e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

3D IMAGE OPTMIZATION USING ADAM ALAGORITHM

J.S. Kanchana*1, T. Venkatesh*2, T.H. Vinod*3

*1Professor, Department Of Information Technology, K L N College Of Engineering, Sivagangai, Tamilnadu, India.

*2,3Student, Department Of Information Technology, K L N College Of Engineering, Sivagangai, Tamilnadu, India.

DOI : https://www.doi.org/10.56726/IRJMETS35191

ABSTRACT

Image segmentation is the process of breaking a picture up into collections of pixel regions that are each represented by a mask or labelled image. U-net is an image segmentation framework developed by CNN. The primary disadvantage of U-net is the slow learning rate at the deeper model's intermediate layers. Adam is a different optimization algorithm that can be used to teach deep learning models instead of stochastic gradient descent. Adam creates an optimization algorithm that can manage sparse gradients on noisy problems by combining the best elements of the Momentum and RMSProp algorithms. The learning rate will be increased by incorporating the Adam algorithm into the U-net design.

Keywords: Image Segmentation, Stochastic Gradient, Learning Rate, U-Net, CNN, Adam Algorithm.

I. INTRODUCTION

Image segmentation involves breaking an image into segments, so that can process only the crucial portions of it rather than the complete picture. Image segmentation entails converting an image into a collection of regions of pixels that are represented by a mask or a labelled image. One of the many interesting uses for machine intelligence is computer vision. It is easy to use and has many applications in today's society. The most frequent computer vision jobs are object detection and image classification. Predicting whether an image fits in class A or class B is the process of image classification for two classes. The complete image is given the predicted label. When we want to determine what class the picture belongs to, classification is useful. Contrarily, object detection goes a step further by foretelling where an object will appear in our input picture. By creating a bounding box around an item in an image, we can localise it there. The contents of the picture can be found and tracked with the help of detection.

Classification and localisation are two concepts that can be combined to form image segmentation. Segmenting an image entails dividing it into smaller, referred-to as segments. To comprehend what is provided in a picture at the pixel level, segmentation is used. It offers detailed knowledge of the image as well as the boundaries and shapes of the items. The result of picture segmentation is a mask, the elements of which each identify the class that each pixel belongs to.

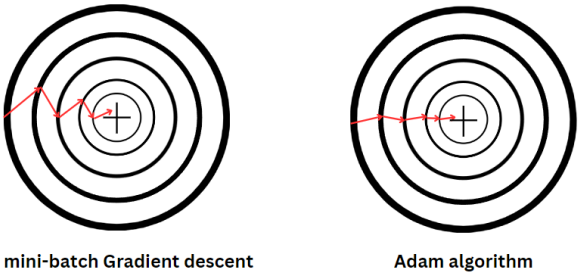

Figure 1: Comparison of other optimizers with Adam Algorithm

Now a days image segmentation are mostly used in the fields of Medical Imaging and Autonomous Driving. Automated medical image segmentation is one of the crucial component of CAD. CAD stands for Computer Aided Diagnosis. Now a days there are various segmentation approaches based on the deep learning, but the main drawback is the learning rate. Most of the algorithms like Gradient, Stochastic gradient, Momentum, etc are more inclined in the vertical direction which indeed increases the learning rate, but the Adam algorithm

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

solves this problem, because it is more inclined to horizontal direction and it combines best properties of RMSprop and Momentum.

For Image segmentation U-Net architecture is mostly used because it consists of two Convolution layers followed and two networks. First is the encoder followed by the decoder. U-Net is so good at image segmentation because it consist of skip connections and decoder networks and it separates it from other CNN's. The major drawback of U-Net architecture is that it has a slow learning rate which makes the model in compactible for a large scale of data, therefore combining the Adam algorithm and U-Net architecture can overcome this drawback and can be used widely.

To test our U-Net architecture we consider the problem statement of "Liver segmentation from CT scan", the data collected are the original CT scan and the mask image consisting of liver. We will be using the Dice coefficient method for the model evaluation.

