Applied Deep Learning
Generative Adversarial Networks

Mark Schutera

Karlsruhe Institute of Technology | ZF Friedrichshafen AG | Hochschule Ravensburg-Weingarten

Course overview

1. Deep Learning Foundations

3. Transfer Learning and Object Detection

5. Segmentation Networks

8. Deep Reinforcement Learning

10. Generative Adversarial Networks

12. Recurrent Neural Networks

Course overview

Project proposal
One page on introduction, methods, dataset
Deadline 3. Lecture

Intermediate presentation
Ten minutes on achievements, problems, next steps
Due 7. Lecture

Final presentation
Screencast (Slides and Audio)
Due 13. Lecture (20.01.2020)

Final documentation
Documentation and code as Jupyter Notebook
Deadline 13. Lecture (20.01.2020)

Course features

Sli.do

Every question matters.

Get the app.

Ask questions (with slide number) or vote on other students' questions during the lecture.

And give direct feedback.

#TOBEDETERMINED

Questions will be covered immediately or in the next lecture in more depth.

Github

Find slides, tutorials, flashcards and references on Github.

https://github.com/schutera/DeepLearningLecture_Schutera

You found typos, additional material such as links, algorithms, papers, literature or want to contribute to the slides and lecture notes..

..Feel free to contribute, e-mail me.

Course features

Sli.do

Every question matters.
Get the app.
Ask questions (with slide number) or vote on other students' questions during the lecture.

#TOBEDETERMINED

Questions will be covered directly or in the next lecture in more depth.

Github

Typos, additional material such as links, algorithms, paper, literature, lecture notes..

..Feel free to contribute.

Grade Bonus .3
Prepare flashcards based on Ian Goodfellow's Deep Learning Book

- Commit to flashcard set by emailing me, first come first serve
- Must be comprehensive

This lecture in one slide

Introduction to Generative Adversarial Neural Networks

Adversarial learning a problem statement

Generative adversarial learning framework

Generative adversarial learning with neural networks

Introduction – Generative Adversarial Neural Networks

Two neural networks contest with each other in a game.
This is similar to an actor-critic setup.

Two neural networks contest with each other in a game.

Given a training set, this technique learns to **generate new data** with the same distribution as the training set.

How would you counterfeit money?

Introduction – Generative Adversarial Neural Networks

How would you counterfeit money?

Forge some of your self-made coins

Introduction – Generative Adversarial Neural Networks

How would you counterfeit money?

Forge some of your self-made coins

Try to deposit your coins at a bank

Introduction – Generative Adversarial Neural Networks

How would you counterfeit money?

Forge some of your self-made coins
Try to deposit your coins at a bank

Get rejected and get feedback why the bank did not take your money

Introduction – Generative Adversarial Neural Networks

How would you counterfeit money?

Forge some of your self-made coins
Try to deposit your coins at a bank
Get rejected and get feedback why the bank did not take your money

Improve your self-made coins accordingly

Introduction – Generative Adversarial Neural Networks

How would you counterfeit money?

Forge some of your self-made coins
Try to deposit your coins at a bank
Get rejected and get feedback why the bank did not take your money
Improve your self-made coins accordingly

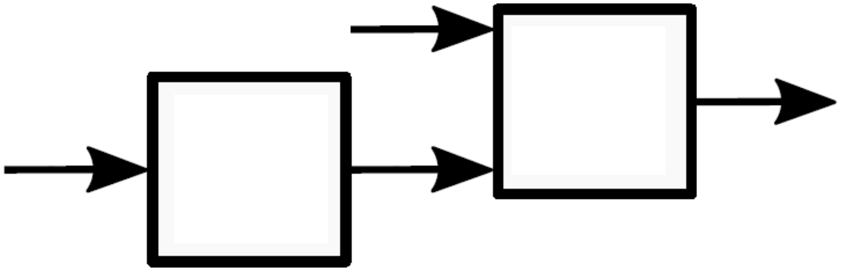
Repeat until the bank takes your coins

This process enables us to **model the distribution** of a given data set.

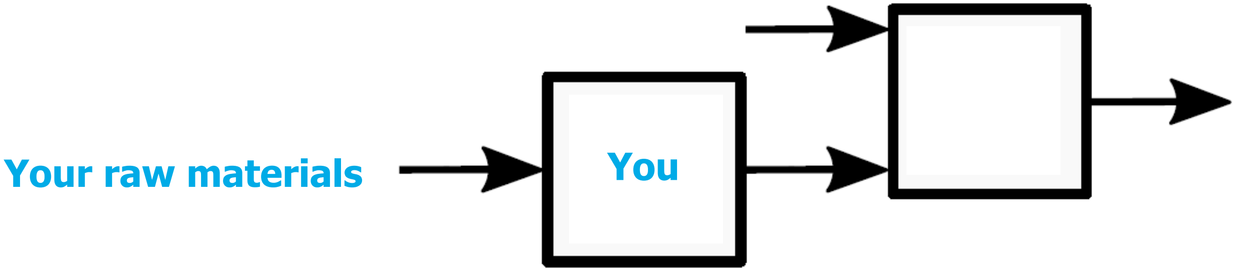
And to **transfer an input** according to the distribution of a given data set.

