FOI

Deep learning with limited data: A synthetic approach

LINUS J. LUOTSINEN, FARZAD KAMRANI,
LUKAS LUNDMARK, JOHAN SABEL,
HARALD STIFF, VIKTOR SANDSTRÖM

FOI-R--5215--SE
ISSN 1650-1942 December 2021

Linus J. Luotsinen, Farzad Kamrani,
Lukas Lundmark, Johan Sabel, Harald Stiff,
Viktor Sandström

Deep learning with limited data: A synthetic approach

Bild/Cover: Photo by Sandeep Arora from FreeImages

Titel	Djupinlärning med begränsade datamängder: Ett syntetiskt angreppssätt
Title	Deep learning with limited data: A synthetic approach
Report no	FOI-R--5215--SE
Month	December
Year	2021
Pages	50
ISSN	1650-1942
Customer	Swedish Armed Forces
FOI Research area	C3 and Human Factors
Armed Forces R&T area	Command and Control
Project no	E85002
Approved by	Cecilia Dahlgren
Division	Defence Technology
Export control	The content has been reviewed and does not contain information which is subject to Swedish export control.

Abstract

This report focuses on how synthetic data, created using simulation or generative models, can be used to address the deep learning data challenge. These techniques offer many advantages: 1) data can be created for rare cases that are difficult to observe in the real world; 2) data can be automatically labeled without errors; and 3) data can be created with little or no infringement on privacy and integrity.

Synthetic data can be integrated into the deep learning process using techniques such as data augmentation or by mixing synthetic data with real-world data prior to training. This report, however, focuses mainly on the use of transfer learning techniques where knowledge gained while solving one problem is transferred to more efficiently solve another related problem.

Besides introducing synthetic data generation and transfer learning techniques, this report presents experimental results that provide valuable insights into the potential of the synthetic data approach in the context of pilot behavior cloning, vehicle detection and face verification tasks. Preliminary results from the experiments show that military simulators and generative models can be used to support deep learning applications. However, the performance is often limited by the fidelity gap between synthetic and real-world data.

Keywords

Artificial intelligence, machine learning, deep learning, deep neural networks, synthetic data, simulation, generative models, transfer learning

Sammanfattning

Denna rapport fokuserar på hur syntetisk data, skapad genom simulering eller generativa modeller, kan användas för att adressera databrist inom djupinlärning. Dessa tekniker erbjuder många fördelar: 1) data kan skapas för fall som är svåra att observera i den verkliga världen, 2) data kan annoteras automatiskt och felfritt, och 3) data kan skapas utan risk för att vara integritetskränkande.

Syntetisk data kan integreras i inlärningsprocessen med hjälp av tekniker som dataaugmentering eller genom att blanda syntetisk med verklig data innan träning. Den här rapporten utforskar främst överföringsinlärningstekniker i vilka kunskap som erhållits från att lösa ett problem kan överföras och återanvändas för att mer effektivt lösa ett annat relaterat problem.

Förutom att introducera syntetiska datagenererings- och överföringsinlärningstekniker presenterar denna rapport experimentella resultat som ger insikter i hur effektiv syntetiskt skapad data kan vara i tillämpningar som kloning av stridspilotbeteende, detektion av militära fordon och ansiktsverifiering. Preliminära resultat från experimenten visar att militära simulatorer och generativa modeller kan användas för att stödja inlärning i dessa tillämpningar. Prestandan begränsas dock ofta av gapet i realism mellan syntetisk och verklig data.

Nyckelord

Artificiell intelligens, maskininlärning, djupinlärning, djupa neuronnät, syntetisk data, simulering, generativa modeller, överföringsinlärning

Contents

1 Introduction

Deep learning (DL) is a technology that has increased the abilities to automate complex tasks in a wide range of real-world applications. Translation, transcription, video surveillance, recommendation systems and self-driving cars represent examples where DL-based solutions have already been developed and deployed for commercial purposes. In the military domain, DL has the potential to support human decision making in all domains and levels of warfare with applications such as automatic target recognition, predictive maintenance and automatic control of unmanned vehicles.

Similar to other machine learning (ML) techniques, DL uses algorithms to extract knowledge from data. The knowledge is in this case encoded in high-capacity deep neural networks (DNNs) that may consist of thousands, millions or even billions of adjustable parameters, depending on the complexity of the considered task. To properly adjust these parameters, the learning algorithm requires large quantities of training data. Without it the DNN will not be able to generalize, hence, it will not perform well when presented with previously unseen data.

Acquiring training data for DL is difficult. This is true in commercial applications but perhaps even more so in the military domain. One of the bottlenecks is that the learning algorithm often requires data that has been manually labeled (i.e., provided with a correct answer to each input data point). Therefore, even in cases where it is relatively cheap to acquire large quantities of input data, it is often expensive and time-consuming to correctly label all the data. For instance, each of the 5,000 samples in the Cityscapes dataset took on average 1.5 hours to label (roughly ten months for the full dataset) [1]. Furthermore, since labeling is performed by humans, the result can be incorrect, biased or even prejudiced, which would also be reflected in the behavior of the trained model.

In addition, training data often suffers from the long tail distribution problem. That is, training data is relatively easy to acquire for a limited number of common cases but inherently difficult to acquire for a large number of important edge cases. For instance, consider a UAV-based military vehicle surveillance and tracking system. In this case, aerial images of friendly vehicles are relatively easy to acquire. Vehicle data can be acquired in a variety of locations, altitudes, angles, weather conditions, environments, etc. Acquiring a similar real-world dataset representing the vehicle fleet of a qualified opponent is typically not possible as such an invasive intelligence operation would result in adversarial actions. A system trained using a dataset that follows a long tail distribution is often of limited practical value as it can only be used when conditions are ideal (i.e., the input data is similar to the common cases). The system will not perform well and cannot be relied upon when presented with real-world data representing edge cases.

1.1 Purpose and scope

The objective of this report is to introduce techniques that can be used to address some of the challenges associated with limited training data in a military context. Specifically, this report focuses on how synthetic data, created using military simulation or generative models, can be used in combination with transfer learning techniques such as fine-tuning, domain adaptation, multi-task learning and meta-learning to accelerate the development and deployment of future DL applications in the military domain.

1.2 Target readership

The target readership of this report is personnel that operate, acquire or develop military systems where AI, ML and DL technologies are used by or embedded in the systems.

1.3 Reading instructions

This report assumes that the reader has basic a knowledge about ML and DL concepts such as supervised learning, reinforcement learning, loss functions, gradient descent and backpropagation. Readers lacking such knowledge are encouraged to read chapter 2 in the FOI-report FOI-R–4849–SE [2] prior to proceeding with this report.

1.4 Outline

Chapter 2 provides an overview of techniques and methods that can be used to generate and integrate synthetic training data in deep learning. Chapter 3 provides an overview of transfer learning techniques that can be used to facilitate the reuse of knowledge from one task to another. In chapter 4, a subset of the techniques is evaluated, and experimental results that provides insight into the potential of the synthetic data approach are provided. Conclusions are presented in chapter 5.

2 Synthetic data generation

This chapter presents synthetic data generation techniques and discusses how to incorporate them into the DL pipeline.

2.1 Data augmentation

Data augmentation [3] is arguably the most popular and widely used data generation technique. It is often applied to improve model performance even in cases where there is no shortage of data. The idea is to expand the size of a training dataset by creating modified versions of existing data. For instance, image datasets are often augmented using transformation functions such as translation, rotation and scaling. Similarly, in the audio domain, functions such as time stretching and pitch scaling can be used.

Data augmentation is a straightforward technique that does not require significant modification to the training pipeline, DL algorithm or the DNN structure. Figure 2.1 illustrates data augmentation in a supervised learning context. Here, the inputs are transformed to create augmented inputs that in turn are fed into the model.

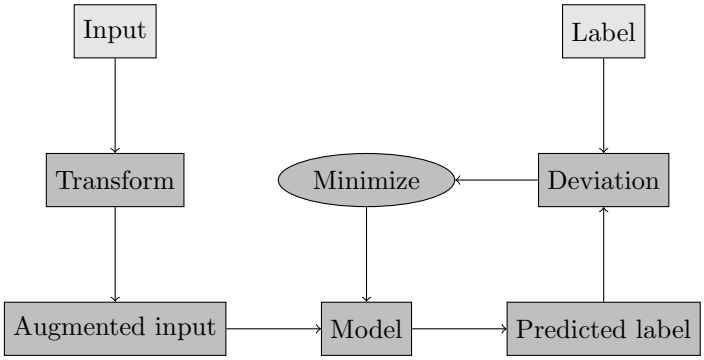
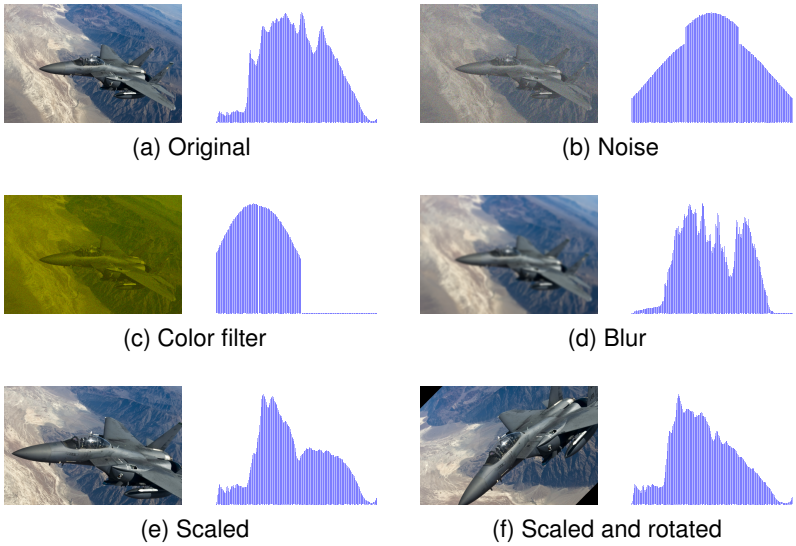


Figure 2.1: Data augmentation in supervised learning. Learning is performed using inputs that have been augmented using various transformation functions. Light gray boxes represent training data (i.e., inputs and their labels).

To better understand why data augmentation improves the performance of a model, consider a DNN that is trained for object classification in images. It is reasonable to assume that the model should be able to classify objects regardless of rotation, scaling, image noise, coloring, etc., as long as the image contains the same object. Figure 2.2 illustrates the effects of applying different transformation functions to an image of a fighter jet. The plots to the right of each image represent the distribution of the pixel values for each augmented image. In this case a human has no problem classifying the object in the images as a fighter jet. However, given that the distributions are very different from each other, a DNN may struggle to correctly identify the fighter jet. To mitigate this problem the model should preferably be trained with as many augmentations or semantically invariant transformations of the image as possible.

Many more augmentation methods exist, developed for specific domains and applications, but all methods rely on the idea of training on a more diverse distribution of data to achieve a more robust model.



(a) Original (b) Noise

(c) Color filter (d) Blur

(e) Scaled (f) Scaled and rotated

Figure 2.2: An image of a fighter jet (2.2a) that is augmented by adding noise (2.2b), color filter (2.2c) and blur (2.2d), and also by scaling (2.2e) and scaling followed by rotation (2.2f). Each image is accompanied by the corresponding plot over the distribution of the average RGB values of all pixels. While all images are semantically invariant, the shapes of the distributions vary significantly.

2.2 Self-supervision

Self-supervised learning is a technique where DNNs are trained using unlabeled datasets. The idea is to learn from auxiliary tasks that are created using the unlabeled data. For instance, reconstruction tasks are relatively easy to create from images or text by occluding parts of the image or text sequence [4]. Given such transformation it is possible to apply the standard supervised learning procedure (Figure 2.3) to train DNNs capable of reconstructing the occluded image or sentence on potentially unbounded unlabeled datasets.

Self-supervised learning results in DNNs that learn how to represent and extract features from data. These models contain general knowledge that is useful when learning other downstream tasks where, for instance, training data is limited. Self-supervised learning has been successfully applied in the text [5], image [6], and speech [7] domains, where there is an abundance of publicly available unlabeled data.

