Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

TibetanGoTinyNet: a lightweight U-Net style network for zero learning of Tibetan Go*#

Xiali LI‡[1,2], Yanyin ZHANG[1,2], Licheng WU[1,2], Yandong CHEN[1,2], Junzhi YU‡[3]

[1]*Key Laboratory of Ethnic Language Intelligent Analysis and Security Governance, Ministry of Education, Minzu University of China, Beijing 100081, China*

[2]*School of Information Engineering, Minzu University of China, Beijing 100081, China*

[3]*Department of Advanced Manufacturing and Robotics, College of Engineering, Peking University, Beijing 100871, China*

E-mail: xiaer_li@163.com; 18560711191@163.com; wulicheng@tsinghua.edu.cn; chenyd2022@163.com; junzhi.yu@ia.ac.cn

Received July 21, 2023; Revision accepted Dec. 17, 2023; Crosschecked June 13, 2024

Abstract: The game of Tibetan Go faces the scarcity of expert knowledge and research literature. Therefore, we study the zero learning model of Tibetan Go under limited computing power resources and propose a novel scale-invariant U-Net style two-headed output lightweight network TibetanGoTinyNet. The lightweight convolutional neural networks and capsule structure are applied to the encoder and decoder of TibetanGoTinyNet to reduce computational burden and achieve better feature extraction results. Several autonomous self-attention mechanisms are integrated into TibetanGoTinyNet to capture the Tibetan Go board's spatial and global information and select important channels. The training data are generated entirely from self-play games. TibetanGoTinyNet achieves 62%–78% winning rate against other four U-Net style models including Res-UNet, Res-UNet Attention, Ghost-UNet, and Ghost Capsule-UNet. It also achieves 75% winning rate in the ablation experiments on the attention mechanism with embedded positional information. The model saves about 33% of the training time with 45%–50% winning rate for different Monte–Carlo tree search (MCTS) simulation counts when migrated from 9×9 to 11×11 boards. Code for our model is available at https://github.com/paulzyy/TibetanGoTinyNet.

Key words: Zero learning; Tibetan Go; U-Net; Self-attention mechanism; Capsule network; Monte–Carlo tree search

https://doi.org/10.1631/FITEE.2300493 **CLC number:** TP39

1 Introduction

Since the introduction of DeepMind's AlphaZero family (Silver et al., 2016, 2017a, 2017b), zero learning research has become the most effective, and now baseline, approach to studying computer games. The programs of games developed by large companies applied mostly immense neural network models, and it is challenging for general research institutions and researchers to afford the large cost of computing resources. As deep learning techniques continue to evolve, more outstanding network structures have emerged, for example, U-Net network (Ronneberger et al., 2015) and its variations (U-Net network is a lightweight and effective network in many fields such as segmentation). Lightweight networks have been studied and applied in many fields, such as object detection (Tian et al., 2022) and medical image processing (Mamoon et al.,

‡ Corresponding authors

* Project supported by the National Natural Science Foundation of China (Nos. 62276285 and 62236011) and the Major Projects of Social Science Fundation of China (No. 20&ZD279)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2300493) contains supplementary materials, which are available to authorized users

ORCID: Xiali LI, https://orcid.org/0000-0001-7950-6204; Junzhi YU, https://orcid.org/0000-0002-6347-572X

2020). However, few people have studied these lightweight networks in computer games. Therefore, it is an essential and challenging task to investigate lightweight networks in the field of computer games drawing on related technologies in the field of deep learning.

Policy learning in board game zero learning has a strong similarity to the image segmentation task, where the categories are all located at the pixel level (see the supplementary materials, Section 1.1). Tibetan Go is an intangible cultural heritage in China and needs to be protected and promoted (see the supplementary materials, Section 1.2). The general game playing (GGP) system serves to research artificial intelligence (AI) in games (see the supplementary materials, Section 1.3).

In this work, we propose a novel lightweight U-Net style network TibetanGoTinyNet. Board game actions are mapped to each coordinate, which is highly analogous to the pixel-level classification problem of image segmentation widely used by U-Net. To adapt U-Net to the zero learning task, we add an input extraction block to the input position and a value head, and a policy head to the output position of the network. Given the small size of the board and the direct mapping of actions to coordinates, we design the feature map size in the model to be the same as the board size, and the model scale is constant. This is different from the multi-scale idea in U-Net. To obtain decent results with easier operations, we introduce GhostNetV2 (Tang et al., 2022) instead of traditional residual blocks (He et al., 2016) typically used in the field of computer games. To enable U-Net to enhance its grasp of global location information, we add the attention mechanism named coordinate attention (CA) (Hou et al., 2021) to the GhostV2 bottleneck to do work on the spatial relative position. To select valuable channels, we employ the squeeze–excitation module to the network's skip connection. We employ a capsule network to overcome some drawbacks of traditional convolutional neural networks (CNNs), which are frequently used by U-Net.

We develop nine-way Tibetan Go using Ludii universal game language (Soemers et al., 2022). We train TibetanGoTinyNet under complete zero learning, and TibetanGoTinyNet is tested against diverse networks with each other under various Monte–Carlo tree search (MCTS) conditions. The network obtains a high winning rate while using fewer parameters and lower computational costs. We conduct ablation experiments on attention modules to corroborate the importance of the CA modules. We also conduct transfer experiments to evaluate the efficiency of the network migrating to larger boards.

Our contributions are summarized as follows:

1. A scale-invariant lightweight U-Net style two-headed output network TibetanGoTinyNet is proposed. We combine the current zero-learning deep reinforcement learning model with TibetanGoTinyNet. TibetanGoTinyNet achieves 62%–78% winning rate against other four U-Net style models.

2. Under the condition of limited computing power, we control the number of model parameters to about one million, and study the role of the self-attention mechanism and lightweight CNN structure in zero learning of game from the perspective of a lightweight network. Compared with the network with residual block as the backbone, TibetanGoTinyNet has only 63% of the parameters of the former under the condition of achieving a better victory rate. It offers beneficial references for computer games under resource constraints.

3. Since research on Tibetan Go zero learning has high requirements on the rate of training data generated by self-playing, we develope 9-way and 11-way Tibetan Go. Through migration experiments on Tibetan Go boards of different sizes, we can effectively reduce the model training time. The models can save about 33% of the training time when migrated from 9×9 to 11×11 boards.

2 Related works

U-Net is an encoder–decoder structure, simple but effective, originally designed to solve image segmentation problems. The lightweight and simple structure of U-Net performs well in small-scale tasks. This is an important reason why we use U-Net as a base. Another feature of U-Net is that low-level features and high-level semantic features can be fused, while all semantic information of the board game is important. Moreover, there is no need to filter useless information, and using feature concatenation is suitable.

Self-play and deep reinforcement learning are the mainstream of AI research and state-of-the-art (SOTA) techniques for the game of Go. Tibetan Go

is a national intangible cultural heritage of China. There are very few players who are good at the game and it is on the verge of extinction. There are very few digitized game records, and very few gaming intelligence and related studies are available.

Self-play and deep reinforcement learning adopted by the Alpha Zero family are SOTA techniques for chess games. It is applicable to apply zero learning and self-play to Tibetan Go. For this reason, we discuss the Alpha Zero family here.