II. METHODOLOGY

In this project we will be completing the task of liver segmentation from CT scan. Firstly we will be collecting the data set from the CT scan and mask image of the liver, the data are collected in the form of NifTi then all the images undergo the process of undersampling. While these process occurs simultaneously we will be building an U-Net architecture form the convolution layer using CNN and also implement the Adam algorithm to it.

U-Net Architecture:

U-Net architecture is widely used for image segmentation because it comprises of convolution layer and two network, the networks are encoder followed by decoder.

The encoder network, also known as a contracting network, learns the feature map of the input picture and attempts to determine what is in the image. It is comparable to the classification job carried out by convolution neural networks, with the exception that in a U-Net, the output that is now required is not the class label but a mask the same size as our input picture. There are 4 encoder units in this encoder network. Each block consists of two convolutional layers with proper padding and a kernel size of 3*3, then a Relu activation function. This is fed into a 2*2 kernel size max pooling layer. By halving the spatial dimensions acquired with the max pooling layer, we have decreased the computational expense of training the model.

The barrier layer sits between the encoder and decoder networks. This is the layer that is at the bottom. It has two convolutional layers and then Relu. The ultimate representation of the feature map is the bottleneck's output.

The expansive network is another name for the decoder network. To make our feature maps fit the size of our input picture, we upsample them. This network uses skip links to create a segmentation mask using the feature map from the bottleneck layer. Our second query, "where," is attempted to be answered by the encoder network. There are 4 decoder units in total. Each block begins with a transpose convolution with a kernel size of 2*2 (designated as up-conv in the illustration). This output is joined to the appropriate encoder block skip layer connection. A Relu activation function is then used, followed by two convolutional layers with a kernel size of 3*3.

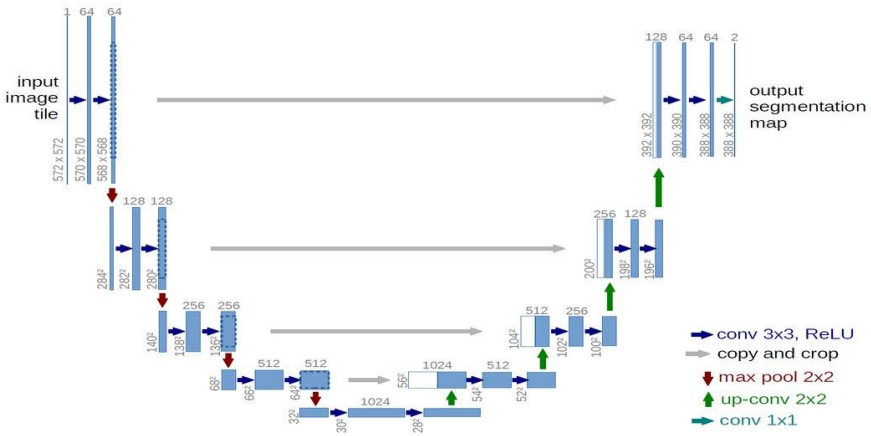

Figure 2: U-Net Architecture

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

Adam Algorithm:

Adam algorithm is majorly used to make the learning path more inclined in the horizontal direction, therefore the time consumed for the learning rate is decreased drastically. It combines best properties of RMSprop and Momentum. It generally calculates the exponential moving average of gradients and square gradients. The formulas of Momentum, RMSprop and Adam are given below

Momentum

$$V_{dw} = \beta_1 V_{dw_{prev}} + (1 - \beta_1)dw$$

$$V_{dB} = \beta_1 V_{dB_{prev}} + (1 - \beta_1)dB$$

$$W = W - \alpha\, V_{dw}$$

$$B = B - \alpha\, V_{db}$$

RMS Prop

$$S_{dw} = \beta_2 . S_{dw_{prev}} + (1 - \beta_2)(dw)^2$$

$$S_{dB} = \beta_2 . S_{dB_{prev}} + (1 - \beta_2)(dB)^2$$

$$W = W - \alpha \left(dW \Big/ \sqrt{S_{dw} + \varepsilon} \right)$$

$$B = B - \alpha \left(dB \Big/ \sqrt{S_{dB} + \varepsilon} \right)$$

Adam (Adam Moment Estimation)

$$W = W - \alpha . \frac{V_{dw}}{\sqrt{S_{dW} + \varepsilon}}$$

$$B = B - \alpha . \frac{V_{dB}}{\sqrt{S_{dB} + \varepsilon}}$$

$$\beta_1 = 0.9$$

$$\beta_2 = 0.999$$

$$\epsilon = 10^{-8}$$

III. IMPLEMENTATION

There is various process to completely build the U-Net architecture and to implement the Adam algorithm and then to making the liver segmentation. Following are the modules that are done in-order to achieve the liver segmentation from the CT scan.

