Generative Adversarial Networks(GANs) for improving classification and Neural style transfer

Yujin CHO, *Master student, University Paris-saclay, France*

***Abstract*—In computer vision, image classification is one of the traditional and basic deep learning problems. In many domains, deep learning technology is being used as a result of the recent rapid development of artificial intelligence technology. However, in general, it is difficult to collect enough data for training, and it is difficult to improve the performance of a deep learning model due to the difference in the number of data for each class. Various studies are being attempted to solve this problem, and one of them is to augment data. In this work, we want to show that the performance of a simple deep learning model classifier with limited number of data can be improved through data augmentation. A lot of research has been done on the Generative Adversarial Networks(GANs) model, which is the principle of this data augmentation for the source domain X and the target domain Y, simply generator G: X $\rightarrow$ Y generates data close to the original data distribution G(X). It is also applicable to various fields such as image to image translation. For unpaired images, an image close to the original can be created, and an image close to the ground truth can be restored F(G(X)) $\approx$ X through inverse mapping F: Y $\rightarrow$X. We can do Neural style transfer of images with this method. In this review, we intend to perform neural style transfer through the Cycle-Consistent Adversarial Networks and show the results.**

I. Introduction

THE aim of this work is to have better understanding in Machine learning/Deep learning concepts which are frequently used in computer vision fields. The project is composed of 2 parts. Convolutional Neural Network(CNN) is a multilayered neural network with a special architecture which allows to detect features of data. It has been used in many computer vision application such as image recognition, segmentation, style transfer and classification. In this paper, we are going to build a VGG-16 [2] based CNN for classifying images, to classify cow and horse with limited numbers of datasets. It is hard to get enough data for sufficient machine training in real life. By doing data augmentation implementing DCGANs [5], we show that we could improve classification accuracy.

In second part of the work, we show image to image translation for example cow to horse and vise versa. There have been many studies on neural style transfer. At the same time, Image to Image translation studies have been progressed from paired images into unpaired images. It is because obtaining paired data for training is expensive and difficult compared to unpaired images. CycleGANs [7] is a technique for training unsupervised image translation models via the GANs architecture using unpaired collections of images from two different domains.

Y. Cho is with the Department of Electrical Engineering, University Paris-Saclay, France, e-mail: (amycho92@naver.com).

II. Related works

Convolution Neural Networks(CNN) is a convolutional neural network with multi-layers neural network as one of the most widely used ML technique. A typical CNN architecture comprises alternative layers of convolution and pooling followed by one or more fully connected layers at the end(or a global average pooling layer). The convolutional layer is a composed of a set of convolutional kernels which convolves with the images using weights by multiplying its elements. Once features are extracted in convolutional layers, pooling layers allows to reduce the dimensions of the features map. It summarizes the features present in a region of the feature map.

Generative Adversarial Networks(GANs) is a deep learning architecture that estimates data distribution in probability point of view. It consists two networks: a generator G and discriminator D, which pits one against the other in order to generate new synthesis data that can be looked like a real data. [4]

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

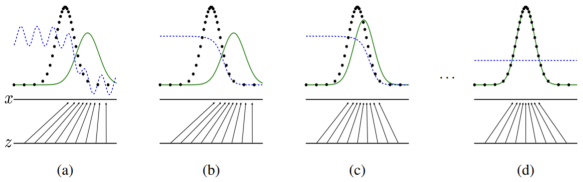

Figure 1: With training data distribution(black dotted line), the discriminator is trained to obtain the distribution of training dataset(green solid line).The discrimination distribution(blue solid line) is the difference of two distribution between generator and discriminator.At the end, The discriminator cannot distinguish two distributions [4]

Image-to-Image translation is a field of computer vision that aims to map input images and output images using a paired image train set. For example, it is possible to add color to a black and white photo, change a day photo to a night photo, or make a photo with only a border look like a real object. The pix2pix method [6] introduced in 2017 enabled image to image translation for unpaired images by competing and learning generators and discriminators using GANs. However, in reality, it is not always possible to construct a paired dataset, and CycleGANs [7] was introduced to overcome this limitation. The principal of CycleGANs is to change the style

of the photo, but only to the extent that it can be restored back to the original image. In other words, a constraint is placed so that the input image can be returned well by considering the return part rather than a simple mapping from the x domain to the y domain.

