Generating Camouflage using Generative Adversarial Networks

ALEX RUNDLE, University of Exeter, UK

This review considers both generative and adversarial techniques within the context of camouflage generation, and how they can be combined as the Generative Adversarial network (GAN). Camouflage intends to form a cognitive illusion to hide an object within a set background or scene, and is most often used within the military. To this end, this paper reviews the original Generative Adversarial network and its flaws, alongside improvements to its instability such as the Deep Convolutional GAN and the Wasserstein distance. Additionally, this paper discusses how the Wasserstein GAN solves almost all flaws in the Generative Adversarial Network, also discussing why it requires a Gradient Penalty. Furthermore, generative and adversarial techniques for generating camouflage are considered, alongside a Generative Adversarial network for generating camouflage. Finally, a project is proposed to determine the extent to which Generative Adversarial Networks can be used to create effective camouflage, by implementing a Generative Adversarial Network and modifying a Deep Convolutional GAN to utilise the Wasserstein Distance. Furthermore, measures to assess the performance of the model are considered, alongside the outline for an observation test using human participants.

I certify that all material in this dissertation which is not my own work has been identified and correctly credited:

Signed: Alex Rundle

1 INTRODUCTION

Camouflage is based on the idea of forming a cognitive illusion in which an object projects the scene behind it to reduce its saliency (how prominent it is against the background) [1]. As such, it is often used in the military to reduce the prominence of personnel and vehicles in active warfare. In this project, I seek to explore how Generative Adversarial networks can be used to generate camouflage and the extent to which effective camouflage can be generated. As it is used by military personnel, it is essential that the camouflage works effectively and conceals them from enemies and, as such, being able to generate effective camouflage which minimises the risk of being spotted could be a large advantage to a military force. Furthermore, Generative Adversarial networks are well-studied and are shown to provide excellent image generation compared to prior generation techniques [2], hence this project provides an interesting research avenue that has well-founded real-world applications.

In this paper, I cover a brief history of camouflage in section 1.0.1, related Generative and Adversarial methods in section 2 and how the ideas behind the two separate methods can be combined into a Generative Adversarial network in section 3. I then proceed to cover further advancements in the Generative Adversarial network architecture in section 4, as well as applications of generative and adversarial techniques in the generation of camouflage in section 5. I specify my own functional and non-functional requirements for my project, as well the evaluation criteria under which I determine the success and effectiveness of the generated camouflage in section 6. Finally, I conclude and summarise this paper in section 7.

1.0.1 History of Camouflage. As the outcome of this project is to generate camouflage, one must understand the reason for which it was created. For hundreds of years, military uniform was designed to aid in telling friend from foe when on the battlefield and often used vivid colours, such as the red uniforms worn by the British Infantry during the Napoleonic wars [3]. As various military's developed more accurate firearms, the vividness of the current uniform meant that infantry were easy to spot by marksmen. In 1800, Colonel Charles Hamilton Smith ran an experiment to determine the effect uniform colour has on a marksman's ability [4]. Over stages, he had a troop of marksmen shoot various targets painted red, green and grey. He had found that the red target had sustained twice the amount of damage as the grey target, such that "it fell to pieces before the last shot was delivered" [4]; the green target was intermediately damaged.

An example of modern camouflage that might be used by military personnel in a forest or woodland setting can be seen in Figure 1.

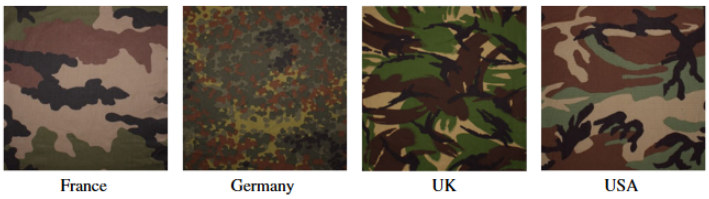

Fig. 1. An example of Woodland camouflage from 4 NATO members during the early 2000s from Talas et al. [5].