Data Pre-Processing:

All the data are collected in the form of NifTi. The images are read by the nibabel library and the undersampling is done. The undersampling is done by making an axial cut along the Z axis. Then only the 2D section containg the liver is recovered. The recovered section (mask images) will also undergo a testing whether it contains only zero, if that is the case then it means there is no liver. These images are at last saved into .npy (numpy) files for ease of access. 70% of data are used as training data and reaming 30% is used as test data. Each training data and testing data are converted into the .npy files separately. Having the .npy files help us for easy access to the data. This file format makes incredibly fast reading speed enhancement over reading from the plain text or a CSV file.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

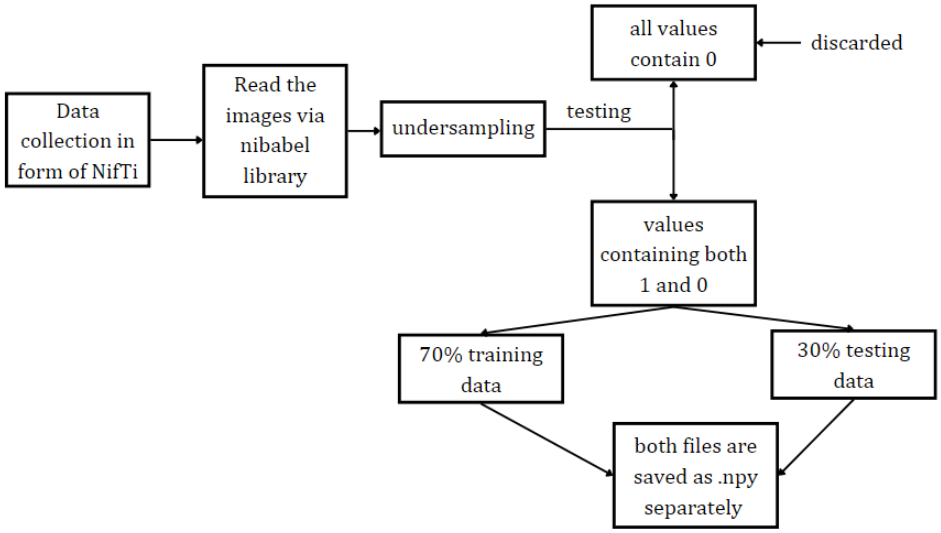

Figure 3: Flow chart for Data Collection

Building U-Net Architecture:

Building an U-Net architecture is done via the convolution layers. It comprises of an expansive path and a contracting path. The contracting path adheres to the standard convolutional network design. Two 3x3 convolutions (unpadded convolutions) are applied repeatedly, and after each one, a rectified linear unit (ReLU) and a 2x2 max pooling operation with step 2 are applied for downsampling. We double the amount of feature channels with each downsampling step. An upsampling of the feature map is followed by a 2x2 convolution ("up-convolution") that cuts the number of feature channels in half, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU, at each stage of the expansive path. Due to the loss of boundary pixels in each convolution, cropping is required. Each of the 64-component feature vectors is mapped to the intended number of classes at the final layer using a 1x1 convolution. The network has 23 convolutional layers in total.

```
img_rows = int(512/2)
img_cols = int(512/2)
smooth = 1.
inputs = Input((img_rows, img_cols, 1))
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)
up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

