Department of Computer Science

Submitted in part fulfilment for the degree of
MSc in Software Engineering.

Predicting Memorable Regions Of Images Using Deep Learning and Adversarial Networks

Babar Khan

Version 1, 2023-April-11

Supervisor: Adrian Bors

Contents

Contents

1 Abstract

2 Executive Summary

3 Introduction

3.1 Background

3.1.1 We see a lot of images

We are constantly surrounded by imagery, on the way to work, on the internet, on TV, in stores, etc. Some of them stick out more than others, we see hundreds, if not thousands, of images a day, and yet culturally and individually we all remember the same ones.

3.1.2 Cultural vs Individual Memorability

Due to cultural significance an image can become memorable. The 2015 dress [1], a country's flag, or the 1932 image "lunch atop a skyscraper" [2], come to mind. I would argue that the memorability of these images is tied to the culture surrounding them, not necessarily due to intrinsic properties within. The focus of this research is on the intrinsic properties within an image that make it memorable to an individual, like seeing an advert on a bus and then later recognising the same advert online.

3.1.3 How do we measure the memorability of an image

It has been shown that the memorability of an image is an intrinsic property, independent from the viewer [3]–[6]. This was achieved by performing a memorability game where participants are presented with a stream of images, and on an interval shown an image that they had already seen. Most participants were able, or unable, to remember the same images. How often an image was recognised is proportional to its memorability. This is taken further by Akagunduz et al. [7] where, in a similar experiment, they measure which regions of images are memorable.

3.1.4 Why do we want to create a memorable image

3.1.5 How does this research help to create a memorable image

3.2 Motivation

3.3 What are my goals

3.4 Structure

4 Literature Review

4.1 Memorability

In R. Breners work [8] subject's abilities to remember a series of units was tested, a unit was different depending on the test,

> In the digit test, for example, each digit was a unit; in the sentence test each sentence was a unit, etc. ... Each nonsense syllable constituted a unit ... Each consonant constituted a unit ... Each [geometric] design constituded a unit. [8, p.468]

It was found that people are not very good at remembering nonsense sylables or sentences but are much better at remembering digits, consonants, and colours. Remembering geometric designs is placed somewhere in the middle. This is of interest because it's very similar to our investigation into what aspects make images memorable. Brener was interested in consonants, colours, geometric designs, etc but the finding that different units can be significantly harder or easier to remember is useful to us. If we swap out units for image properties such as texture, contrast, saturation etc, or even more abstract features that a CNN may recognise, then it should be possible to find how memorabilty is impacted by these.

The mean score of 5.31 found for geometric designs seems to indicate that people are not great at recalling images, however in R. Nickerson's work [9] we can see the opposite, subjects are found to be extremely good at recognising images. Nickerson found that subjects shown a series of images are, with great accuracy, able to recall if the photo is one they have seen before or not. An item is referred to as 'new on its first occurrence and old on its second occurrence'[9, p.156], it was found that subjects shown one image at a time, each with equal chance of being old or new, are able to correctly distinguish them with 95% accuracy.

Like in [9], R. Shephards work [10] also found that subjects are able to distinguish with great accuracy new and old images, the mean percentage of images correctly identified was 99.7% after a delay 2 hours, 92.0% after a delay of 3 days. 87.0% after a delay of 7 days, and 57.7% after a delay of 120 days. The introduction of a delay into Nickerson's experiment allows

us to see that regardless of whether the image is in our short-term memory (2 hour delay) or our long-term memory (3-120 days) subjects are able to correctly identify an incredible amount of images.

L. Standing built on the work of Nickerson, Shepard, and Brener in [11]. Four experiments were ran which tested memory capacity and retreival speed for pictures and words, I am interested in the performance related to pictures specifically. He found that the capacity for image recognition from memory is almost limitless, when measured under testing conditions. In Standing's first experiment he tested 'normal' and 'vivid' images. He describes 'normal' images as those that 'may be characterized as resembling a highly variegated collection of competent snapshots'[11, p.208], and 'vivid' images are described as 'striking pictures ... with definitely interesting subject matter'[11, p.208]. Much like how in Brener's work [8] the memorability of an object was variable based on what classification of unit was tested, this work shows that on an even more specific level, the classification of the image into normal or vivid, has an impact on subjects memorabilty, Standing found that the vivid images were more likely to be remembered by subjects.