2.1 U-Net structure improvement variations

Since U-Net (Ronneberger et al., 2015) was proposed, the lighter U-Net network optimization exists mainly in the backbone design, combining U-Net with self-attention mechanisms and bottleneck enhancements. Many researchers have focused on studying one of these aspects to achieve higher performance in their related domain tasks. TibetanGoTinyNet proposed in this work is based on U-Net with improvements suitable for board games in all these aspects mentioned earlier.

It is common to use various forms of backbones in newer U-Net extensions. The backbone specifies how the layers in the encoder are arranged, and its counterpart is therefore used to describe the decoder architecture (Azad et al., 2022a). There are many classic backbones for the U-Net architecture, such as ResNet (He et al., 2016), MultiResUNet (Ibtehaz and Rahman, 2020), and DenseNet (Huang G et al., 2017). For example, GNET (Xue et al., 2019) adopts a bilaterally symmetrical structure modified from U-Net's classic backbone. Nowadays, many U-Net style networks use mainly MobileNets (Howard et al., 2017), EfficientNet (Tan and Le, 2019), and GhostNet (Han et al., 2020) as the backbone in lightweight network research. For example, Mobile-Unet (Jing et al., 2022) and RMU-Net (Saeed et al., 2021) use MobileNets as the backbone. Networks with EfficientNet as the backbone are used in medical image segmentation tasks, like EUnet-DGF (Ding and Wang, 2021) and ghost tied block (GTB) (Guo YH et al., 2022). Ghost-UNet (Kazerouni et al., 2021) combines GhostNet with U-Net. Ghost-Unet (Xu et al., 2022) proposes a deep architecture for semantic segmentation from scratch based on an asymmetric encoder–decoder architecture using GhostNet and U-Net. In our proposed TibetanGoTinyNet, we use SOTA lightweight module GhostNetV2 (Tang et al., 2022) to reduce the computational burden, and it contains a decoupled fully connected (DFC) attention mechanism to enhance the control of the overall information. DFC attention is ideal for the global view of the board game feature information extraction, and overcomes the loss of spatial information caused by the use of cheap operation.

The commonly used attention mechanisms are squeeze excitation (SE) (Hu et al., 2018), convolutional block attention module (CBAM) (Woo et al., 2018), and CA (Hou et al., 2021). SmaAt-UNet (Trebing et al., 2021), PDAtt-Unet (Bougourzi et al., 2023), SA-UNet (Guo CL et al., 2021), HED-UNet (Heidler et al., 2022), GCAUNet (Huang Z et al., 2021), and GomokuNet (Gao et al., 2021) perform self-attention mechanism related work in the U-Net. In particular, GomokuNet was designed based on U-Net and with a spatial attention mechanism suitable for Gomoku. In TibetanGoTinyNet, we apply more than one self-attention mechanism to solve the problems in channel selection, spatial position relations, and spatial long-range relations in board games.

The bottleneck is used to force the model to learn a compressed representation of the input data which should contain only the essential and useful information required to restore the input in the decoder. Such models include ASPP-FC DenseNet (Hai et al., 2019), FRCU-Net (Azad et al., 2021), SMU-Net (Azad et al., 2022b), SA-UNet (Guo CL et al., 2021), and JCS (Wu et al., 2021). The latest improvements on the bottleneck are usually combined with the attention mechanism. A position-wise attention block (PAB) is used in MA-Net to characterize spatial dependencies between pixels in the bottleneck feature maps with self-attention. A spatial attention module is incorporated into the bottleneck of SA-UNet architecture. It is very rare to use a capsule network (Sabour et al., 2017) at the U-Net bottleneck. 3DConvCaps (Tran et al., 2022) uses three-dimensional (3D) capsule networks as the encoding layer of bottleneck in 3D image segmentation. In TibetanGoTinyNet, as our task is on a two-dimensional (2D) level, we use 2D capsule convolution instead of traditional convolutional layers.

2.2 AlphaGo family and its improvements

The AlphaGo family, including AlphaGo, AlphaGo Zero, and AlphaZero (Silver et al., 2016,

2017a, 2017b), has had tremendous success in many complex board games such as Go and chess (see the supplementary materials, Section 2.1).

3 Preliminaries

3.1 Zero learning

Zero learning is a learning method that does not apply expert knowledge and generates training data through self-play. The zero learning architecture is shown in Fig. 1. The intelligent agent performs self-play through MCTS guided by neural networks and selects corresponding actions through four steps, namely, selection, expansion, simulation, and backpropagation. When a game ends, a series of tuples containing chess game states s, action probabilities $\boldsymbol{\pi}$, and rewards z are generated to train the strategy value network, so that the strategy output $\boldsymbol{p}_s$ and value output v_s of the neural network approach the training data at state s. Then the latest network weights are used to participate in the next game of self-play. The two processes of self-play and model training are continuously iterated.

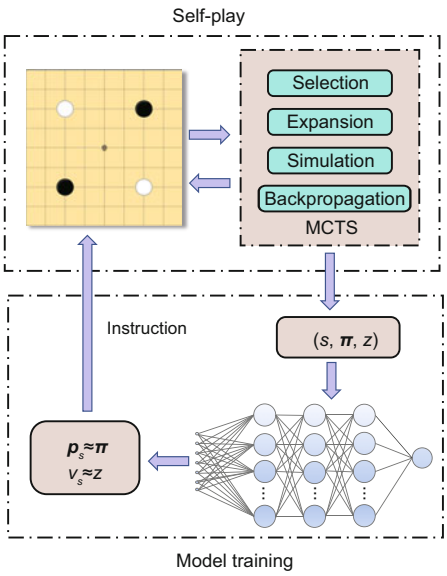

Fig. 1 Zero learning architecture (MCTS: Monte–Carlo tree search)

3.1.1 Monte–Carlo tree search

By averaging random simulations, MCTS entails approximating values. In MCTS, we enhance the frequency of moves that appear outstanding by leveraging the data from earlier simulations to bias these random simulations. The upper confidence tree (UCT), the most well-known variation of MCTS (Kocsis and Szepesvári, 2006), employs a formula derived to bias the simulations. A move is chosen if it maximizes a score as Eq. (1):

$$\text{score}_{\text{uct}}(s,a)=\bar{Q}(s,a)+c_{\text{uct}}\sqrt{\frac{\log(n(s))}{n(\text{next}(s,a))}}, \quad (1)$$

where s denotes a state, a refers to an action, $n(s)$ denotes the number of simulations at state s, next(,) denotes a game's dynamics, $\bar{Q}(s,a)$ represents the average reward for next status, and c_{uct} is a hyperparameter that governs the exploration and exploitation.

3.1.2 MCTS instructed by neural networks

Let f represent the neural network used for guiding the internal steps of an MCTS. f obtains a state s as the input and outputs a probability vector $\boldsymbol{p}_s$ for all possible moves and a scalar $v_s\in[-1,1]$ which corresponds to the network confidence regarding the present player's chances of winning the game. The MCTS instructed by neural networks follows the polynomial upper confidence tree (PUCT) formula (Silver et al., 2017a) as Eq. (2):

$$\begin{aligned}&\text{score}_{\text{puct}}(s,a)\\&=\bar{Q}(s,a)+c_{\text{puct}}P_{\text{NN}}(\text{tensor}(s))(a)\frac{\sqrt{n(s)}}{n(\text{next}(s,a))},\end{aligned} \quad (2)$$

where $\text{tensor}(s)$ is a tensor representation of the state, and $P_{\text{NN}}(\text{tensor}(s))(a)$ is a real number, representing the logit of the probability of choosing action a according to the neural network at state s.

