Obj2Obj GAN: Masked image to image translation for data augmentation

Aayam K. Shrestha, Hung Phan Manh Nguyen, Zhengxian Lin
Oregon State University
{shrestaa,nguyehu5,linzhe}@oregonstate.edu

Abstract

Effective data augmentation has been a long-standing problem in deep learning. One promising approach for this would be an image to image translation. Recent advancements in image-to-image translation for two domains have shown remarkable success for multiple target instances and signicant changes in shape. However, these approaches have limited scalability in handling more than two domains since different models should be built independently for every image domain pair. To this end, we propose a novel method, coined obj-obj GAN (oobj-GAN), that incorporates object features and target shape and background to generate object to object replacements. The proposed method employs an object conditional GAN with cyclic loss that encourages the network to preserve the object's important features while transforming the object to the target mask. We demonstrate our model's capability for data augmentation by generating plausible in-painted objects for the animal and fruits domain.

1. Introduction

Tremendous effort has been invested for efficient use of training data in different areas; including but not limited to, network architecture design, optimization methods, sampling methods among others. Nonetheless, data augmentation has been a center piece in the quest of extracting the most information from data-sets. Data augmentation has been shown to help models generalize better , remove biases [5], and converge faster. Case in point, the state of the art on CIFAR-100, without augmentation, the error rate of DenseNet [8] is lower than that of Resnet [7] by a large margin. However, after augmenting data, results of both methods are close.

Researchers use many strategies such as rotation, cropping, flipping, Shifting, elastic transformation, adding noise, PCA jittering, Color jittering, and some combinations. The common property of those methods is that they slightly modify the images, and hence the generated images are highly correlated to the original one. There are other approaches which try to crop objects based on their segmentation and copy those objects to different background. While creating much more distinct images from the original datasets, these approaches produce images that may look unreal, i.e images that come from other distributions rather than data's distribution. These approaches also require lots of engineering skills and sometimes domain knowledge to create images without accidentally changing important properties, e.g medical images, terrain images, etc.

Recent studies have shown that the power Generative Adversarial Network (GAN) [6] can also be exploited in Data augmentation task. [5, 2, 4, 16]. Approaches like [5] are specially interesting as it allows us to learn more about the mechanics of the model while exploring different data augmentation methodologies. What makes GANs stand out from previous methods is GANs can learn to generate new realistic data from the original one with the same properties in an unsupervised manner. This opens up a whole new dimension to data augmentation.

Given the current gap for object driven data augmentation methods and recent success of GANs in data augmentation, we propose a novel object driven data augmentation conditional Generative Adverserial Networks. Our approach allows us to swap any two objects in the same or different images while keeping the integrity of the background.

Our main contribution is two fold: an object conditional Generative Adverserial Network, Augmentation of target mask/instance using multimodal learning framework. First, we propose a neural network architecture that that is able to generate a new object with all relevant features of the object that it is conditioned upon. It is a domain independent approach, hence can generate any object that it is trained on. Second, we propose a multimodal approach to augment the target instance information into the GAN architecture. More specifically we encode the target mask as well as the conditioned object using the same encoding architecture in order to get a visually discriminative and more consistent embedding.

2. Related Work

CycleGan. [17] This architecture, depicted in figure 1, is used to convert data from source domain **X** to target domain **Y** and vice versa. For each domain, we have a generator to convert data from this domain to the other's and a discriminator to tell whether a sample come from this domain or not. However, because CycleGan can only convert data between 2 specific domain, in multi domain setting, the number of Generators and Discriminators increases drastically. **StarGAN**[3] takes one step further by incorporate target class embedding as a guide for Generator to know which domain G should convert the input to.

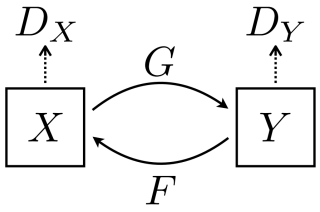

Figure 1. CycleGan architecture

InstaGan. [11]Both frameworks mentioned above share a weakness when the translation task requires significant objects' shape shifting. InstaGan addresses this challenge by introduce instance information, specifically segmentation maps of the objects. This information can help the generator pay more attention to the objects when translating and preserve background better.

