(12) INNOVATION PATENT
(19) AUSTRALIAN PATENT OFFICE

(11) Application No. **AU 2018100325 A4**

(54) Title
A New Method For Fast Images And Videos Coloring By Using Conditional Generative Adversarial Networks

(51) International Patent Classification(s)
H04N 9/43 (2006.01) ***G06T 5/00*** (2006.01)
G06N 3/00 (2006.01) ***H04N 19/00*** (2014.01)

(21) Application No: **2018100325** (22) Date of Filing: **2018.03.15**

(45) Publication Date: **2018.04.26**
(45) Publication Journal Date: **2018.04.26**
(45) Granted Journal Date: **2018.04.26**

(71) Applicant(s)
YE ZHAO;Xilai Nian;MENGYU SUN;SHINAN WANG;JIAFAN XUE;JIATONG ZHU

(72) Inventor(s)
ZHAO, YE;WANG, SHINAN;Nian, Xilai;SUN, MENGYU;XUE, JIAFAN;ZHU, JIATONG

(74) Agent / Attorney
Gloria Li, 65b Hattaway Ave, Buckland Beach, Auckland, 2012, NZ

Abstract

This new invention presents a method for coloring black-and-white films and photos in real time by using conditional Generative Adversarial Network(cGAN). Using the YUV channel instead of RGB channel to train the network makes the training more effective. The cGAN which is made up of a "U-Net"-based architecture for generator and a convolutional "PatchGAN" classifier avoids shortages of traditional encoder-decoder models.

