z}^l &= \operatorname{MHCA}(z^{l-1}) + z^{l-1}, \\ z^l &= \operatorname{LFFN}(\tilde{z}^l) + \tilde{z}^l, \end{aligned} \tag{5}$$

where z^{l-1} is the input from the $(l-1)^{th}$ block, $\tilde{z}^l$ and z^l is the output of MHCA and ECB in the l^{th} block.

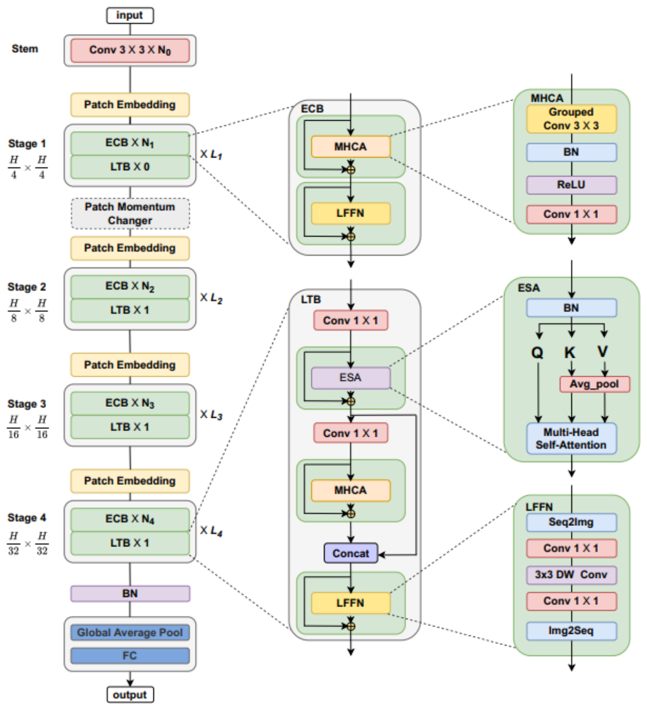


Fig. 10: MedViT model.

The LTB captures low-frequency signals with an Efficient Self-Attention (ESA) and also captures parallel information with the MHCA. The formulation of ESA is:

$$\begin{aligned} \operatorname{ESA}(x) &= \operatorname{Concat}\left(\mathrm{SA}_1\left(x_1\right), \ldots, \mathrm{SA}_h\left(x_h\right)\right) \mathbf{W}^O \\ \mathrm{SA}(X) &= \operatorname{Attention}\left(X \cdot \mathbf{W}^Q, P_s\left(X \cdot \mathbf{W}^K\right), P_s\left(X \cdot \mathbf{W}^V\right)\right), \end{aligned} \tag{6}$$

where $X = [x_1, \ldots, x_h]$ is the input feature X divided into multi-head form; $\mathbf{W}^O$ is the output parameter matrix; $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ are query, key, value matrix for the self-attention, respectively. P_S is an average pooling operation with a stride s for downsampling the spatial dimensions.

The LTC can be summarized as follows:

$$\begin{aligned} \bar{z}^l &= \operatorname{Proj}\left(z^{l-1}\right), \\ \ddot{z}^l &= \operatorname{ESA}\left(\bar{z}^l\right) + \bar{z}^l, \\ \dot{z}^l &= \operatorname{Proj}\left(\ddot{z}^l\right), \\ \tilde{z}^l &= \operatorname{MHCA}\left(\dot{z}^l\right) + \dot{z}^l, \\ \hat{z}^l &= \operatorname{Concat}\left(\ddot{z}^l, \tilde{z}^l\right), \\ z^l &= \operatorname{LFFN}\left(\hat{z}^l\right) + \hat{z}^l, \end{aligned} \tag{7}$$

where z^l is the output of the l^{th} LTB; $\tilde{z}^l$, $\ddot{z}^l$ denote the output of MHCA and ESA, respectively; and Proj refers to the point-wise convolutional layer .