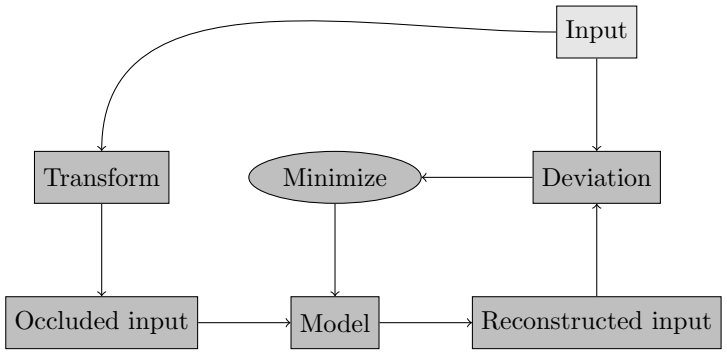


Figure 2.3: Similar to supervised learning, the objective of the self-supervised learning process is to create a model that minimizes the deviation from the presented training examples. In this case the task is to learn how to reconstruct (automatically created) occluded versions of the original inputs.

2.3 Simulation and domain randomization

Simulation provides a convenient way to generate data that can be used to create or expand training datasets for DL. The main advantage of using simulators, as opposed to data augmentation and self-supervision, is that they can be used to create arbitrary data representing edge cases that rarely occur in the real world. However, since simulators only represent simplified versions of the real world, the created data will also be of lower fidelity compared to data observed in the real world. Bridging this fidelity gap is perhaps the greatest challenge of the simulation-based synthetic data approach.

Domain randomization [8] is a technique that has been proposed to address this challenge. It is based on the idea that models will generalize to the real world with no additional training, assuming that the variability in the synthetic data is rich enough. In other words, the goal is to introduce sufficient variability in the data during training so that, when deployed, the real world will appear as just another variation to the model.

DL applications that take advantage of simulators and domain randomization mainly originate from the robotics and computer vision research communities. Despite similarities, these two communities employ synthetic data and simulators in rather different manners and are therefore discussed separately in the remainder of this section.

2.3.1 The robotics approach

Deep reinforcement learning (DRL) is a learning strategy that can be used to solve complex robotic control problems in simulated domains [9, 10]. DRL typically requires that the robot randomly explores the environment and gathers hundreds of thousands of samples. Clearly, such an approach is not feasible in terms of time and cost in physical environments. The ideal setting is that the policy (i.e., a function approximated by a DNN that maps the robot's different states to appropriate actions) is learned entirely in a simulation and later transferred to the physical domain, preferably without any additional training [8]. While simulation has been used for education and testing purposes for many years, significant progress has only recently been made in transferring capabilities learned in simulation to reality [11].

The domain randomization technique has been used in a wide range of robotic applications (e.g., robotic control training [12, 13, 14] and drone racing [15]). One issue with the original domain randomization, which uses uniform sampling to create data, is that not all possible simulation settings contribute equally to training. Some settings might even negatively affect the robot's ability to learn. One approach that has been proposed to address this challenge is automatic domain randomization [14]. Here, the learning algorithm is able to automatically determine the data it needs to learn, as opposed to the unguided random approach taken in standard domain randomization. In addition, the difficulty of the task gradually increases alongside the robot's ability to learn. Using only simulation, automatic domain randomization has been successfully applied to train a five-fingered robotic hand to solve the Rubik's cube [14] in the physical domain. The robot is able to manipulate the Rubik's cube even when two of the fingers are tied together. It can also solve the cube when pushed by external forces. None of these scenarios are encountered during training.

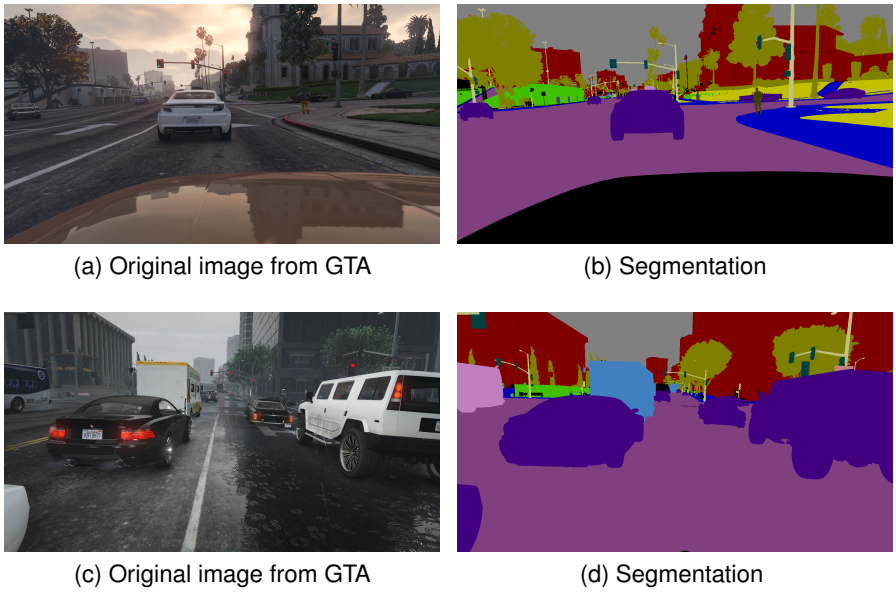
2.3.2 The computer vision approach

In recent years, computer vision researchers have started to use synthetic images created by various rendering tools for DL tasks such as object recognition [16, 17, 18, 19], semantic segmentation [20, 21, 22, 23, 24], and, more recently, face-related tasks such as face parsing (assigning a label to each pixel in an image, e.g., eyes, mouth or nose), and facial landmark localization (finding the position of facial points of interest in 2D) [25].

Semantic segmentation lays the foundation for many vision applications such as autonomous driving and medical image analysis. It involves assigning a label to every pixel in an image so that semantically equivalent pixels have the same label. Large training datasets are crucial for training semantic segmentation systems. However, manual segmentation of real images is a time-consuming and labor intensive task.

One way to overcome this problem and create large-scale training image data is to use commercial video games. The scale, appearance, and content of many video games have significant advantages over open-source 3D animation tools that lack this extensive content. As shown in [24], it is possible to create pixel-accurate semantic label maps for images (Figure 2.4) extracted from games without access to the source code or the internal operation of the game by using only the communication between the game engine and the graphics hardware.

Domain randomization in computer vision tasks has been used in two seemingly contrary manners:



(a) Original image from GTA (b) Segmentation

(c) Original image from GTA (d) Segmentation

Figure 2.4: Segmentation of two sample images from the popular game Grand Theft Auto V (Figure 2.4a and 2.4c), and the corresponding segmentation map (Figure 2.4b and 2.4d) [24].

- The parameters of the simulator (such as lighting and object textures) are randomized indiscriminately, without any efforts to make the rendered images realistic [26].
- Structured domain randomization, which takes into account the context of the scenes when synthesizing them. Instead of sampling different objects uniformly, each aspect of a synthetic image is generated given a probability that preserves the structure, or context, of the scene (e.g., the probability of cars, trucks or pedestrians being present on a road lane) [21].

The structured domain randomization is developed further in Meta-Sim [22] and Meta-Sim2 [23], in which creating the scene is automated. In Meta-Sim, it is assumed that the structure of each scene is correct (e.g., in a driving scene there is a road, a sidewalk and a number of cars), and the goal is to learn the attributes of each object (the location of cars, etc.), so that the gap between rendered images and real-world scenes is minimized. In Meta-Sim2, both the structure of the scene and the attributes of the objects are learned.

Another conceptually similar approach, albeit technically different from Meta-Sim, is *learning to simulate* [27]. Here, the idea is to learn how to configure the simulator used to create training data. Using this approach, the data generation process itself is optimized so that it can create the most relevant data needed by the model to maximize its performance. The approach reduces the amount of human expert effort required to configure the simulator and may also reduce the amount of training data needed to learn. As illustrated in Figure 2.5, the *learning to simulate* approach uses, reinforcement learning to learn the data generation policy.

In essence, methods such as structured domain randomization, Meta-Sim and Meta-Sim2 aim to reduce randomization in the data creation process. Whether more or less randomization yields better performance is considered to be an open question [23].

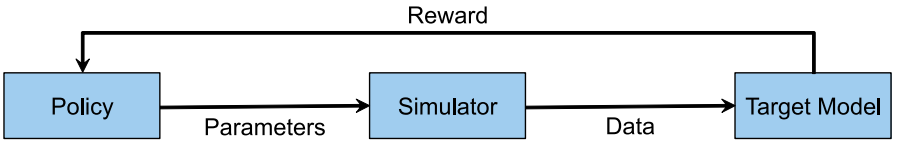


Figure 2.5: An overview of the learning to simulate approach. Using reinforcement learning a policy learns to tune the simulation parameters so that the generated data increases the accuracy of the target model.

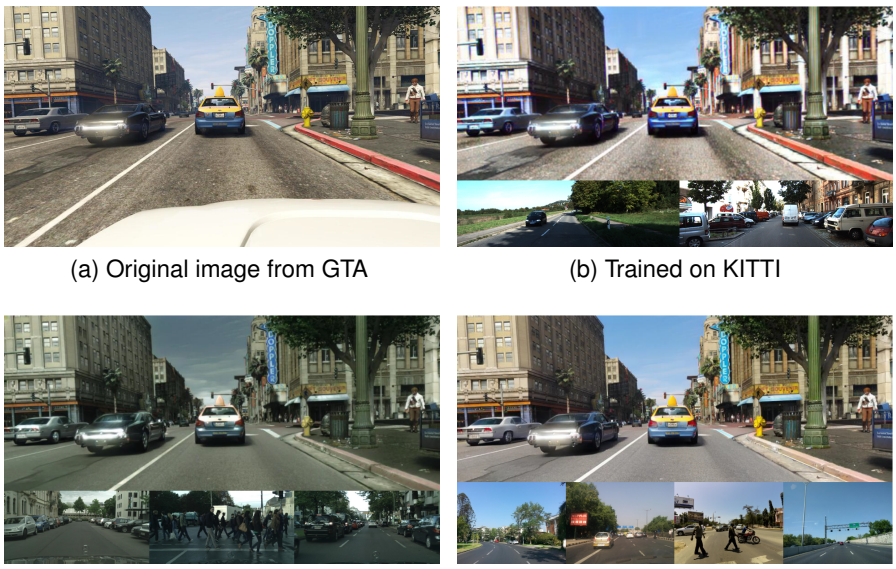
2.4 Generative models

Generative models have in recent years significantly improved the ability to generate synthetic yet highly realistic data to the degree that they are now a potential threat to society, primarily its digital and political security [28]. Regardless of such malicious use, these techniques also have the potential to accelerate the development and deployment of other DL applications in the military domain. The reader is referred to [29] for an in-depth introduction to generative models from a security and defense perspective. The remainder of this section focuses on a subset of the techniques, specifically those that have been applied to address the data challenge.

Many generative models for synthesis and manipulation of high-dimensional data such as images, texts, voices and videos have been introduced. A non-exhaustive list of applications that use generative models includes generating synthetic photographs of human faces [30], generating realistic photographs [31], image-to-image translation [32], text-to-image translation [33, 34], face-aging [35], 3D object generation [36], super-resolution [37], and video face swap [38].

Image synthesis methods that rely on generative adversarial networks (GANs) capture the underlying distribution of some large training datasets and generate new samples based on the learned distribution and bypass physical modeling of objects, material appearance, and lighting. One problem with these approaches is that it can be hard to control the details of the generated images, which is often required when addressing the data challenge.

Recently, Richter et al. [39] present a technique capable of enhancing the photorealism of synthetic images, which are produced by modern video games, using GANs (i.e., synthetic-to-real translation). They also introduce several architectural improvements in the DNN modules used for photorealism enhancement and report progress in stability and realism in comparison to a variety of other image-to-image translation approaches. The method is tested for transferring images acquired from the popular game Grand Theft Auto V [24] (source domain) to the real-world datasets KITTI [40], Cityscapes [1], and Vistas [41] (target domains). It is demonstrated that the model is able to transfer synthetic source images to visual styles of the real-world target datasets (Figure 2.6).



(a) Original image from GTA

(b) Trained on KITTI

(c) Trained on Cityscapes

(d) Trained on Mapillary Vistas

Figure 2.6: A rendered image from the popular game Grand Theft Auto V (Figure 2.6a) is enhanced using real images from KITTI, Cityscapes, and Vistas as target datasets (Figure 2.6b, 2.6c and 2.6d, respectively). The characteristic appearance of these datasets, (e.g., sensor noise in KITTI, saturation in Cityscapes, and fine textures in Vistas are reproduced while the structure of the original GTA image is kept. Sample images from the respective target dataset are inset beneath each image [39].

2.5 Summary

In this chapter, four different techniques to address the data challenge have been introduced:

- Data augmentation: This is an approach that expands the size of a training dataset by creating modified versions of existing data. The method is well elaborated and is already widely used to prevent overfitting.
- Self-supervision: In this approach DNNs are trained from auxiliary tasks and synthetic data that are automatically created using unlabeled data.
- Simulation: This approach offers a safe and relatively low-cost way to acquire labeled data representing rare edge cases. The goal is often to create data using high fidelity simulators so that the synthetic data is highly similar to the real world. However, high fidelity simulators are not always required, instead domain randomization can be used to create a diverse training dataset that, despite appearing unrealistic to humans, results in better models.
- Generative models: This approach can be used to create highly realistic data or even to improve the realism of data created using simulators.