Lots of work has been done to further build on that of Standing, Nickerson, Brener et al. T. Brady et al. [12] performed research into how much information is retained in long term memory and found that subjects are able to remember not just the gist of an image, but also fine grain information such as the state of objects within an image, and that they are able to distinguish between variants of objects shown in images. An example shown is an abacus in two different states, 13/14 subjects were able to distinguish which one they had seen before.

T. Konkle et al. build further on the work in [11], [12] by studying the impact that categorical distinctness has on memorability. This was tested by creating a dataset of images where composed of categories such as tables, cameras, and bread etc. Each category had between 1 and 16 images, a memory test like those performed in [11], [12] was performed and the percentage correctly identified was found to decrease as the number of images within a category increased. From this we can see that categorically distinct images are more likely to be remembered.

Studies by Isola et al. [3], [4] were performed with the goal of identifying a collection of visual attributes that make images memorable, and to use those to predict the memorability of an image. It was found that properties such as mean saturation, and the log number of objects, has less impact on the memorability score than object statistics. Categories such as: person, person sitting, and floor, were most helpful image memorability. The categories: ceilings, buildings, and mountains, were least helpful. Their approach was limited by the fact that the object statistics were annotated by

hand, this would both make automating the process of determining image memorability impossible, and limit them from finding any abstract properties that helped/hindered memorability.

Khosla et al. [13] built on the work in [3], [4] by, instead of determining memorability of an entire image, creating a model that discovers memorability maps of individual images without human annotation. These memorability maps are able to distinguish which areas of an image are remembered, forgotten, or hallucinated. Their approach, similar to Isola et al. in [3], [4], is limited by the arbitrarily picked list of features that define memorability.

4.2 Using Deep Learning to Predict Memorability

4.2.1 Memorability Datasets

Khosla et al. have created a dataset, LaMem [5], containing 60,000 annotated image and memorability pairs. They use deep learning to predict these memorability maps.

This is similar to the VISCHEMA plus dataset [7], which I will be working with. In [7] the network output utilises fully connected layers which I believe is unnecessary, and may even hurt performance. More modern computer vision network structures, such as those in [14]–[18], use fully convolutional networks. These maintain spatial locality which allows them to generalise better.

In both [5], [7] they present models that predict memorability maps of the images in the datasets presented, they do this by training a classifier to predict the memorability score of an image, and then to split the image into multiple sections which they then predict the memorability of producing a lower resolution memorability map.

I instead propose to use an Image-to-Image model such as those presented in [14], [16], [17] to take an image from the dataset and predict the memorability map of it from that.

Using an Image-to-Image model we should be able to predict memorability maps of images by training a fully convolutional network across a dataset such as those presented in [5], [7].

4.2.2 Image Synthesis Models

Autoencoders

The autoencoder model is composed of 3 parts, an encoder, a bottleneck, and a decoder, these are typically used for compression. The U-Net [14] is a variation of an autoencoder which has also found great success in image segmentation within the medical field. It makes use of residual connections between corresponding layers in the encoder and decoder blocks, these allow the model to preserve data for use in the decoding stage. This architecture is the backbone of many image to image models [17]–[19], and has seen many extensions [20], [21]

Generative Adversarial Networks

GAN architectures[15] use competitive co-evolutionary algorithms where a Generator and a Discriminator compete. The Generator is typically given a latent space vector and uses that to produce an image, the Discriminator has to determine if that image belongs to a given distribution. These networks have seen great success [17], [22], [23] but are typically unstable and require a lot of parameter fine tuning. They can also suffer from: non-convergence, mode collapse, and diminished gradients [24].

Diffusion Models

Diffusion based generative models work by learning to iteratively remove Gaussian noise from a sample T times until it produces an image from the training domain. The model was first proposed by Sohl-Dickstein et al. [25], and has been further developed by Ho et al. [16], Dhariwal and Nichol [19], and Saharia et al. [18].