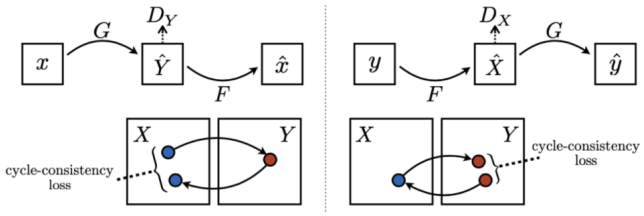

Figure 2: The model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$ with two discriminator D_X, D_Y. [7]

Total loss function is represented as Equation 2 considering minimize the loss between generated image and original image.

$$\begin{aligned}\mathcal{L}\left(G, F, D_X, D_Y\right) &= \mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) \\ &+ \mathcal{L}_{\text{GAN}}\left(F, D_X, Y, X\right) \\ &+ \lambda \mathcal{L}_{\text{cyc}}\left(G, F\right)\end{aligned}$$

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}\left(G, F, D_X, D_Y\right) \quad (2)$$

III. Implementation and Results

Classifier Network Architecture VGGNet(2014) is a widely used convolutional neural networks. It is similar to AlexNet [1] but with more layers(16 convolutional layers) and use 3x3 kernel size. In this work, we adapted the architecture for our purpose changing kernel size. Our classification problem is simple (find cow and horse), even though there are many kinds of CNNs have powerful performance, we choose VGG-16 thanks to good performance and simple and uniform architecture for adapting our datasets. As dataset images are 256×256 size, 2nd convolution layer of kerner size and stride size are adapted as 17 and 2 compared to the original VGG architecture. By using equation , size of convolution layer can be calculated where W is the input volume, K is the kernel size, P is the padding and S is the stride.

$$\frac{(W - K + 2P)}{S} + 1 \quad (3)$$

Classifier training details Datasets contain 41 images for each class for the training, and 41 images for each class for the test dataset. For the loss calculation, cross entropy is used, which allows to compare two distributions. Cross entropy loss increases as the predicted probability diverges from the actual label. Stochastic Gradient descent(SGD) is used for our linear classifiers. It has advantages of efficiency and easy to implement. However, it is known as sensitive to feature scaling. Due to limited number of small dataset and dissimilarity between train and test dataset, with 50 epochs and learning rate = 0.005 of training the accuracy is low about 74.39%. (Table I) 50 epochs of training was enough to converge loss near 0 and static learning rate is used as there are not many layers and model converge fast enough. We could tackle the bottleneck of limited dataset by data augmentation.

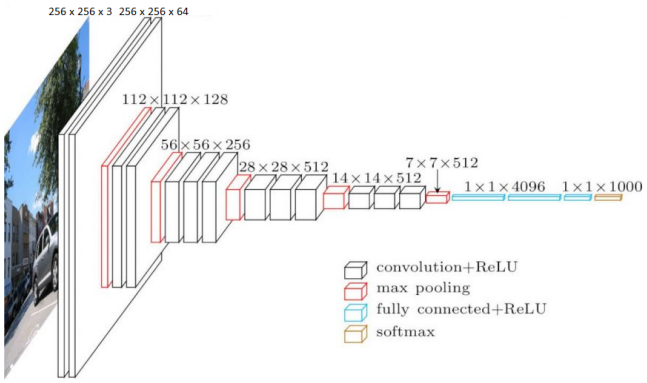

Figure 3: It represents VGG-16 based CNN architecture. The input conv1 layer of original VGG-16 is fixed to 224 x 224 RGB image and small kernel size(3x3x3) is used to have a deep layers. To adapt our dataset size, cov2 lyaer of kernel size and strides are adpated. [3]

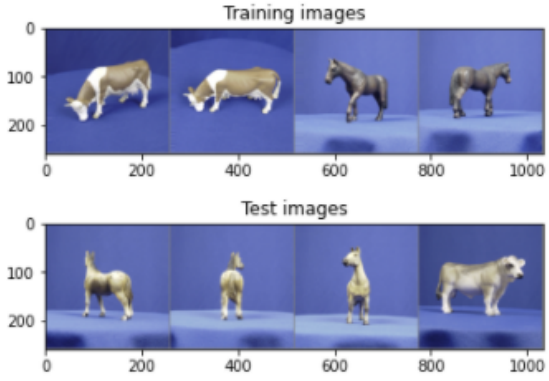

Figure 4: This figure shows training and test images of original dataset before data augmentation. As Training and Test images of cow and horse are dissimilar, it leads to low accuracy of classification.