3 Transfer learning

Transfer learning (TL) [42, 43, 44] is a broad research topic in ML that focuses on applying knowledge gained while solving one problem to another related problem. TL is typically used to train models that could otherwise not be trained successfully due to insufficient access to training data or limited computational resources (e.g., embedded systems and laptops). Although there are many variants of TL, this report focuses primarily on techniques that have been evaluated and applied in a DL context.

Specifically, this chapter presents TL techniques that can be used to extract and transfer knowledge from models trained using synthetic datasets, which are relatively easy to acquire, to more efficiently train other models using datasets that are limited in size and difficult to acquire. A subset of the TL techniques identified in this chapter are evaluated for such purposes in chapter 4.

3.1 Notation and definition

The notation and definition that follow originate from [42]. TL can be defined in terms of domains and tasks. A domain, $\mathcal{D}$, consists of two components: 1) a feature space, $\mathcal{X}$, and 2) a marginal probability distribution, P_X over $\mathcal{X}$. For a given domain, $\mathcal{D}$, a task, $\mathcal{T}$, also consists of two components: 1) a label space, $\mathcal{Y}$, and 2) a predictive function $f(\cdot)$ parameterized by θ, that are learned from pairs of feature vectors and labels $(x^{(i)}, y^{(i)})$, where $x^{(i)} \in \mathcal{X}$ and $y^{(i)} \in \mathcal{Y}$. Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ on $\mathcal{T}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

Domains cannot be observed directly, although sampled data from domains can be observed. In this report, source and target domain samples are collected into datasets $D_S = \{(x_S^{(i)}, y_S^{(i)})\}_{i=1}^{n_S}$ and $D_T = \{(x_T^{(i)}, y_T^{(i)})\}_{i=1}^{n_T}$, respectively. In general, TL techniques are useful when the amount of source domain data is much greater than the amount of target domain data, $0 \leq n_T \ll n_S$.

3.2 Pre-training and fine-tuning

Pre-training and fine-tuning are arguably the most popular and commonly used TL techniques in DL. Both techniques assume that there exists either a large labeled training dataset, D_S, that can be used to train a model, $f_S(\cdot)$, from scratch or more commonly that the model, $f_S(\cdot)$, has already been trained for a given source domain and task. Typically, the label spaces of the source and target tasks are different from each other, $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$. The main idea, as illustrated in Figure 3.1, is to:

1. Copy parts of the source model, $f_S(\cdot)$, into a target model, $f_T(\cdot)$.
2. Add one or more randomly initialized (untrained) layers to $f_T(\cdot)$ so that the last layer matches the target label space, $\mathcal{Y}_\mathcal{T}$.
3. Train the target model, $f_T(\cdot)$, using target domain data, D_T.

Pre-training and fine-tuning differ only in step 3. In pre-training the model parameters in $f_T(\cdot)$, which were copied from the source model, are frozen. That is, only the parameters in the additional, randomly initialized, layers are updated during training. In this case, the source model is merely a feature extractor used by the target model. When fine-tuning is used some of the

layers originating from the source model are also updated, or fine-tuned, during training.

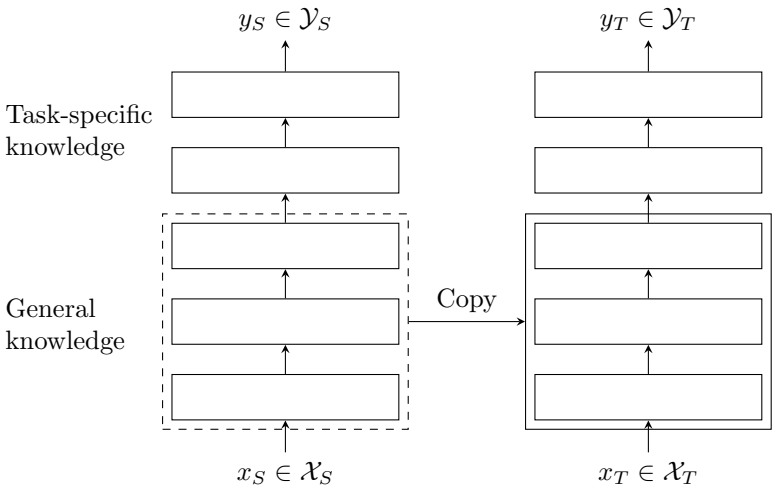


Figure 3.1: Pre-training and fine-tuning is performed by copying layers from a source model (left) to a target model (right). Only the task-specific layers closest to the output in the target model need to be trained. When successfully applied the target training data, D_T, is typically much smaller than the source training data, D_S.

A major advantage of the pre-training and fine-tuning techniques is that they are relatively easy to apply. Another upside is that there are many downloadable pre-trained models on the internet that have already been trained for weeks or months using specialized high-end computational resources. There are, however, also several downsides associated with the pre-training and fine-tuning techniques:

- Reusing models or even datasets that are available online may introduce vulnerabilities that are unacceptable in many military applications [45].
- Pre-trained models typically only exist for common domains such as text, images and audio. Military models trained using sensor data from LIDAR, IR, SAR, hyperspectral and other military specific domains are typically not shared.
- There is no guarantee that a source model is suited for a given target task. The techniques only perform well when the source and target domains are similar and related to each other. Hence, in practice many different models need to be empirically evaluated to identify good candidates.
- Selecting which layers to freeze and which layers to fine-tune is important to ensure that only useful knowledge from the source task is transferred to the target task [46]. Performance drops are expected if specific knowledge from the source task is also transferred to the target model.

Similar to other TL techniques many improvements have been proposed. Adaptive fine-tuning [47] is one such approach, where the layers to freeze and update are automatically selected by the learning algorithm for each training sample. This approach improves the model's ability to adapt to the target domain and may also reduce the amount of human effort required to train models using the fine-tuning approach.

3.3 Domain adaptation

Domain adaptation (DA) [48, 49, 50] is another common approach to TL where, in contrast to pre-training and fine-tuning, the source and target domains are different, $\mathcal{D}_S \neq \mathcal{D}_T$, but the tasks are the same, $\mathcal{T}_S = \mathcal{T}_T$. Specifically, the marginal probability distributions of the data are different in the domains, $P_{X_S} \neq P_{X_T}$. DA can be either homogeneous or heterogeneous. In the homogeneous case the feature spaces remain the same, $\mathcal{X}_S = \mathcal{X}_T$, whereas they are different in the heterogeneous case, $\mathcal{X}_S \neq \mathcal{X}_T$. Furthermore, DA can be supervised, semi-supervised and unsupervised where target domain data, D_T, is represented by labeled data, a mix of labeled and unlabeled data and unlabeled data, respectively.

This report focuses primarily on homogeneous and unsupervised DA (UDA) because of its intriguing capability to support the synthetic data approach presented in this report. The UDA approach assumes that labeled data is cheap and easy to acquire for the source, D_S, and that only unlabeled data is required for the target, D_T. UDA can be used to manage domain shifts that occur as a result of sensor variations. For instance, [51] addresses the challenge of recognizing faces in the video domain, where labeled data is limited and expensive to acquire, by transferring knowledge using labeled datasets that have been acquired and used to recognize faces in the image domain.

UDA has already been proposed as a technique for bridging the gap between synthetic and real-world data. This is actively being pursued [52, 53, 54, 55, 56, 57] using a variety of techniques, such as adversarial domain loss [58], generative adversarial networks (GANs) [59, 60, 61], and divergence minimization [62, 63].

The UDA approach taken in [58] is often used as a baseline when comparing and evaluating other UDA techniques. Figure 3.2 illustrates the concept of this approach. Here, the network is separated into three trainable modules, where the parameters are optimized using different objectives. The primary task is to minimize the error given the source dataset, D_S. The domain classifier is trained using inputs originating from both the source, D_S, and target, D_T, datasets, with the objective of minimizing the domain classification error (i.e., its goal is to determine if an input originates from D_S or D_T). Finally, a domain invariant feature extractor, ϕ, is implicitly trained to minimize the error of the primary task, but also to maximize domain classification error. The maximization objective is in this case achieved using a gradient reversal layer that negates the gradients originating from the domain classifier during training. As a result, the feature extractor is trained to map source and target data to a feature space where both datasets have a similar distribution (i.e., $P_{X_S} \neq P_{X_T}$, but $\phi(P_{X_S}) \approx \phi(P_{X_T})$). The domain classifier and the feature extractor are adversarially trained to optimize domain classification and misclassification (i.e., domain confusion), respectively.

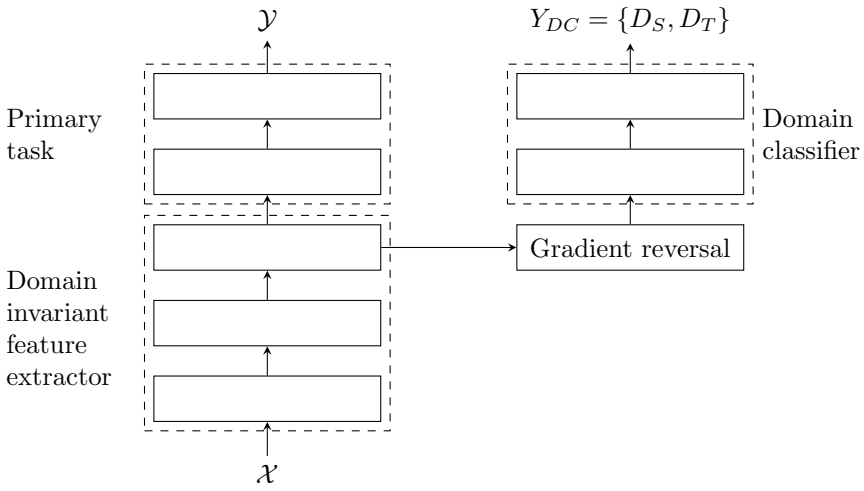


Figure 3.2: UDA using an adversarial multi-task learning approach [58].

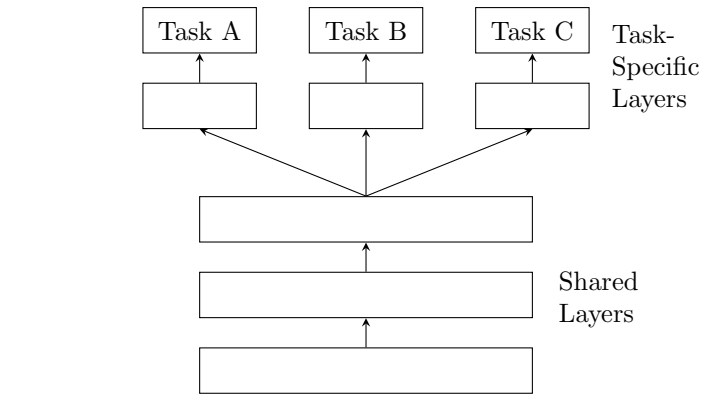
3.4 Multi-task learning

In multi-task learning (MTL), multiple tasks are trained simultaneously in a single DNN. Thereby, knowledge from one task is reused in another task [64, 65]. The main goal in MTL is to improve generalization by using the domain information contained in the training signals of related tasks. MTL accomplishes this by training tasks in parallel while using a shared representation. As a result, the training signals for extra tasks serve as an inductive bias (a set of hints that the learner uses to predict outputs of unseen inputs) [65].

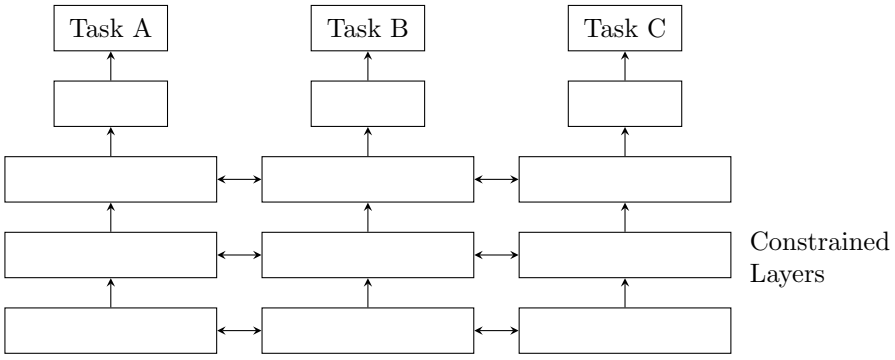
To provide a sense of how MTL works, consider Figure 3.3a, where a single DNN with three outputs aims to solve three different (but related) tasks. Each of the three outputs corresponds to one of the tasks. These three outputs are fully connected to a shared hidden layer and the backpropagation is performed in parallel on the three outputs in the DNN. Since the three outputs share common hidden layers, it is possible for one task to use the internal representations that have been learned in the hidden layer by other tasks. The central idea in MTL is sharing what is learned by different tasks while tasks are trained in parallel [66, 65]. This can be compared with a scenario where three distinct DNNs are trained separately for the same tasks, without the possibility of sharing what they learn.