Adversarial Text to Image Synthesis. [12] CycleGan, StarGan, and InstaGan all work on domain level, i.e as long as the translated image/object belonging to the target domain, the discriminator will accept the result. In contrast, our problem concerns more about the identity of the objects. Simply put, for an object **x** in Domain *A*, we want our model to translate **x** to a specific object **y** in Domain *B* rather than a random object **z** belonging to *B*. Hence, besides domain information encoded in the Generator, we still need to provide instance-specific/ identity information of targeted object. We can do this by simply conditioning both our Generator and Discriminator on the object's embedding. This is similar to the idea in *Adversarial Text to Image Synthesis*, where the author conditions the generator and discriminator on the text embedding in order to generate an object with desired properties in the description.

Object Driven Text to Image Synthesis (ObjGan).[9] takes an object-centered approach for text-to-image synthesis in complex domains. More specifically, it uses an object-driven attentive image generator along with an object-wise discriminator along with the objectdriven attention mechanism. At every stage, the generator synthesizes the image region within a bounding box by focusing on words that are most relevant to the object in that bounding box. The object-wise discriminator checks every bounding box to make sure that the generated object indeed matches the pre-generated semantic layout.

3. Background

3.1. Generative Adverserial Network

usually includes 2 parts, Generator and Discriminator. The Generator takes noise as input and tries to produce sample data. In contrast, the Discriminator takes both generated and real data and tries to build a policy to distinguish which distribution the data comes from. Both networks are optimized jointly under a minimax game based on function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (1)$$

In practice, **z** can come from any distribution other than just noise. In conditional gan [10], G takes both **z** and class embeddings as input to create sample data coming from desired classes. D also takes generated data and class embeddings. By doing this, we can mitigate the mode collapse problem, where the variation of generated data is small, but not completely .

3.2. Autoencoder

The autoencoder [1] architecture has been extensively used in many areas including nlp and computer vision. This architecture includes two parts: encoder and decoder - figure 2. Initially, the purpose of autoencoder is to learn a good representation called code of the input by forcing the decoder to reconstruct the input from the code generated by the encoder. Recently, the idea of autoencoder has been extended in a way that we can use the decoder to decode the intermediate code to a point in different domain rather than the input's domain. For example, we can use this architecture to generate segmentation masks for images [13], or translation [14].

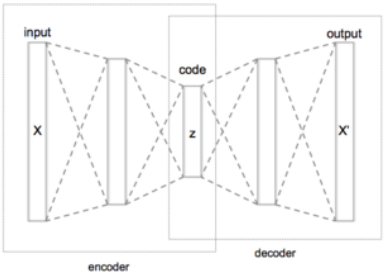

Figure 2. Autoencoder architecture

4. Method

In this section we briefly discuss several approaches we experimented with along with the approach that worked best for our work.

4.1. Vanilla Auto Encoder with swapped latent space

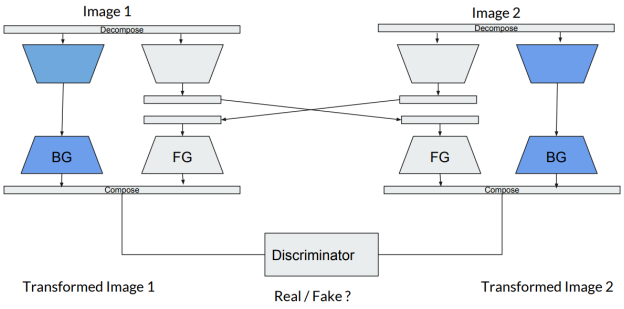

Figure 3. Auto Encoder with swapped latent space

In this approach, we use autoencoders to extract background and foreground features from each image. The foreground features are randomly permuted within a batch and matched with new background. The old background features with new foreground features then fed into the decoders separately and output is composed at the later endt. Since we don't have the ground truth for the new generated images, we can't use losses such as mean square error between the generated images and ground truth. To address this problem, we decide to add a Discriminator to guide our autoencoder to generate more realistic images.