2 RELATED WORK

2.1 Generative Models

Generative models are an important part of supervised learning techniques, allowing for the sampling of new data formed from estimated models. There are two classes of generative model: explicit density and implicit density [6]. Explicit density models provide explicit parametric specifications of the distribution of data and are easy for humans to understand [7]; however, they incur higher computational cost and increased complexity. For example, the Boltzmann machine uses Markov chains to generate new samples, but has a very high computational complexity [8]. Other such models include maximum likelihood estimation, during which parameters are updated according to data samples producing a smooth generative model [9]–[11]. Implicit density models, on the other hand, do not directly estimate or even fit the data to a distribution, they define stochastic procedures that directly generate data [6]. The Generative Stochastic Network learns the transition operator of a Markov chain (which is an estimate of the data distribution) and can be used on sets with missing inputs and can be used to sample subsets of data [12]. Unlike the Boltzmann machine [8], the Generative Stochastic Network [12] does not provide a likelihood function nor make any explicit assumptions about the data distribution, instead providing a generation procedure [6]. Explicit models experience intractability from the likelihood estimate on high-dimensional data, with it being very difficult to compute; whereas, implicit models experience complexity due to the lack of the likelihood term, causing the available learning methods to be significantly reduced [6].

2.2 Adversarial Techniques

A key component of the Generative Adversarial Network is the so-called adversarial technique [13]. Using knowledge from game theory, networks can be set to compete against one another during the training process, such as the predictability minimization technique developed by Schmidhuber [14] where one network aims to minimise prediction errors, whilst the other maximises the total objective function T. Adversarial networks have also been used within domain adaptation, where a feature generator transforms source and target domain data into abstract features, making it difficult to discriminate between the domains. A domain discriminator tries to discriminate the domains accurately; the two networks work as adversaries, trying to minimise the other's accuracy [15].

3 COMBINING GENERATIVE AND ADVERSARIAL TECHNIQUES

3.1 Original Generative Adversarial Network Structure

The main basis of a Generative Adversarial Network (GAN) is the Nash equilibrium from game theory, where, in an n-player non-cooperative game, each player's strategy is optimal against the other players and that no other player can increase their own payoff, even if they change their own strategy [16]. In the case of a GAN, a 2-player non-cooperative game is held, with the two players consisting of the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$. The generator $\mathcal{G}$ aims to correctly learn the distribution of the real data (such that it can produce new samples fitting the distribution), whilst the discriminator $\mathcal{D}$ aims to correctly distinguish real data from generated data. [2]. Winning the game requires the players to continuously optimise themselves to improve their own abilities, being generation and discrimination

respectively; this optimisation is done in the aim of finding the Nash equilibrium between the generator and discriminator.

Goodfellow et al. [2] define the generator as a differentiable function in the form of a multilayer perceptron $\mathcal{G}(z;\theta_g)$, with parameters θ_g. The discriminator is defined as a second multilayer perceptron $\mathcal{D}(x;\theta_d)$ with parameters θ_d. Their inputs are a random noise vector z and real data x, respectively. Formally, the generator $\mathcal{G}$ maps a noise vector z in latent space Z, to an image x: $\mathcal{G}(z) \rightarrow x$. The discriminator $\mathcal{D}$ maps an image x to a binary classifier: $\mathcal{D}(x) \rightarrow [0,1]$, with a real image being classified as true (close to 1) and a fake image classified as false (close to 0) [17]. $\mathcal{G}(z)$ represents a generated image from $\mathcal{G}$, using the noise vector z, and $\mathcal{D}(x)$ represents the probability that an image x comes from the real data, p_{data}, as opposed to the generated data, p_g [18]. As the discriminator, $\mathcal{D}$ wants to achieve a high rate of correct classification of data, while $\mathcal{G}$ wants to make the performance of generated data (i.e. $\mathcal{D}(\mathcal{G}(z))$) consistent with that of real data (i.e. $\mathcal{D}(x)$). Once $\mathcal{D}$ has a high discrimination ability, but cannot still discriminate between $\mathcal{G}(z)$ and x, it is considered that $\mathcal{G}$ has successfully captured the distribution of the real data.

3.2 Learning and Training Method:

$\mathcal{D}$ is trained to maximise the probability of assigning the correct class label to the real data and generated samples from $\mathcal{G}$. Simultaneously, $\mathcal{G}$ is trained to minimise $\log(1-\mathcal{D}(\mathcal{G}(z)))$, the probability that $\mathcal{D}$ correctly labels generated data $\mathcal{G}(z)$. This forms a two-player mini-max relationship between the two networks, using the following value function $V(\mathcal{D},\mathcal{G})$:

$$\min_{\mathcal{G}}\max_{\mathcal{D}} V(\mathcal{D},\mathcal{G}) = \min_{\mathcal{G}}\max_{\mathcal{D}} \mathbb{E}_{x\sim p_{data}(x)}[\log \mathcal{D}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))] \tag{1}$$

$V(\mathcal{D},\mathcal{G})$ is a binary cross entropy value function, which measures how correctly real data and generated data is classified.