3.1.3 Loss function and training pipeline

The training process consists of loops between self-play and optimization.

1. Self-play: Using MCTS and the most recent neural network weights, the participant competes against itself. Each time a game is over, we have a family of three-tuples $(s,\boldsymbol{\pi}_s,z_s)$, one for each state that was played. $\boldsymbol{\pi}_s$ is the probability vector obtained from MCTS, and z_s is the final result of the game.

2. Optimization: The neural network is trained to reduce the disparity between v_s and z_s and to

increase the similarity between $\boldsymbol{p}_s$ and $\boldsymbol{\pi}_s$. c is a parameter controlling the weight regularization, and θ is the parameter of the deep neural network. The loss function employed to do this is expressed as

$$\begin{aligned}\text{Loss}(s) =&(z_s - v_s)^2 + \text{CrossEntropy}\left(\boldsymbol{p}_s, \boldsymbol{\pi}_s\right)\\ &+ c\|\theta\|^2,\end{aligned} \tag{3}$$

where $\|\cdot\|^2$ is the regularization to prevent overfitting of the model.

3.2 A brief introduction to Ludii and the rules of Tibetan Go

3.2.1 Ludii

Ludii is a general game system designed to play, evaluate, and design a wide spectrum of games (see the supplementary materials, Section 3.1).

3.2.2 Rules of Tibetan Go

Tibetan Go, also known as "chess with many eyes" or the "war of eyes," is called "Mimang" in Tibet. Tibetan Go has its own character although most rules are similar to those of modern Go. The regulation size of a Tibetan Go board is 17×17, meaning that there are 289 intersection points, where black and white pieces can be placed on the board. Before beginning a game, six black and six white pieces are preplaced on the board as shown in Fig. 2a.

Similar to modern Go, the purpose of Tibetan Go players is to surround as much area as possible. However, there are some rules that differ from those used in modern Go (see the supplementary materials, Section 3.2).

However, considering that such a large board is not conducive to AI research with limited computing resources, we subsequently make experiments on a 9×9 board and migrate the model to larger boards, such as 11×11. In these sizes of boards, the number and position of the preplaced pieces are redesigned as shown in Figs. 2b and 2c.

The redesigned versions shown in Figs. 2b and 2c are inspired by the practice of simplifying the board size from 17 lines to 11 or 9 lines during the research process of Go games. This approach facilitates the verification of the performance of proposed models under conditions of small game data and limited hardware resources. Compared to the original Tibetan Go, the simplified version has fewer intersections and lower complexity, but the rules of play and win/loss determination remain unchanged.

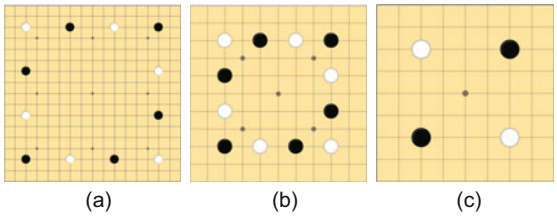

Fig. 2 Distribution of the opening pieces on Tibetan Go boards of different sizes: (a) traditional 17×17 board; (b) the simplified 11×11 board used in the experiment; (c) the simplified 9×9 board used in the experiment

Specifically, the 9×9 Tibetan Go can still well embody the characteristics of the 17×17 Tibetan Go (see the supplementary materials, Section 3.3).

4 Network structure

4.1 Overview of TibetanGoTinyNet

Fig. 3 illustrates the overall architecture of TibetanGoTinyNet. The network is a U-Net style network with a structure divided into five parts: pre-feature extraction, encoder, capsule encoder, decoder, and policy–value dual output headings. Fig. 3 shows the processing of the feature map in the network and the various operations for processing the feature map. The input feature matrix is transformed from the state s with dimensions $H \times W \times C$, where H, W, and C are the height, width, and number of channels of the feature map, respectively. The input size is $9 \times 9 \times 36$ for the dense miniature version Tibetan Go 9×9 board used in the experiment.

Unlike U-Net where feature maps are upsampled and downsampled, TibetanGoTinyNet keeps the network scale and the number of channels constant when arranging layers. The feature map size is maintained the same as the board size from the outset to the end.

Different from other variants on U-Net, which focus on improvements from one or two aspects, the proposed TibetanGoTinyNet's enhancements are from three aspects. SE is applied to skip connection to enhance the weight of the channels with more valuable information, and GhostNetV2 is used as the backbone of the network. Besides, a capsule network is used at the bottleneck to encode high-level feature information. A CA is employed in GhostNetV2 to

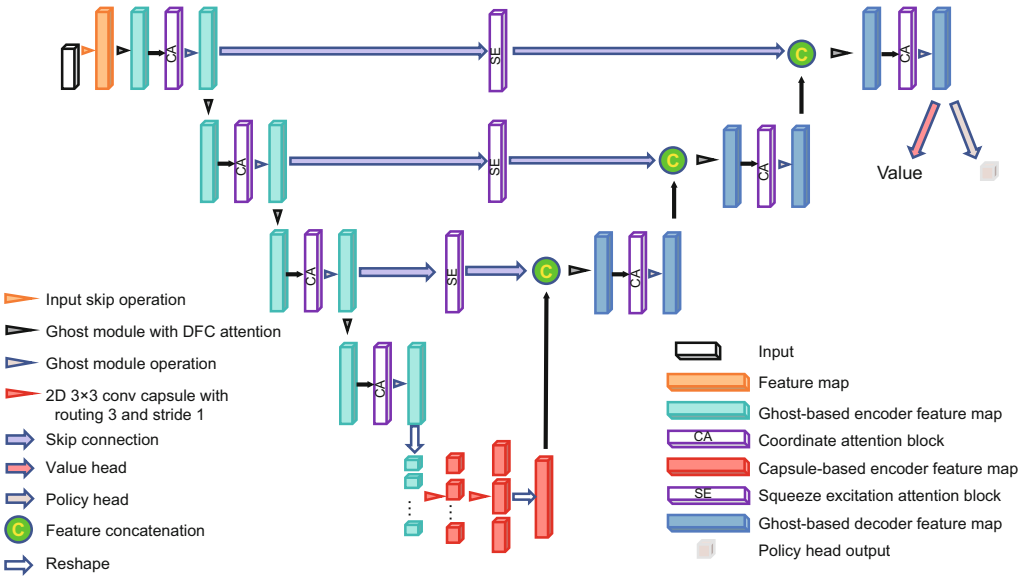

Fig. 3 TibetanGoTinyNet architecture (DFC: decoupled fully connected)

capture information in the direction of spatial location in the bottleneck.

4.2 Backbone for encoder and decoder

4.2.1 GhostNetV2

Compared with a normal convolution module and a traditional backbone used in the U-Net style network, the ghost module requires fewer parameters and lower computation cost for the same number of input and output feature maps. GhostNet (Han et al., 2020) significantly reduces the computation cost compared to traditional network architectures like MobileNetV3. Specifically, GhostNet can reduce computation cost by approximately 50% to 70% while maintaining the same level of accuracy.