Note: This approach does not perform well and fails to produce realistic looking results. Our inital guess is since the background and foreground generation process is running parallelly and one cannot affect the other, it is not able learn to generate coherent images. Moreover, it does not have a strong prior information about the target mask.

Algorithm 1 Vanilla Auto Encoder with swapped latentspace

procedure AE_SWAP(Img_1, Img_2)
  $Obj_1, BG_1 \leftarrow Decompose(Img_1)$
  $Obj_2, BG_2 \leftarrow Decompose(Img_2)$
  $EObj_1, EBG_1 \leftarrow Encode(Obj_1), Encode(BG_1)$
  $EObj_2, EBG_2 \leftarrow Encode(Obj_2), Encode(BG_2)$
  $\hat{Img_1} \leftarrow Compose(Decode(EObj_2), Decode(EBG_1))$
  $\hat{Img_2} \leftarrow Compose(Decode(EObj_1), Decode(EBG_2))$
  return $\hat{Img_1}, \hat{Img_2}$

4.2. Pix to pix with self attention

The previous approach, however, doesn't work quite well. We argue that simply swapping foreground latent code and adding a discriminator don't give enough guide for the decoder to construct realistic images from background and foreground information. The problem perhaps is because the new object is not well aligned with the old object, it's really hard for the decoder to figure out a way to modify the new foreground latent code and combine it with the background latent code. By saying 2 objects are not aligned to each other, we mean that they can be extremely different in shape, size, pose, or viewing angle.

In order to address this alignment problem, we propose using a attention mechanism to adaptively choose a proper region from object image to fill in the hole left by old object in our background based on where we want to copy to in the background.

We borrow the idea from the Non-local Neural Networks paper [15] and modify the structure to allow it to use background feature as the query and both background and foreground as the value.

However, the model only fill every pixels of the hole in the background with the same thing. Our assumption is that maybe because the black region is not visually discrimnative from each other, in order to add the new object to in the background, the black pixels doesn't provide concrete information. And hence the features learnt are not discriminative enough for the attention module to choose different pixels on the foreground.

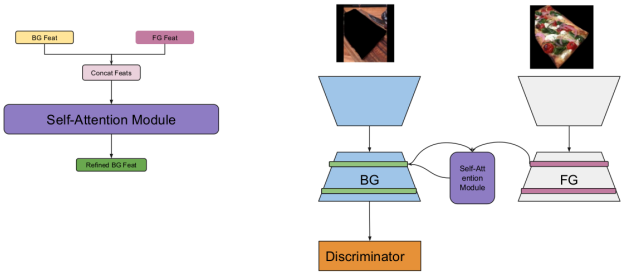

Figure 4. Pix2Pix with self attention

Algorithm 2 Pix to pix with self attention

procedure AE_SWAP_ATTENTION(Img_1, Img_2)
  $Obj_1, BG_1 \leftarrow Decompose(Img_1)$
  $Obj_2, BG_2 \leftarrow Decompose(Img_2)$
  $EObj_1, EBG_1 \leftarrow Encode(Obj_1), Encode(BG_1)$
  $EObj_2, EBG_2 \leftarrow Encode(Obj_2), Encode(BG_2)$
  $\hat{Img_1} \leftarrow Compose(DwA(EObj_2, EBG_1), Decode(EBG_1))$
  ▷ DwA: Decode with Attention
  $\hat{Img_2} \leftarrow Compose(DwA(EObj_1, EBG_2), Decode(EBG_2))$
  return $\hat{Img_1}, \hat{Img_2}$

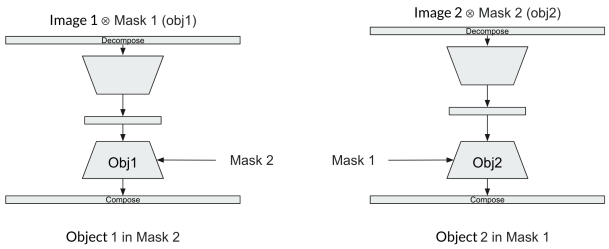

Figure 5. Auto-encoder with feeding mask

4.3. Auto-encoder architecture with feeding modified mask

This is our approach three. After we try the previous two approaches,we are intent to start at a simple model as baseline. The fig.4 is our architecture. We create an auto-encoder that takes the object as input. After got the latent layers, we feed the modified mask into each layer of decoder. The way we modified the mask to randomly do the translation, resizing, and rotating. Also, the target object is corresponding to the shape of mask.