Framework – Generative Adversarial Neural Networks

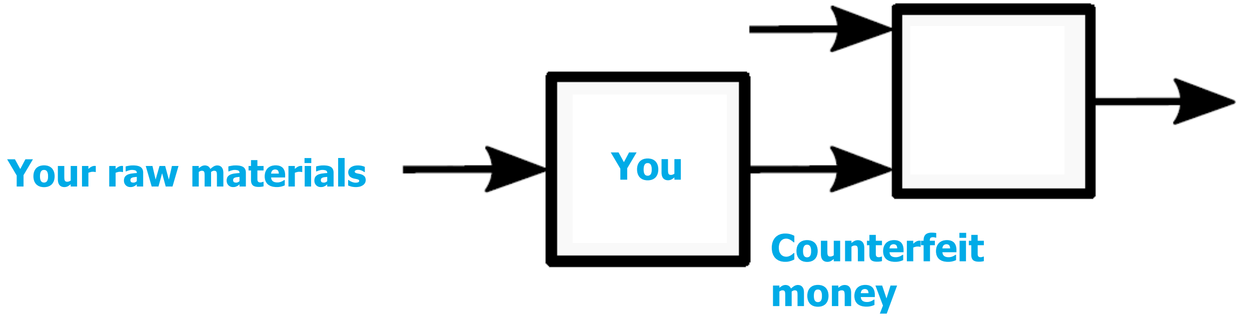
Your raw materials



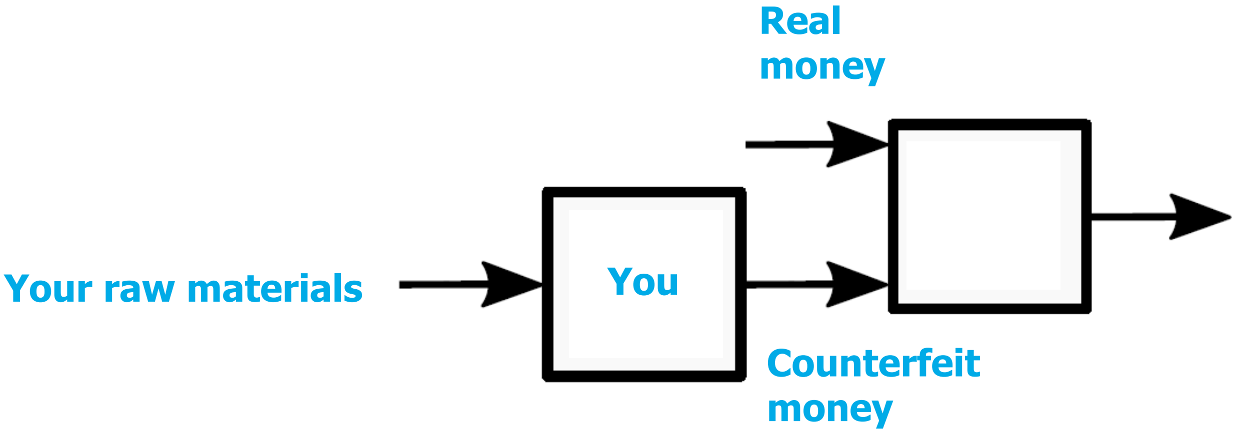
Framework – Generative Adversarial Neural Networks