TABLE I: Classification accuracy using VGG16 based

Accuracy	Before Augmentation	After Augmentation
Network	74.39%	100%
Cow	85.366%	100%
Horse	63.415%	100%

GAN Network Architecture DCGANs(Deep convolutional GANs) is used for the generator architecture. By introducing convolutional layer on GAN model, DCGAN allows us to train model with stability. There are some characteristics of DCGANs for example, there are no pooling layers to prevent not to lose data instead, uses convolutional stride. The generator layer is composed of ConvTranspose2d layers. These layers upsample the noise vector and transform noise into an image.(Figure 5) Batch normalization is used for every layer except output layer for the generator and the input layer of the discriminator. It helps to get stable learning and prevents overfitting. FC layer is removed because generator is not used as a classifier. All layers use ReLU except the last layer that

uses Tanh. Tanh allows the model to learn more quickly to saturate by adjusting range of the output.

DCGANs of discriminator has inverse architecture of the generator. However, it doesn't use the same 2d transpose layers because the discriminator is doing supervised task determine whether generated images are real or fake, not like generate images. For discriminator, the pre-trained(Resnet18 [10]) is used in this work. (Table II) The last FC layer of number of features in pre-trained model is linked into the output '1' to determine whether images are real or fake, not the classification task. By using pre-trained discriminator, it already have trained feature extraction layer for easier training.

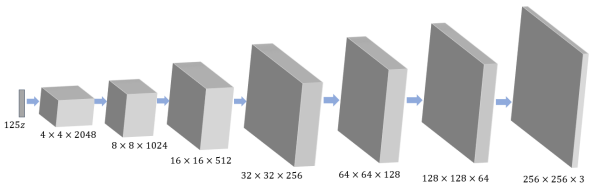

Figure 5: The architecture of DCGANs generator. 125 dimensional uniform distribution(z) is converted into $256 \times 256 \times 3$ pixel images by passing through fractionally-strided convolution layer.

TABLE II: Resnet18 architecture and output size
*mxpl stands for maxpool

Layer name	Output size	Resnet 18
conv1	$128 \times 128 \times 64$	7×7, stride 2
conv2	$64 \times 64 \times 64$	3×3 mxpl, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3	$32 \times 32 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4	$16 \times 16 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5	$8 \times 8 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
avg pool	$1 \times 1 \times 1$	7×7 average pool
fully connected	1	$512 \times 1000 \times 1$

GAN training details GANs are known for hard to train. The reason is that both generator and discriminator model are trained simultaneously in a game. There are several tips for the GAN training through many research work. First, Adam optimizer is widely used. It allows to achieve good performance with little hyper-parameter tuning and converge learning rates faster. Second, Batch norm layers are recommended for both generator and discriminator except the output of the generator and input to the discriminator. Third, It is known that activation function(ReLU) are used to address the vanishing gradient problem in deep convolutional neral networks. ReLU is recommended for the generator, and Leaky ReLU is preferred in the discrminators. It prevents dying state by allowing some negative values to pass through for gradient descent step being more efficient. To get a good image with various aspects, training was done 5000 epochs for 2 times for both cow and horse. Figure 6 shows generator loss and discriminator loss during training. We could observe that generator loss tends to increase as epoch increases, and discriminator loss tends to decrease close to zero. Thanks to the pre-trained discriminator, the discriminator loss is significantly lower from the start.