When we was training the model, we feed the modified mask that is corresponding to the input image, and the ground true is the modified image. We was hoping that the decoder will learn how to deal with shape of object, so that if we give the different mask, it will generate the same object with different shape. Then, we can pass object from image A and mask from image B into our model. The output can be paste into image B, and also do the same way with image B to image A.

This approach kind of works. If the shape and pattern is easy to be deal with, the result is not bad. However, our goal is not just trying to swap the simple object, we want to swap real object that has many different shape and noises. It is successful to generate a good shape, but not a reasonable object.

Algorithm 3 Auto-encoder architecture with feeding modified mask

procedure AE_MASK($Img_1, Img_2, Mask_1, Mask_2$)
  $BG_1, Obj_1 \leftarrow Decompose(Img_1, Mask_1)$
  $BG_2, Obj_2 \leftarrow Decompose(Img_2, Mask_2)$
  $EObj_1 \leftarrow Encode(Obj_1)$
  $EObj_2 \leftarrow Encode(Obj_2)$
  $TObj_1 \leftarrow Decode(EObj_1, Mask_2)$
  $TObj_2 \leftarrow Decode(EObj_2, Mask_1)$
  $\hat{Img_1} \leftarrow Compose(BG_1, TObj_2)$
  $\hat{Img_2} \leftarrow Compose(BG_2, TObj_1)$
  return $\hat{Img_1}, \hat{Img_2}$

4.4. object conditional GAN with object autoencoder

The auto-encoder is very powerful architecture to compress state into a latent feature map vector. The feature map have to include enough information about the input data, So that the decoder can figure out how to reconstruct the input data. Our approach four is to use these feature map as input for a conditional GAN. Hopefully, the generator will figure out how to generate a image base on the condition.

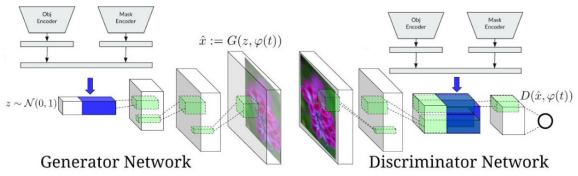

Figure 6. Object Conditional GAN

Finally, we approach the problem from a different perspective exploiting a few ideas from multi-modal learning. Here, we treat the object and the target images as two different modes of input and employ a multi-modal framework. In this approach, we train a deep convolutional generative adverserial network (DC-GAN) conditioned on object encoding and target mask encoding from pre-trained autoencoders. both the generator network and the discriminator network is performing inference conditional upon the concatenated and compressed encoding of the object and the target mask.

Object/Mask AutoEncoder. We crop and resize all objects/mask in the training dataset to a dimension of 224X224, then train an autoencoder on it. The small bottleneck layer of the autoencoder along with augmentation in training dataset encourages the autoencoder to learn only the important features of the object. The encoder network comprises of Resnet-18 backbone layers where the first and last layer is swapped for a new convlutional layer and a fully connected layer respectively. The convlutional layer has a channel size of 24 with a kernel size 7. The fully connected layer takes input from the adaptive average pool layer and uses a Tanh activation function. The encoder network as a whole converts a 224X224 image to a 128 bit vector encoding. The decoder network is relatively simple and comprises of 2 fully connected layer complemented with 5 convolutional layers. It takes the 128 vector embedding and tries to recreate the actual 224X224 input image.

Generator. Here, we start by encoding the object as well as target mask using the object and mask encoder respectively. Then we compress the concatenated encoding to a 128 bit vector using a fully connected layer. Following this,

the compressed representation along with a 100 bit sample from noise prior $z \in R^Z \sim N(0,1)$ is fed to the feed forward deconvolutional network; the output of which is a generated image of the object conditioned on the noise and latent representation of the object and the target mask.

$$\hat{x} \leftarrow G(z; [\chi(O_s), chi(M_t)])$$

Discriminator. the discriminator applies several layers of stride-2 convolution with spatial batch normalization followed by a leaky ReLU. We also apply the same compression method as generator. when the spatial dimension of the discriminator is 8X8 we perform a depth concatenation with the embedding and perform a 8X8 convolution to compute the final score from D. Batch normalization is performed on all convolutional layers.