While early literature discussed MTL in non-neural models and shallow neural networks, most recent literature has mainly focused on how MTL can be used in DNNs. In this context, MTL is typically achieved with either hard or soft parameter sharing of hidden layers [67] (Figure 3.3):

- Hard parameter sharing: This is, in fact, the original MTL as suggested by [64, 65]. It is applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers as shown in Figure 3.3a.
- Soft parameter sharing: Here, different tasks have their own models with their own parameters. However, to share extra information in the training signals between tasks, the distance between the parameters in corresponding layers are regularized using some regularization method (e.g., ℓ_2 distance). See Figure 3.3b for a high-level outline of the soft parameter sharing MTL.



(a) Hard parameter sharing.



(b) Soft parameter sharing.

Figure 3.3: MTL parameter sharing approaches in DNNs [67].

MTL provides a solution for applications in which there is an interest to predict multiple tasks simultaneously. However, in many situations there is only concern about one task. To make use of the benefits of multi-task learning, one can still find suitable auxiliary tasks and use extra information signal during training to generalize over a wider set of data even though these predictions are not required. Hence, when deployed the auxiliary networks are completely discarded. Examples of auxiliary tasks and auxiliary networks in the context of self-driving cars are presented in [68, 69, 70].

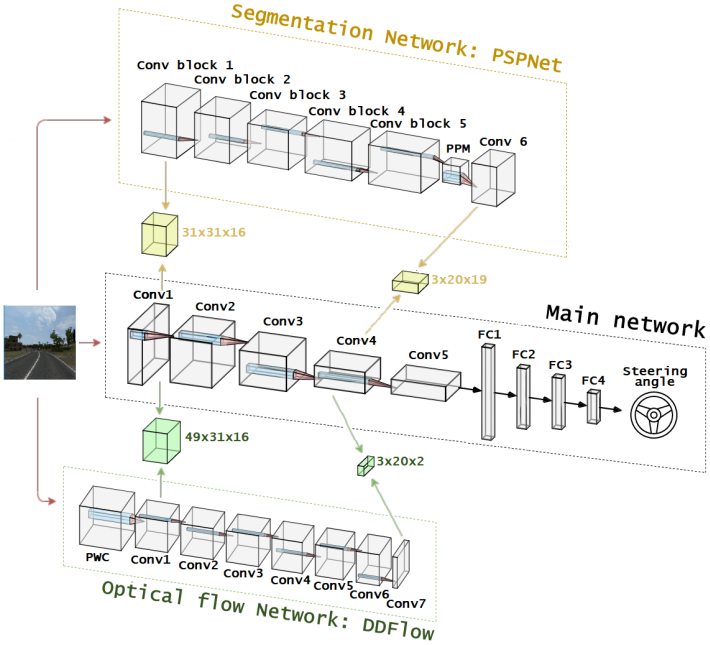


Figure 3.4: A DNN is trained to predict steering wheel angle of a self-driving car using images, where a segmentation network and an optical flow network contribute with their extracted features with soft parameter sharing. The two auxiliary networks are discarded after training [71].

3.5 Meta-learning

Meta-learning [72, 73], or learning to learn, techniques are specifically designed to extract meta-knowledge (i.e., knowledge on how to learn) from one or more source tasks. That is, in contrast to the other TL techniques introduced in this chapter, a meta-learner typically consists of an outer learner, where a meta-objective function is explicitly defined for the purpose of updating one or more, task-specific, inner learners so that the outer meta-objective gradually improves throughout training. When successfully applied the model will not necessarily perform well on any specific task, instead it will quickly adapt and learn new tasks given only limited training data.

Meta-learners are often applied and evaluated using few-shot, or k-shot, learning (FSL) problems, where multiple tasks, each represented by only a limited number (e.g., $k = 1$, $k = 5$, $k = 10$ and $k = 20$) of task-specific training examples, are used to train a model. After training, evaluation is performed using a different, but related, set of tasks to measure the model's ability to extract useful knowledge (i.e., generalize across task) to rapidly learn new tasks. Note that, in the case of FSL classification, the n-way k-shot notation is used, where n is the number of classes in the task, and k is the number of training

examples per class.

Training data for FSL originates from multiple source tasks and is therefore also represented by multiple training datasets. Each task's training dataset is separated into support and query sets, $D_S = \left\{D_S^{support}, D_S^{query}\right\}$. The support set consists of k training examples, or $k \times n$ in the classification case, and is used to learn the specific task. The query set consists of q examples, or $q \times n$ examples in the classification case, and is used to learn the meta-objective. Furthermore, multiple test datasets (i.e., target tasks), $D_T = \left\{D_T^{support}, D_T^{query}\right\}$, are typically used to evaluate the learner's performance. During testing the support dataset represents the knowledge required to learn a new task, whereas the query data is used to gather performance statistics.

Meta-learning can be implemented using a wide range of techniques. This section introduces two different approaches. The first approach learns a meta-model where the parameters have been optimized for fast adaptation using gradient descent. The second approach learns embedding functions that capture meta-knowledge. Yet another approach, not covered in this work, uses memory augmented neural networks (MANN) [74], e.g., the neural Turing machine (NTM) [75], where attention and external memory mechanisms are used to facilitate meta-learning.

3.5.1 Optimization

Model agnostic meta-learning (MAML) [76] is an influential meta-learning approach that can be applied to any learning problem and model that is trained using backpropagation and variants of the gradient descent procedure. Backpropagation is the standard approach used in DL to find the directions (or gradients), $\nabla_\theta \mathcal{L}(f_\theta)$, required to minimize an error or loss function, $\mathcal{L}(f_\theta)$, using gradient descent. Thus, MAML can be applied in both supervised, unsupervised and reinforcement learning problems using different DNN architectures (e.g., fully connected, convolutional and recurrent).

The meta-objective of MAML is to find the optimal initialization of the trainable parameters, θ, for a given DNN architecture, f_θ. Learning is performed iteratively, where the outer loop initially provides a randomly initialized θ to the inner loop. The inner loop iterates over multiple tasks, where each task i is represented by a training dataset, $D_{S_i} = \left\{D_{S_i}^{support}, D_{S_i}^{query}\right\}$, with only k training examples (or $k \times n$ in the classification case) to promote fast adaptation. Given f_θ and $D_{S_i}^{support}$ a new set of parameters, θ_i', can be acquired using backpropagation and gradient descent. The outer loop then applies backpropagation and gradient descent again using the updated DNNs from the inner loop, $f_{\theta_i'}$, and the query dataset, $D_{S_i}^{query}$. That is, the outer loop literally finds the gradients of the gradients and uses this information to update θ. As a result, the outer loop updates θ so that the average performance over all tasks is increased. This process is repeated until θ has converged to a satisfactory solution. The MAML algorithm is visualized in Figure 3.5.

Many improvements to the MAML algorithm have been proposed. First order MAML (FOMAML) [76] and the Reptile algorithm [77] approximate the second order derivative of MAML to reduce the number of computations needed. MAML++ [78] improves the ability of MAML to generalize and converge and also reduces the computational overhead of the algorithm. The ANIL algorithm demonstrates that MAML can learn with almost no inner loop [79].

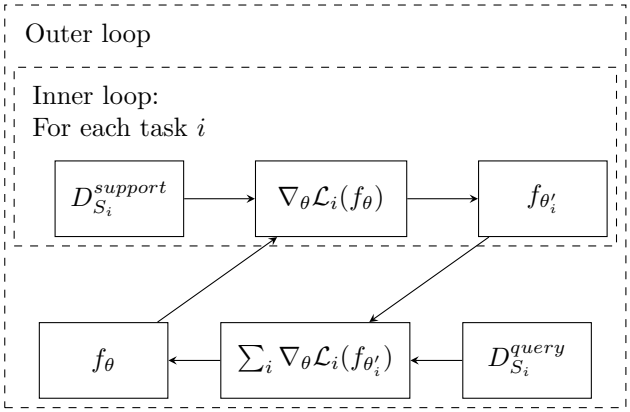


Figure 3.5: MAML consists of two loops. The inner loop learns specific tasks using a support dataset, $D_{S_i}^{support}$, that consists of only k training examples (or $k \times n$ in the classification case) to promote fast adaptation. The outer loop then uses a query dataset, $D_{S_i}^{query}$, and the information gained in the inner loop (i.e., θ_i') to update the trainable parameters in θ. The outer loop updates θ so that the meta-objective, in this case the average performance over all tasks, is increased.

3.5.2 Metric

The objective of metric-based meta-learning is to learn an embedding that, combined with a similarity function, can be used to determine how close inputs are to each other. For instance, in a one-shot learning context where each example in the support set represents a unique class, a query example is classified by calculating similarity values for all examples or classes in the support set and then picking the class it is most similar to.

One simple approach, often used as a baseline to evaluate other FSL algorithms, is the siamese network [80]. In this case inputs are, as illustrated in Figure 3.6, processed in pairs using two identical DNNs (i.e., two bodies) that output an embedding or feature vector representing each input. The embeddings are then merged (i.e., into one head) using a distance layer (e.g., Euclidean distance or cosine simularity) and a sigmoid-based output layer that is used to squash the distance value into a continuous similarity value between 0 (i.e., not similar), and 1 (i.e., similar). Training is performed using positive and negative sample pairs, where the two inputs belong to the same (i.e., $y = 1$) and, different (i.e., $y = 0$) class, respectively. Given that this is now a binary classification task, the siamese network can be trained using the binary cross entropy loss function. Improvements to the siamese networks approach include replacing the binary cross entropy with contrastive [81] or triplet [82] loss functions (see section 4.3 for a case study where triplet loss is used for face verification purposes).

Recent work has shown that metric-based approaches can be used in combination with pre-training to achieve state-of-the-art FSL results [83, 84]. The idea is to transfer knowledge acquired in a pre-trained network by removing the output layer and using the remainder of the network to produce embeddings (see section 3.2). In the case where the support set contains multiple embeddings for each class (i.e., $k > 1$), the average value of all embeddings are used to create a single centroid embedding. The cosine similarity function is then applied given the query set and the centroid embeddings to generate a prediction. The approach has also been extended to include a meta-learning phase using fine-tuning that improves performance even more [84].

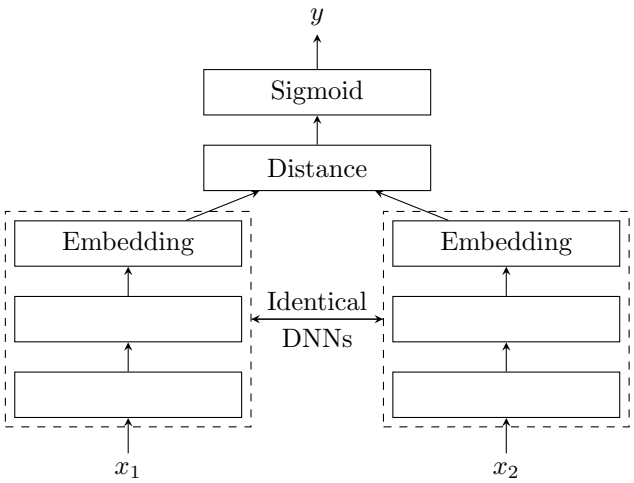


Figure 3.6: Siamese networks consist of two identical networks that share a single head. The network is capable of calculating a similarity metric given an input pair. Each input is first processed by the twin networks to generate an embedding. The embedding is then merged into a single head using a distance layer whose output is then processed by an output layer where a sigmoid activation function is used to produce a similarity value that can be thresholded to determine if the two inputs are from the same class or not. Siamese networks can be trained using binary cross-entropy, contrastive or triplet loss functions.

3.6 Domain generalization

Domain generalization (DG) aims at learning from a set of domains, and from this learning procedure extract a model that generalizes well on related unseen domains [85]. That is, the model must learn domain-invariant features from the source domains in order to be successful on the unseen target domain. Unlike DA and meta-learning, the target data is not accessible in DG.