3.2.1 Global Optimum: Goodfellow et al. [2] theorise that, if $\mathcal{G}$ and $\mathcal{D}$ have enough capacity and $\mathcal{D}$ is allowed to reach its optimum $\mathcal{D}^*$ for a fixed $\mathcal{G}$ (defined in equation (2)), p_g converges to p_{data}.

$$\mathcal{D}^*_{\mathcal{G}}(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)} \tag{2}$$

Substituting the optimal discriminator $\mathcal{D}^*$ from equation (2) into equation (1), they discover the equation becomes the Jensen-Shannon Divergence [19] between $p_{data}(x)$ and $p_g(x)$. The optimal generator which minimises $JSD(p_{data} \parallel p_g)$, is $p_{data}(x)$, with $\mathcal{D}$ becoming $\frac{1}{2}$ by substituting the optimal generator into equation (2) [2]. In general terms, an equilibrium between $\mathcal{G}$ and $\mathcal{D}$ occurs when $p_{data}(x) = p_g(x)$ and, in this case, $\mathcal{D}$ will always produce a value of $\frac{1}{2}$ [18].

For learning the parameters, θ_g and θ_d respectively, $\mathcal{D}$ must be trained to maximise the accuracy of discriminating the generated data $\mathcal{G}(x)$ from real data x (e.g. maximising $\mathcal{D}(x)$ and minimising $\mathcal{D}(\mathcal{G}(z))$). Alongside this, $\mathcal{G}$ must be trained to minimise $\log(1-\mathcal{D}(\mathcal{G}(z)))$. A training method proposed by Goodfellow et al. [2] begins by fixing $\mathcal{G}$ and optimising $\mathcal{D}$ to maximise the accuracy of $\mathcal{D}$; subsequently, $\mathcal{D}$ is fixed and $\mathcal{G}$ is optimised to minimise the accuracy of $\mathcal{D}$. By alternating the process, the global optimum solution can be found (again, if and only if $p_{data}(x) = p_g(x)$); during the training process, the parameters of $\mathcal{D}$ are updated k times, after which the parameters of $\mathcal{G}$ are updated once [13].

Compared with the Auto-Encoding Variational Bayes algorithm, another generative model which optimises a recognition model to efficiently learn model parameters and features [20], the images are usually less blurred and appear more realistic. However, the original technique is prone to mode collapse, in which the generator $\mathcal{G}$ generates the same image x for many noise values z, causing a limited set of outputs (in the case of partial collapse) or producing a single sample only (complete collapse) [21], [22].

4 VARIANTS OF THE GAN:

4.1 Conditional GANs

To address the lack of control over the outcome of training a GAN, Mirza et al. [23] expand the traditional GAN model to include extra information over which the network is trained. In the Conditional Generative Adversarial Network (CGAN), both the generator and discriminator are conditioned on some extra information y (usually class labels); y is fed into the generator and discriminator as an additional input layer [23]. By feeding the discriminator the condition information y, it can distinguish real and fake samples given the information in y [18]. Compared with the original model, CGANs have the added benefit of being able to control (in the case of the MNIST dataset [24]) the number of digits generated where the original GAN was previously unable [18]. Alongside this, the CGAN is better suited for multi-modal data generation [21]. Using the conditional information found in a CGAN [23] to generate camouflage could allow for generation of camouflage better suited to different ecological environments.

4.2 DCGAN

Radford et al. [25] comment that images generated by the original GAN architecture [2] suffer from being noisy and incomprehensible, hence prompting their proposal of a GAN using aspects of a deep Convolutional Neural Network (CNN). The DCGAN (Deep Convolutional GAN) uses a pair of deep convolutional networks for the generator and discriminator, developed by adopting and modifying multiple changes to the CNN architecture [25]. Firstly, the all convolutional network [26] replaces the pooling functions with strided convolutions allowing for the network to learn its own spatial sampling operators; these handle the changes in sampling rates and locations [21]. They attempt to eliminate fully connected layers by employing global average pooling [27] but find that it decreases convergence speed. Instead, they directly connect the highest convolutional features to the input of the generator and the output of the discriminator, increasing model stability [25]. They also use Batch Normalization [28], which provides more stability to learning by normalising the inputs to each unit in the network; doing this they demonstrate that it is critical in preventing the mode collapse of the generator [2], but should not be applied to the generator's output layer or discriminator's input layer to prevent oscillation and model instability [25].