F. Our Vision Transformer

Figure 11 illustrates the architecture of our ViT for medical image classification. Given an input image, we first create patch embeddings, where each embedding can be viewed as a token and we pass these tokens through three VIT blocks. The VIT Block 1 contains two blocks of `Basic Block 1` without attention layer, while VIT Block 2 and VIT Block 3 contain four and five blocks of `Block 2&3` with single-head self-attention, respectively. The reason behind the absence of attention in VIT BLock 1 is due to the head redundancy as proven in [25], that attention heads have minimal impact on performance and low speed-accuracy trade-off compared to CNN-only layers in the first stage. We employ down sampling after VIT Block 1 and VIT Block 2 to reduce tokens by keeping only useful ones. Finally, there is a global average pooling followed by the fully connected layer at the end of the model to produce predictions.

1) Patch Embedding: The Patch Embedding blocks have four consecutive `Conv2D_BN`, each of which is a normal `Conv2D` layer followed by a `Batch Normalization`. All `Conv2D` layers have a kernel size of 3×3 and a stride of 2, so the image size will be reduced by half after each `Conv2D_BN`. Given an input image of size $H \times H \times 1$, we will have patch embeddings of size $\frac{H}{16} \times \frac{H}{16} \times C_1$, where C_1 is the embedding dimension, and each of these C_1 channels is a $2D$ embedding. We then pass these patch embeddings to VIT Block 1.

2) VIT Block 1: The VIT Block 1 consists of two `Basic Block 1`s, each of which is a residual `Conv2D_BN` followed by a `Feedforward` network. The Feedforward network is basically two `Conv2D_BN` layers having a kernel size of 3×3 and a stride of 1 with zero padding, connected to each other by a ReLU activation function. The first `Conv2D_BN` reduce number of filters in data by half, and the second one double the number filters, so that the data size does not change after the Feedforward network. The purpose of this network is for channel interaction only. The data size of the output of VIT Block 1 is still $\frac{H}{16} \times \frac{H}{16} \times C_1$ and will be down sampled in the next step.

3) Down Sampling: The Down Sampling block is composed of two `Basic Block 1`s with three `Conv2D_BN` layers placed between them. The middle `Conv2D_BN` has a kernel size of 3×3 and a stride of 2, while the other two `Conv2D_BN`s have a kernel size of 1×1 and a stride of 1. There are also ReLU activation functions between `Conv2D_BN` layers. We do not apply zero padding here, so the output of Down Sampling block will have a size of $\frac{H}{32} \times \frac{H}{32} \times C_2$, where $C2$ is the custom embedding dimension. The purpose of Down Sampling is to reduce token size, so that we only keep informative parts of our tokens.

4) VIT Block 2, VIT Block 3, and Single-head Self-attention: VIT Block 2 and VIT Block 3 are composed of `Block 2&3`s, each of which is similar to `Basic Block 1` but has a `1-Head Attention` layer between the `Conv2D_BN` and the `Feedforward` layers. Inside the `Block 2&3`, the beginning `Conv2D_BN` has kernel size of 3×3 and a stride of 1 with padding to keep the data size unchanged. Let's say the data size right before the `1-Head Attention` is $D \times D \times C_a$.

Because we will apply the self-attention to only a part of data block, we need to split the data along the channel dimension into two portions of size $D \times D \times rC_a$ and $D \times D \times (1-r)C_a$. Here the value of r is set to $1/4.67$