To demonstrate the application of DG, consider the Photo-Art-Cartoon-Sketch (PACS) dataset [85]. This dataset contains roughly 10,000 images in seven categories (person, horse, elephant, guitar, dog, giraffe and house) in the four different styles suggested by the name (as illustrated in Figure 3.7).

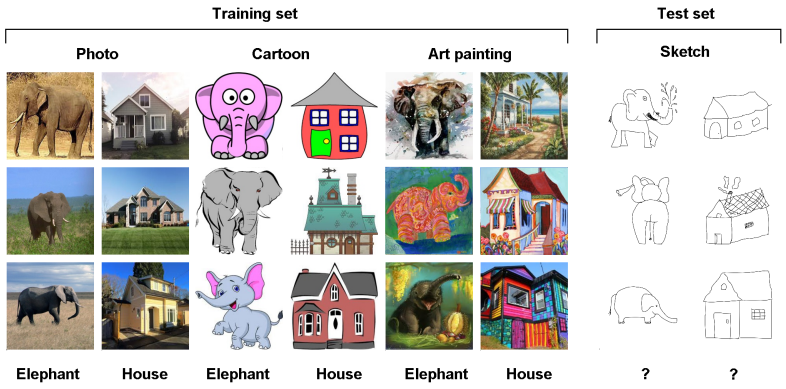


Figure 3.7: Examples from the dataset PACS [85] for domain generalization. The training set is composed of images belonging to domains of photos, cartoons, and art paintings. The aim of the learning is to learn a generalized model that performs well on the unseen target domain of sketches.

Suppose that the goal is to correctly classify the images of one specific style, but only have access to the remaining styles at training time. The problem of DG is then to use the remaining datasets to train a model that performs well on the style in question. This could be to classify sketch images using photo, art and cartoon images (Figure 3.7). This benchmark problem has been widely studied [86, 87, 88], reaching up to 88% average accuracy.

A model capable of generalizing in unseen domains is highly desirable. A comprehensive survey of existing DG methods is provided in [89]. In this survey, DG is categorized into three groups:

1. Data manipulation: These methods attempt to increase the diversity and quantity of existing training data using techniques such as interpolation, data augmentation, domain randomization and generative models. For instance, interpolation has shown promising results on the PACS problem, reaching 83.7% [87].
2. Representation learning: These methods are based on decomposing the predictive function into two parts, $f_\theta = g \circ h$, where g is a classifier function and h is a representation learning function. The latter function is designed to either learn a domain-invariant feature representation of the data or to decompose a feature representation into domain-invariant and domain-specific features. Kernel methods and multi-component analysis are techniques that can be used to learn h.
3. Learning strategy: These methods focus on exploiting the general learning strategy to boost generalization. It mainly includes two different approaches: 1) ensemble learning DG, which uses an ensemble of models to learn a unified and generalized model, and 2) meta-learning DG, which uses a mechanism to learn general knowledge by constructing meta-learning tasks to simulate domain shifts (see section 3.5).

3.7 Summary

The TL techniques introduced in this chapter highlight a subset of the most popular and promising approaches used in DL to more efficiently learn new tasks through the transfer of previously acquired knowledge. The remainder of this report focuses on how these TL techniques can be used in combination with synthetic data generation to address the DL data challenge.

4 Case studies

This chapter introduces and provides experimental results from case studies where synthetic data and TL techniques have been applied to train DNNs for behavior cloning (section 4.1), vehicle classification (section 4.2) and face verification (section 4.3) tasks. In the first two cases, data was generated using a military combat simulator (Virtual Battlespace 3 or VBS3), and in the last case a generative model was used. In all cases the majority of the training data was synthetic. Hence, the results represent baselines where improvements were expected as more real-world data was acquired and integrated into the learning process over time.

4.1 Fighter pilot behavior cloning

DL techniques can potentially be used to construct realistic models of human behavior for fighter pilots [90]. Such models are expected to improve the realism and also the training effects of simulation-based training facilities. The present case study examined the fine-tuning TL technique (see section 3.2) in the context of one such fighter pilot model. As a part of an ongoing research project at the Swedish Defence Research Agency (FOI), the main idea was to investigate how well this model can learn different flight dynamics models (FDMs). An FDM is a mathematical function used in flight simulators to calculate the succeeding state given the current state and some input [91].

The experiments were designed to resemble a scenario in which a fighter pilot model is trained using a low-fidelity flight simulator where large datasets are cheap to produce, and subsequently transferred and fine-tuned in a different high-fidelity flight simulator environment where training data is scarce. The work was conducted at FOI in 2021 as part of a master thesis [92].

4.1.1 Concept

A large number of problems exist where predictions affect the environment in some way. These problems are generally of a sequential nature, where each state is followed by an action that affects the next visited state. Such problems are regularly addressed in reinforcement learning and imitation learning. The latter represents methods to train models by imitating experts. These expert demonstrations are generally comprised of sequences of state-action pairs, meaning that the expert demonstrates the correct action given the current situation. An imitation learning model (commonly referred to as a learner policy) is said to guide an agent in an environment, such as, drive a van (agent) on a test track (environment) [93]. Imitation learning is generally a difficult problem, but is often approached with some simplifying assumptions. The simplest assumption reduces the imitation learning problem into a supervised learning problem, an approach commonly called behavior cloning. Although this reduction makes training easier, it can result in poor performance, as explained in [94].

Suppose that a trained learner policy is available and performs well, then it is natural to investigate how well this specific policy performs in a related environment. It is conceivable that a policy capable of acting in one environment can learn the new environment more efficiently compared to a randomly instantiated policy. In this study, a policy was pre-trained to fly a Jas Gripen 39 FDM on a track in the VBS3 combat simulator. This policy was subsequently

queried to fly an F22 represented by a significantly different FDM. The effect on the performance as a function of the amount of training data was measured and compared against a model that was not pre-trained.

4.1.2 Military applications

Fighter pilots conduct much of their training in simulator facilities. In these environments, the pilots often train various combat scenarios using realistic computer generated forces (CGFs). DL has the potential to improve the realism of CGFs, but this also requires that the facility has the capability to to train and deploy such models. Typically, adding DL capabilities to an existing training facility requires significant effort. A proposed solution is to develop CGFs using an alternative flight simulator that was specifically designed for DL, and subsequently use TL to incorporate the models into the target facility.

4.1.3 Experimental setup

This section describes the data acquisition infrastructure and how to enable aircraft control for both learner and expert policies. The considered task is explained in detail, as is how to measure the model performance and TL efficiency. The final policy is represented by a three-layered DNN trained using behavior cloning (i.e., supervised learning).

Flight scenario

The policy is trained to control and navigate the agent around a track consisting of five waypoints placed at a distance of 15 km (Figure 4.1). A waypoint is considered complete if the aircraft passes within a sphere of radius 250 meters, an error margin corresponding to about one degree from the straight line connecting the two waypoints. Given three runs, the objective was to complete a full lap on each run.

Using the TL notation from section 3, the source and target domain originate from the same simulator but using two different FDMs, and the source and target task is identical, i.e., $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$.

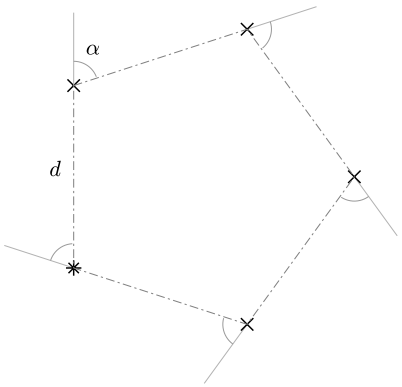


Figure 4.1: Task illustration. Each waypoint is placed a distance $d = 15\,km$ with $\alpha = 72°$. A cross marks a waypoint and the star marks the starting position. The track is oriented clockwise.

State vector

To complete the flight scenario, the policy must receive enough information to be able to determine the correct action in a given state. This means that it must be able to navigate to the next waypoint, orient the aircraft with respect to the waypoint and the surface of the earth and, ideally, predict some future state to avoid catastrophic errors. The navigation and orientation quantities are: 1) a vector pointing towards the waypoint, and 2) a unit vector pointing in a cardinal direction, both in a local reference frame. These vectors are calculated using Tait-Bryan angles and basic matrix transformations [95]. The last quantity are derivatives. The main reason for including this are as follows. Consider two different scenarios: 1) a snapshot of the aircraft at level flight, and 2) a snapshot of the aircraft at the exact moment where the aircraft is at level flight but is rotating at high velocity. If the derivatives are not included, the policy would consider the two scenarios as identical, but the second scenario would clearly require a compensating action to compensate for this rotation. The state vector at time t could thus be written abstractly as

$$x_t = \big(\text{navigation}, \quad \text{orientation}, \quad \text{derivatives}\big)^\top,$$

for this specific problem. Note that this was a sufficient, but not necessarily optimal, choice of state vector. If the scenario had been different, then a different state vector might have been needed.

Measuring performance

Two metrics are used to measure performance. This first and primary metric is the ratio of completed waypoints, RoC. If $RoC = 0.8$, then 80% of all waypoints were completed.

The second metric is called relative closeness, $\underline{C_R}$, and measures how close to the completed waypoint the aircraft passes, compared to the expert demonstrations. Assuming that the learner perfectly imitates the expert, then the relative closeness would be $\underline{C_R} = 1$ and form a strict lower bound. If $\underline{C_R} > 1$, then the agent (on average) performed worse than the expert.

Control setup

In previous work [96], a software infrastructure was developed to enable both human and AI control of the aircraft in the simulator. The setup uses two control loops: 1) a control loop that allows a human pilot to control the aircraft, and 2) a control loop allowing the policy to steer the aircraft. The first control loop uses a manual joystick, which is mapped to a virtual joystick software to steer the aircraft. The virtual joystick feeds the simulator with control data and a human pilot can operate the aircraft through computer screen visuals.

The second control loop uses code that extracts the data from the simulator, converts it to the desired state space representation and subsequently feeds it into the trained policy. The policy outputs control signals to the virtual joystick, which in turn controls the agent in the simulator. Figure 4.2 illustrates the full setup. In the figure, a block denoted *Data Manager* calculates the state vector from raw data and also produces a dataset for training and performance analysis.

4.1.4 Results

The results of the experiments are summarized in Table 4.1. Both policies performed better as a function of the amount of data used in training, and the pre-trained model (Policy B) was able to complete the full track ($RoC =$

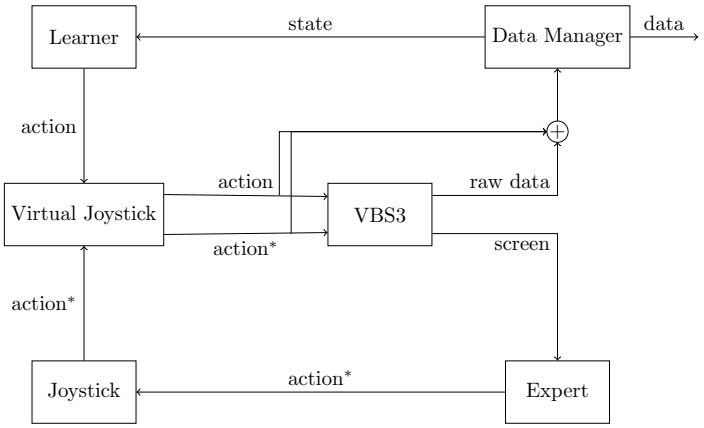


Figure 4.2: Illustration of control and data gathering setup. The star superscript marks the expert (human) pilot.

1) after training on 75% of the data, whereas the other model (Policy A) required training on the full dataset to reach $RoC = 1$. The results indicate that fine-tuning can be used to reduce the amount of data to reach a certain level of performance in the target domain by about 25%. Most likely, more sophisticated TL techniques could increase this number significantly.

Table 4.1: Performance for each percentage of the dataset used in training. Recall that Policy B was pre-trained in the source domain.

% of data	Policy A		Policy B	
	RoC	$\underline{C_R}$	RoC	$\underline{C_R}$
0	0.000	-	0.200	1.193
25	0.267	3.168	0.600	2.417
50	0.267	2.725	0.600	2.412
75	0.467	1.882	1.000	2.208
100	1.000	2.357	1.000	1.063

As for the relative closeness metric, it is more difficult to draw conclusions. The relative closeness for Policy B reached a level almost as good as the expert ($\underline{C_R} = 1$), but the values show a high variance. For this to be a more decisive number, the sample size must be increased to reduce the statistical error. However, if one compares the relative closeness for the runs where $RoC = 1$, it seems that Policy B reached Policy A's closeness value already at 75% and the last 25% reduced the value close to one. The resulting trajectories for Policy B are shown in Figure 4.3.