Alongside this, Radford et al. [25] use the ReLU (for Rectified Linear Units [29]) activation function which has a linearity that avoids the vanishing gradient problem (a cause of training instability when using sigmoid activation functions) [30]. It is used for all but the output layer of the generator, where the Tanh function is used [25]. In contrast to the maxout activation function used by the original GAN [2], Radford et al. [25] use the Leaky ReLU activation function [31], [32] for all layers of the discriminator, as they find it better suited for higher resolution modelling [25].

I believe that a DCGAN would prove to be a very useful technique within this project as it is very stable in training and produces higher quality images as a result [25] and could provide a useful starting point for more advanced GAN techniques which modify the DCGAN.

4.3 LAPGAN

Investigating the instability of the original GAN, Denton et al. [33] formulate the Laplacian Pyramid of Generative Adversarial Networks (LAPGAN), expanding a conditional GAN into the framework of a Laplacian pyramid, an image representation for image detail enhancement and manipulation comprised of high-pass difference images of decreasing resolution. Each difference image is created by subjecting an image to a blurring function and subtracting it from its original form [34]. The original GAN proves unstable and unreliable when generating larger resolution images [2], thus the LAPGAN seeks to provide a more stable framework for generating images of higher resolution [33]. Where the DCGAN innovated by incorporating a CNN into the underlying structure of the generator and discriminator [26], the LAPGAN instead finds innovation in a pyramid framework where, at each level, a CGAN [23] is found.

At each level of the Laplacian pyramid, a conditional generator takes the current image as a conditioning variable alongside some noise vector z_K. The image generated by the generator is combined with the conditioning variable to create a residual image $\tilde{I}_K$. This residual image is upsampled and provided to the next generator at the next level as the conditioning variable. This process will continue for all K levels of the pyramid. For training the LAPGAN model, a 64x64 training image I_0 is downsampled, blurred, and upsampled again to create a low-pass version of itself, i_0. With equal probability, i_0 is used to create either a real difference image h_0 (by subtracting i_0 from I_0) or generated difference image (by supplying i_0 and a noise vector z_0 to the generative network). In both cases, the resultant difference images are supplied to the discriminator network, alongside the low-pass image i_0, where the discriminator determines whether the sample is real or generated [33].

By training each level independently, mode collapse becomes significantly less of a problem than in the original GAN literature [2], owing to the course-to-fine manner of the LAPGAN [17]. Through human evaluation, it is found that a regular LAPGAN can generate samples of which are realistic enough to fool a human around 30% of the time; the original GAN model produces images realistic enough to fool a human $\leq$ 10% of the time [33]. However, due to the chaining of models, noise begins to build over each network on each layer of the pyramid, which makes the resultant images look 'wobbly' and noisy [25].

4.4 Wasserstein GAN

As shown by Arjovsky and Bottou [22], the original GAN model is both delicate and unstable in training, with mode collapse creating the need to maintain a careful balance between training the generator and discriminator. Prompted by this, Arjovsky et al. [35] investigate the distance and divergence measures used in prior GAN literature and formulate the Wasserstein Generative Adversarial Network (WGAN). Where the DCGAN [25] and LAPGAN [33] focus on the structure of the GAN itself, the WGAN instead seeks to find a more stable objective function [35].

First, they detail the Earth-Mover (EM) distance (also referred to as Wasserstein-1), a modification of the Kantorovich-Rubinstein metric [36]; the EM distance is the cost of an optimal plan to transport one

mass x (a probability distribution) to another mass y, where the cost is assumed to be the amount of mass moved times the distance moved [36]. They demonstrate that the EM distance produces better gradient behaviour compared to other distance metrics [17], showing through gradient descent that a sequence using the EM distance converges due to it being continuous, whereas neither the Jensen-Shannon divergence [19] nor Kullback-Leibler divergence [37] converge. The continuousness of the EM also alleviates it of the vanishing gradients found in the Jensen-Shannon divergence [35].