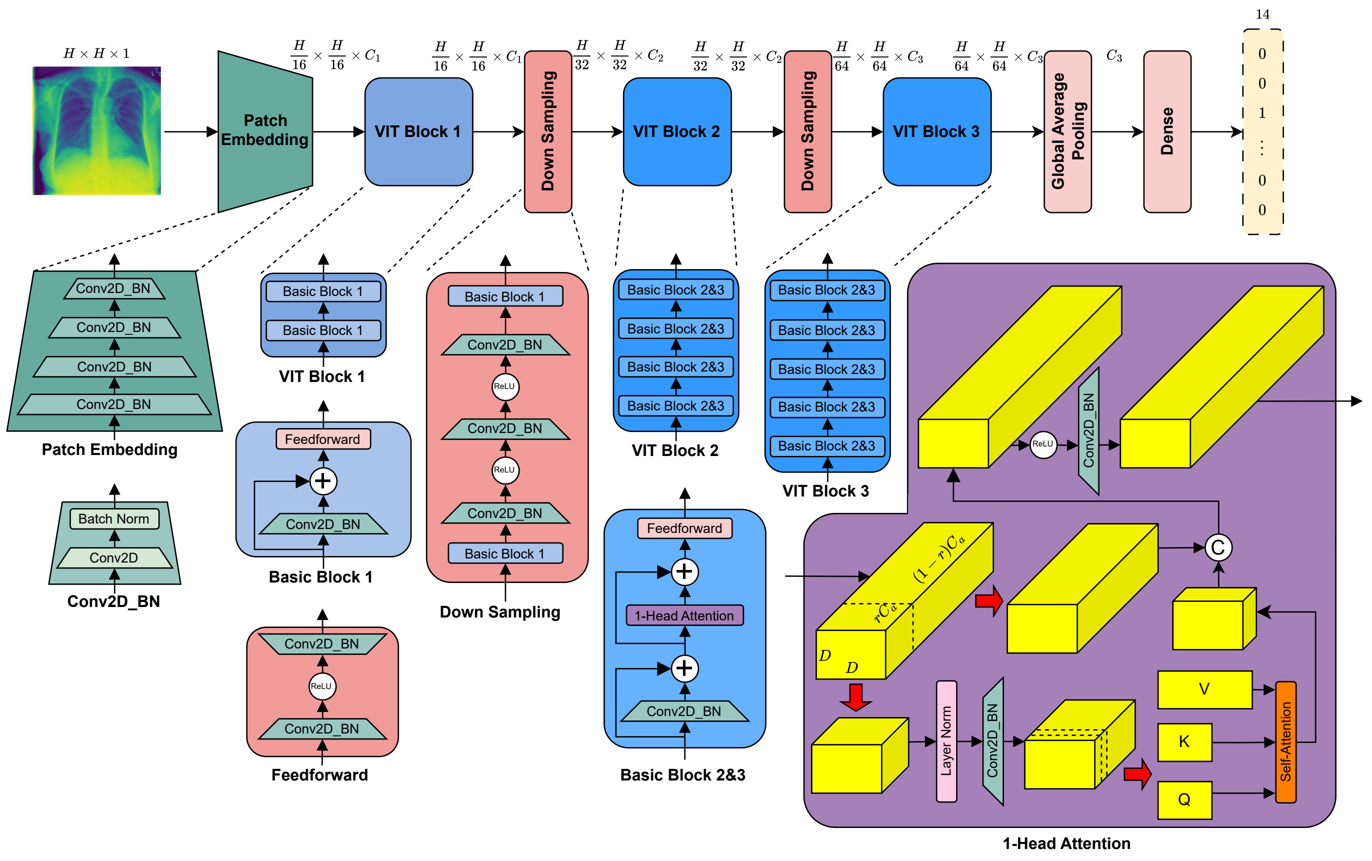


Fig. 11: Overview of the Vision Transformer for Medical Image Classification.

as suggested in [25]. For example, with input image of size 64×64, the data size in the VIT Block 2 will be $2 \times 2 \times 224$ where 224 is our custom embedding size. With $r = 1/4.67$, we split the data into two portions of size $2 \times 2 \times 48$ and $2 \times 2 \times 176$. We do not process the second portion any further, but we apply Layer Normalization and `Conv2D_BN` layer (kernel size=1×1, stride=1, with padding) to the first portion. After that we split and reshape the resulting data block into a query matrix $\mathbf{Q} \in \mathbb{R}^{2D \times d_{qk}}$, a key matrix $\mathbf{K} \in \mathbb{R}^{2D \times d_{qk}}$, and a value matrix $\mathbf{V} \in \mathbb{R}^{2D \times d_v}$, where q_{kv} is the dimension of query and key, and d_v is the dimension of value. We perform the self-attention as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_{qk}}}\right)\mathbf{V} \quad (8)$$

The resulting Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ have a size of $2D \times d_v$, which is then reshaped back to a 3D block of size $D \times D \times d_v$. Now we concatenate this data block to the second portion of size $D \times D \times (1-r)C_a$ that we split earlier, getting a data block of size $D \times D \times (d_v + (1-r)C_a)$. We apply a ReLU activation function and a `Conv2D_BN` (kernel size=3×3, stride=1, with zero padding) to this data, creating the output of the `1-Head Attention` layer. After the VIT Block 3, the size of our data block is $\frac{H}{64} \times \frac{H}{64} \times C_3$, where $C3$ is the custom embedding size.