The input extraction block is an extraction block employing an input_skipextract structure to extract the input channels from 36 to 72, followed by four GhostNetV2 bottleneck structures for various levels of feature encoding. The structure of the GhostNetV2 bottleneck is shown in Fig. 4a. Ghost module can replace the original convolution operation with simple operations, such as depth-separable convolution.

Given the input feature $\boldsymbol{X} \in \mathbb{R}^{H\times W\times C}$, the ghost module divides the output channels into two parts (typically in half). The first portion is the

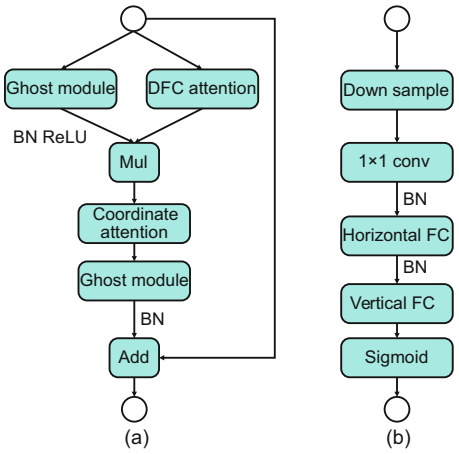

Fig. 4 Structures of the GhostNetV2 bottleneck that contains the coordinate attention and the structure of the DFC attention module: (a) GhostNetV2 bottleneck with coordinate attention; (b) DFC attention (DFC: decoupled fully connected; BN: batch normalization; ReLU: rectified linear unit; mul: multiplication; FC: fully connected)

regular convolution operation, as shown in Eq. (4):

$$\boldsymbol{Y}' = \boldsymbol{X} * \boldsymbol{F}_{1\times 1}, \tag{4}$$

where "$*$" represents the convolution operation, $\boldsymbol{F}_{1\times 1}$ is the point-wise convolution, $\boldsymbol{Y}' \in \mathbb{R}^{H\times W\times C'_{\text{out}}}$ is the partial output feature, and C'_{out} is the number of intrinsic features.

The final output is generated by a simple linear transformation with cheap operations, and the

results of the two sections are concatenated together to acquire the final output $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \mathrm{Concat}([\boldsymbol{Y}', \boldsymbol{Y}' * \boldsymbol{F}_{\mathrm{dp}}]), \tag{5}$$

where $\boldsymbol{F}_{\mathrm{dp}}$ is the depth-wise convolution, $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times C_{\mathrm{out}}}$ is the final output feature, and C_{out} refers to the number of output features.

GhostNetV2 has a DFC attention mechanism to deal with the drawback of the convolution operation which can model only local information within a window. The DFC attention structure is shown in Fig. 4b. This type of self-attention mechanism is suitable for board games, and there are few self-attention mechanisms integrated into other U-Net style networks for board game applications.

4.2.2 CA

CA is integrated into ghost bottleneck to combine position information with channel attention. The CA's structure is shown in Fig. 5a. First, the average pooling operation in horizontal and vertical directions is done on the feature map to obtain two one-dimensional (1D) vectors. We first concatenate the two vectors and then perform a 1×1 convolution operation to compress the number of channels.

After applying batch normalization (BN) and nonlinear operations to encode spatial information in both vertical and horizontal directions, the vector is split into two tensors and each of them gets the same number of channels as the input feature maps by the 1×1 convolution operation.

Finally, the spatial information is merged using channel-weighted aggregation. In other U-Net style networks, CBAM is often used as the self-attention mechanism. However, for board games, spatial location information is very important, and appropriate attention mechanisms can help networks better obtain spatial location information. CA outperforms CBAM in many tasks (Hou et al., 2021). Thus, we integrate CA into TibetanGoTinyNet to make it better guide the network in terms of spatial location. This approach is flexible and lightweight and not only captures cross-channel information but also contains direction-aware and position-sensitive information, which enables the model to locate and identify the target region more accurately.

4.3 Bottleneck and skip connection

To overcome the shortcomings of traditional convolution in spatial information availability by using the characteristics of capsule networks, and to better extract features while controlling network parameters and maintaining the lightweight characteristics of the network, we use only the capsule network layer in the extraction of the highest level features (see the supplementary materials, Section 4.1).

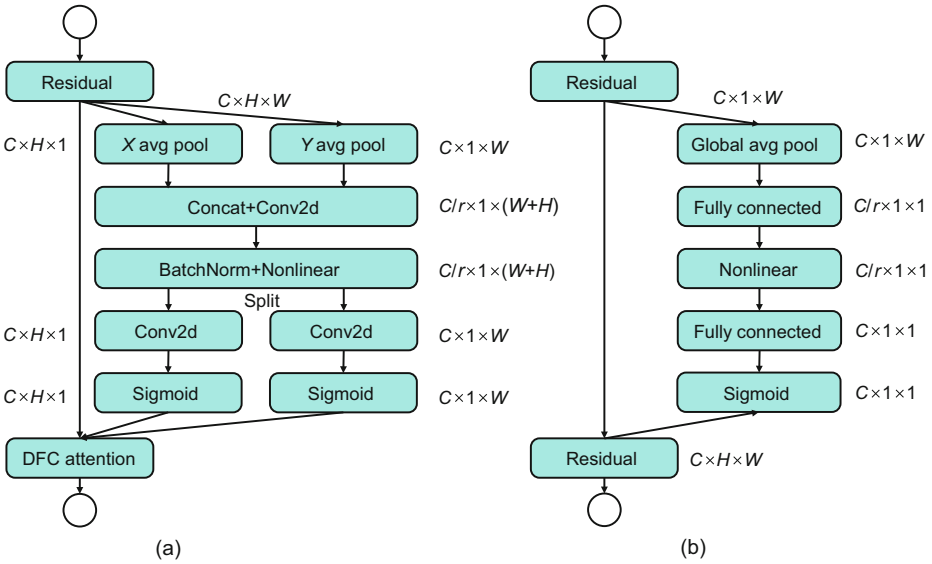

Fig. 5 Two types of attention mechanism modules used in TibetanGoTinyNet: (a) coordinate attention block; (b) squeeze excitation channel attention block (*X* avg pool: one-dimensional (1D) horizontal global pooling; *Y* avg pool: 1D vertical global pooling; DFC: decoupled fully connected)

We use the channel attention mechanism SE in skip connection to increase the weight of some of the important channels (see the supplementary materials, Section 4.2). The structure of SE is shown in Fig. 5b.

4.4 Dual-output network

Different from the mostly used output design in general U-Net style networks, our proposed network has dual outputs (see the supplementary materials, Section 4.3).

5 Experiments and discussion

5.1 Overview

The experiment aim of this work is to explore whether the network structure is applicable to novel computer game networks with limited computing resources. The results are investigated in a network with fewer parameters and lower computation cost, with limited computing resources and training time.