Three main differences can be identified between the original GAN [2] and the WGAN, the first being that no logarithm is used in the objective function [35]. Secondly, the discriminator is used to approximate the EM distance, where the original GAN used the discriminator as a binary classifier [35]. This results in the output of WGAN being a regression task and, consequently, means that the Sigmoid layer can be removed and that the output of the WGAN is unconstrained [35]. Third and finally, the discriminator in the WGAN is required to be K-Lipschitz for some K, for which the WGAN uses weight clipping to achieve [7], [35]. As the cost function does not experience vanishing gradients, the discriminator can be trained till optimality and, thus, avoids the mode collapse which occurs due to optimising a generator for fixed discriminator [35].

Arjovsky et al. [35] test WGAN by creating two Deep Convolutional GANs (DCGANs [25]), one using the EM distance and the other using the original Jensen-Shannon divergence [35]; it is seen that both algorithms produce high quality samples, meaning WGAN can produce samples of a quality as high as the regular DCGAN. The two DCGANs are further trained without the use of Batch Normalization [28] with the standard DCGAN failing to learn whilst the one using the EM distance still learns and produces quality samples, demonstrating that the WGAN is more robust in training compared to the original DCGAN [35]. However, the WGAN can still produce low-quality samples and fail to converge under specific circumstances [13], further discussed in section 4.5.

4.5 WGAN-GP

Although the WGAN is seen to be more stable in training [2], [35], instabilities still remain; one of which is the failure to converge. Noticing this failure to converge, Gulrajani et al. [38] investigate the discriminator's need for weight clipping. They find that although the weight clipping provides stability in a lower-depth WGANs [35], it proves to be problematic in deeper WGAN discriminators with many discriminators pushing the weights towards the two ends of the clipping range. Vanishing and exploding gradients occur as the weights reach the lower and upper ends of the clipping range respectively, causing low quality images and instability as a result [38]. In this paper, the authors formally demonstrate that using weight clipping to enforce the k-Lipschitz constraint biases the discriminator towards learning simple functions and, thus, reduces the capacity of the discriminator itself [21], [38]; this produces an instability in training similar to the original GAN [2].

In light of this, Gulrajani et al. [38] instead propose the Wasserstein GAN with Gradient Penalty (WGAN-GP), which removes the weight clipping and instead directly constrains the gradient norm of the discriminator's output with respect to its input; by directly constraining the gradient norm of the output , the authors remove the possibility of vanishing or exploding gradients caused by the original weight clipping [35], [38]. Prior GAN implementations such as the DCGAN [25] and WGAN [35] used Batch Normalization [28], which maps a batch of inputs to a batch of outputs [39], to better aid the stability in training. However, the newly penalised training objective instead penalises the norm of

the discriminator's gradient with respect to each individual input and, as such, batch normalization would not be valid within the discriminator and is omitted [38]. Comparing the WGAN-GP to the original WGAN [35] and the DCGAN [25], they show that the WGAN-GP significantly outperforms WGAN in terms of convergence speed and produces higher quality images; in real-time, WGAN-GP is shown to converge faster than the WGAN but slower than the DCGAN, however it is more stable in its convergence [38]. Although the WGAN-GP converges slower than the original DCGAN [25], I believe it would be the best fit for generating camouflage as it is the most stable during training and converges to a more stable state and provides higher quality, better defined images than the original DCGAN [38].

5 APPLICATIONS OF GENERATIVE AND ADVERSARIAL TECHNIQUES

In this section, I cover generative and adversarial techniques used to generate camouflage.

Du et al. [40] design a camouflage generation approach by utilising two-scale decomposition; coefficients of a large-scale layer of the background and a structure layer of the foreground are blended to create a camouflage effect [40]. Alongside this, Du and Shu [41] also present a camouflage generation technique using mean value interpolation and alpha blending operations [41]; however, Yang and Yin [42] note that these two techniques are hard to utilise in the active field for military purposes. Instead, they propose a generation scheme utilising colour similarity to create a 'full fusion' [42] of the target and background. They do this by calculating two colour similarity metrics on a selected polygonal region on an image, measuring the similarity of the pixels within the polygon to those outside of the polygon. They then create mosaic blocks inside the target polygon, colouring it using the two combined similarities. [42]. These techniques provide insight into how camouflage can be generated using the overall colour and textures of an image [40]–[42], however they only consider single images and do not have the ability to generalise to a set of images like a GAN does [2]. Due to this, I will not implement these specific techniques.