5) Output Layers and Loss Function: Similar to conventional CNNs, we adopt a Global Average Pooling to our data to convert it into a C_3-dimensional vector. Finally, the output part of our model is two fully connected layers, where the second one has 14 units with Sigmoid activation function.

VI. Experiments and Resuls

A. Evaluation Metrics

To evaluate our model performance, we measure Accuracy (ACC) and Area under the ROC Curve (AUC) as the evaluation metrics. To measure ACC, we set a threshold of 0.5 to prediction vectors to get the actual predictions. Let $\widehat{\mathbf{y}}_i = \{\widehat{y_{i,1}}, \ldots, \widehat{y_{i,14}}\} \in \mathbb{R}^{14}$, $\widehat{\mathbf{y}'}_i = \left\{\widehat{y'_{i,1}}, \ldots, \widehat{y'_{i,14}}\right\} \in \{0,1\}^{14}$, and $\mathbf{y}_i = \{y_{i,1}, \ldots, y_{i,14}\} \in \{0,1\}^{14}$ be the Sigmoid output prediction vector, the actual prediction vector, and the true label vector of the i^{th} test sample, respectively. Then we have $\widehat{y'}_{i,j} = 1$ if $\widehat{y_{i,j}} > 0.5$, otherwise $\widehat{y'}_{i,j} = 0$, and the ACC is:

$$\text{ACC} = \frac{1}{N_t C} \sum_{i=j}^{C} (TP_j + TN_j) = \frac{1}{N_t C} \sum_{j=1}^{N} \sum^{C} \mathbf{1}_{(\widehat{y'_{i,j}} = y_{i,j})} \quad (9)$$

where N_t is the number of test samples, TP_i is the true positive number of class i (i.e. how many images are correctly predicted to have disease i), and TP_i is the true negative number of class i (i.e. how many images are correctly predicted not to have disease i). $\mathbf{1}_{(\widehat{y'_{i,j}}=y_{i,j})}$ is the indicator function that counts the number of correctly classified values $\widehat{y'_{i,j}} = y_{i,j}$.

ACC depends on the specified threshold and is sensitive to class distribution, so it alone is not enough to evaluate the general performance. If a class has few positive (because the associated disease is rare), then simply predicting all zero values can have a very high ACC but it makes no sense. On the other hand, AUC measures the classification performance on all possible thresholds, thus making a general evaluation.

$$\text{AUC} = \frac{1}{C}\sum_{i=j}^{C}\text{AUC}_j, \tag{10}$$

where AUC_j is the AUC of class j across all test samples.

B. Implementation Details

Our experiments are implemented in Python with Tensorflow[1] framework. We set the embedding dimensions C_1, C_2, C_3 to $128, 224, 320$, respectively. We train our ViT model using the Adam optimizer [27] for 30 epochs with Early Stopping strategy. Specifically, we stop the training if ROC metric is not improved after five consecutive epochs. We use a batch size of 128, the learning rate and weight decay are both set to 0.001. We use the NVIDIA A100 provided by Google Colaboratory to train the model and measure GPU throughput. We run experiments for all three image sizes: 64×64, 128×128, and 224×224. For VGG and Resnet models, we download the models provided by Tensorflow and train the whole models from scratch. For MedViT, we use its public implementation[2] and follow its provided settings.

C. Experimental Results

Table I compares the performance of basiline models and our ViT model. All models obtain a high accuracy, but the their AUC varies among models. The results are consistent across three image sizes, and the metrics are a little better if the image size is bigger. = In all scenarios, MedViT outperforms all other models, and our ViT is the second best one. However, the high performance of MedViT comes at the cost of its complexity. To evaluate models fairly, we need to consider the trade-off between model size and inference time.