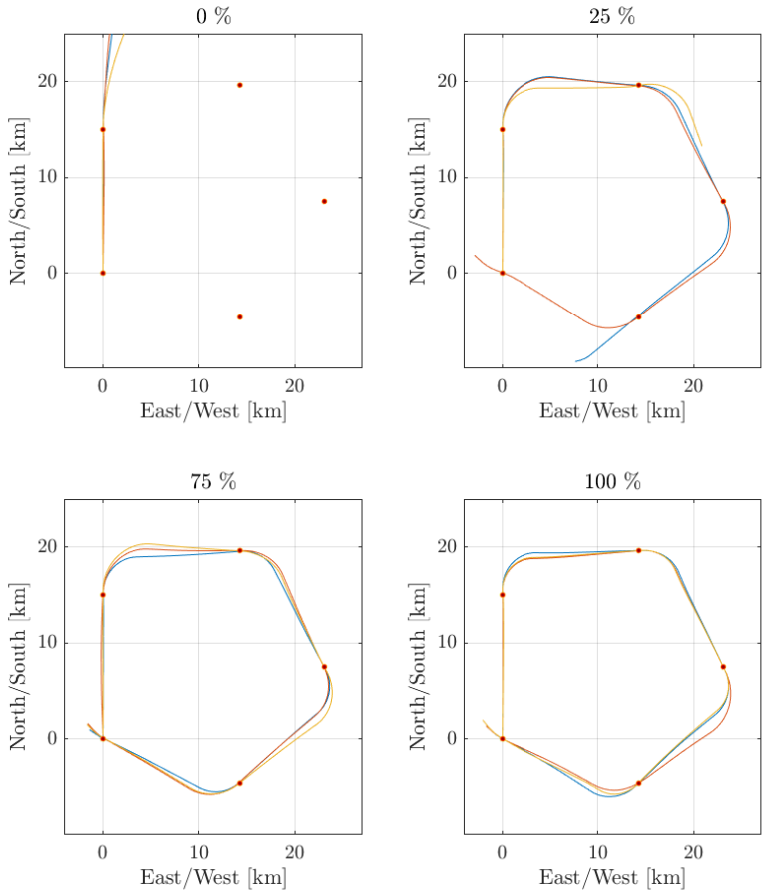


Figure 4.3: Learning process for Policy B: Plots showing the resulting trajectories for different amounts of data used in training. The red dots are spheres of radius $250\,m$, which indicates the error margin.

4.2 Vehicle classification using synthetic meta-learning

Meta-learning and FSL (see section 3.5) techniques are used to train models that are fast learners. These models are capable of learning complex, downstream tasks using only a handful of examples. However, meta-learning requires that the model is pre-trained using a large dataset. In the present case study, the VBS3 simulator was used to generate such a dataset, and several meta-learning algorithms were implemented and evaluated in the context of military vehicle classification tasks.

4.2.1 Data generation

The present case study used an in-house developed data generation tool capable of generating millions of automatically labeled vehicle images using the VBS3 simulator. VBS3 contains more than 2,000 vehicle models. However, many models are variants of the same real-world vehicle, differing only in coloring or weapons mounted, and are therefore grouped into a single class. Figure 4.4 represents an example image of a military truck generated using the tool.

Figure 4.4: Synthetic image of a military truck, including its bounding box and segmentation annotation.

4.2.2 Experimental setup

The evaluation was performed using four different meta-learning approaches: 1) model agnostic meta-learning (MAML); 2) meta-baseline (MB); 3) rethinking few-shot (RFS), and; 4) feature reconstruction networks (FRN). The first approach is optimization-based and the last three approaches are metric-based.

In this experiment, learning was performed using the synthetic dataset and evaluation was performed using a real-world dataset. The synthetic dataset consisted of 400,000 images covering 407 different vehicles. The evaluation dataset consisted of 40 different real-world military vehicle types with roughly 10-30 examples per class.

In an attempt to compare the synthetic meta-learning approach with the ideal scenario, where real-world data is abundant, a dataset of 10,000 images covering 64 different real-world vehicles was also used to represent an estimated upper bound baseline. Furthermore, a lower bound baseline was included in

the experiments using randomly initialized DNNs where no pre-training was performed.

4.2.3 Results

The results are presented in Table 4.2. The metric learning approaches failed to learn and were not much better than the randomly initialized DNNs. The only exception is FRN that outperformed even the upper baseline, but it was still significantly worse than MAML. This result is most likely due to FRNs non-standard method of pre-training, making it less likely to overfit to the training data.

Table 4.2: Accuracy for the four meta-learning approaches in % over three different 5-way few-shot settings. The *upper* and *synth* subscripts denote models pre-trained using the real-world and synthetic datasets, respectively. The *lower* subscript represents the randomly initialized model.

Models	**1-shot (%)**	**5-shot (%)**	**10-shot (%)**
$\text{MAML}_{\text{lower}}$	31.95	49.92	52.54
$\text{MAML}_{\text{synth}}$	46.41	67.83	72.92
$\text{MAML}_{\text{upper}}$	**62.97**	**72.88**	**73.70**
$\text{RFS}_{\text{lower}}$	29.40	36.32	40.10
$\text{RFS}_{\text{synth}}$	30.28	41.21	44.16
$\text{RFS}_{\text{upper}}$	37.67	52.01	55.87
MB_{lower}	29.31	36.73	40.80
MB_{synth}	29.54	39.14	44.01
MB_{upper}	37.59	51.55	55.13
$\text{FRN}_{\text{lower}}$	27.95	34.51	36.80
$\text{FRN}_{\text{synth}}$	36.31	52.70	58.67
$\text{FRN}_{\text{upper}}$	35.20	50.48	54.35

In the present case study MAML was the best approach. This result is surprising since in recent research, MAML has often been outperformed by the metric approaches. However, MAML has some advantages that may contribute to the results:

- The fine-tuning step of MAML offers more ways to compensate for the domain shift between synthetic and real data. MAML updates the entire DNN on the query set with 1-10 gradient updates, which neither of the metric approaches does. This allows the DNN to adjust its feature representation to the real-world data.
- MAML computes its batch normalization statistics over the given query batch. This is called transient learning and can make the DNN more robust to the domain shift.
- MAML uses a smaller DNN making it less prone to overfitting. The metric approaches uses larger DNNs based on the ResNet12 architecture, which seems to require more data to generalize.

The goal of the case study was to demonstrate that meta-learners can compensate for the lack of real-world data by pre-training on synthetic data. Unfortunately, the fidelity of the synthetic data seemed to be too low in this case, ultimately making the gap to the real-world data too large to bridge for most meta-learning approaches.

4.3 Face verification using synthetic data from generative models

The present case study examined the problem of building a face recognition system without having access to training images of real people. *Face recognition* is commonly divided into verification and identification. *Face verification* is the task of determining whether two given images contain the face of the same person (Figure 4.5). Hence, face verification systems can be used to determine whether a person is who he/she claims to be. *Face identification* is the task of determining whether a given facial image matches one of the identities in a database containing facial images of known persons (Figure 4.6). Hence, face identification systems can be used to find out the identity of a person.

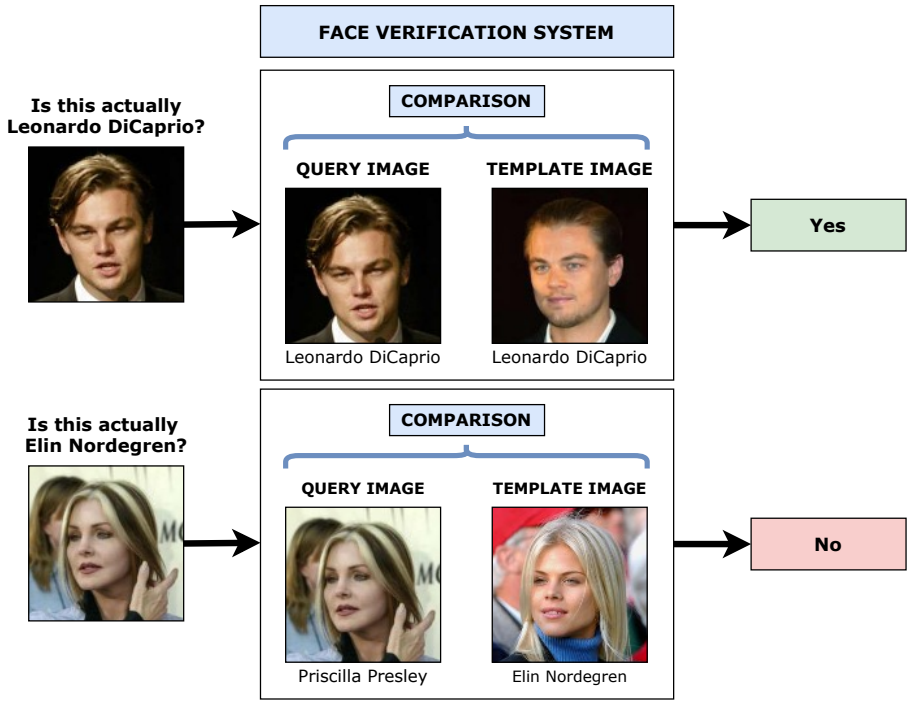


Figure 4.5: Face verification system. During verification, the query image is compared with an existing template image of the person whose identity is being claimed (one-to-one matching). All facial images have been obtained from [97].

In this study, focus was on face verification. One important aspect when it comes to face verification systems is the ability to handle new identities that were unknown at the time of system deployment. One does not want to recalibrate the whole system every time a new identity is introduced. For instance, a website might require newly registered users to verify their identity by providing two facial images: a live webcam image and a scanned passport image. This is an example of *one-shot learning*, since a model must be able to determine whether the person in the webcam image is who he/she claims to be based only on *one* single template image (the passport).

DL-based methods have been shown to perform well when applied to face verification tasks. For instance, the FaceNet [82] model achieves over 99% accuracy when evaluated on the Labeled Faces in the Wild (LFW) [97] benchmark dataset. Given a facial image x as input, the model outputs a face embedding vector $f(x)$. Two different vectors can be compared to obtain a distance metric, which indicates the similarity of the two corresponding images. A short

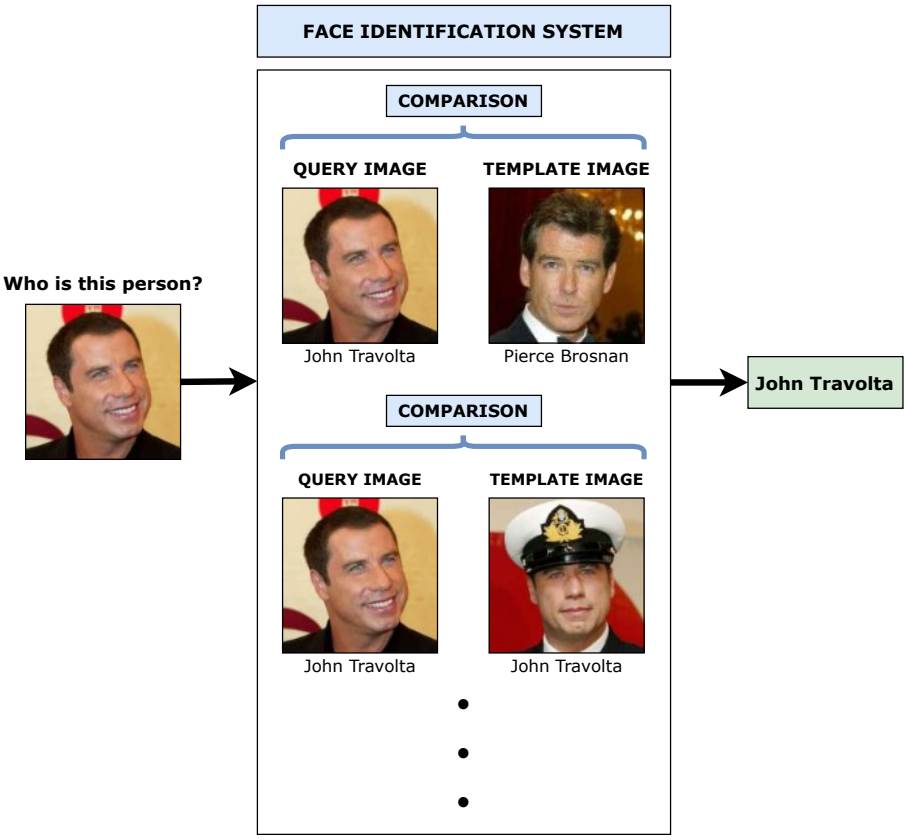


Figure 4.6: Face identification system. A person is identified by comparing the query image with all template images in a database (one-to-many matching). All facial images have been obtained from [97].

distance (i.e., similar vectors) indicates that the images contain the face of the same person. The ability to produce high-quality face embeddings for previously unseen persons is learned by minimizing a *triplet loss function* during training of the model. The word *triplet* comes from the use of an anchor image x^a (an identity), a positive image x^p (same identity but a different image), and a negative image x^n (another identity). The loss function encourages the model to minimize the distance $\|f(x^a) - f(x^p)\|_2^2$ between images of the same identity (anchor and positive), and maximize the distance $\|f(x^a) - f(x^n)\|_2^2$ between images of different identities (anchor and negative):

$$\mathcal{L} = \max(\|f(x^a) - f(x^p)\|_2^2 - \|f(x^a) - f(x^n)\|_2^2 + \alpha, 0),$$

where α is a margin enforced between positive and negative pairs [82].

