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 Introduction

1.1 Background

1.1.1 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.

1.1.2 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.

1.1.3 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.

1.1.4 Why do we want to create a memorable image

1.1.5 How does this research help to create a memorable image

1.2 Motivation

1.3 What are my goals

1.4 Structure

2 Literature Review

2.1 Memorability

In R. Breners work [An experimental investigation of memory span] 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.

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 [Short-term memory for complex meaningful visual configurations] 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"[Short-term memory for complex meaningful visual configurations 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 [Short-term memory for complex meaningful visual configurations], R. Shephards work [Recognition Memory for Words, Sentences, and Pictures]

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 [Learning 10,000 pictures]. 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"[Learning 10,000 pictures, p.208], and 'vivid' images are described as "striking pictures ... with definitely interesting subject matter"[Learning 10,000 pictures, p.208]. Much like how in Brener's work [An experimental investigation of memory span] 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. [Visual long-term memory has a massive storage capacity for object details] 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 [Learning 10,000 pictures] and [Visual long-term memory has a massive storage capacity for object details] 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 [Learning 10,000 pictures] and [Visual long-term memory has a massive storage capacity for object details] etc 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. [1, 11] 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. [4] built on the work in [1, 11] 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 [1, 11], is limited by the arbitrarily picked list of features that define memorability.

2.2 Using Deep Learning to Predict Memorability

2.2.1 Deep learning has seen great success with image synthesis

Recently, the use of deep convolutional networks [12,13] has proven exceptionally successful in image classification and object recognition. It stands to reason that a combination of these systems could perform well at this task. It could, in the worst case, recognise which objects are within an image, and to use that to help inform how memorable the image is. In the best case, it could learn to extract the features from categories that make them memorable and use this to predict the memorability of categories not yet seen. The weights within a convolutional layer act as filters to recognise patterns within images, through the stacking of these filters we are able to build networks that can recognise complex objects. The main limitation within [1,11,4] is that the features are arbitrarily picked by humans and manually annotated. The use of convolution should make it possible to learn these features automatically.

Khosla et al. have created a dataset, LaMem [5], containing 60,000 annot-

ated 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 the U-Net [8] or ResNet [9], use fully convolutional networks. These maintain spatial locality which makes them less prone to overfitting, and also allows them to generalise better. Using deep learning we should be able to automate the process of learning features like those in [1,4,11], by training a fully convolutional network with a sufficiently large dataset such as those presented in [5,7].

I propose that predicting memorability maps is an extension of image segmentation, modern machine learning has found many network architectures that would be worth exploring for this problem

2.2.2 Auto-Encoders

The U-Net [8] has also found great success in image segmentation, this system however relies on the output being a classification. Something in between these two could work very well.

I will use a U-Net [U-Net paper] as my image to image network. The U-Net makes use of an Encoder block and a Decoder block, there are residual connections between different sections of the blocks that help to preserve image data in the decoding stage. The network has seen great success in the medical field. [U-net success paper].

2.2.3 Generative Adversarial Networks

Conditional GAN architectures have seen lots of success and produce high quality images, they work through the use of competitive co-evolutionary algorithms where a Generator and a Discriminator will 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. In conditional GANS The weights of the two networks are updated through backpropagation based on if they win or lose. The Generator wins by fooling the Discriminator into predicting that the image is from the distribution. The Discriminator wins if it can accurately predict whether an image is from the distribution or not. These networks have seen great success but typically suffer from instability due to their adversarial nature, and often suffer from mode collapse. Despite this, lots of research

has gone into producing GANs that produce high quality work.

Pix2Pix by Isola et al. [10], is an image-to-image deep learning architecture based on Conditional Generative Adversarial Networks. It is able to convert images from one style into another. For example, it can convert labels into a street scene or photos of daytime into night time. It makes use of a UNet backbone and has found great success.

2.2.4 Diffusion Models

Diffusion based image generation is inspired by nonequilibrium thermodynamics, it works through a forward diffusion process of adding Gaussian noise to a sample until it has been transformed such that it is no longer distinguishable to Gaussian noise, this should follow a diffusion schedule where noise is applied T times. We then train a model to compute a reverse diffusion process. That is, given a value 1 < t < T, and a sample that has been transformed t times, predict the sample at t-1 transformations. When we want to produce a sample, we create $x_0 \sim N(0,1)$ and apply reverse diffusion to it T times, following our diffusion schedule.

Diffusion models were first introduced by J. Sohl-Dickstein et al. (Deep Unsupervised Learning using Nonequilibrium Thermodynamics) as a flexible and tractable machine learning model. They describe the forward diffusion process of 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.