Table II compares the model sizes in terms of number of parameters (# P) and measures the throughput in images per second. It is clear that the throughput decreases as the image size increases. For the baseline CNN, its number of parameters increases by the image size because there is a Flatten layer before the fully connected layer at the end of the model to convert 3D data block into 2D for calculating outputs. In contrast, for the remaining models, they use GlobalAveragePooling layer instead of Flatten layer for this purpose, so the model size remains unchanged across model sizes. As expected, MedViT is the largest model, and its throughput is also the lowest. The baseline is CNN is the worst one in performance, but it processes images the fastest

[1]https://www.tensorflow.org/

[2]https://github.com/Omid-Nejati/MedViT

TABLE I: Comparison of performance results.

Img Size	**64x64**		**128x128**		**224x224**	
Model	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
CNN	0.933	0.674	0.938	0.678	0.941	0.686
VGG-16	0.947	0.712	0.947	0.718	0.947	0.721
VGG-19	0.947	0.729	0.947	0.733	0.947	0.744
ResNet-18	0.947	0.722	0.947	0.728	0.947	0.735
ResNet-50	0.948	0.747	0.948	0.752	0.948	0.76
MedViT	**0.956**	**0.776**	**0.957**	**0.782**	**0.959**	**0.789**
Our ViT	0.948	0.755	0.948	0.767	0.948	0.772

TABLE II: Comparison of model sizes (number of parameters) and throughputs (images/s).

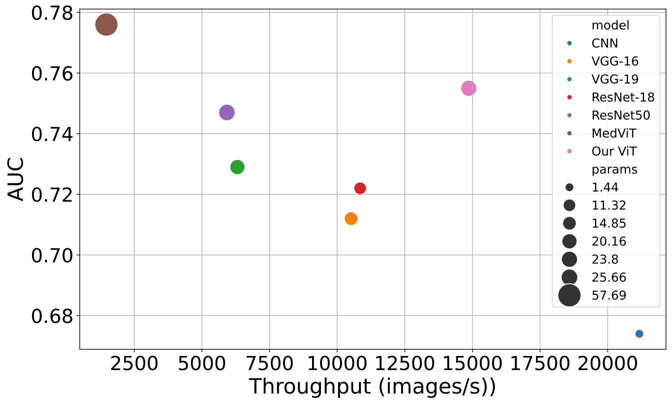
Img Size	**64x64**		**128x128**		**224x224**	
Model	# P	Thrp	# P	Thrp	# P	Thrp
CNN	1.44M	21169	4.29M	15674	12.94M	5339
VGG-16	14.85M	10516	14.85M	8393	14.85M	5228
VGG-19	20.16M	6312	20.16M	4481	20.16M	2433
ResNet-18	11.32M	10849	11.32M	8567	11.32M	5469
ResNet-50	25.66M	5923	25.66M	4146	25.66M	2245
MedViT	57.69M	1469	57.69M	1063	57.69M	492
Our ViT	23.8M	14864	23.8M	9216	23.8M	4473

due its simplicity. The number of parameters of our ViT is a bit lower than that of ResNet-50, but the throughput is over double of ResNet-50 throughput. This result is consistent to the findings of Single-head Attention ViT in [25].