Diffusion models, first introduced by J. Sohl-Dickstein et al. [25], are a machine learning model that work through a forward diffusion process systematically destroying the structure in a data distribution, and the learning of a backwards process to restore the structure. The method uses a Markov chain to convert x_t into x_{t-1}. Starting with x_T, a sample of Gaussian noise, a generative Markov chain converts this into x_0, which is a sample from the target data distribution. Because the model only estimates small pertubations of noise, x_t given x_{t-1} , rather than an entire transformation from x_0 to x_T, it is tractable to train.

The DDPM, a UNet based diffusion model,[16] is capable of producing

high quality images and achieves state of the art FID scores across the CIFAR10 dataset. Dhariwal and Nichol [19] show tweaks that allow for diffusion models to achieve state of the art FID scores across the ImageNet dataset and when used in combination with upsampling diffusion further improve FID scores. They do this by using improvements proposed in [23], [26]–[29]. These improvements also reduce the number of noise steps required from thousands to (in some cases) 50. Through the decrease in noise steps they are able to reduce the amount of time that it takes to generate an image. Chen discusses in [30] how changing the resolution of an image has an impact on the noise scheduling required, he finds that the optimal scheduler at a smaller resolution may cause under training for higher resolution images. Multiple strategies are proposed to adjust noise scheduling. Firstly, changing the noise schedule functions to those based on cosine or sigmoid, with temparature scaling. Secondly, reducing the input scaling factor from 1 increases the noise levels which destroys more information at the same noise level. They then combine these into a compound noise scheduling strategy.

5 Methodology

5.1 Requirements Capture

The VISCHEMA dataset[Vischema paper] contains image to vms mappings. I aim to use a deep learning model to learn a mapping of these images to their corresponding labels. Our model should learn a general understanding of the mapping such that when it is provided with an image that matches our distribution, it can accurately create a VMS label for it. Our model will need to learn to create accurate mappings and we can test that through a loss function, such as L1, and through qualitative analysis.

5.2 Motivation

As standard backpropagation, GANs, and diffusion all produce images in different ways I think it would be interesting to compare how the 3 of them perform when asked the same task. I will run three experiments: I will train a UNet, a GAN, and a diffusion model to produce a VMS label given an image from the dataset. I hope to be able to find the strengths and weaknesses of each of these systems in this application.

For each system I will experiment with network parameters and training hyperparameters to fine tune models that produce the best output. I will experiment with training the models to produce the sum of the images and the VMS labels, and with training the models to produce just the VMS label. In the former the VMS label can be calculated from the output.

In each experiment I will automate a system to: test the use of different values for the number of layers and channels in each network, vary the normalisation method, optimisation function, learning rate, and batch size to find the best training environment and model parameters. As it is not be feesible to test every combination of these variables, I will employ a training strategy to test every variation of a single parameter/hyperparameter while keeping the rest constant, pick the best scoring variation of that parameter, and move on to the next. I will start with choices that I estimate to be good,

as this should only improve on them. This will bring the size our search space down by an order of magnitude, however we won't be exploring the entire search space and will potentially miss out on good values. This will be especially necessary in experiment 2 as training within a GAN is typically unstable [papers showing instability in GANs]. In experiment 3 I will also vary the noise scheduler and the beta values.

5.3 Experiment 1

Model: This network is a UNet autoencoder that takes as input a 64x64x3 tensor of floating point values in the range [-1,1], these are the images in our dataset. It outputs a 64x64x3 tensor of floating point values in the range [-1,1], these are our label estimates. This model can be described as:

$$L_{pred} = model(I)$$

Model Loss: This is simply the L1 loss between L_{pred} and L_{real}, describing how closely the output of our network matches the corresponding label. At each epoch we will calculate this over the training dataset, use that for backpropagation, and then calculate it across the validaton dataset to test how well we have generalised.

$$L = L1(model(I), L_{real})$$

Training Strategy: Please see algorithm 1 for the training loop.