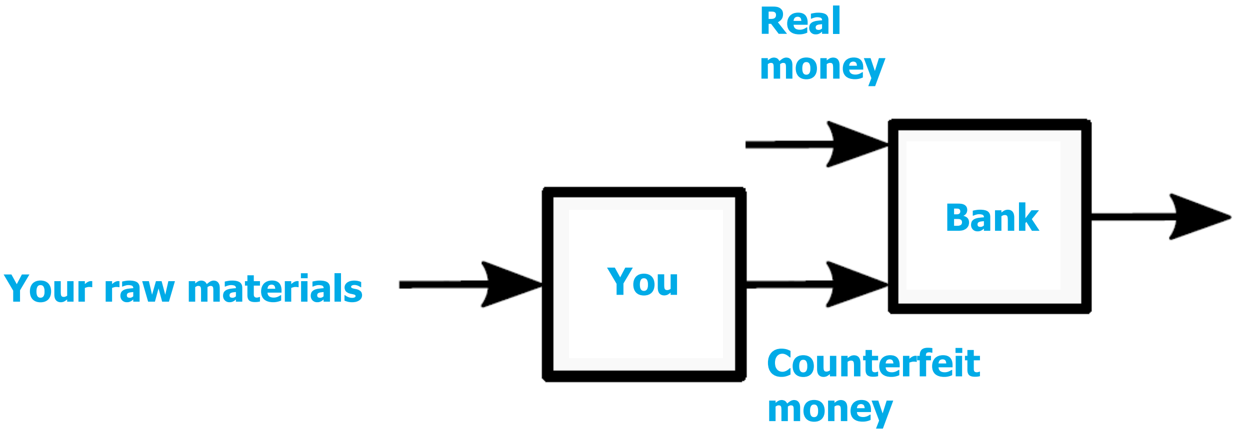
Framework – Generative Adversarial Neural Networks



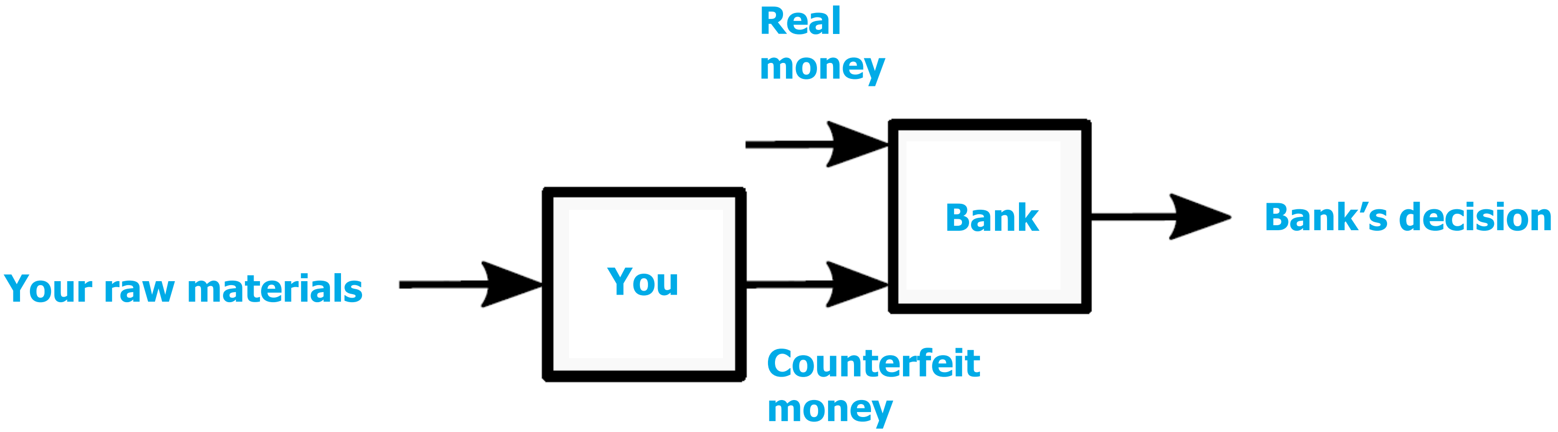
Framework – Generative Adversarial Neural Networks



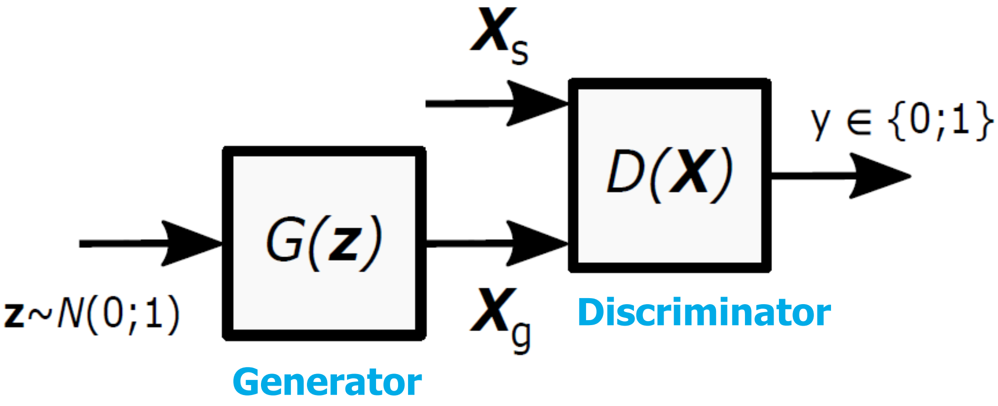
Framework – Generative Adversarial Neural Networks