```
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)
up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)
up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)
up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)
model = Model(inputs=[inputs], outputs=[conv10])
```

Implementing the Adam Algorithm

Adam algorithm combines the best properties for Momentum and RMS prop. Implementing this algorithm can speed up the learning rate by increasing the learning path in horizontal direction. With this algorithm we can reduce the movement in vertical direction. The learning rate is lower in gradient descent because there is a greater tendency for zigzag patterns, but it is higher in the Adam algorithm since the learning path is horizontal.

For β1 we take the value 0.9 and for β2 we take the value 0.999 and then for epsilon, $\epsilon=10^{-8}$

Then the formula for both weight and bias are

$$W = W - \alpha . \frac{V_{dw}}{\sqrt{S_{dW} + \varepsilon}}$$

$$B = B - \alpha . \frac{V_{dB}}{\sqrt{S_{dB} + \varepsilon}}$$

Model evaluation

We will be using "Dice coefficient" method for evaluating our model. Dice coefficient are used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. x-axis will have the number of epoch and the y-axis will have the Dice coefficient.

The formula of dice coefficient are :

$$\frac{2*|X \cap Y|}{|X|+|Y|}$$

Along with this we also use time module to get a precise time taken to build the model and for image processing.

Web app integration

The web app will be created using the Flask framework. All the business logic are written in the python language, along with Flask, HTML, CSS and BOOTSTRAP to enhance the user interactions and user experience.

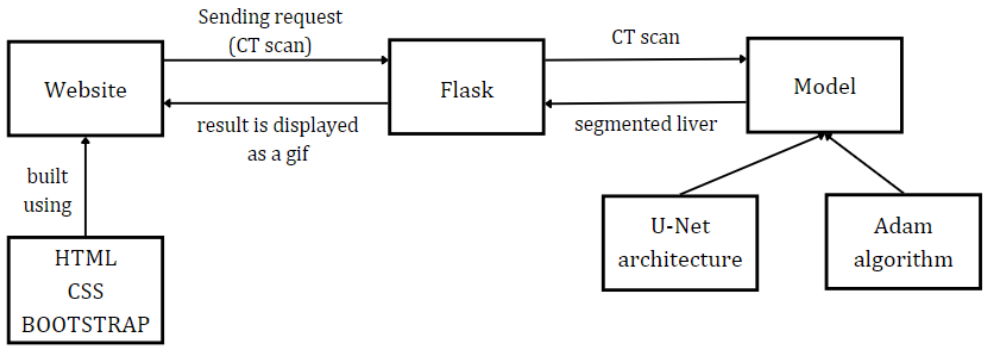

Figure 4: Flow chart for Webapp integration

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

IV. RESULTS AND DISCUSSION

The CT scan is sent to the U-Net architecture and the liver is segmented in a yellow declined line. All the actions are carried out via a webapp for ease of interaction. The result is displayed via gif in the webapp.

Below is the table explaining the total time consumed for each Epoch the time per step with the dice coefficient value for each epoch.

Table 1. All the value of Dice coefficient with respect to the epoch

SN.	Epoch	Total time	Time per step	Dice coefficient
1	Epoch 1/20	1069s	9s/step	0.5361
2	Epoch 2/20	959s	8s/step	0.8383
3	Epoch 3/20	932s	8s/step	0.8643
4	Epoch 4/20	802s	7s/step	0.8882
5	Epoch 5/20	895s	8s/step	0.9038
6	Epoch 6/20	791s	7s/step	0.9221
7	Epoch 7/20	789s	7s/step	0.9313
8	Epoch 8/20	858s	7s/step	0.9398
9	Epoch 9/20	787s	7s/step	0.9410
10	Epoch 10/20	786s	7s/step	0.9623
11	Epoch 11/20	792s	7s/step	0.9652
12	Epoch 12/20	785s	7s/step	0.9648
13	Epoch 13/20	782s	7s/step	0.9654
14	Epoch 14/20	777s	7s/step	0.9704