Expanding camouflage creation into the area of Generative Adversarial networks, Talas et al. [43] propose a novel method based on the natural evolution of protective camouflage for a simulated prey, which they call CamoGAN. They use a modified DCGAN [25], in which a triangular target is continuously optimised by the generator to match the given background images. Compared with other GAN implementations, such as CycleGAN [44], the CamoGAN does not optimise the whole image and, instead, acts much like an inpainting network [45], [46]; the target is optimised to fit the context of the background image, with the background image remaining untouched [43]. To test the efficacy of the generated samples, they recruit 45 participants to act as predators to the simulated prey (for which the targets represent), displaying a 64x32 target at a random position on a background image from the test set. Each participant is required to click on the detected target as quickly and as accurately as possible, with each image displayed for a maximum of 10 seconds [43]. Two control samples were introduced, the average colour of the background and a sample produced by Fourier analysis of the background, which has been shown to produce convincing military camouflage [47]. They found that the mean reaction times increased over more epochs, with the mean reaction time after 500 epochs equal to that of the average colour control set. Furthermore, after 2,500 epochs, the mean reaction time is equal to that of the Fourier control set. After 10,000 epochs, they found that these samples were significantly more difficult to detect than the Fourier set [43].

Although the CamoGAN uses a DCGAN to provide high quality generated camouflage [43], Yang et al. [48] instead use a WGAN to generate spot-based deformation camouflage, which serves to reduce the saliency of a moving target. They demonstrate through observation experiments with 30 participants that the spots generated have a probability of discovery less than 20%. Alongside this, they compare Mean-Square loss to the Wasserstein loss; they show that Mean-Square loss continues to oscillate and remains unstable past 2,500 epochs, evidenced by the generated images becoming increasingly monotonous [48]. Moreover, the Wasserstein loss is shown to converge after 2,000 epochs, with no further change or oscillation in the loss; the images generated maintain a good level of diversity and a high definition, showing that the Wasserstein loss is better for the generation of spot-based camouflage than the Mean-Square loss.

Furthermore, I believe that due to the nature of the CamoGAN it could be very useful as a starting reference for my own GAN, using the open-source code base and further expanding it using the Wasserstein loss [35] (and even further with the gradient penalty [38]), as it is shown by Yang et al. [48] to be greatly advantageous in terms of convergence speed, image quality and image diversity. Talas et al. [43] mention that the CamoGAN converges too quickly in some cases due to hyperparameter choice [43], meaning it would benefit from the stability found with the Wasserstein loss and gradient penalty [38]. This would allow me to leverage the existing work within the field of camouflage generation, benefit from the increased training stability and higher resolution images from the WGAN [35] and allow me to build deeper models using the gradient penalty found within WGAN-GP [38].

6 SPECIFICATION

In this section, I detail my plan for generating camouflage using Generative Adversarial Networks. The central hypothesis for this project is to investigate the extent to which Generative Adversarial networks can be used to generate effective camouflage. I begin by describing the functional and non-functional requirements of my project; finally, I discuss my evaluation criteria for the central hypothesis of my project.

6.1 Functional Requirements

I will be using Windows 10 and Python 3.9 to implement and test the system. In terms of the dataset being used, I intend to use the Scene Classification dataset [49], which contains 25,000 images of resolution 150x150 pixels. This dataset contains 6 classes, one of which being 'Forests' which I intend to explore. This is done in an attempt to generate camouflage as if it were being used in a forest or woodland setting, like those found in Figure 1. I will begin by training the networks to produce a 32x32 pixel image to fill in the square target hole. This will be my standard for training and testing, however camouflage is used often for less linear-shaped objects such as humans, so it would be pertinent (if there is the time to do so) to further explore arbitrary shapes, sizes and locations to better emulate real-world application:

- The Scene Classification Dataset [49] must be used to train and test the GANs
- All GANs must be trained to generate camouflage for a 32x32 central hole
- If the above requirement is met, the GANs should be further trained to generate camouflage for holes of arbitrary shapes, sizes or locations

I shall start by implementing two basic methods. The original GAN architecture [2] will be implemented despite the multiple flaws already mentioned previously (see sections 3 and 4) as it provides a good baseline for what is to be expected of generation. Furthermore, the open-source CamoGAN [43] shall be modified to fit the context of this project, demonstrating how a DCGAN [25] can be used. Finally, I have set two requirements which are not essential to be met but, if I have the time, I would like to explore as they introduce more complex concepts (the Wasserstein distance [35] and Gradient penalty [38]), both of which have been shown to provide more stability and better results:

- A basic GAN based on Goodfellow et al.'s method [2], must be implemented to use the Scene Classification dataset [49]
- The CamoGAN from Talas et al. [43] must be modified to utilise the Scene Classification dataset [49]
- Time permitting, the CamoGAN should be modified to utilise the Earth-Mover distance from the WGAN [35]
- If the above requirement is met, the modified Wasserstein CamoGAN should be further modified to utilise the Gradient Penalty from Gulrajani et al. [38].