4.3.1 Concept

The purpose of this study was to investigate to what extent it is possible to perform face verification on real images of faces after training a face verification system on synthetic images of faces generated by generative adversarial networks (GANs). Using synthetic training data is not only interesting given the focus of this report, but could also help overcome privacy and data protection issues related to the use of datasets containing images of real people. A few years ago the face recognition dataset MS-Celeb-1M [98], which contained approximately 10 million facial images of 100,000 different persons, was deleted.

The reason for doing this is unclear, but it is not unlikely that data protection laws such as GDPR influenced the decision. Nevertheless, there is no guarantee that large datasets of real faces will continue to be publicly available, and permitted to use, in the future.

GAN models such as StyleGAN2 [30] can be used to generate synthetic images of faces (i.e., facial images of non-existing persons) that are perceived as authentic by humans. StyleGAN2 is able to produce an unlimited number of faces after being trained on a relatively small dataset of real faces. This could potentially help reduce the need for large amounts of real data in many applications. However, at least two different images of each unique person are required in order to train a face verification system. This is problematic since the GAN does not provide full user control over the output. Even small changes in input might result in a completely different face.

With all this in mind, experiments were performed for the present study, attempting to 1) generate multiple facial images of the same identity; 2) train a face verification system on such images; and, 3) evaluate its performance on facial images of real persons.

4.3.2 Military applications

Face recognition technology can be applied to various fields, such as surveillance, security, and biometric authentication. There are both commercial and military use cases, ranging from facility and airport security systems to drone surveillance systems and websites requiring users to verify their identity. For instance, a face recognition system could be integrated with the door access control system to grant authorized personnel access to restricted facilities, or even be used on the battlefield to determine whether an individual is part of the own military personnel or the enemy.

4.3.3 Experimental setup

The description of the experimental setup used in this case study can be divided into: 1) the process behind generating synthetic training images; 2) how FaceNet models were trained on the images; and, 3) the evaluation protocol used to measure the performance of the models when tested on real facial images.

Generating synthetic training data

StyleGAN2 [30] currently yields state-of-the-art results for image synthesis. Therefore, it was decided to use a pre-trained StyleGAN2 model to build a dataset of synthetic facial images. However, it is not trivial to generate two different images of the same person when using GAN models out of the box, and only changing details such as the hair or freckles is not sufficient. During training of the face verification model, the anchor image and the positive image should ideally look quite different, as long as one can make sure that they contain the same identity. Otherwise, the model cannot be expected to perform well when tested on real images, since two images might have been captured at different locations with different camera angles, showing different poses and facial expressions of the same person.

Luckily, Härkönen et al. [99] have demonstrated methods to augment pre-trained GANs with control variables. For StyleGAN2, they used principal component analysis (PCA) to find principal components (principal directions) in the intermediate latent space. Loosely speaking, a component could be seen as corresponding to a specific image attribute when restricted to certain

StyleGAN2 layers. For instance, the authors managed to rotate the head of a person by moving along the second component only at the first three layers. This type of edit can be done for various attributes without them interfering with each other (although some attributes are still entangled to some degree). In other words, it is possible to edit specific facial attributes of an identity without accidentally changing the entire identity in the process.

The authors provided an interactive tool for exploring principal directions and changing attributes. In this case study changes were made to their code to automatically generate two images of the same person. Specifically, functionality was added to generate an image of a person as well as a new version of the same image, but with randomly edited attributes (e.g., hair color, hair length, head rotation, facial expression, background and light direction). This made it possible to automatically build a synthetic training dataset containing 12.5M identities (i.e., 25M facial images). Some example facial images are shown in Figure 4.7.

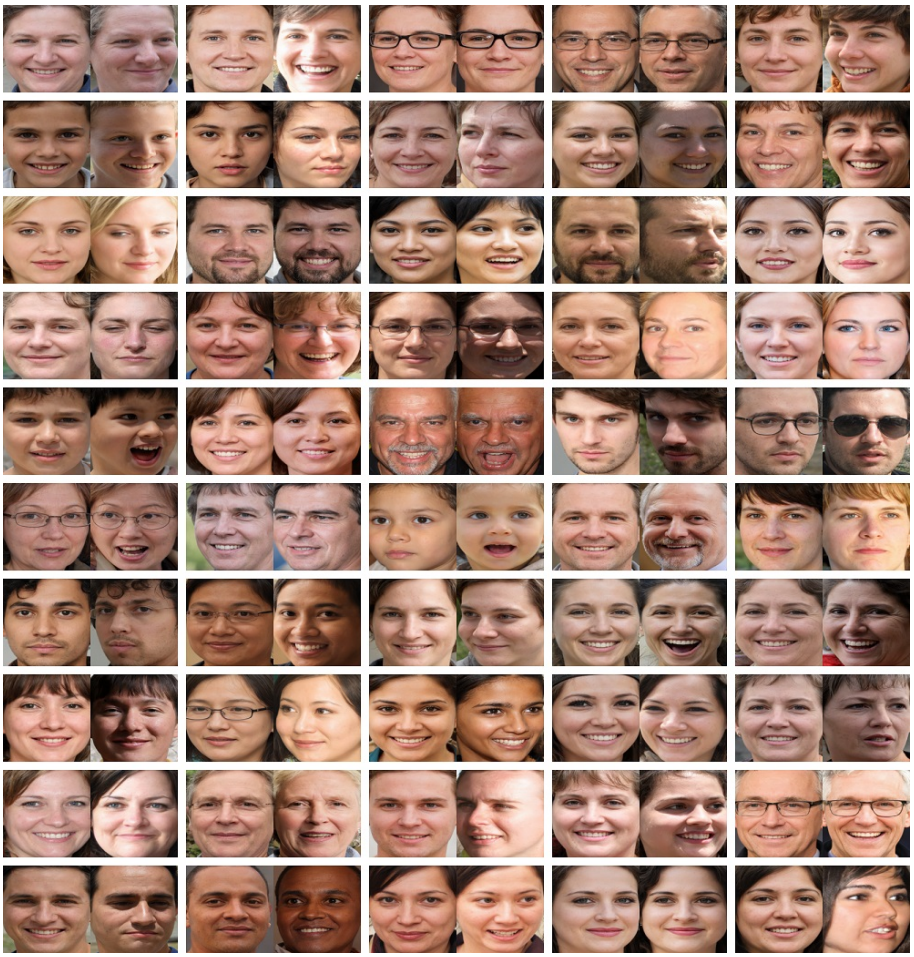


Figure 4.7: Synthetic images randomly selected from our training dataset. For each image pair, the face that was originally generated is shown to the left while the edited face is shown to the right. Note that all images have been cropped around the face.

Training FaceNet

Three separate FaceNet models were trained with the same architecture[1] and randomly initialized parameters, using triplets from the generated training dataset:

- The first model was trained with randomly selected triplets, i.e., for each epoch and anchor-positive pair a negative image was randomly selected. All images were cropped around the face before being fed as input to the model.
- The second model was trained in the same way as the first model. However, this time each input image was augmented on the fly using a random combination of the post-processing techniques presented in Figure 4.8. Specifically, for every image, each post-processing technique was applied with 30% probability, in random order, using a randomly chosen intensity. Data augmentation is commonly used to expand datasets with altered samples, which helps prevent overfitting, especially when there is limited training data. It is also well known that DL-based computer vision models tend to be susceptible to image distortions [100], which is important to keep in mind when a model trained on synthetic data is intended to be used on real images that might have been distorted during acquisition, transmission, or storage. Therefore, although it was possible to generate a large number of synthetic images, they were still augmented during training in an attempt to improve model robustness and generalizability.
- The third FaceNet model was trained in the same way as the second one. However, this time a simple technique was also used to select more ideal negative images. Focusing on difficult triplets during training has been shown to be important in order to increase the performance of the model [82]. Therefore, for each training epoch and anchor-positive pair, 1,000 randomly selected negative images were compared to the anchor image. Specifically, the negative image x_i^n with the smallest distance $\|f(x^a) - f(x_i^n)\|_2^2$ was chosen as input (the distance can be viewed as a dissimilarity metric, which in this case should be minimized to select similar faces that represent a difficult case). It was also required that $\|f(x^a) - f(x^p)\|_2^2 < \|f(x^a) - f(x_i^n)\|_2^2$, since the model indeed became stuck in a local minimum without this requirement [82].

In all three experiments, a validation loss was computed on unseen synthetic images after each training epoch. The model state with the lowest validation loss was saved as the final state of the model (i.e., a total of three models were saved since one model was trained in each experiment). In every experiment, the model with the lowest validation loss computed on the CASIA-WebFace [101] dataset, which contains 10,575 real identities, was also saved. Note that all models were trained exclusively on synthetic images, but in this case real images were used to choose which models should be used later during evaluation.

Measuring performance

The trained FaceNet models were evaluated on real faces from the Labeled Faces in the Wild (LFW) [97] database, which is commonly used for benchmarking face verification in unconstrained settings. The standard evaluation protocol [97] was followed and accuracy was measured using 6,000 testing image pairs, i.e., 3,000 matched pairs (same identity) and 3,000 mismatched pairs (different identities). Accuracy here refers to the fraction of image pairs that

[1] https://github.com/timesler/facenet-pytorch

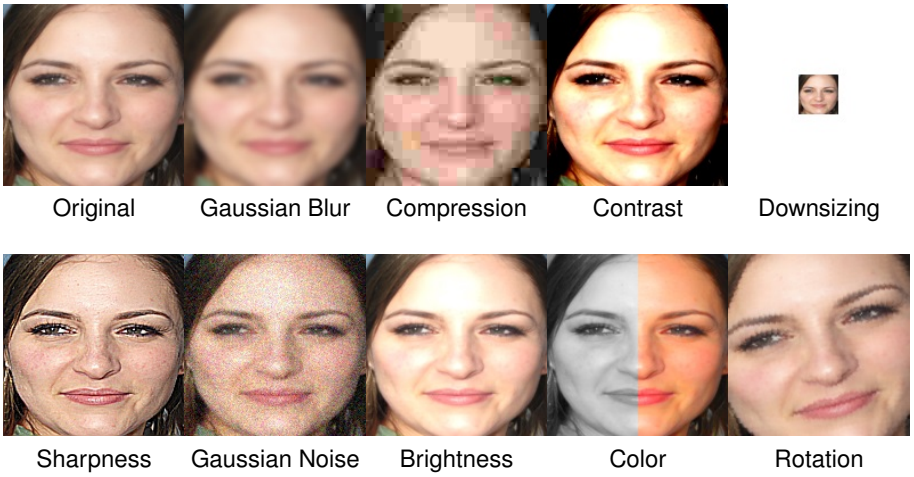


Figure 4.8: GAN-generated facial image post-processed using the highest intensities considered in this case study. Since it is possible to decrease and increase the image color, both alternatives are shown in the same image (bottom row fourth column).

was correctly classified by FaceNet. In other words, for every image pair, FaceNet predicted whether or not both images contained the face of the same person. As mentioned, when FaceNet outputs an embedding vector for each image in a pair, face similarity corresponds to the distance between the two vectors. It is therefore necessary to specify at what distance two images should be considered a matched pair. Since a small distance indicates similar faces, it is suitable to set an upper threshold distance value that should not be reached.

The LFW testing images were divided in 10 subsets of equal size. Following the standard protocol, 10 separate experiments were performed in a leave-one-out cross validation scheme. In each experiment, 9 subsets were used to select the distance threshold rendering the highest accuracy on those subsets, while the remaining (left out) subset was used for measuring the final testing accuracy. Note that a different subset was left out in every experiment, i.e., each subset was left out exactly one time. Hence, the final performance was reported as the mean accuracy computed across all 10 left out subsets.

4.3.4 Results

The face verification performance of FaceNet on LFW is presented in Table 4.3. As mentioned, two FaceNet models were evaluated in each experiment: one selected using synthetic validation images from StyleGAN2, and another one selected using real validation images from CASIA-WebFace. Recall that all FaceNet models were trained exclusively on synthetic images.