Algorithm 1 UNet Autoencoder Training Strategy

for `every epoch` **do**
  for I, L_{real} `in training dataloader` **do**
    $L_{pred} = model(I)$
    $loss = L1(L_{pred}, L_{real})$
    `Update weights of model with backpropagation`
  end for
end for

5.4 Experiment 2

I will train a conditional generative adversarial network to generate images of VMS maps given images from the VISCHEMA dataset. I will use an

adapted Pix2Pix network [17], making tweaks that I think will increase performance. Pix2Pix by Isola et al.[17], is an image-to-image conditional generative adversarial model based on the UNet [14]. Designed to translate images from one style into another. In [31] M. Arjovsky et al. introduce a GAN variant based on the Wasserstein distance between the output distribution and the image distribution, the benefit of this is that the Wasserstein distance is continuous and differentiable almost everywhere, reducing the risk of dimminishing gradients. In [32] N. Makow investigated the use of Wasserstein Distance in the Pix2Pix model but unfortunately found that it does not perform much better, I would still like to experiment with it as VISCHEMA plus is different from any dataset used in [17] and as Makow states they were unable to perform a complete hyperparameter search, meaning that its possible we could achieve greater results than vanilla Pix2Pix. I will be adapting the following Pytorch implementation [33].

Lots of work has gone into making GAN models more stable and I will use these findings in my own models. In [29] A. Brock et al. found that when 'increasing the batch size by a factor of 8 ... models reach better final performance in fewer iterations, but become unstable and undergo complete training collapse'. Because of this I will experiment with early stopping and low batch sizes. In [34] Z. Shengyu et al. show how using differentiable augmentation on your images increases the quality of the outputted images. With a dataset with as few as 100 images they are able to produce high quality outputs, this is useful to us because the VISCHEMA plus dataset only has 1280 training images, which typically wouldn't train very well on a GAN.

Generator: This network takes as input a 64x64x3 tensor of floating point values and outputs a 64x64x3 tensor of floating point values in the range [-1,1]. This model can be described as:

$$L_f = G(I)$$

Discriminator: This network takes as input a 64x64x6 tensor of floating point values. Channels 1,2, and 3 store the image and channels 4,5, and 6 are its corresponding label, either real or generated. It outputs a 4x4x1 tensor of boolean values. Each value in this output represents a 16x16x6 region of the input.

The loss for each network is computed as described in [pix2pix paper], across an entire batch of images and then the weights are adjusted with backpropagation. The optimiser used is one of the hyperparameters that we will search for.

Generator Loss: The generator loss describes how well it can trick the discriminator, and how closely its output matches the real label for the given

image.

$$L = MSE(D(L_f, I), 1) + L1(L_f, L_r)$$

Discriminator Loss: The discriminator loss describes how accurately it is able to predict, given a label and an image, if the label is real or not.

$$L = 0.5 \times (MSE(D(L_f, I), 0) + MSE(D(L_r, I), 1))$$

Because these two loss values are relative to the performance of eachother they can't be used to see if our generator has converged on a solution. Therefore it is necessary, at each epoch, to also compute the L1 loss of the fake labels and real labels across the training and the validation dataset, as these values are independent of the discriminator. This will inform allow us to see if the generator is overfitting, underfitting, or training well.

Training Strategy: Please see algorithm 2 for the training loop.

Algorithm 2 GAN Training Strategy

for `every epoch` **do**
  for I, L_r `in training dataloader` **do**

    $L_f = G(I)$

    $pred_fake = D(L_f, I)$
    $loss_G = MSE(pred_fake, 1) + L1(L_f, L_r)$
    `Update weights of G with backpropagation`

    $pred_real = D(L_r, I)$
    $loss_D = 0.5 \times (MSE(pred_fake, 0) + MSE(pred_real, 1))$
    `Update weights of D with backpropagation`

  end for
end for

5.5 Experiment 3

I will adapt a PyTorch implementation of Palette [35] and train it to predict labels from the VISCHEMA dataset. Palette is a versitile conditional image to image diffusion model proposed by Saharia et al. [18] that is able to uncrop, inpaint, colorize, and remove JPEG artifacts from images. Because of this versitility I think it could learn to predict VMS maps from images. The conditional image passed to the model will be the image from the dataset,

the ground truth will be the label. This network takes as input a 128x128x6 tensor of floating point values, this is the concatenation of an image and the noise added to the ground truth after t steps. It outputs a 128x128x3 tensor of floating point values, this is the models estimate of the noise added at time $t-1$. This model can be described as

$$noise_{pred} = model(I, t)$$

Model Loss: The model loss describes how well it can estimate the noise added to an image between time steps $t-1$ and t, it is computed across an entire batch of images and then the model weights are adjusted with backpropagation.