In terms of the camouflage itself, I seek to explore the generation of Woodland camouflage (as seen in Figure 1) by using images from the 'Forests' class only. Alongside this, a non-essential requirement is defined, which is to explore other types of camouflage such as Alpine camouflage (by using the 'Mountains' class) or Urban camouflage (by using the 'Buildings' or 'Streets' classes); military personnel utilise more than one type of camouflage and should be investigated once the networks have been trained on the 'Forests' class.

6.2 Non-Functional Requirements

The number of epochs a network is trained for often affects the quality of generated samples (such as is demonstrated in Yang et al. [48]), so I aim to train each network to a maximum of 10,000 epochs or for a maximum time of 24 hours, depending on which one is the shortest. I make this distinction as I do not want to be training a single network for any longer than 24 hours, as extended time spent training the networks is, more often than not, extended time not spent working on other aspects of the project.

Another requirement for this project is that there is a user-friendly user interface for the human observation tests. I believe that having a clear, easy to use user interface for the tests will reduce the likelihood of an incorrect answer (as in, a user's answer is not what they intended) due to poor design and lack of clarity. The testing application will be local as to reduce development time and costs (due to web-hosting and data storage).

6.3 Evaluation Criteria

In this section, I detail how I will evaluate the success of my Generative Adversarial networks and how I will determine the extent to which the generated camouflage is effective.

As mentioned previously in section 6.1, I will be using the Scene Classification dataset [49]. In terms of the networks themselves, I shall track and compare the convergence speed of each network, such as is done in Gulrajani et al. [38]; this shall done by tracking the Loss metric from both the discriminator and generator of each respective network and graphing it with respect to the number of epochs. By tracking the convergence speed, I can determine which network is the most stable when applied to

generating camouflage and allow me to demonstrate instabilities in the networks; this can also allow me to demonstrate the overall performance of each network in both the number of epochs and the amount of real-world time.

6.3.1 User Testing. As camouflage is a cognitive illusion designed to trick an observer, the most conclusive way of testing my own generated camouflage is to test it on human participants. I intend to gather a minimum of 5 participants to visually test the generated camouflage. I shall show the participants, at random, either an original image from the Scene Classification dataset [49] or an image with a generated camouflage patch on it; the participant then has to answer whether or not they see a camouflaged patch (similar to the human evaluation seen in Denton et al. [33]). The participants will be given a random amount of time from a set of times (between 1 second and 60 seconds) to see the image, after the allotted time the image is removed and the answer inputs remain. This technique should allow me to determine the effectiveness of the generated camouflage by comparing the number of correct identifications across each network.

7 CONCLUSION

In this paper, I have reviewed generative and adversarial methods respectively, the combination of the two as Generative Adversarial networks, their architecture, generative and adversarial methods for generating camouflage and how Generative Adversarial networks can be applied to the problem of camouflage generation.

GANs have provided solutions to problems held by generative models for years, but researchers still face ongoing challenges whilst developing them. Complete convergence of the GAN model and the existence of a Nash equilibrium point are yet to be formally shown [13] and, as such, remain strong research points currently. The training process requires a careful balance between training the generator and the discriminator, otherwise problems with the quality of generated samples begin to appear [2]; however, controlling the synchronization of the two networks is very difficult and often involves fixing the generator whilst training the discriminator [2], causing the training process to become unstable [13], [17]. Although the instability of the original GAN [2] (and subsequent improvements [33]) proves to be a constant research challenge, newer techniques have shown improvements in the stability of the GAN, such as the DCGAN using ReLU to prevent the vanishing gradient problem in the regular GAN [25] or the WGAN-GP removing the instability in the original WGAN by replacing weight clipping with a gradient penalty [38].

Alongside this, the samples in GANs are usually as diverse as the image set provided, yet remain uncontrollable in most cases [23] and prone to mode collapse [22], in which case the generated samples are no longer diverse and provide little semantic difference [13]. However, techniques such as the WGAN partially solves the mode collapse problem [35], but mode collapse still remains a further research direction.