The lightweight and simple structure of U-Net performs well in small-scale tasks. Therefore, under the condition of limited computing resources, we selected the lightweight U-Net style networks and other models we explored as the baseline models to verify the performance of TibetanGoTinyNet. To verify the performance of the lightweight CNN structure in TibetanGoTinyNet, we selected Res-UNet as one of the baseline models. To verify the effectiveness of the self-attention mechanism that can handle channel information in the skip connection position, we selected Res-UNet Attention as one of the baseline models. To verify whether the DFC mechanism in GhostNetV2 is suitable for handling game tasks by grasping global location information, we selected Ghost-UNet as one of the baseline models. To verify whether adding a spatial CA mechanism can significantly improve the model's winning rate, we selected Ghost Capsule-UNet as one of the baseline models. The training parameters of these network models were scaled to the same tier, and learning was conducted for a specified number of epoch cycles under the same hyperparameters. To conduct MCTS self-play more efficiently and reduce the search path of MCTS, we used a 9×9 scaled-down variant of Tibetan Go on the board for the experiments. Since the number of Tibetan Go board dot arrays is far from the number of pixels in normal image tasks, it is not suitable for upsampling and downsampling small Tibetan Go board dot arrays. So, we did not upsample any of the aforementioned networks and transformed them to keep the scale constant, and the size of the feature map was equal to the size of the Tibetan Go board. Furthermore, to reduce the computation cost, the number of channels was maintained as 72 in most cases of the network for the experiments.

For the baseline models mentioned earlier, we trained 160 epochs, which is called the training completion state of the model. TibetanGoTinyNet was trained from zero, and a checkpoint was saved for every 10 epochs. Each checkpoint played against the rest of the model training completion states to observe and analyze the winning rate trend.

To observe the effect of CA on Tibetan Go's game winning rate, we conducted ablation experiments for CA. We also performed experiments on migration learning to explore the performance of the model trained on a smaller 9×9 board and migrated to a larger 11×11 board.

Our experiments were conducted on Ubuntu 18.04.4 with a limited computing capacity, including 64 GB of RAM, 16 CPU cores, a Tesla T4 graphics card, and 16 GB of GPU memory. Despite the low computing capacity, we managed to play eight games simultaneously during the training process. On average, each checkpoint completed 70–80 games. To address the issue of insufficient training data, we implemented a replay buffer while ensuring that the number of views for an example never exceeded four.

In cases where the self-play speed was not high enough to fill the replay buffer, we introduced learning delay to overcome the lack of training data. The model training in this study used the Polygames framework, with several key hyperparameters set as follows: a batch size of 128, a learning rate of 0.001, eight parallel games, and 200 rollouts for the MCTS algorithm.

To further highlight the superiority of the TibetanGoTinyNet model, we conducted experiments under various training conditions by primarily modifying the learning rate and MCTS rollouts.

5.2 Baseline models

In this work, we compare TibetanGoTinyNet with U-Net, Res-UNet, Res-UNet Attention,

Ghost-UNet, and Ghost Capsule-UNet (see the supplementary materials, Section 5.1).

5.3 Results and analysis

5.3.1 Indicator analysis

We first observe the loss, pi_err, v_err, and grad_norm curves of the aforementioned models in the training process. Fig. 6a shows the loss value during the training process, representing the difference between the simulation results of MCTS and the outputs of the policy–value heads of the neural network. The loss is made up mainly of pi_err and v_err, and their variation curves are shown in Figs. 6b and 6c respectively. pi_err represents the difference between the output of the action head of the neural network (a tensor shaped as the output space) and the tensor of frequencies in the MCTS simulations, and is the main component of loss. v_err is the difference between the evaluation of the level of optimism in a Tibetan Go game by a neural network and the actual results. The grad_norm curve, as shown in Fig. 6d, is the gradient norm of the model obtained after gradient clipping. The size of the gradient norm can be used to determine the stability and convergence degree of network training. These indicators were selected to analyze the changes and convergence of the model during training and to analyze them in conjunction with experimental results.

In Fig. 6a, the curves of almost all the models have a slight rise, and then drop rapidly. When the training process reaches about 80 epochs, the curves show a stable decline trend. Res-UNet loss and Res-UNet Attention loss have the highest decline speed. The loss of Ghost-UNet is relatively high compared to those of the other modules. The loss curves of U-Net, Ghost Capsule-UNet, and TibetanGoTinyNet are relatively compromised. In Fig. 6b, the trend of pi_err curves is similar to that presented in Fig. 6a, due to the fact that pi_err is the main component of the loss. In Fig. 6c, each curve has a large degree of oscillation, except for the models of Res-UNet and Res-UNet Attention, which converge too fast, and Ghost-UNet which converges too slow, and the v_err curve of U-Net even appears to rise. In Fig. 6d, the Res-UNet and Res-UNet Attention grad_norm

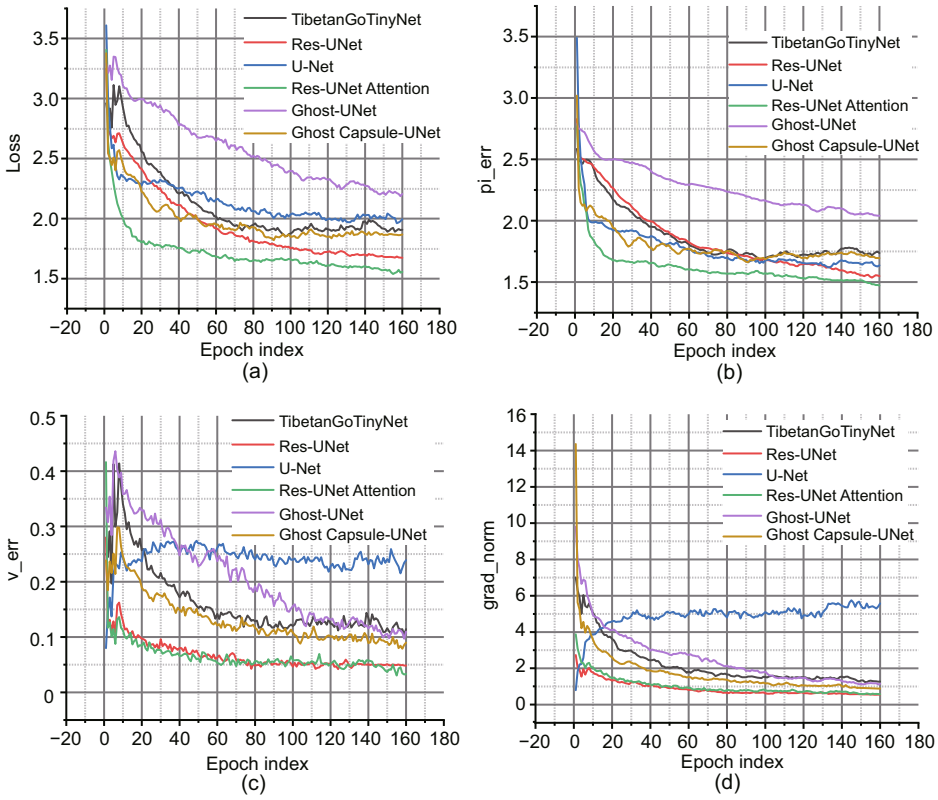

Fig. 6 Loss (a), pi_err (b), v_err (c), and grad_norm (d) curves generated during model training

curves are still at a lower level, the Ghost-UNet curve is at a higher level, and U-Net curve is at an elevated state.