20181003325 15 Mar 2018

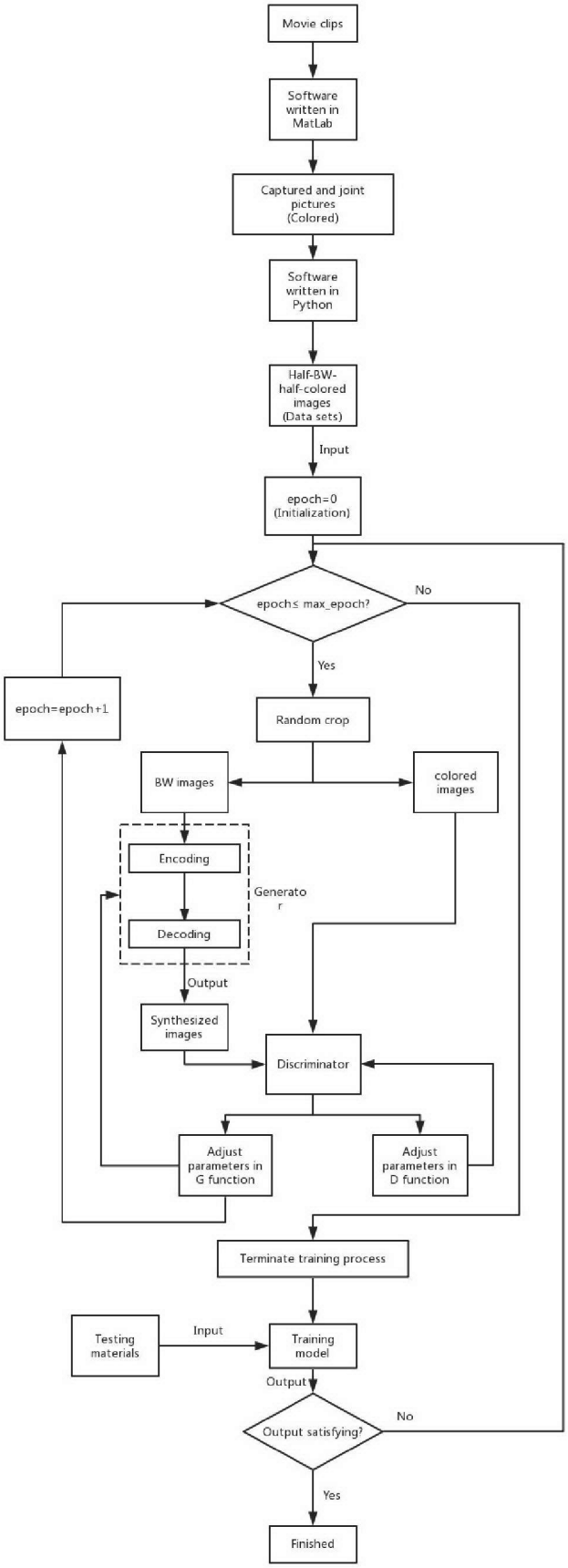

Figure 1

DESCRIPTION

TITLE

A New Method For Fast Images And Videos Coloring By Using Conditional Generative Adversarial Networks

FIELD OF THE INVENTION

The present invention relates to a technique which colorizes black-and-white films and photos. In particular, this invention can color the movies and the photos automatically and in a fast speed, which guarantee the possibility of video processing. The method can give full expression to high-level semantics through using conditional Generative Adversarial Networks (cGANs).

BACKGROUND OF THE INVENTION

Before the middle of 20 century, constrained by manufacturing cost and technology, films and photos almost were black and white. The foundation of film colorization is image colorization. Initial image coloring was all done by hand which costs of efforts and time. As the computer technology is stepping forward to mature, the film has been digitized and colored with the help of computer. However, it still takes a large number of manpower to fine-tune details to ensure quality. With revolution of computer performance, in the area of Deep Learning, encoder- decoder model uses regression to predict color of each pixel

which is widely used in image colorization which is much more advanced than the previous method.

Nevertheless, the only link between encoding and decoding is a fixed length semantic vector. In other words, the encoder will compress the information of the entire feature into a fixed length vector. Semantic vector can't express features in, which makes decoding can't get enough input sequences information at the beginning. Which means the performance of extracting high-level semantic features is disappointed. In this case, the decoding accuracy reduces and the color is dull. The "semantic gap" distance between low-level visual features and high-level semantic features is necessary by traditional encoder-decoder model.

Therefore, a basic premise is to find an independent model which has a complete mapping from the input image to the output image. The model should learn high-level semantic features and ensure consistency between input and output.

The related references are:

[1] Zhang R, Isola P, Efros A A. Colorful Image Colorization[J]. 2016:649-666.

[2] Iizuka S, Simo-Serra E, Ishikawa H. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification[M]. ACM, 2016.

2018100325 15 Mar 2018

[3] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.

SUMMARY OF THE INVENTION

In order to avoid the shortages of traditional encoder-decoder model, this invention applies GANs in the conditional setting. Generative combat network is a kind of learning model, and judging by the generator is composed of two parts, the goal of GAN is to learn the relationship between random noise vector Z and Y output image generator, to generate output can deceive false discriminator as the goal, to distinguish the true image for the task. The two parts are antagonistic to each other and are trained together. The loss function of cGAN is more complex than traditional GAN network. It is to learn the mapping relationship between the observed image x, the random noise vector Z and the output vector y. The joint distribution is used to make the network more precise. The cGAN generator uses the U-Net model, and the discriminator uses the Patch GAN network. We use a "U-Net"- based architecture for our generator to make setup simpler and the "Patch GAN" classifier is for discriminator we use a convolutional, which only penalizes structure at the scale of image patches. The U-Net network is composed of encoders and decoders, and the false pictures are generated by convolution and deconvolution. In the convolution process, the size of the filter and the step size determine the degree of reduction. Deconvolution is the opposite

of the convolution, which is used to expand the dimension. The Patch GAN network structure of discriminator and generator is slightly different. The network structure of discriminator is only composed of encoders, that is, a convolution network. We define the picture generated by the generator as fake B, define the color map originally used for practice as real B, and enter the fake B and real B at the same time, according to the result, we can know whether the generated fake B can be false or true by discriminating.