5. Results

5.1. Tie

We first experimented with Tie dataset. Our assumption was that it is a easy dataset to generate as most of them play around with texture and patterns. the results shown are from approach 3.

Original Image A	Original Object A	Original Background A	Generated Object B	Composed Generated Image B
Original Image B	Original Object B	Original Background B	Generated Object A	Composed Generated Image A

Figure 7. Result Grid Semantics

Figure 8. [TOP]: Original Image (Red Tie) –>Composed output Image (Blue Tie) [Bottom]: Original Image (Blue Tie) –>Composed output Image (Red Tie)

5.2. Pizza

The second experiment we performed was on pizza, the notion was to capture more texture and variation in the data. As seen in results the texture in the source object is captured well but the model is over-fitting to the spatial characteristics of the object and target mask.

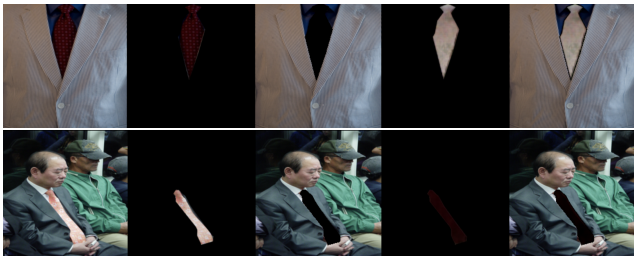

Figure 9. [TOP]: Original Image (Black Tie) –>[Bottom]: Composed output Image (White Tie) [Bottom]: Original Image (White Tie) –>Composed output Image (Black Tie)

Figure 10. [TOP]: Original Image (Berry Pizza) –>Composed output Image (Vege Pizza) [Bottom]: Original Image (Vege Pizza) –>Composed output Image (Berry Pizza)

Figure 11. [TOP]: Original Image (Salami Pizza) –>Composed output Image (Mushroom Pizza) [Bottom]: Original Image (Mushroom Pizza) –>Composed output Image (Salami Pizza)

5.3. Horse and zebra

In our third experiment tried a difficult problem of Horses and Zebras, to test the effectiveness of our model. it perfoms fairly well but the results are blurry and lacks details.

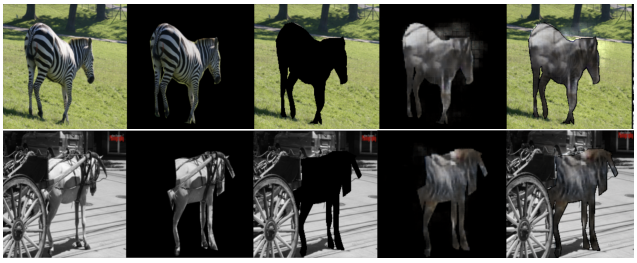

Figure 12. [TOP]: Original Image (Zebra) –>Composed output Image (Horse) [Bottom]: Original Image (Horse) –>Composed output Image (Zebra)

Figure 13. [TOP]: Original Image (Horse) –>Composed output Image (Zebra) [Bottom]: Original Image (Zebra) –>Composed output Image (Horse)

5.4. apple vs orange vs Donuts

Here the model performs fairly well for this domain and can generate the best results among the experiments.

Figure 14. [TOP]: Original Image (Orange) –>Composed output Image (Apple)[Bottom]: Original Image (Apple) –>Composed output Image (Orange)

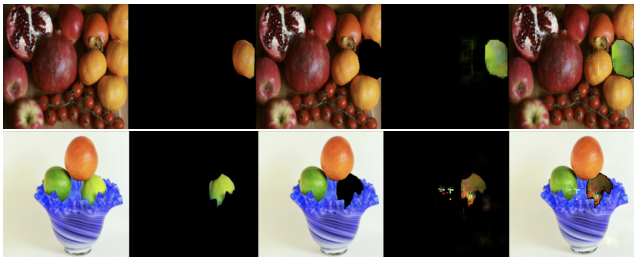

Figure 15. [TOP]: Original Image (Orange) –>Composed output Image (Apple)[Bottom]: Original Image (Apple) –>Composed output Image (Orange)