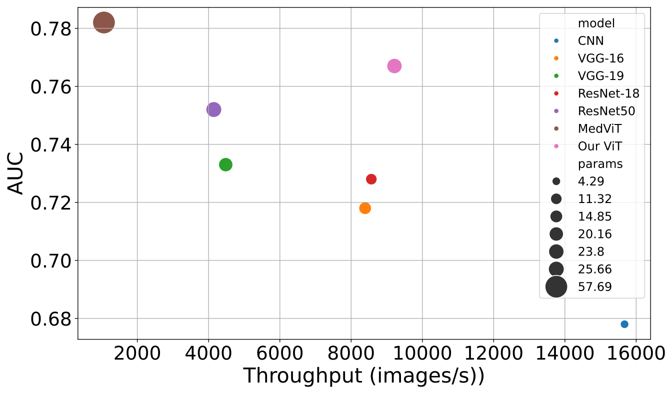
To better observe the complexity-throughput tradeoff, Figure 12 visualizes the joint comparison of number of parameters size, AUC, and throughput of models. The marker area represents the model size. Our ViT (pink) stands second in AUC, but the throughput is comparable to ResNet-18 (red) and is nearly ten times higher than throughput of MedViT (brown). Another observation is that the number of paprameters of of our ViT and ResNet-50 (purple) is similar, but the throughput of ResNet-50 is only half of our ViT's throughput. This is because ResNet-50 has much more convolutional layers than our ViT, therefore it process the information flow in the network more slowly. Overall, our ViT makes a good tradeoff between performance and throughput.

D. Prediction Visualization

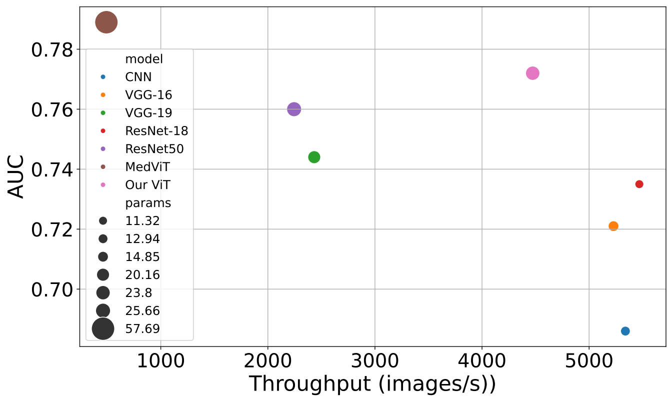
We employ GradCAM [28] to visualize the areas of X-ray image that the model focus on when making predictions. Figure 13 shows the gradients of the last convolutional layer for a healthy image and an image with two diseases. We can see that the model focus on the lung area, which makes sense as we are predicting lung diseases. Moreover, for the image with diseases, the gradient heatmaps are concentrated in the left areas, which can explains for the diseases in the image. This proves that the model was able to learn where to focus on to predict diseases in X-ray images.



(a) 64x64

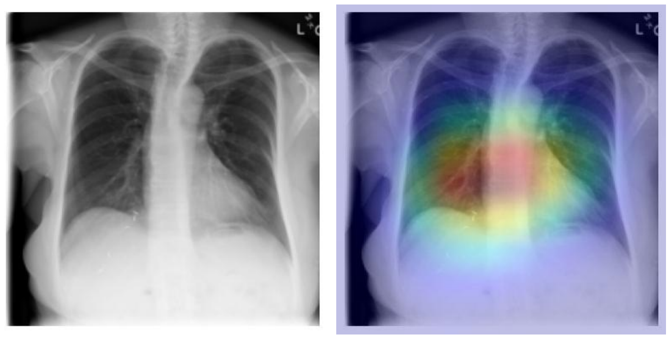


(b) 128x128

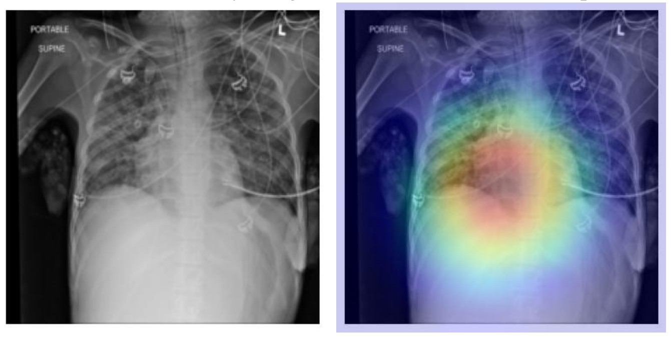


(c) 224x224

Fig. 12: Complexity-Throughput tradeoff.



(a) A raw healthy image and its Grad-CAM heatmap.



(b) An image with infiltration and nodule diseases and its Grad-CAM heatmap.