DESCRIPTION OF DRAWING

The following drawings are only for the purpose of description and explanation but not for limitation, where in:

Fig.1 is the flow diagram of the core algorithm of our invention, which shows the specific step in our work step by step;

Fig2 shows two types of the architecture of the generator. The "U-Net" is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks;

Fig3 shows the architecture of the U-Net model in detail. We input a grayscale image which size is 512*256. The left leg consists of eight convolution-layer-blocks and the input of each block is the output of the previous block. The right leg is composed of eight deconvolution-layer-blocks whose inputs consist of two parts: the output

of the former block and the output of the symmetric convolution-layer-block.

Fig4 shows the process of the test. We input an image to the procedure, and then the procedure transforms it to the tensor. After the regularization, the tensor goes through the model in forward direction. Finally, the procedure output the image;

Fig5 is the process flow diagram of our website, which provides another fully visual and interactive user interface to users. With the guidance of this figure, users could easily transform black-and-white images and GIF to colorful images and GIF through the website. Users can choose to input images or GIF in the index, and then select a file to coloring. After that, users could back to the index in this page;

Fig6 is the process flow diagram of our EXE file, which provides a fully visual and interactive user interface .With the guidance of this figure, users could easily transform black-and-white images and GIF to colorful images and GIF through the EXE file. Users can choose to input images or GIF in the index, and then select a file to coloring. After that, users could download the color images and GIF, empty the interface, or back to the index;

201810032515 Mar 2018

Fig.7 shows the results of our invention. The left column lists the original images we used in our model, while the right column shows the colorful images.

DESCRIPTION OF PREFERRED EMBODIMENTS

The invention adopts the cGAN, which is more convenient and faster than the traditional color method. In order to make the invention easier to be understood, an embodiment of the invention will be described in detail in description, and the process of the process is shown in Figure

Step 1:

In order to train our network, we need to collect a training set for the network to adjust its parameters. The training set should be large enough, usually thousands of pictures, that is capable for training a decent network. To get these images, we intercept some movies and selected the appropriate sampling frames. To prevent the network from over-fitting, we take 1 frame every 50 frames instead of all of the frames for the training process. The images in the training set is supposed to be the combination of a gray image and an RGB image of a same image(required in the network structure). The proportion of the height and width of the image must be 2:1, for example, 1920:960.In our demo, the training set contains about 6500 images.

Step 2:

After obtaining the training sets, we need to use the training images to train the network. To begin with, we create a data loader class to load the images in the training set and convert the data type to tensor. Next, we divide an input image in half, respectively named A and B. Then, we convert the A image, a RGB image, to gray image, and keep B unchanged. When using YUV channel for output, we convert B to YUV scale. To convert the color space of the images, we multiply the value of different channel by a certain factor. Lastly, we randomly take a part of A image which size is 512*256, and cut the corresponding part of B image. By randomly cutting the images, we can reuse a same training image several times.

After the data processing, we need to create a pix-to-pix network model. Firstly, we initialize a generator network. The generator network contains 16 blocks, 8 for convolution layer and 8 for deconvolution layer. The convolution layer calculate all the values of the input image with the kernal matrix that stores weight parameters of the pixels around the calculating pixel point in order to extract the feature from the image. After the convolution, use a function named ReLu to set the value in range (0,1).To lower the dimension of the output image of each layer, we set the convolution stride as 2, so height and width of the convolution result image would be half of the input image. We add skip connections between the corresponding deconvolution layers to shuttle the original

image directly to the deconvolution process. After we input a gray image into this network, it will pass the 8 blocks convolution layer, and then be restored to the original size by passing 8 blocks deconvolution layer.

Then we need to initialize a discriminator network to identify the fake image generated by the generator. We use a network structure similar to PatchGAN. The discriminator gets a image first and convolve it with convolution matrix(stride 2).After 3 convolution layers, we could get 256 images which size are 64*32.Then we convolve them with a convolution matrix(stride 1) and get 512 images which size are 63*31.Lastly, input these output feature maps into a full-connection layer to squeeze the image to 1*30*30.The discriminator will use the final output to distinguish real and false.