This was then built on by Y. Song and S. Ermon in (Generative Modeling ny Estmating Gradients of the Data Distribution) by using Langevin dynamics

The DDPM a diffusion model based on a UNet, introduced by J. Ho et al. (Denoising Diffusion Probabilistic Models) is capable of producing high quality images, they show that it is capable of achieving state of the art FID scores across the CIFAR10 dataset. Dhariwal and Nichol (Diffusion Models Beat GANs on Image Synthesis) 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 by Song et al. [Denoising diffusion implicit models], Nichol and Dhariwal [Improved denoising diffusion probabilistic models], Song et al. [Score-

based generative modeling through stochastic differential equations], [Large scale gan training for high fidelity natural image synthesis] [A style based generator architecture for generative adversarial networks]. Through these improvements they are able to improve image quality while reducing the number of noise steps from 1000 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 [On the importance of noise scheduling] how changing the resolution of an image has an impact on the noise scheduling required, he finds that at a smaller resolution the optimal schedule 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.

2.2.5 Alternatives

3 Methodology

3.1 Requirements Capture

We have a dataset of images that we will aim to accurately to map onto their corresponding labels. Through learning to map these images onto their corresponding VMS labels, we hope that our system will gain a general understanding of the data, such that when another image that matches our distribution is presented, the system can accurately create a VMS label for that. Ideally our system should create accurate mappings, we can test this by using a loss function such as L1 Loss and through qualitative analysis.

3.2 Analysis/Design

I will create a deep learning image to image model that will be trained on the training data portion of the VISCHEMA PLUS dataset, it will be evaluated on the validation data portion of the dataset. The network will be trained to directly output the corresponding VMS map.

I will implement this network in Python using the PyTorch machine learning framework. This is because I am familiar with it rather than it having any advantages over other existing frameworks, and the methodology that I describe should produce the same results in any programming language or framework.

I will also experiment with training the network to output the composition of the image and the label, and then manually subtract the image after. Because the difference between the image and the label is much greater than the difference between the image and the image + label I suspect that this should be easier for the generator to learn the weights for. This may produce better results and is something I will experiment with, I'm not entirely convinced however because GANs have been shown to struggle when one network is much better than the other, I suspect that the discriminator would have a harder time being able to tell the difference and may make learning impossible.

3.3 Experiment 1

3.3.1 Implementation

3.3.2 Model Architectures

3.3.3 Training Strategy

3.3.4 Results Analysis/Testing

3.4 Experiment 2

3.4.1 Model Architectures

I will train a conditional Generative Adversarial Network to generate images of VMS maps given images from the VISCHEMA dataset. The Generator will implemented using a UNet and the Discriminator will be a fully convolutional PatchGAN classifier as described by Isola et al. [Pix2Pix paper], producing a 4x4 array of boolean values given a label, describing whether a region of the image is from the datset, or if it is generated.

Generator: This network takes as input a 64x64x3 tensor of floating point values in the range [-1,1] It outputs a 64x64x3 tensor of floating point values in the range [-1,1]. This model can be described as $Label_{fake} = Generator(Image)$

Discriminator: This network takes as input a 64x64x3 tensor of floating point values in the range [-1,1] It outputs a 4x4x1 tensor of boolean values. Each value in this output represents a 16x16x3 region of the input.

Training within a GAN is typically unstable [papers showing instability in GANs], if one performs much better than the other it can become impossible for the other to improve, therefore I will experiment with different values for the number of layers and channels in each network.

The loss for each network is computed differently.

Generator Loss: The generator loss is the sum of our loss when tricking the discriminator and the pixelwise difference between our produced label and the real label. Our network will produce a fake label, Lf.

L = MSE(D(Lf, image), 1) + L1(Lf, Lr)

Discriminator Loss: When training the Discriminator the loss is calculated as the mean value of how close our prediction was to the correct answer across a fake and real label.

L = 0.5 * (MSE(D(Lf, image), 0) + MSE(D(Lr, image), 1))

In both networks the loss is computed across an entire batch of images and then the weights are adjusted with backpropagation.

When tracking the progress of the GAN the L1 loss of the produced label vs the real label can be used, as it is independent of the discriminator, and therefore should improve over time. Every epoch I will track the value of this across the validation dataset and use that to inform me about how well our network is training. It should inform me of whether the network is overfitting, underfitting, or training well.

3.4.2 Training Strategy

My training loop will be composed of an image label pair being taken from the dataset, the image will be passed into a generator, and then a fake label will be produced. The Discriminator will be passed the input image and a corresponding VMS, either produced by the generator or the real one, and through backpropagation its weights will be updated depending on if it answers correctly or not. The generator will have its weights updated through backpropagation depending on if it can fool the discriminator or not.

Typical training (gradient descent steps using the loss of the output vs the label) vs GAN (Training a generator to produce the labels and a discriminator to determine if they are from the dataset or not)

Variables that I will modify: 1. Normalisation method: a. Batch Norm b. Layer Norm c. Instance Norm 4. Dropout 2. Optimisation function: a. Adam b. SGD c. Adagrad d. Adadelta 3. Other Hyperparameters: a. Learning Rate b. Batch Size

Mention the fact that when testing it is not feasible to test every combination of these variables. Instead I will test every variation of normalisation method, pick the best, and then use that when testing for the best optimisation function, then use both of those etc.