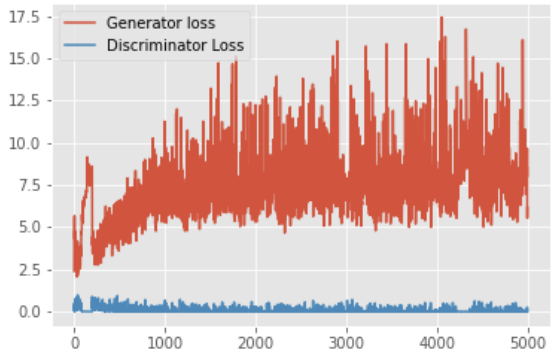

Figure 6: Generator loss and Discriminator loss according to increase of epochs. Generator loss tends to increase and discriminator loss tends to decrease near the 0.

The final result of images are saved that discriminator is fooled by fake data. The reason many epochs were needed for training was that the pre-trained CNN, discriminator (Resnet18), did not follow the recommended structure of the discriminator, and the number of datasets was significantly small compared to training epochs. Another cause is that the structure of the discriminator and generator is usually symmetrical, but it is estimated that training was difficult because it was different from the generator structure of DCGANs by using Resnet18. At the end of the training, 40 cow images and 58 horse images were augmented through training as shown in Figure 7, 8. As horse accuracy (63.415%) was lower than cow (85.366%) before data augmentation, more data of horse is added for training. Therefore accuracy improved into 100%. (Table I)

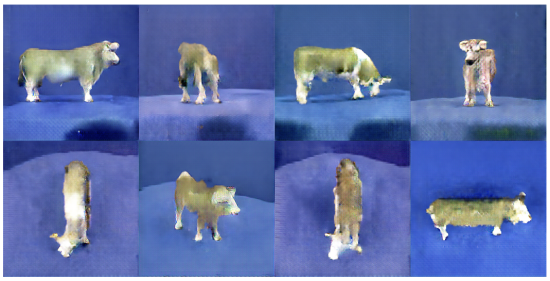

Figure 7: Cow images generated by GANs

Figure 8: Horse images generated by GANs

CycleGANs Architecture Genearator generates converted image as output based on input image. In CycleGANs, generator consists of three sections: Encoder, Transformation, and Decoder. (Figure 9). The encoder increases the number of channels while reducing the size of the input image. It is composed of 3 layer blocks including 2D convolution layer, Instance Normalization layer and Leaky ReLU layer. The first layer block does not change the size of the image and creates only 64 channels in the input. Each of the next two layer blocks doubles the number of channels while halving the input size. Then the adjusted input is passed to the Transformer. Transformer does not change its size while adding the desired function to the input. Residual block is included and is called ResNet. In this work, 9 residual blocks are used to fit with 256×256 size of images. Each ResNet has two layer blocks, where the transformed input is passed to the decoder. (Figure 10). It enables stable learning and produces good results for ill-posed problems. As a result of learning with ResNet, there is almost no change in the shape of the image, so the result image with good resolution can be obtained.

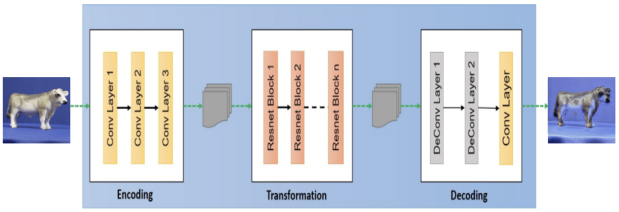

Figure 9: Generator architecture using Resnet for CycleGANs [9]. It is composed of 3 parts : encoder, transformer, decoder.

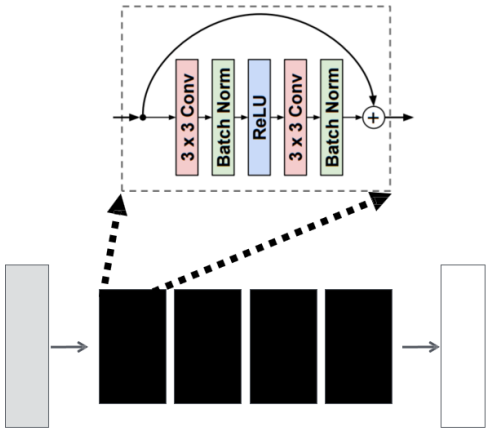

Figure 10: Resnet architecture used for CycleGANs

The decoder enlarges the size of the input and reduces all channels to RGB to form the final image. In general, as the number of channels decreases, the size of the input increases, and the 64 channels of 256×256 pixel data generated by two transpose convolution layers are supplied to the output layer, and the channels are reduced to RGB and output as the final output image.