Step 3:

After created the network model and initialized the generator and discriminator network, we need to start the training process, the main task is to find the best parameters of the convolution matrix. For each of the input image A, the generator network would generate a fake image after it passes through all layers of the network. Then, we input the fake image and the real image that we hope the generator could generate into the discriminator network. The discriminator will answer whether the image is real or not while the supposed answer is false. If the outcome value of

the discriminator is not totally false, for example, 0.8 false, it would adjust its parameters of each convolution layer with the output error. The parameter adjustment method is similar to back-propagation. When input the error, multiply the error by its corresponding gradient, take the outcome as the new input error and pass it to the next node. After the parameter adjustment, the discriminator network would be more capable for identifying the fake image that the network generates.

As for the generator, if it gets a 'fake' answer from the discriminator, it would also adjust its parameters with back-propagation process. After the adjustment, the generator is supposed to generate images that could fool the discriminator in order to get the 'real' answer. As a result, the discriminator would be optimized if generator could fool discriminator, and the generator would be optimized if discriminator could identify the fake image. The network would be optimized to a decent one in the training process above. The objective (loss) function of previous cGAN can be expressed by:

$$\min_G \max_D V(G,D) = E_{x,y\sim p_{data}(x,y)}[\log D(x,y)] + E_{x\sim p_{data}(x),z\sim p_{data}(z)}[\log(1-D(x,G(x,z)))]$$

When training, the generator network tries to minimize the objective function while the discriminator tries to maximize it.To optimize the difference comparison better, we defined a new loss function that can be expressed by:

$$L = \lambda_p L_p + \lambda_a L_{GAN}$$

Where L_p is pix loss which refers to the standard deviation of the pixel between the generated image and the authentic image. L_{GAN} is adversarial loss which is from discriminator. λ_p and λ_a are pre-defined weights for perceptual loss and adversarial loss.

The L_p can be expressed by:

$$L_p(G) = \frac{1}{C \cdot W \cdot H} \sum_{c=1}^{C} \sum_{x=1}^{W} \sum_{y=1}^{H} \left\| \phi_E(x^{c,w,h}) - (y_b^{c,w,h}) \right\|_2^2,$$

where $\{x, y_b\}$ are the image pair we input with C channels, width W and height H, where x is the input image and y_b is the corresponding ground truth.

As for the discriminator, the binary cross-entropy loss for the discriminator D is defined as:

$$L_A = -\frac{1}{N} \sum_{i=1}^{N} (l_i \log(D(y_i)) - (1 - l_i) \log(1 - D(y_i))))$$

where N is the input images count (ground truth or de-rained result) and l_i are the corresponding labels (1 for “real”, 0 for “fake”).

During the training process, the generator and the discriminator would use the value L as reference to optimize their parameters.After the training is over, the network could produce a colored image that similar to the authentic image.

Step 4:

After all functions have been completed, our network demo is about to invoke them to make the website being connected to the background which is made by python. First, our website is rendered by "flask" in our background, then we can log on the website. Second, our website can load the image which is uploaded by our users, and get it filename as a parameter which is transformed into the function" transformIMG( ) ". Last, the website invoke the function to make the picture become colorful through the model we have trained, our users can see the colorful image on the website immediately and find it in the folder later.

Step 5:

We create an executable file to complete the interaction by using C#, a method differs from the website in step 4. Firstly, to guarantee to invoke python files in C#, we should configure the environment- download IronPython, pytorch and conda 8.0. Secondly, design the process which is shown in Fig 6.Click the" Color" button, the program will invoke transfer.py which contains the function " transformIMG( )". At last, encapsulate the program as an executable file which users can download and execute in their own computers. The result of executable file is that you choose an image locally, click the "Color" button and a colorful

image will be showed which can be download.The colored result in "Video" part is exhibited by GIF mode.