$$L = MSE(model(I, t), noise_{real})$$

The loss calculated for backpropagation is relative to the performance in estimating noise added, not for the performance when calculating VMS labels. Because of this I will also have to calculate the L1 loss of the generated labels against the real labels. This will take a lot of computational time so I will do it at the end of training. I will be able to see if our model has converged by observing the backpropagation loss.

Training Strategy: Please see algorithm 3 for the training loop.

Algorithm 3 Diffusion Model Training Strategy

for `every epoch` **do**
  for I, L `in training dataloader` **do**

    $t = random(0, noise_steps)$
    $noise_{real}, L_{noisy} = noise\_image(L, t)$ ▷ Apply t steps of noise and return noise added at step t

    $input = concatenation(I, L_{noisy})$
    $noise_{pred} = model(I, t)$
    $loss = MSE(noise_{pred}, Noise_{real})$
    `Update weights of model with backpropagation`
  end for
end for

5.6 Results Analysis/Testing

After performing all 3 experiments I will compare the results across the validation dataset qualitatively and using the L1 loss. The L1 loss will tell

me how statistically close the outputs are but through qualitative analysis I can see if the outputs have cheated. I can also compare the L1 loss across the training and validation sets to see if any models have generalised well, if the loss across the training dataset is much smaller/larger than the validation dataset then it will imply that the model has become overfit/underfit respectively. Ideally they should be similar.

6 Experimental Results

6.1 Experimental Environment

I have implement the experiments in Python using the PyTorch machine learning framework, however the methodology that I describe should produce the same results in any programming language or framework.

This model was trained on a computer using an RTX 3070 with 8GB of VRAM and an AMD Ryzen 3600 with 32GB of system RAM. Testing the hyperparameter options for experiment 1 took approximately X days and I trained the final model over the course of Y hours Testing the hyperparameter options for experiment 2 took approximately 3 days and I trained my final model over the course of 5 hours. Testing the hyperparameter options for experiment 1 took approximately X days and I trained the final model over the course of Y hours

All experiments were performed with differentiable augmentation as described in [34]. I performed translation and cutout, each with a 90% chance.

6.2 Hyperparameter tuning

For experiments 1 and 2 I automated the process of exploring the parameter and hyperparameter search space. In my search I varied the following parameters: The normalisation layer used, the channel layouts used, the optimiser used, the learning rates for the optimiser, and, if the Adam optimiser was used, the beta values. Because training a diffusion model is far more computationally expensive than training an Autoencoder or a GAN I was unable to automate the process of tuning the hyperparameters in experiment 3. Training a GAN model on the VISCHEMA dataset would typically take around 100 minutes, but training a diffusion model would take around 20 hours due to the greater number of epochs required.

In experiment 2 I allowed for the generator and discriminator models to use different normalisation functions, optimisers, and learning rates.

Exploring this search space exhaustively is unfortunately not feesible, there are over 5000 different combinations possible, and if I tested each combination once for 100 epochs then it would take approximately 300 days to test per experiment. Instead I will have to explore a subset of this search space. For each experiment I estimated some good default parameter and hyperparameter values, by varying these values I can lower the scope of the search space to approximately 100 combinations, which took $\sim$3 days per experiment to test. Unfortunately this does mean that not every combination has been tested, however we should achieve a good approximation of the best parameters and hyperparameters.

In experiment 3 I was only able to explore a small range of possible hyperaparameters and as such my results do not emulate the recent success found in diffusion based image generation. However, with a greater number of computing resources it may be possible to do so.

6.3 Experiment 1

In this experiment I trained a UNet to predict the VMS maps of images, I trained the model using backpropagation of the L1 loss of the model output given an image, and the corresponding label.