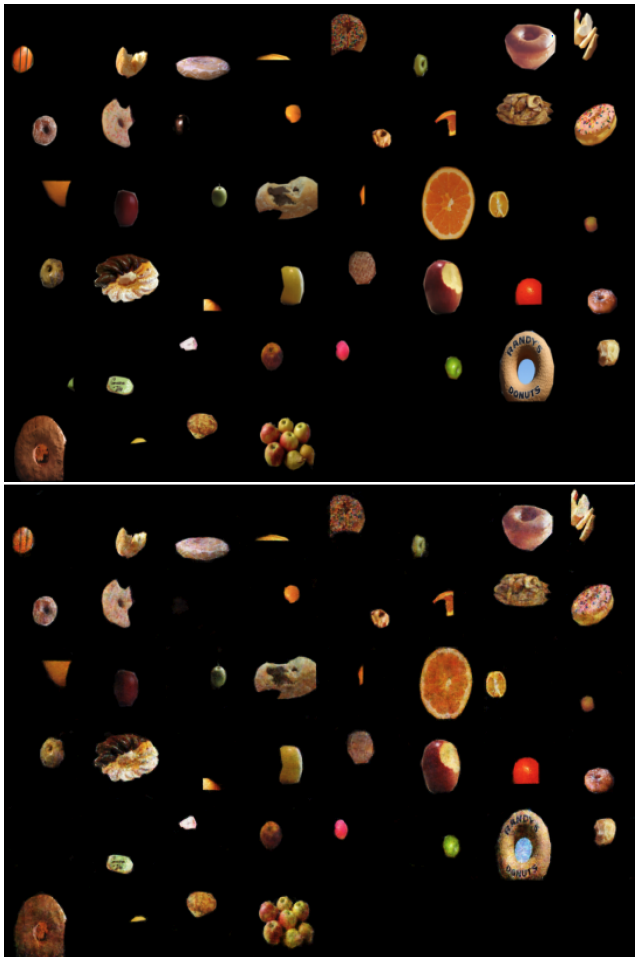

Figure 16. [TOP]: Original Images (Mixed Examples) [Bottom]: Generated Images (Mixed Examples)

6. Conclusions

In this project, we introduce 4 different approaches to tackle the problem of object driven data augmentation. The first two approaches do not show any good results. Although approach 3 shows reasonable results it was not robust enough to handle spatial data well. Approach 4 with multimodal framework and conditional DCGAN shows the most promising results among all 4. Since the last approach, uses conditional GAN; conditioning on object's embedding as well as target mask, it provides larger room for future enhancements; among them, addition of cyclic GAN looks most promising. Unfortunately, this method in its current from can't still handle large shift in object's shape and pose; and fails to capture a bigger spectrum of object features in the transformation. In the future, we expect to improve the results by including additional losses such as cycle loss and identity loss and develop an architecture to smooth the transition between the background and new object to make it more natural.

References

[1] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *ICML Unsupervised and Transfer Learning*, 2012.

[2] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. N. Gunn, A. Hammers, D. A. Dickie, M. del C. Valdés Hernández, J. M. Wardlaw, and D. Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *CoRR*, abs/1810.10863, 2018.

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. 2017.

[4] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 289–293, 2018.

[5] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR*, abs/1811.12231, 2018.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

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

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016.

[9] W. Li, P. Zhang, L. Zhang, Q. Huang, X. He, S. Lyu, and J. Gao. Object-driven text-to-image synthesis via adversarial training. *CoRR*, abs/1902.10740, 2019.

[10] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014.

[11] S. Mo, M. Cho, and J. Shin. Instagan: Instance-aware image-to-image translation, 2018.

[12] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis, 2016.

[13] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[14] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[15] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks, 2017.

[16] Y. Wu, Y. Yue, X. Tan, W. Wang, and T. Lu. End-to-end chromosome karyotyping with data augmentation using gan. *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2456–2460, 2018.

[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.