3.4.3 Results Analysis/Testing

Using all of the different combinations of parameters that I have outlined in my methodology, I will train models to map from the image set to the label set, and then I will evaluate these in multiple different ways. The most important will be the L1 loss across the validation set, this will measure the mean absolute difference between each pixel in the output of the network and the corresponding label. The smaller this number the closer the network is to achieving a direct mapping from the image set to the label set.

I can also use the L1 loss across the training and validation sets to determine if my model has become overfit to the training data. If the L1 loss has become significantly smaller across the training dataset vs the validation dataset, this will imply that the model has become overfit to the training set, and instead of learning the general pattern of the distribution of images, it is instead memorising the mappings. This will be a useful tool during development, and will allow me to make changes to prevent it. If the L1 loss across my train and validation data is similar then it will imply that my network is able to learn a general mapping of the data. If they both stop improving it may mean that some improvements could be made by making the network more complex (either by adding more layers or adding more channels to my layers).

I will also examine the images qualitatively, do the images look like they are the same, are there any large issues (for example the large blue blobs or yellow areas)

3.5 Experiment 3

3.5.1 Implementation

3.5.2 Model Architectures

3.5.3 Training Strategy

3.5.4 Results Analysis/Testing

3.6 Analysis/Design

4 Experimental Results

This model was trained on a computer using an RTX 3070 with 8GB of VRAM, testing all of my hyperparameter options took approximately 3 days and I trained my final model over the course of days.

4.1 Best Training Hyperparameters and Model Parameters

I wanted to find which parameter and hyperparameter combinations would perform the best. I decided to automate this process to save time. I wanted to vary the following:

1. The normalisation layer for both the generator and the discriminator.
2. The channel layouts for the generator and the discriminator.
3. The optimiser used for the generator and the discriminator.
4. The learning rates used for each optimiser.
5. If the Adam optimiser was found to be ideal for either the generator or discriminator, then to use different beta options.

I explored the following options:

Normalisation layers:

- Batch Normalisation
- Instance Norm

Generator channel layouts:

- encoder:(3, 64, 128, 256, 512, 1024), decoder:(1024, 512, 256, 128, 64),
- encoder:(3, 32, 64, 128, 256, 512), decoder:(512, 256, 128, 64, 32),

- encoder:(3, 50, 100, 200, 400, 800), decoder:(800, 400, 200, 100, 50),
- encoder:(3, 100, 200, 400, 800, 1600), decoder:(1600, 800, 400, 200, 100)

Discriminator channel layouts:

- (6, 64, 128, 256, 512, 1024),
- (6, 32, 64 , 128, 256, 512),
- (6, 50, 100, 200, 400, 800),
- (6, 100, 200, 400, 800, 1600)

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 ((0.9, 0.999), (0, 0.999), (0.5, 0.999))
- Adadelta, using the following learning rates (0.5, 1, 2)

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. Instead I will have to explore a subset of this search space. 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 to test. Unfortunately this did mean that every combination hasn't been tested, however it should give us a good approximation of the best combination.

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:

- Encoder Channel Layout: (3,64,128,256,512,1024)
- Decoder Channel Layout: (1024,512,256,128,64)
- Optimiser: Adam, betas = (0.9, 0.999)
- Learning Rate: 0.001

Default Discriminator Parameters and Hyperparameters:

- Channel Layout: (6,64,128,256,512,1024)
- Optimiser: Adam, betas = (0.9, 0.999)
- Learning Rate: 0.001

I then 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 parameter options performed better than the default then they would be used going forward. This meant that I only had to iterate $\sim$100 combinations. This took 3 days to test, and at the end I found that the following combination was the best:

Best Generator Parameters and Hyperparameters:

- Normalisation Layer: Batch Normalisation
- Encoder Channel Layout: (3,32,64,128,256,512)
- Decoder Channel Layout: (512, 256, 128, 64,32)
- Optimiser: SGD
- Learning Rate: 0.01

Best Discriminator Parameters and Hyperparameters:

- Normalisation Layer: Batch Normalisation
- Channel Layout: (6,100,200,400,800,1600)
- Optimiser: Adam, betas = (0.9, 0.999)
- Learning Rate: 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 Adam optimiser for both the generator and discriminator achieved a loss of $\sim$1.04 across the validation dataset. 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.

Varying Normalisation

Varying Optimiser

Varying Loss Function

PatchGAN vs Boolean Discriminator

Benefits of Residual Connections

4.2 Results

4.3 Evaluation Metrics

4.4 Analysis

5 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*, 2011.

[5] A. Khosla, A. S. Raju, A. Torralba and A. Oliva, 'Understanding and predicting image memorability at a large scale,' in *International Conference on Computer Vision (ICCV)*, 2015.

[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.