Fig. 13: Last convolutional layer visualization using Grad-CAM method.

VII. Conclusions

This project implements seven computer vision models to predict lung disease from X-ray images. We design our Vision Transformer with single-head self-attention, which is a combination of covolutional layers with attention mechanism to guide the model where to focus on when learning. In general, MedViT outperforms the remaining models, but its inference speed is the lowest. On the other hand, our Vision Transformer has a bit lower performance, but the throughput is much higher compared to MedViT and is comparable to ResNet-18. The model size is also smaller than MedViT and is similar to ResNet-50 in terms of number of parameters. In conclusion, our ViT does not have the best classification performance but it is the best one in balancing the complexity and inference speed.

References

[1] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “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*, pp. 2097–2106, 2017.

[2] P. J. LaMontagne, T. L. Benzinger, J. C. Morris, S. Keefe, R. Hornbeck, C. Xiong, E. Grant, J. Hassenstab, K. Moulder, A. G. Vlassenko, *et al.*, “Oasis-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer disease,” *medrxiv*, pp. 2019–12, 2019.

[3] L. N. Koenig, G. S. Day, A. Salter, S. Keefe, L. M. Marple, J. Long, P. LaMontagne, P. Massoumzadeh, B. J. Snider, M. Kanthamneni, *et al.*, “Select atrophied regions in alzheimer disease (sara): An improved volumetric model for identifying alzheimer disease dementia,” *NeuroImage: Clinical*, vol. 26, p. 102248, 2020.

[4] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, “Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification,” *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[5] S. Azizi, B. Mustafa, F. Ryan, Z. Beaver, J. Freyberg, J. Deaton, A. Loh, A. Karthikesalingam, S. Kornblith, T. Chen, *et al.*, “Big self-supervised models advance medical image classification,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3478–3488, 2021.

[6] Y. Dai, Y. Gao, and F. Liu, “Transmed: Transformers advance multimodal medical image classification,” *Diagnostics*, vol. 11, no. 8, p. 1384, 2021.

[7] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, “Transfer learning for medical image classification: a literature review,” *BMC medical imaging*, vol. 22, no. 1, p. 69, 2022.

[8] O. N. Manzari, H. Ahmadabadi, H. Kashiani, S. B. Shokouhi, and A. Ayatollahi, “Medvit: a robust vision transformer for generalized medical image classification,” *Computers in biology and medicine*, vol. 157, p. 106791, 2023.
[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[10] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.
[11] R. Togo, H. Watanabe, T. Ogawa, and M. Haseyama, “Deep convolutional neural network-based anomaly detection for organ classification in gastric x-ray examination,” *Computers in Biology and Medicine*, vol. 123, p. 103903, 2020.
[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
[13] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 568–578, 2021.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*, pp. 213–229, Springer, 2020.
[16] B. Cheng, A. Schwing, and A. Kirillov, “Per-pixel classification is not all you need for semantic segmentation,” *Advances in neural information processing systems*, vol. 34, pp. 17864–17875, 2021.
[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
[20] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, “Fastvit: A fast hybrid vision transformer using structural reparameterization,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5785–5795, 2023.
[21] Y. Li, J. Hu, Y. Wen, G. Evangelidis, K. Salahi, Y. Wang, S. Tulyakov, and J. Ren, “Rethinking vision transformers for mobilenet size and speed,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16889–16900, 2023.
[22] N. Kitaev, Ł. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” *arXiv preprint arXiv:2001.04451*, 2020.
[23] Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan, and Z. Liu, “Mobile-former: Bridging mobilenet and transformer,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5270–5279, 2022.
[24] S. Mehta and M. Rastegari, “Separable self-attention for mobile vision transformers,” *arXiv preprint arXiv:2206.02680*, 2022.
[25] S. Yun and Y. Ro, “Shvit: Single-head vision transformer with memory efficient macro design,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5756–5767, 2024.
[26] L. Wang, Z. Q. Lin, and A. Wong, “Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images,” *Scientific reports*, vol. 10, no. 1, p. 19549, 2020.
[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.