CLAIM

1. A method for fast images and videos coloring by using conditional generative adversarial networks, which applies GANs in the conditional setting, said generative combat network is a kind of learning model, and judging by the generator is composed of two parts, the goal of GAN is to learn the relationship between random noise vector Z and Y output image generator, to generate output can deceive false discriminator as the goal, to distinguish the true image for the task.

2. A method for fast images and videos coloring by using conditional generative adversarial networks as claim 1, which said the two parts are antagonistic to each other and are trained together.

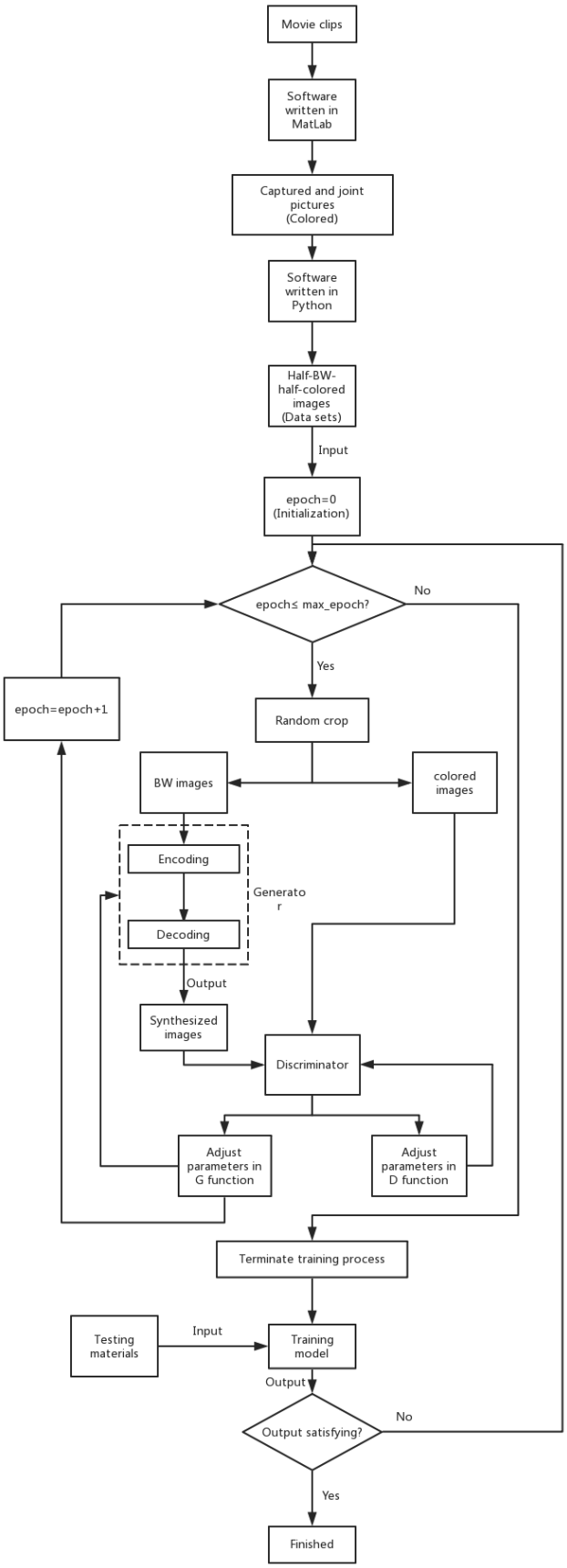
Movie clips
Software written in MatLab
Captured and joint pictures (Colored)
Software written in Python
Half-BW-half-colored images (Data sets)
Input
epoch=0 (Initialization)
epoch≤ max_epoch?
No
Yes
epoch=epoch+1
Random crop
BW images
colored images
Encoding
Generator
Decoding
Output
Synthesized images
Discriminator
Adjust parameters in G function
Adjust parameters in D function
Terminate training process
Testing materials
Input
Training model
Output
Output satisfying?
No
Yes
Finished