In terms of generating camouflage, there are few techniques which use Generative Adversarial techniques, with notable examples being the CamoGAN [43] for generation of natural prey camouflage and Yang et al. [48] exploring the generation of spot-based deformation camouflage using a WGAN; I believe that this project could serve to further explore the application of Generative Adversarial networks to the generation of camouflage, by expanding the work of those before into techniques of higher stability and quality.

REFERENCES

[1] M. A. Hogervorst, A. Toet, and P. Jacobs, "Design and evaluation of (urban) camouflage," in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXI*, International Society for Optics and Photonics, vol. 7662, 2010, p. 766 205.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[3] T. Newark and J. Miller, *Camouflage*. Thames & Hudson London, 2007.

[4] Great Britain. Army. Royal Engineers, *Aide-mémoire to the Military Sciences: Framed from Contributions of Officers of the Different Services*, ser. Aide-mémoire to the Military Sciences. J. Weale, 1853-1862, vol. 1, pp. 257–259.

[5] L. Talas, R. J. Baddeley, and I. C. Cuthill, "Cultural evolution of military camouflage," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 372, no. 1724, p. 20 160 351, 2017.

[6] S. Mohamed and B. Lakshminarayanan, "Learning in implicit generative models," *arXiv preprint arXiv:1610.03483*, 2017.

[7] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *arXiv preprint arXiv:2001.06937*, 2020.

[8] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

[9] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[10] H. White, "Maximum likelihood estimation of misspecified models," *Econometrica: Journal of the econometric society*, pp. 1–25, 1982.

[11] F. S. Richards, "A method of maximum-likelihood estimation," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 23, no. 2, pp. 469–475, 1961.

[12] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *International Conference on Machine Learning*, PMLR, 2014, pp. 226–234.

[13] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[14] J. Schmidhuber, "Learning factorial codes by predictability minimization," *Neural computation*, vol. 4, no. 6, pp. 863–879, 1992.

[15] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[16] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[17] X. Wu, K. Xu, and P. Hall, "A survey of image synthesis and editing with generative adversarial networks," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 660–674, 2017.

[18] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–43, 2019.

[19] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[22] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[23] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[26] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[27] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.
[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International conference on machine learning*, PMLR, 2010.
[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
[31] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.
[32] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
[33] E. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *arXiv preprint arXiv:1506.05751*, 2015.
[34] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*, Elsevier, 1987, pp. 671–679.
[35] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, PMLR, 2017, pp. 214–223.
[36] L. V. Kantorovich and S. Rubinstein, "On a space of totally additive functions," *Vestnik of the St. Petersburg University: Mathematics*, vol. 13, no. 7, pp. 52–59, 1958.
[37] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
[38] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.
[39] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.
[40] H. Du, X. Jin, and X. Mao, "Digital camouflage images using two-scale decomposition," in *Computer Graphics Forum*, Wiley Online Library, vol. 31, 2012, pp. 2203–2212.
[41] H. Du and L. Shu, "Camouflage images based on mean value interpolation," in *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, Springer, 2013, pp. 775–782.
[42] H. Yang and J. Yin, "A digital camouflage generation algorithm using color similarity," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 6, pp. 159–164, 2015.
[43] L. Talas, J. G. Fennell, K. Kjernsmo, I. C. Cuthill, N. E. Scott-Samuel, and R. J. Baddeley, "Camogan: Evolving optimum camouflage with generative adversarial networks," *Methods in Ecology and Evolution*, vol. 11, no. 2, pp. 240–247, 2020.
[44] 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 international conference on computer vision*, 2017, pp. 2223–2232.
[45] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.
[46] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6721–6729.
[47] A. Toet and M. A. Hogervorst, "Urban camouflage assessment through visual search and computational saliency," *Optical Engineering*, vol. 52, no. 4, p. 041 103, 2012.
[48] X. Yang, W.-d. Xu, Q. Jia, L. Li, W.-n. Zhu, J.-y. Tian, and H. Xu, "Research on extraction and reproduction of deformation camouflage spot based on generative adversarial network model," *Defence Technology*, vol. 16, no. 3, pp. 555–563, 2020.
[49] N. Bharathi. (Dec. 2018). "Scene classification dataset," [Online]. Available: https://www.kaggle.com/nitishabharathi/scene-classification (visited on 11/22/2021).