The best parameters and hyperparameters that I found for my model were the following:

My training graphs:

After X training epochs I was able to achieve an L1 loss across my validation dataset of Y. I was also able to achieve a FID score of Z

A sample of output images:

When performed without differentiable augmentation the results looked like this:

FID score of Z, L1 Loss of Y.

6.4 Experiment 2

I explored the following parameter and hyperparameter options:

Normalisation layers: Batch Normalisation, Instance Norm

Channel layouts: I iterated over generator encoder and decoder, and discriminator channel layouts of following form, with values of c from {32,64,50,100}:

- Encoder: (3, c, 2c, 4c, 8c, 16c),
- Decoder: (16c, 8c, 4c, 2c, c),
- Discriminator: (6, c, 2c, 4c, 8c, 16c).

Optimisers:

- SGD, using the following learning rates: 0.005, 0.01, 0.02,
- Adam, using the following learning rates: 0.0005, 0.001, 0.002, and using the following betas values: (0.9, 0.999), (0, 0.999), (0.5, 0.999),
- Adadelta, using the following learning rates: 0.5, 1, 2.

Note: I didn't need to specify default values for the normalisation layer as these were the first variables I tested.

Default Generator Parameters and Hyperparameters:

- C = 64
- Optimiser: Adam, betas = (0.9, 0.999)
- Learning Rate: 0.001

Default Discriminator Parameters and Hyperparameters:

- C = 64
- Optimiser: Adam, betas = (0.9, 0.999)
- Learning Rate: 0.001

I iterated over each different parameter and hyperparameter and varied each one sequentially. I tested each combination for 100 epochs and used the L1 loss across the validation dataset as the score, if any new parameter options gave a better L1 score the it would become the default used going forward. This meant that I only had to iterate $\sim$100 combinations. I found the following best options.

Best Generator: Batch Normalisation, C = 32, SGD Optimiser with a learning rate of 0.01.

Best Discriminator: Batch Normalisation, c = 100, Adam optimiser with betas = (0.9, 0.999) and a learning rate of 0.001.

With this combination we achieved a loss of $\sim$1.02 across the validation dataset. I found other good results using similar combinations. Using the Adatelta optimiser for the generator and the Adam optimiser for the discriminator achieved a loss of $\sim$1.03 across the validation dataset. Using the Adam optimiser for both the generator and discriminator achieved a loss of $\sim$1.04 across the validation dataset.

FID score of Z, L1 Score of Y

Here are the images generated when I dont use differentiable augmentation

FID score of Z, L1 Score of Y

Wasserstein GAN: Using a Wasserstein image-to-image GAN as described in [32] I was able to achieve a best score of

FID score of Z, L1 Score of Y

Here are the images generated when I dont use differentiable augmentation

FID score of Z, L1 Score of Y

6.5 Experiment 3

6.6 Results, Evaluation Metrics, and Analysis

a graph of the train/val loss over time across the different experiments

a graph of the FID score at the end of the different experiments

a selection of the images with the different training methods

Some discussion about them.

7 Conclusion

Bibliography

[1] BBC. 'Optical illusion: Dress colour debate goes global.' (2015), [Online]. Available: https://www.bbc.com/news/uk-scotland-highlands-islands-31656935.

[2] M. Gambino. 'Lunch atop a skyscraper photograph: The story behind the famous shot.' (2012), [Online]. Available: https://www.smithsonianmag.com/history/lunch-atop-a-skyscraper-photograph-the-story-behind-the-famous-shot-43931148/.

[3] P. Isola, J. Xiao, A. Torralba and A. Oliva, 'What makes an image memorable?' In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 145–152.

[4] P. Isola, D. Parikh, A. Torralba and A. Oliva, 'Understanding the intrinsic memorability of images,' in *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[5] A. Khosla, A. S. Raju, A. Torralba and A. Oliva, 'Understanding and predicting image memorability at a large scale,' in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2390–2398. DOI: 10.1109/ICCV.2015.275.

[6] P. Isola, J. Xiao, D. Parikh, A. Torralba and A. Oliva, 'What makes a photograph memorable?' *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1469–1482, 2014.