As shown in Table 4.3, FaceNet did not achieve satisfying performance when trained directly on synthetic images. The mean accuracy on real testing images from LFW was below 60%, which can be compared to FaceNet models that achieve over 99% accuracy when trained on real images. An untrained FaceNet model with randomly initialized parameters would yield approximately 50% accuracy. However, when trained on randomly post-processed synthetic images, FaceNet achieved around 73% accuracy. This is a significant improvement, although far from perfect. Finally, accuracy did not increase significantly when using a simple technique to select more difficult triplets during training. In all experiments, the performance only improved marginally when using real images

Table 4.3: FaceNet performance on real images from the LFW database, reported as the mean accuracy obtained when following the standard LFW evaluation protocol for face verification. Accuracy is presented for three experiments: *Default* (FaceNet trained directly on StyleGAN2 images), *Post-Processing* (FaceNet trained on post-processed StyleGAN2 images), and *Triplet Selection* (FaceNet trained on post-processed StyleGAN2 images when using a technique to select more difficult triplets).

Experimental Setup / *Validation Images*	Default	Post-Processing	Triplet Selection
StyleGAN2	$57.67\% \pm 0.85$	$\mathbf{72.62}\% \pm 0.74$	$72.28\% \pm 0.66$
CASIA-WebFace	$58.28\% \pm 0.75$	$73.02\% \pm 0.70$	$\mathbf{73.38}\% \pm 0.84$

from CASIA-WebFace for model selection.

The experiments performed in this study indicate that face verification models trained exclusively on synthetic data tend to lack sufficient generalization capability when applied to real data, at least when using FaceNet and the experimental setup specific to this study. The models turned out not to be reliable enough. However, the results are still rather promising since there are some experimental details that could help improve performance further. As mentioned, when selecting more difficult triplets, each anchor image was compared with 1,000 randomly selected negative images *after* computing the embedding vectors. The original authors of FaceNet, on the other hand, performed the comparison using almost twice as many images [82]. In other words, the triplet selection approach in the present study could probably be improved, but limited computational resources did not allow for this. One should also keep in mind that the LFW validation dataset is commonly used for model selection, which might help improve performance on the corresponding testing dataset. It was decided to use the CASIA-WebFace validation dataset since the LFW validation and testing sets overlap, i.e., some of the images occur both in the validation and testing set, which might lead to biased results.

5 Conclusions

Acquiring sufficient training data for DL is a major challenge when solving complex tasks. These tasks require large DNNs and, ultimately, large datasets to learn how to generalize to real-world inputs. Fortunately, many techniques have been proposed to enable learning even if the available training dataset is limited.

This report's specific focus is on the use of synthetic data created using simulators, generative models or a combination of the two (e.g., generative models that are used to enhance data created by simulators). The report presents several techniques that can be used to integrate the synthetic data into the DL training pipeline. For instance, simply augmenting the real-world training dataset with synthetic data often has a positive, but limited, impact on overall performance. For more challenging cases where, for instance, the majority of training data is synthetic it is possible to use transfer learning techniques such as multi-task learning, meta-learning and domain adaptation.

Empirical results from case studies where DNNs have been trained using synthetic data and transfer learning are presented in the report. In the first case it was shown that a model trained to navigate between waypoints using a JAS-39 simulator can be reused using fine-tuning to more efficiently, with 25% less training data, learn to perform the same task in a F22 simulator where the aircraft is governed by a different flight dynamics model. The second case shows that meta-learning can be used to train a model, which learns how to learn, given a large synthetic dataset and a relatively small real-world dataset. Results show that the model can learn a 5-way vehicle classification task at an accuracy of approximately 73% using only 5 real-world observations of each vehicle. Although these results are promising, there is a significant loss in performance when compared to models trained using only real-world data. In the face verification case, where generative models are used to create highly realistic data that even humans struggle to recognize as synthetic, the performance is approximately 73% compared to 99% when trained using only real-world data.

In this report, it is concluded that the synthetic data approach is promising for military DL applications where training data is limited and difficult to acquire. However, more research is needed to better bridge the fidelity gap between synthetic and real-world data to, ultimately, improve performance.

Bibliography

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Linus J. Luotsinen, Daniel Oskarsson, Peter Svenmarck, and Ulrika Wickenberg Bolin. Explainable artificial intelligence: Exploring XAI techniques in military deep learning applications. Technical Report FOI-R--4849--SE, Swedish Defence Research Agency (FOI), 2019.

[3] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. In *Journal of Big Data*, volume 6, 2019.

[4] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[6] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised Pretraining of Visual Features in the Wild. *arXiv e-prints*, page arXiv:2103.01988, March 2021.

[7] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 12449–12460, 2020.

[8] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[10] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 1329–1338, 2016.

[11] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Florian Golemo, Melissa Mozifian, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, C. Karen Liu, Jan Peters, Shuran Song, Peter Welinder, and Martha White. Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop. *arXiv e-prints*, page arXiv:2012.03806, December 2020.

[12] Rika Antonova, Silvia Cruciani, Christian Smith, and Danica Kragic. Reinforcement Learning for Pivoting Task. *arXiv e-prints*, page arXiv:1703.00472, March 2017.

[13] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018.

[14] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a robot hand. *arXiv e-prints*, page arXiv:1910.07113, October 2019.

[15] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep Drone Racing: From Simulation to Reality with Domain Randomization. *arXiv e-prints*, page arXiv:1905.09727, May 2019.

[16] Tomás Hodan, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N. Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 66–70, 2019.

[17] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015.

[18] Weichao Qiu and Alan L. Yuille. UnrealCV: Connecting computer vision to unreal engine. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 9915 of *Lecture Notes in Computer Science*, pages 909–916, 2016.

[19] Yue Yao, Liang Zheng, Xiaodong Yang, Milind Naphade, and Tom Gedeon. Simulating content consistent vehicle datasets with attribute descent. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[20] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 7249–7255, 2019.

[22] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-Sim: Learning To Generate Synthetic Datasets. *arXiv e-prints*, page arXiv:1904.11621, April 2019.

[23] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-Sim2: Unsupervised learning of scene structure for synthetic data generation. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, volume 12362 of *Lecture Notes in Computer Science*, pages 715–733. Springer, 2020.

[24] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[25] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake It Till You Make It: Face analysis in the wild using synthetic data alone. *arXiv e-prints*, page arXiv:2109.15102, September 2021.

[26] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[28] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, Hyrum Anderson, Heather Roff, Gregory C. Allen, Jacob Steinhardt, Carrick Flynn, Seán Ó hÉigeartaigh, Simon Beard, Haydn Belfield, Sebastian Farquhar, Clare Lyle, Rebecca Crootof, Owain Evans, Michael Page, Joanna Bryson, Roman Yampolskiy, and Dario Amodei. The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation. *arXiv e-prints*, page arXiv:1802.07228, February 2018.

[29] Fredrik Johansson, Andreas Horndahl, Harald Stiff, and Marianela García Lozano. Data synthesis using generative models. Technical Report FOI-R--5041--SE, Swedish Defence Research Agency (FOI), 2020.

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[33] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1316–1324, 2018.

[34] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Xiao-Yuan Jing, Fei Wu, and Bingkun Bao. DF-GAN: Deep Fusion Generative Adversarial Networks for Text-to-Image Synthesis. *arXiv e-prints*, page arXiv:2008.05865, August 2020.

[35] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face Aging With Conditional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1702.01983, February 2017.

[36] Edward Smith and David Meger. Improved Adversarial Systems for 3D Object Generation and Reconstruction. *arXiv e-prints*, page arXiv:1707.09557, July 2017.

[37] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv e-prints*, page arXiv:1609.04802, September 2016.

[38] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7183–7192, 2019.

[39] Stephan R. Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing Photorealism Enhancement. *arXiv e-prints*, page arXiv:2105.04619, May 2021.

[40] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[41] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The Mapillary Vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5000–5009, 2017.

[42] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2010.

[43] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1), 2016.

[44] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 270–279, 2018.

[45] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. *arXiv e-prints*, page arXiv:1611.02770, November 2016.

[46] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, volume 27, 2014.

[47] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. SpotTune: Transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[48] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[49] Wouter M. Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv e-prints*, page arXiv:1812.11806, December 2018.

[50] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5), 2020.

[51] Kihyuk Sohn, Sifei Liu, Guangyu Zhong, Xiang Yu, Ming-Hsuan Yang, and Manmohan Chandraker. Unsupervised domain adaptation for face recognition in unlabeled videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[52] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. VisDA: The Visual Domain Adaptation Challenge. *arXiv e-prints*, page arXiv:1710.06924, October 2017.

[53] Jaeyoon Yoo, Yongjun Hong, YungKyun Noh, and Sungroh Yoon. Domain Adaptation Using Adversarial Learning for Autonomous Navigation. *arXiv e-prints*, page arXiv:1712.03742, December 2017.

[54] Yuhua Chen, Wen Li, and Luc Van Gool. ROAD: Reality oriented adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7892–7901, 2018.

[55] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[56] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12627–12637, 2019.

[57] Ajay Kumar Tanwani. Domain invariant representation learning for sim-to-real transfer. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[58] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, volume 37, pages 1180–1189, 2015.

[59] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, volume 29, 2016.

[60] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 597–613, 2016.

[61] Minghao Chen, Shuai Zhao, Haifeng Liu, and Deng Cai. Adversarial-learned loss for domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3521–3528, 2020.

[62] Baochen Sun and Kate Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 443–450, 2016.

[63] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(4):801–814, 2019.

[64] Richard A. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning (ICML)*, pages 41–48. Morgan Kaufmann, 1993.

[65] Richard A. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[66] Steven C. Suddarth and Yannick L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. *Lecture Notes in Computer Science*, 412, 1990.

[67] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv e-prints*, page arXiv:1706.05098, June 2017.

[68] Lukas Liebel and Marco Körner. Auxiliary Tasks in Multi-task Learning. *arXiv e-prints*, page arXiv:1805.06334, May 2018.

[69] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018.

[70] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning to steer by mimicking features from heterogeneous auxiliary networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 8433–8440, February 2019.

[71] Jonny Sparrenhök. Influential learning: Knowledge sharing between artificial neural networks for autonomous vehicles. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), April 2021.

[72] Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[73] Mike Huisman, Jan N. van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 2021.

[74] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 48, pages 1842–1850, 2016.

[75] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. *arXiv e-prints*, page arXiv:1410.5401, October 2014.

[76] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70, pages 1126–1135, 2017.

[77] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv e-prints*, page arXiv:1803.02999, March 2018.

[78] Antreas Antoniou, Harrison Edwards, and A. Storkey. How to train your MAML. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[79] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[80] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proceedings of the International Conference on Machine Learning (ICML), Deep learning workshop*, 2015.

[81] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742, 2006.

[82] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[83] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[84] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A New Meta-Baseline for Few-Shot Learning. *arXiv e-prints*, page arXiv:2003.04390, March 2020.

[85] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5543–5551, 2017.

[86] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, pages 124–140. Springer, 2020.

[87] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain Generalization with MixStyle. *arXiv e-prints*, page arXiv:2104.02008, April 2021.

[88] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain Generalization by Seeking Flat Minima. *arXiv e-prints*, page arXiv:2102.08604, February 2021.

[89] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[90] Farzad Kamrani, Mika Cohen, Fredrik Bissmarck, and Peter Hammar. Behaviour modelling with imitation learning. Technical Report FOI-R--4890--SE, Swedish Defence Research Agency (FOI), 2019.

[91] Bernard Etkin and Lloyd D Reid. *Dynamics of Flight*, volume 2. Wiley New York, 1959.

[92] Viktor Sandström. On the efficiency of transfer learning in a fighter pilot behavior modelling context. Master's thesis, KTH, School of Engineering Sciences (SCI), 2021.

[93] Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPS)*, pages 305—313. MIT Press, 1988.

[94] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[95] Paulo Flores. Euler angles, bryant angles and euler parameters. In *Concepts and formulations for spatial multibody dynamics*, pages 15–22. Springer, 2015.

[96] Andreas Horndahl, Daniel Oskarsson, and Linus Luotsinen. Machine learning in FLSC. Technical Report FOI-D--0954--SE, Swedish Defence Research Agency (FOI), 2019.

[97] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[98] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. *arXiv e-prints*, page arXiv:1607.08221, July 2016.

[99] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. GANSpace: Discovering interpretable GAN controls. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9841–9850, 2020.

[100] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *Proceedings of the Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2016.

[101] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning Face Representation from Scratch. *arXiv e-prints*, page arXiv:1411.7923, November 2014.

FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se