The discriminator of CycleGANs was implemented using the PatchGAN method [8]. To determine whether the real image is fake or real. (Figure 11) Unlike a general discriminator that takes a whole, it first splits multiple input images into multiple patches and then judges each individually. The first four layer blocks cut the image size in half and double the number of channels. The input after the fourth layer has a size of 512 channels of 16×16. The last output layer block reduces the 512 channels to a single 16×16 matrix as the final output.

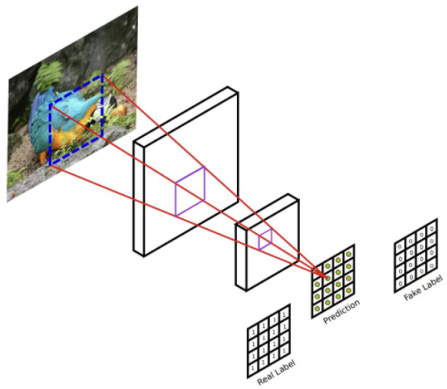

Figure 11: Discriminator architecture for CycleGANs. It judges the authenticity of images created by Generator in units of a specific size rather than the entire area. In other words, a patch size of an appropriate size that falls within the range in which the correlation relationship is maintained is determined, and the patches are mostly learned in the real direction. Patch size can be considered a hyperparameter because it must cover the overall image size and the appropriate range in which there is an association between a specific pixel and other pixels throughout the image.

CycleGANs training details Loss function is defined as Equation II. Adding identity loss in total loss function helps to preserve original data of color information. Identity loss is defined as Equation 4. Detail training description is as followed, all of GAN loss, Cycle loss and Identity loss are used. For the hyper parameters, 200 epochs and learning rate 0.0002 are used. Every 100 epoch, learning rate is set to be decreased for more efficient training. There are some remarkable techniques to implement CycleGANs. First, The weights are initialized with normal distribution(mean 0, std 0.02) to prevent exploding gradients and vanishing gradients problem. Second is to resize image with bicubic method. It is one of methods for interpolation to upsample images. When reading images, we use bicubic interpolation with resize. Third, replay buffer is used in discriminator part. As we know that training GANs are unstable, by showing the images generated by generator to discriminator periodically, we can stabilize the model.

$$\mathcal{L}_{\text{identity}}(G, F) = E_{y\ p_{\text{data}(y)}}[\|G(y) - y\|_1] + E_{x\ P_{\text{data}(x)}}[\|F(x) - x\|_1] \quad (4)$$

For training, dark brown horses and cow with patterns are used to learn features of horse and cow. After training 200

epochs, we evaluate with test datasets and Figure 12 is the result of style transfer. As shown in the picture, the cow has clearly changed to a dark horse color, the horse has a bright back and a speckled pattern of the cow. In Figure 13 case, cows are transferred into a horse style(dark brown) and horses are changed into a cow style(with pattern). The reason we don't see many difference for the horse results, it is because the feature of cow learned by our model is similar with the light color horse for discriminator to think it is a real. We can observe that CycleGANs doesn't change shape of the image, it only can change texture or colors.

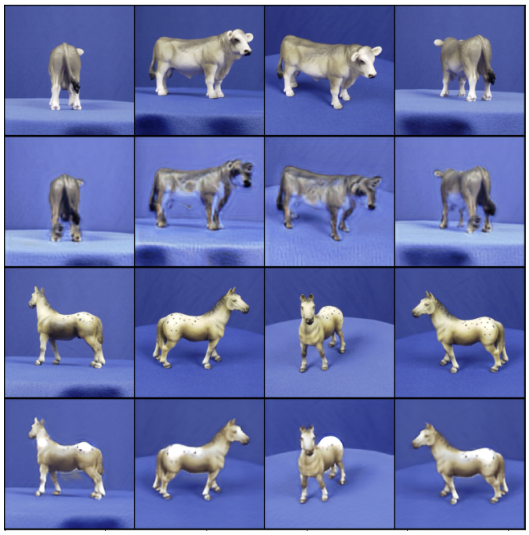

Figure 12: : Style transfer between cow and horse. First row pictures are original images of cow(train data) and second row shows generated cows with horse style. Likewise, Third row pictures(train data) are original image of horse and the final row is the result of generated horse with cow style.