According to the information in Fig. 6, we can make the following analysis: (1) The loss curve enters a steady decline when the model is trained 80 epochs. However, this does not mean that the model has reached a high level of Tibetan Go power. As the experiments do not involve supervised learning and acquire training data through self-play games, the model has low power and produces an average level of training data in the early stage of training. The decrease in the loss value only means that the model can fit the available training data well. As the training continues, the model is still able to improve its power. (2) Although Res-UNet and Res-UNet Attention converge faster in each graph, this is not an advantage for deep reinforcement learning. In the beginning of training, deep reinforcement learning may overly adapt to low-level model actions and lose motivation to explore higher-level actions. As demonstrated by the comparative experiments between models, since the initial skill level of the Tibetan Go model agent is relatively low, the training data generated from self-play are at a lower level at the initial stages. Overfitting to this lower-level training data is not conducive to long-term improvement. ResNet is a mature network that effectively addresses the gradient vanishing problem and enables deeper network architectures with strong stability. Other networks use mostly lightweight modules as the foundation, which balance accuracy while employing inexpensive operations to reduce reliance on high computational power for improved efficiency. For example, the GhostV2 module reduces the number of parameters and computation cost required but introduces instability, resulting in a slower reduction of loss values. The Res-UNet Attention model has 1.59 times the number of parameters compared to TibetanGoTinyNet. Due to the lack of spatial attention mechanisms, it struggles to capture and use the crucial spatial information in Tibetan Go, leading to mediocre performance in this task. On the other hand, models with the CA module can fully leverage spatial information, achieving this goal with fewer parameters. Although models with a CA module may have slightly higher loss values, they align with the characteristics of reinforcement learning, where exploratory actions are prevalent at the initial stages. In conclusion, Res-UNet Attention performs poorly in terms of learning effectiveness despite having low loss values, while the model with CA module compensates for this deficiency.

(3) Fluctuations in v_err represent the intelligence's inability to accurately assess the situation on the field during the early stages of training. Both v_err and pi_err show a brief upward movement at the beginning of the training. Therefore, it is easy to choose less favorable actions at the early stages of model training. This is the "painful" period for the model to learn logic. (4) The v_err and grad_norm curves of U-Net appear upward, and it may be inferred that some overfitting has occurred. The grad_norm of Res-UNet and Res-UNet Attention is very low, and there is a possibility of too slow learning. (5) The curve of Ghost-UNet is on the high side, it may be inferred that the composition in the model is too homogeneous, and the cheap operation of the lightweight CNN leads to unstable learning. This problem is alleviated in Ghost Capsule-UNet and TibetanGoTinyNet, where other structures have been employed.

5.3.2 Battle results of TibetanGoTinyNet against other models

We trained each model with 160 epochs, and the TibetanGoTinyNet model played 100 games against the final version of other models every 10 epochs commencing from the beginning. The rollout number of the MCTS was set to 400, and the winning rate derived is shown in Fig. 7 and Table 1. In Table 1, we can see that the parameter quantity of the model has been scaled to around one million. Compared with Res-UNet Attention, TibetanGoTinyNet has only 62.2% of the number of parameters of the former while achieving a better victory rate.

Fig. 7a shows the results of TibetanGoTinyNet versus Res-UNet, with the winning rate fluctuating from 40% at the beginning to a maximum of 78%. Fig. 7b shows the results of TibetanGoTinyNet versus Res-UNet Attention, with the winning rate fluctuating from 47% at the beginning to a maximum of 75%. This confirms that SOTA lightweight CNN GhostNetV2 applied in the Tibetan Go game scene has a lighter structure and a higher performance.

Figs. 7c and 7d show the results of TibetanGoTinyNet vs Ghost-UNet and Ghost

Capsule-UNet, and the winning rate is increased from 51% at the beginning to 78% and 62%, respectively. The images had a significant drop in winning rate for a while at the beginning of the training and got a substantial improvement at the 80^{th} epoch. This confirms that our addition of the capsule encoder layer effectively deals with the spatial deficiency of traditional convolution operations. This confirms that CA can effectively capture positional relationships and improve the performance of the model.

At the same time, the decline in the winning rate at the beginning of training confirms the existence of the "pain period," and the winning rate is significantly improved at the 80^{th} epoch, indicating that the intelligence does not reach the highest level when the loss value converges, but it is also a very critical time point for improvement.

5.3.3 Performance of TibetanGoTinyNet model with different hyperparameters

To study the impact of the learning rate and rollouts on the model, we conducted experiments with different hyperparameters (see the supplementary materials, Section 5.2).

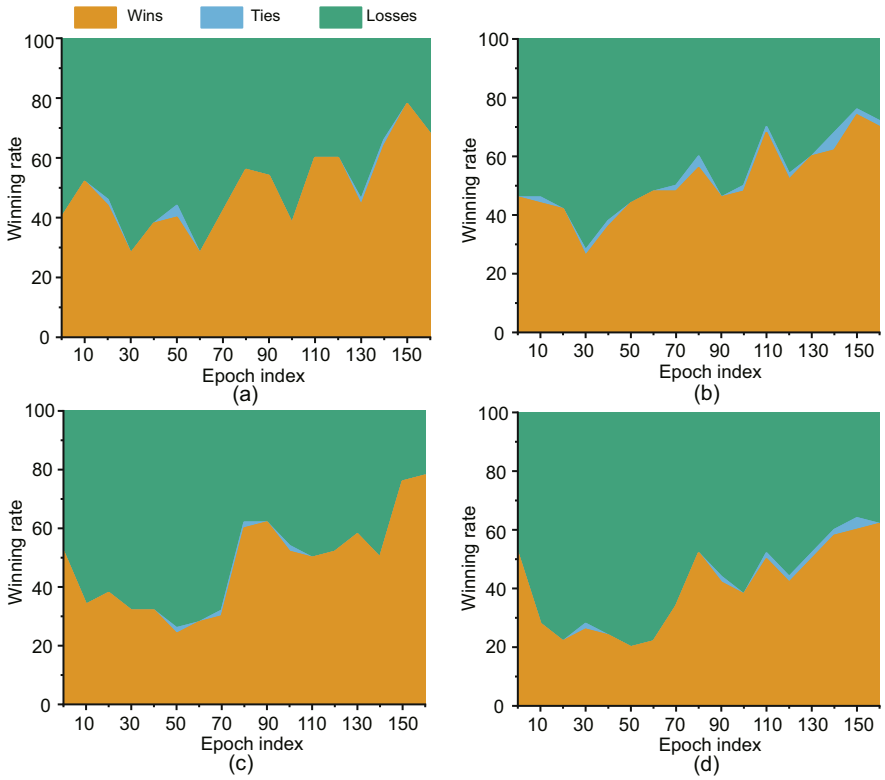

Fig. 7 Results of TibetanGoTinyNet against other models: (a) TibetanGoTinyNet versus Res-UNet; (b) TibetanGoTinyNet versus Res-UNet Attention; (c) TibetanGoTinyNet versus Ghost-UNet; (d) TibetanGoTinyNet versus Ghost Capsule-UNet