Figure 1

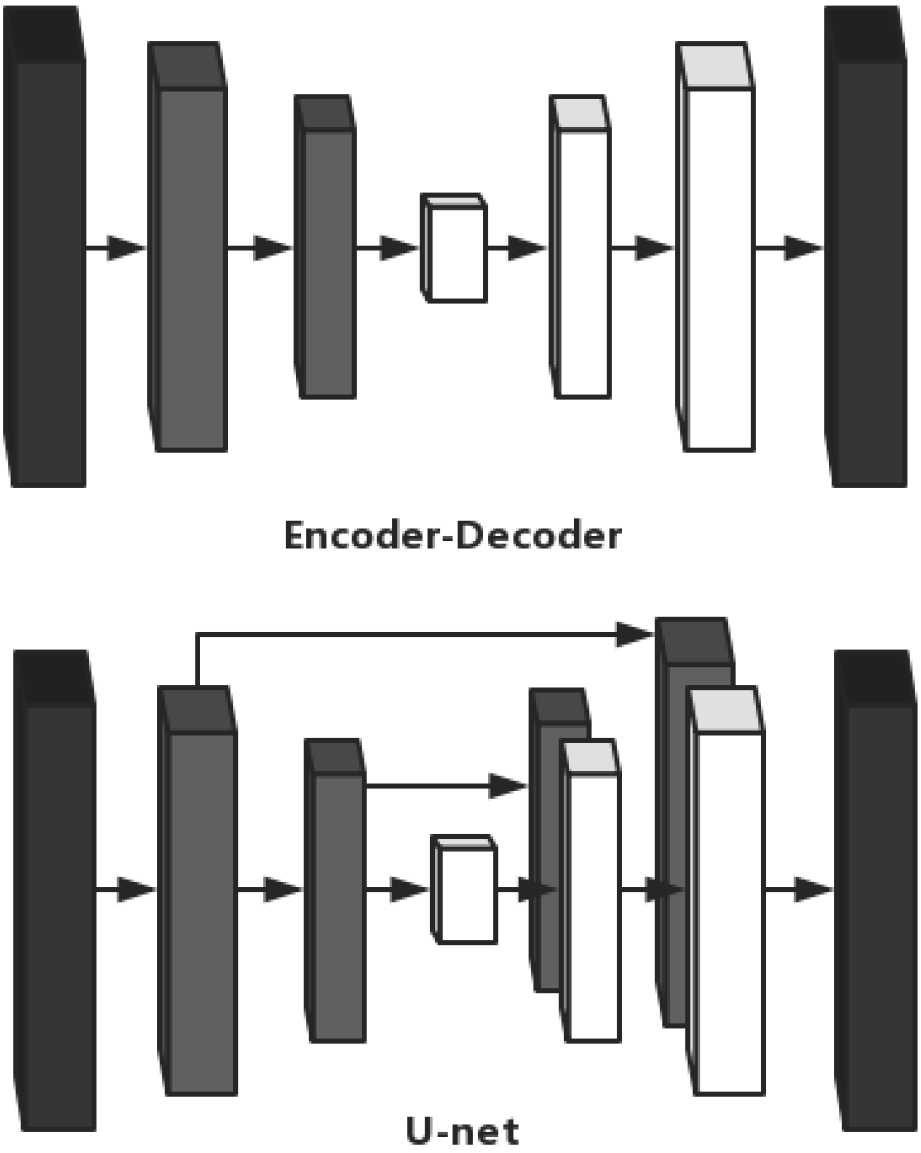

Figure 2

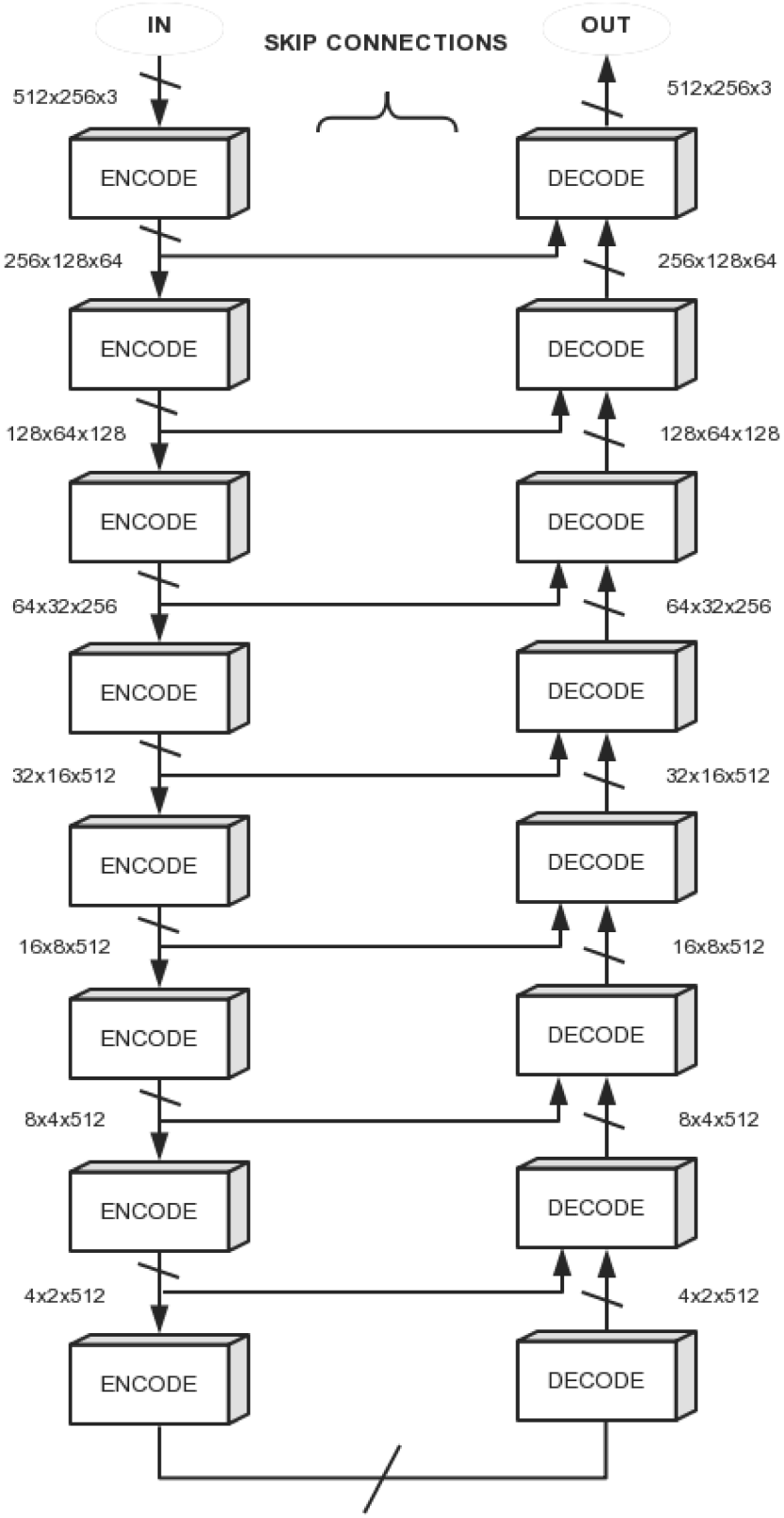
IN
SKIP CONNECTIONS
OUT
512x256x3
512x256x3
ENCODE
DECODE
256x128x64
256x128x64
ENCODE
DECODE
128x64x128
128x64x128
ENCODE
DECODE
64x32x256
64x32x256
ENCODE
DECODE
32x16x512
32x16x512
ENCODE
DECODE
16x8x512
16x8x512
ENCODE
DECODE
8x4x512
8x4x512
ENCODE
DECODE
4x2x512
4x2x512
ENCODE
DECODE