[7] E. Akagunduz, A. G. Bors and K. K. Evans, 'Defining image memorability using the visual memory schema,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 9, pp. 2165–2178, 2020. DOI: 10.1109/TPAMI.2019.2914392.

[8] R. Brener, 'An experimental investigation of memory span,' *Journal of Experimental Psychology*, vol. 26, no. 5, pp. 467–482, 1940. DOI: 10.1037/h0061096.

[9] R. S. Nickerson, 'Short-term memory for complex meaningful visual configurations: A demonstration of capacity.,' *Canadian journal of psychology*, vol. 19, pp. 155–60, Jun. 1965.

[10] R. N. Shepard, 'Recognition memory for words, sentences, and pictures,' *Journal of Verbal Learning and Verbal Behaviour*, vol. 6, pp. 156–163, 1967.

[11] L. Standing, 'Learning 10,000 pictures,' *Quarterly Journal of Experimental Psychology*, vol. 25, pp. 207–222, 1973.

[12] T. F. Brady, T. Konkle, G. A. Alvarez and A. Oliva, 'Visual long-term memory has a massive storage capacity for object details,' *Proceedings of the National Academy of Sciences*, vol. 105, no. 38, pp. 14 325–14 329, 2008.

[13] A. Khosla, J. Xiao, A. Torralba and A. Oliva, 'Memorability of image regions,' in *Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, Dec. 2012.

[14] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[16] J. Ho, A. Jain and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. arXiv: 2006.11239 [cs.LG].

[17] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, 2018. arXiv: 1611.07004 [cs.CV].

[18] C. Saharia, W. Chan, H. Chang *et al.*, *Palette: Image-to-image diffusion models*, 2022. arXiv: 2111.05826 [cs.CV].

[19] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, 2021. arXiv: 2105.05233 [cs.LG].

[20] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, *Unet++: Redesigning skip connections to exploit multiscale features in image segmentation*, 2020. arXiv: 1912.05074 [eess.IV].

[21] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane and M. Jagersand, 'U2-net: Going deeper with nested u-structure for salient object detection,' *Pattern Recognition*, vol. 106, p. 107 404, Oct. 2020. DOI: 10.1016/j.patcog.2020.107404. [Online]. Available: https://doi.org/10.1016%2Fj.patcog.2020.107404.

[22] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, 2020. arXiv: 1703.10593 [cs.CV].

[23] T. Karras, S. Laine and T. Aila, *A style-based generator architecture for generative adversarial networks*, 2019. arXiv: 1812.04948 [cs.NE].

[24] I. Goodfellow, *Nips 2016 tutorial: Generative adversarial networks*, 2017. arXiv: 1701.00160 [cs.LG].

[25] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, 2015. arXiv: 1503.03585 [cs.LG].

[26] J. Song, C. Meng and S. Ermon, *Denoising diffusion implicit models*, 2022. arXiv: 2010.02502 [cs.LG].

[27] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, 2021. arXiv: 2102.09672 [cs.LG].

[28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, *Score-based generative modeling through stochastic differential equations*, 2021. arXiv: 2011.13456 [cs.LG].

[29] A. Brock, J. Donahue and K. Simonyan, *Large scale gan training for high fidelity natural image synthesis*, 2019. arXiv: 1809.11096 [cs.LG].

[30] T. Chen, *On the importance of noise scheduling for diffusion models*, 2023. arXiv: 2301.10972 [cs.CV].

[31] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein gan*, 2017. arXiv: 1701.07875 [stat.ML].

[32] N. Makow, *Wasserstein gans for image-to-image translation*, 2018.

[33] T. W. Jun-Yan Zhu Taesung Park, *Cyclegan and pix2pix in pytorch*, Commit 9f8f61e. [Online]. Available: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

[34] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu and S. Han, *Differentiable augmentation for data-efficient gan training*, 2020. arXiv: 2006.10738 [cs.CV].

[35] L. Jiang, *Palette: Image-to-image diffusion models*, Commit 136b29f. [Online]. Available: https://github.com/Janspiry/Palette-Image-to-Image-Diffusion-Models.