Table 1 Part of the winning rate data of the TibetanGoTinyNet model playing 100 games against the final version of other models every 10 epochs starting from 0

Model	Number of parameters	Winning rate (%)								
		Checkpoint=10	30	50	70	90	110	130	150	160
Res-UNet	1.226M	52	28	40	42	54	60	43	78	68
Res-UNet Attention	1.242M	44	27	44	49	46	69	60	75	72
Ghost-UNet	0.672M	34	32	32	28	31	62	54	76	78
Ghost Capsule-UNet	0.765M	28	27	20	34	43	52	52	64	62

The number of parameters of TibetanGoTinyNet is 0.772M

5.4 Ablation analysis

Through our ablation experiments, we aimed to test the indispensability of the CA module in TibetanGoTinyNet. First, we sought to verify whether models containing the CA module can achieve beter training results with less training time. To do so, we trained a special version of TibetanGoTinyNet without the CA module in the network for 160 epochs. Additionally, we trained a normal TibetanGoTinyNet and saved four checkpoints at the 70^{th}, 100^{th}, 130^{th}, and 160^{th} epochs. We then had the four checkpoints playing against the model without the CA module under 100, 200, 300, and 400 rollouts. The results of these experiments are shown in Fig. 8a.

We conducted experiments by setting all training conditions to be the same for both versions of the models. The training epochs and MCTS counts were identical for each game experiment. We set the rollout number to be 100, 200, 300, and 400 at the 70^{th}, 100^{th}, 130^{th}, and 160^{th} epochs, respectively, resulting in a total of 16 experimental groups. The winning rates achieved by the models with the CA module are shown in Fig. 8b.

The results indicate that the CA structure has a significant impact on the winning rate. Under the same training duration of 160 epochs, when the rollout number was set to 400, the winning rate was 73%. As the rollout number decreases, the difference between the two models narrows, indicating that the small number of rollouts fail to fully leverage the power of the neural network. The results demonstrate that combining lightweight networks with attention mechanisms related to spatial structures is highly effective. This work provides strong evidence that designing a CA module to capture spatial information is a valid approach.

5.5 Board size expansion experiment

In this work, it takes roughly 4000 s to train each epoch on a 11×11 board, compared to roughly 3000 s for a 9×9 board, which is a 33% increase in training time on large boards, let alone larger ones. The four U-Net style models and the TibetanGoTinyNet model trained on a 9×9 board with 160 epochs were transferred to a 11×11 board, and TibetanGoTinyNet still achieved 65%–75% winning rate against them. By training TibetanGoTinyNet on 9×9 and 11×11 boards for 80 epochs, we finally migrated 9×9 to 11×11 board directly with a 25% reduction of the training time for self-play. The migrated model achieved a 45%–50% winning rate, achieving better results with less training time. It shows that the network can effectively complete the migration, successfully completing the migration from small Tibetan Go boards to large Tibetan Go boards and saving training time while ensuring winning rate.

6 Conclusions and future work

In this work, we studied the zero learning model of Tibetan Go under completely limited computing

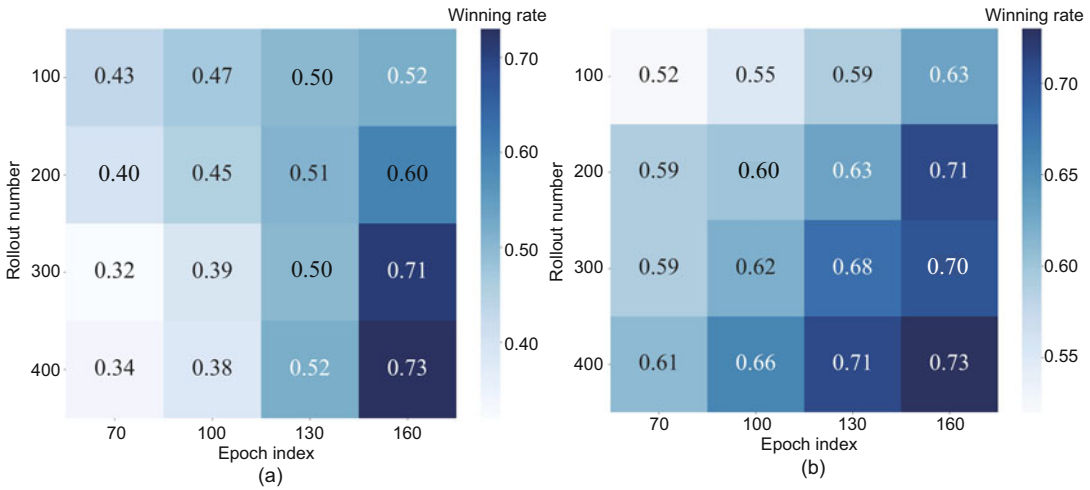

Fig. 8 Winning rates of models with (a) and without (b) the coordinate attention module

power resources and proposed a novel lightweight U-Net style network model TibetanGoTinyNet. We encoded the game Tibetan Go to perform zero learning training. TibetanGoTinyNet is a scale-invariant two-headed output network that suits the board dimensions. We used attention mechanisms, state-of-the-art lightweight network structures, and capsule networks to enhance U-Net's skip connection backbone and bottleneck. We used TibetanGoTinyNet to conduct combat experiments with other models and achieved good winning rate. The ablation experiments verify the effectiveness of the spatial position attention mechanism on board games. The board size expansion experiment obtained good winning rate in a more efficient migration way. TibetanGoTinyNet achieved 62%–78% winning rate against four other U-Net style models including Res-UNet, Res-UNet Attention, Ghost-UNet, and Ghost Capsule-UNet. It also achieved 75% winning rate in the ablation experiments on the attention mechanism with embedded positional information. The models can save about 33% of the training time with 45%–50% winning rate for different MCTS counts when migrated from 9×9 to 11×11 boards.

Future work is to design more effective attention mechanisms and to design more effective two-headed output structures. Optimizing the loss function is also a promising method to improve the performance of the model.

Contributors

Xiali LI designed the research. Yanyin ZHANG processed the data. Xiali LI and Yanyin ZHANG drafted the paper. Yandong CHEN helped process the data. Junzhi YU helped organize the paper. Licheng WU and Junzhi YU revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding authors upon reasonable request. The code for the model is available at https://github.com/paulzyy/TibetanGoTinyNet.