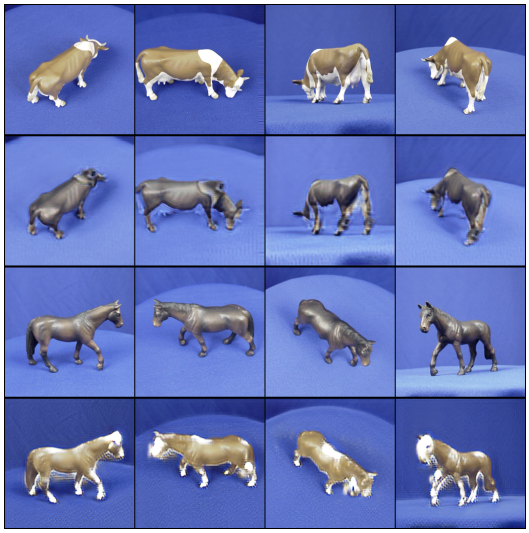

Figure 13: Second row is the result of style transfer of first row images(cow) and forth row is the result of the third row(horse).

IV. Discussion and Future work

In this project, we looked over deep learning techniques applied to the field of computer vision. In summary, in the first part, a simple VGG16-based CNN architecture was directly implemented to test the classifier accuracy. The accuracy was not high(74.39%). In general, there are two ways to improve the performance of the CNN model. As more layers are stacked, a deep structure with overlapping convolutional layers appears. The second method is to reinforce the training data with image augmentation. It is also possible to transform the training data using horizontal flip, rotate, etc., and include the data in the training data. There are also ways to augment data using GANs. Since there are many cases where the number of data is limited in real life, how to obtain and augment data is an important part of deep learning classification. CNN trains through a data set and learns features, so it makes good judgments about the already trained data, However when different data that have not been learned come in, it shows bad results. Therefore, through GANs, fake images that follow a distribution similar to the original were created, and the accuracy of the classifier could be improved. While training GANs, it is important to train the discriminator and generator in balance with each other. GANs are known to be difficult to learn, but it was difficult to always produce the same good output during training. Since the number of given datasets was insufficient, training of many epochs was required. With the augmented data (40 cows, 58 horses) through GANs, we could achieve 100% of the accuracy.

In the second part, style transfer between cow and horse was implemented by learning the image characteristics of cow and horse. It is called CycleGANs because style transfer is possible in both directions. In general, it is similar to the principle of GANs, but the image is transformed only to the extent that it can be restored to the original image from the additionally created image. In addition, good results were achieved by adding an identity loss to preserve the color of the original data described by the author of the paper [7]. For further improvement that can be done in the future is to optimize GANs part. As explained in the previous section, we could train easily by using a pre-trained discriminator, but it did not completely follow the structure of the original discriminator. We can apply LeakyReLU to the discriminator next time by following the tips of GAN training. Furthermore, we can think about the evaluation index of images generated through GANs. Among the images generated by the generator, images that the discriminator judges to be real are stored, and among them, pictures that look real to the eyes of real people are extracted. For the next extension of the study, it will be good to applying metrics to see whether the results are good or not.

V. Conclusions

Today, the field of computer vision is one of the most notable technologies in deep learning and artificial intelligence, and has developed into a unique field thanks to AI. Deep learning technology can be used to classify images into multiple categories, and images can be created to enrich the data as well as style transfer. By performing this project, it was a good opportunity to learn interesting algorithms. Moreover to have a good results, we need to fine tune well the hyper parameters.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in neural information processing systems, vol. 25, no. 2, 2012.

[2] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Computer Science, 2014.

[3] https://neurohive.io/en/popular-networks/vgg16/

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems, 2014

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in ICLR, 2016

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-toimage translation with conditional adversarial networks. In CVPR, 2017

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-toimage translation with conditional adversarial networks. In CVPR, 2017

[9] https://towardsdatascience.com/a-gentle-introduction-to-cycle-consistent-adversarial-networks-6731c8424a87

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition.", 2015