15	Epoch 15/20	782s	7s/step	0.9765
16	Epoch 16/20	774s	7s/step	0.9765
17	Epoch 17/20	776s	7s/step	0.9787
18	Epoch 18/20	777s	7s/step	0.9674
19	Epoch 19/20	779s	7s/step	0.9787
20	Epoch 20/20	0s	-	0.9797

As the result is in a gif format, we will be showing the most different stages of segmentation from the CT scan

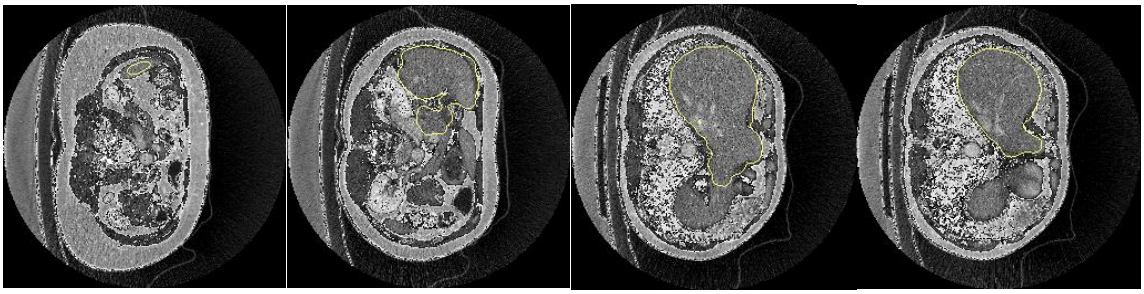

Figure 5: The main 4 stages of the segmentation

The relation between the dice coefficient and Epoch for both training and testing data. The relation is shown by a line graph for ease of understanding and it is generated via matplotlib library.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:03/March-2023 Impact Factor- 7.868 www.irjmets.com

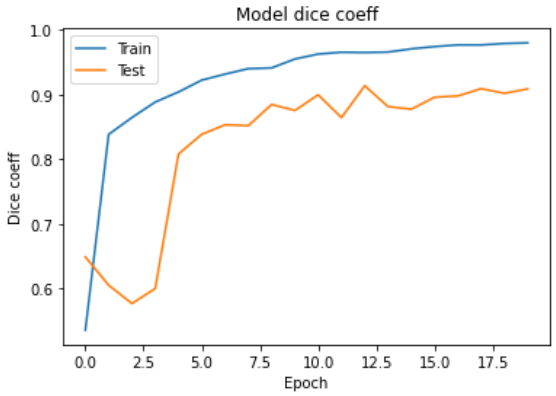

Figure 6: comparison of training and testing data with line graph

V. CONCLUSION

The Adam algorithm and U-Net architecture combined produce a higher learning rate and can therefore be applied to a wide range of datasets, particularly in the medical industry. This can also be extended to projects involving autonomous driving and other forms of automation. The paradigm is made accessible to even naive users (those without programming experience) through the creation of a webapp. This project has a significant impact on the automated segmentation of medical images.

VI. REFERENCES

[1] Kai Han, Lu Liu , Yuqing Song, Yi Liu, Chengjian Qiu, Yangyang Tang, Qiaoying Teng , and Zhe Liu, "An Effective Semi-Supervised Approach for Liver CT Image Segmentation", IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 26, NO. 8, AUGUST 2022, pp. 4001-4003

[2] W. Bai et al., "Semi-supervised learning for network-based cardiac MR image segmentation," in Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv., 2017, pp. 253–260.

[3] D.-P. Fan et al., "Inf-Net: Automatic COVID-19 lung infection segmentation from CT images," IEEE Trans. Med. Imag., vol. 39, no. 8, pp. 2626–2637, Aug. 2020.

[4] Y. Zhou et al., "Semi-supervised 3D abdominal multi-organ segmentation via deep multi-planar co-training," in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2019, pp. 121–140.

[5] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2517– 2526.

[6] Y. Zhang and J. Zhang, "Dual-task mutual learning for semi-supervised medical image segmentation," in Proc. Chin. Conf. Pattern Recognit. Comput. Vis., 2021, pp. 548–559.

[7] Y. Xia et al., "3D semi-supervised learning with uncertainty-aware multiview co-training," in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis., 2020, pp. 3646–3655.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 3431–3440.

[10] X. Zhu, K. Guo, S. Ren, B. Hu, M. Hu, and H. Fang, "Lightweight image super-resolution with expectation-maximization attention mechanism," IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 3, pp. 1273–1284, Mar. 2022.