Figure 3

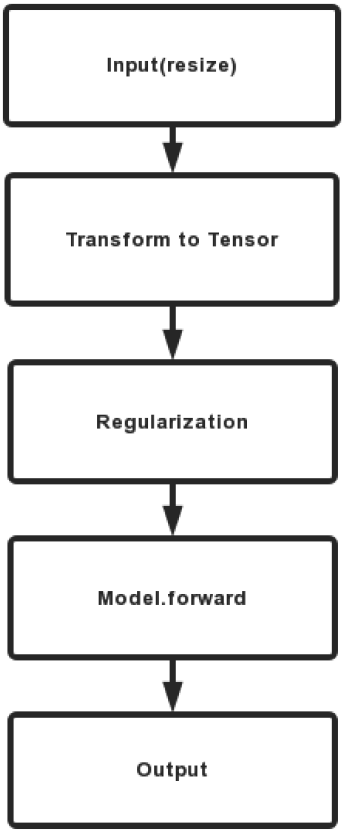

Figure 4

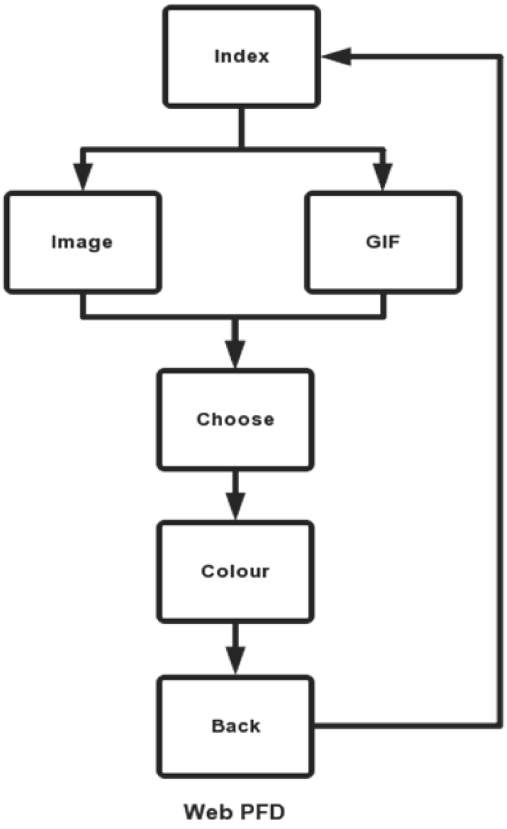

Figure5

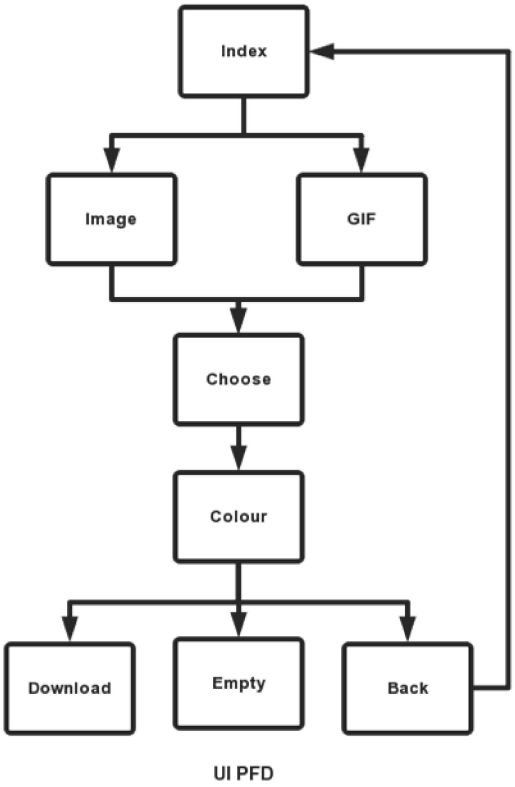

Figure 6

Figure 7