References

Azad R, Bozorgpour IA, Asadi-Aghbolaghi M, et al., 2021. Deep frequency re-calibration U-Net for medical image segmentation. IEEE/CVF Int Conf on Computer Vision Workshops, p.3267-3276. https://doi.org/10.1109/ICCVW54120.2021.00366

Azad R, Aghdam EK, Rauland A, et al., 2022a. Medical image segmentation review: the success of U-Net. https://doi.org/10.48550/arXiv.2211.14830

Azad R, Khosravi N, Merhof D, 2022b. SMU-Net: style matching U-Net for brain tumor segmentation with missing modalities. https://arxiv.org/abs/2204.02961v1

Bougourzi F, Distante C, Dornaika F, et al., 2023. PDAtt-Unet: pyramid dual-decoder attention Unet for Covid-19 infection segmentation from CT-scans. *Med Image Anal*, 86:102797. https://doi.org/10.1016/j.media.2023.102797

Ding XW, Wang SS, 2021. Efficient Unet with depth-aware gated fusion for automatic skin lesion segmentation. *J Intell Fuzzy Syst*, 40(5):9963-9975. https://doi.org/10.3233/JIFS-202566

Gao YF, Wu LZ, Li HY, 2021. GomokuNet: a novel UNet-style network for Gomoku zero learning via exploiting positional information and multiscale features. IEEE Conf on Games, p.1-4. https://doi.org/10.1109/CoG52621.2021.9619111

Guo CL, Szemenyei M, Yi YG, et al., 2021. SA-UNet: spatial attention U-Net for retinal vessel segmentation. 25th Int Conf on Pattern Recognition, p.1236-1242. https://doi.org/10.1109/ICPR48806.2021.9413346

Guo YH, Cai B, Liang PP, et al., 2022. Efficient network with ghost tied block for heart segmentation. Proc SPIE 12032, Medical Imaging 2022: Image Processing, Article 120320A. https://doi.org/10.1117/12.2605538

Hai JJ, Qiao K, Chen J, et al., 2019. Fully convolutional DenseNet with multiscale context for automated breast tumor segmentation. *J Healthc Eng*, 2019:8415485. https://doi.org/10.1155/2019/8415485

Han K, Wang YH, Tian Q, et al., 2020. GhostNet: more features from cheap operations. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.1577-1586. https://doi.org/10.1109/CVPR42600.2020.00165

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Heidler K, Mou LC, Baumhoer C, et al., 2022. HED-UNet: combined segmentation and edge detection for monitoring the Antarctic coastline. *IEEE Trans Geosci Remote Sens*, 60:4300514. https://doi.org/10.1109/TGRS.2021.3064606

Hou QB, Zhou DQ, Feng JS, 2021. Coordinate attention for efficient mobile network design. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.13708-13717. https://doi.org/10.1109/CVPR46437.2021.01350

Howard AG, Zhu ML, Chen B, et al., 2017. MobileNets: efficient convolutional neural networks for mobile vision applications. https://doi.org/10.48550/arXiv.1704.04861

Hu J, Shen L, Sun G, 2018. Squeeze-and-excitation networks. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.7132-7141. https://doi.org/10.1109/CVPR.2018.00745

Huang G, Liu Z, Van Der Maaten L, et al., 2017. Densely connected convolutional networks. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2261-2269. https://doi.org/10.1109/CVPR.2017.243

Huang Z, Zhao YW, Liu YH, et al., 2021. GCAUNet: a group cross-channel attention residual UNet for slice based brain tumor segmentation. *Biomed Signal Process Contr*, 70:102958. https://doi.org/10.1016/j.bspc.2021.102958

Ibtehaz N, Rahman MS, 2020. MultiResUNet: rethinking the U-Net architecture for multimodal biomedical image segmentation. *Neur Netw*, 121:74-87. https://doi.org/10.1016/j.neunet.2019.08.025

Jing JF, Wang Z, Rätsch M, et al., 2022. Mobile-Unet: an efficient convolutional neural network for fabric defect detection. *Text Res J*, 92(1-2):30-42. https://doi.org/10.1177/0040517520928604

Kazerouni IA, Dooly G, Toal D, 2021. Ghost-UNet: an asymmetric encoder-decoder architecture for semantic segmentation from scratch. *IEEE Access*, 9:97457-97465. https://doi.org/10.1109/ACCESS.2021.3094925

Kocsis L, Szepesvári C, 2006. Bandit based Monte-Carlo planning. 17th European Conf on Machine Learning, p.282-293. https://doi.org/10.1007/11871842_29

Mamoon S, Manzoor MA, Zhang FE, et al., 2020. SPSSNet: a real-time network for image semantic segmentation. *Front Inform Technol Electron Eng*, 21(12):1770-1782. https://doi.org/10.1631/FITEE.1900697

Ronneberger O, Fischer P, Brox T, 2015. U-Net: convolutional networks for biomedical image segmentation. 18th Int Conf on Medical Image Computing and Computer-Assisted Intervention, p.234-241. https://doi.org/10.1007/978-3-319-24574-4_28

Sabour S, Frosst N, Hinton GE, 2017. Dynamic routing between capsules. Proc 31st Int Conf on Neural Information Processing Systems, p.3859-3869.

Saeed MU, Ali G, Bin W, et al., 2021. RMU-Net: a novel residual mobile U-Net model for brain tumor segmentation from MR images. *Electronics*, 10(16):1962. https://doi.org/10.3390/electronics10161962

Silver D, Huang A, Maddison CJ, et al., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484-489. https://doi.org/10.1038/nature16961

Silver D, Hubert T, Schrittwieser J, et al., 2017a. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. https://doi.org/10.48550/arXiv.1712.01815

Silver D, Schrittwieser J, Simonyan K, et al., 2017b. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354-359. https://doi.org/10.1038/nature24270

Soemers DJNJ, Piette É, Stephenson M, et al., 2022. The Ludii game description language is universal. https://doi.org/10.48550/arXiv.2205.00451

Tan MX, Le Q, 2019. EfficientNet: rethinking model scaling for convolutional neural networks. Proc 36th Int Conf on Machine Learning, p.6105-6114.

Tang YH, Han K, Guo JY, et al., 2022. GhostNetV2: enhance cheap operation with long-range attention. Proc 36th Int Conf on Neural Information Processing Systems.

Tian MJ, Li XL, Kong SH, et al., 2022. A modified YOLOv4 detection method for a vision-based underwater garbage cleaning robot. *Front Inform Technol Electron Eng*, 23(8):1217-1228. https://doi.org/10.1631/FITEE.2100473

Tran M, Vo-Ho VK, Le NTH, 2022. 3DConvCaps: 3DUnet with convolutional capsule encoder for medical image segmentation. 26th Int Conf on Pattern Recognition, p.4392-4398. https://doi.org/10.1109/ICPR56361.2022.9956588

Trebing K, Staùczyk T, Mehrkanoon S, 2021. SmaAt-UNet: precipitation nowcasting using a small attention-UNet architecture. *Patt Recogn Lett*, 145:178-186. https://doi.org/10.1016/j.patrec.2021.01.036

Woo S, Park J, Lee JY, et al., 2018. CBAM: convolutional block attention module. Proc 15th European Conf on Computer Vision, p.3-19. https://doi.org/10.1007/978-3-030-01234-2_1

Wu YH, Gao SH, Mei J, et al., 2021. JCS: an explainable COVID-19 diagnosis system by joint classification and segmentation. *IEEE Trans Image Process*, 30:3113-3126. https://doi.org/10.1109/TIP.2021.3058783

Xu YH, Li Q, He SY, et al., 2022. Ghost-Unet: an efficient convolutional neural network for spine MR image segmentation: lightweight segmentation method for spine MRI. Proc 4th Int Conf on Robotics, Intelligent Control and Artificial Intelligence, p.1159-1163. https://doi.org/10.1145/3584376.3584581

Xue LY, Lin JW, Cao XR, et al., 2019. A saliency and Gaussian net model for retinal vessel segmentation. *Front Inform Technol Electron Eng*, 20(8):1075-1086. https://doi.org/10.1631/FITEE.1700404

List of supplementary materials