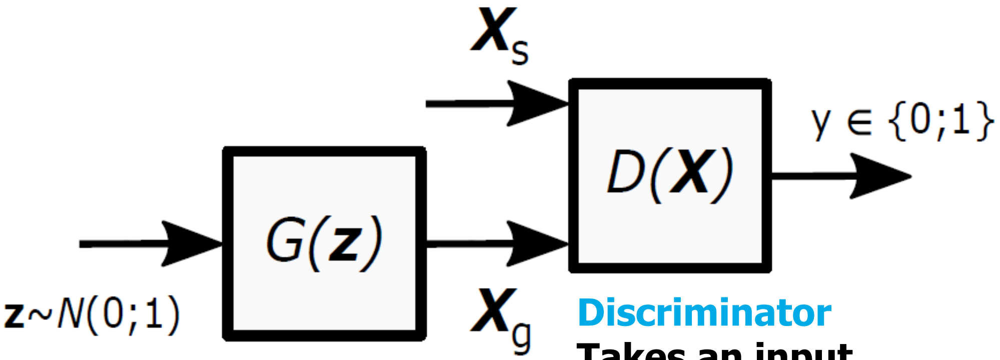
Framework – Generative Adversarial Neural Networks



Framework – Generative Adversarial Neural Networks



Framework – Generative Adversarial Neural Networks



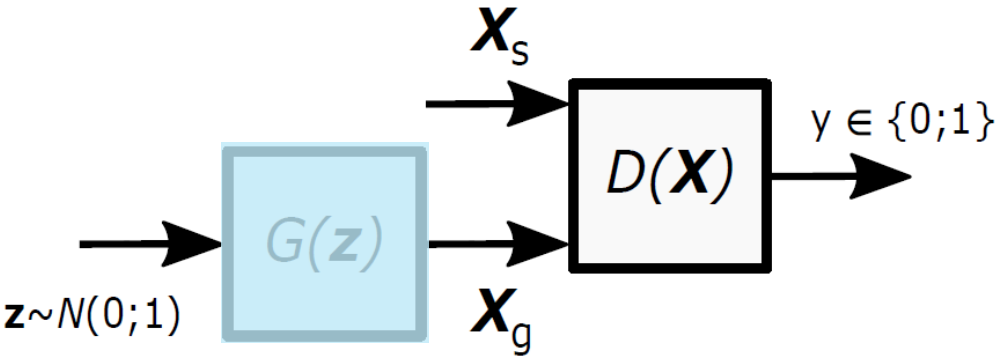
Generator
Takes random input and generates a sample

Discriminator
Takes an input sample and predicts the domain

Training – Generative Adversarial Neural Networks

PASS 1

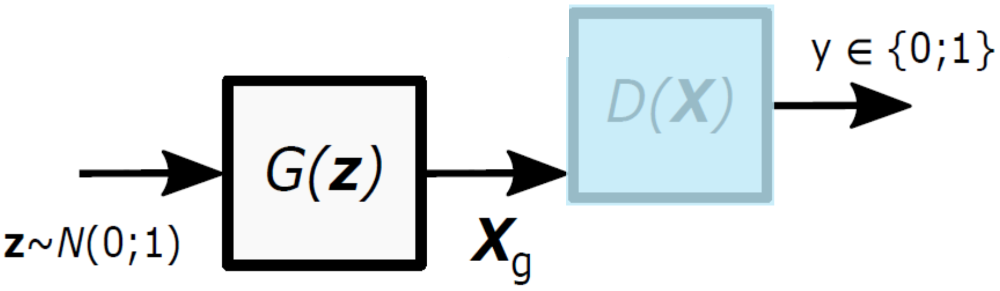
Freeze Generator
Train Discriminator



Training – Generative Adversarial Neural Networks

PASS 2

Freeze Discriminator
Train Generator



This lecture in one slide

Introduction to Generative Adversarial Neural Networks

Generative adversarial learning with neural networks

Generator

Discriminator

Objective Function or Min-Max-Game

Overview State of the Art GANs

Deep Dive MNIST GAN

Why Generative Adversarial Networks do not work yet?

Datasets and Benchmarking

Generator

Task is mapping into the source domain

Generator

Task is mapping into the source domain

Starting from a **latent feature space**
Such as a random variable

$$\mathbf{z} \sim N(0;1)$$

Generative Adversarial Learning with Neural Networks

Generator
Task is mapping into the source domain

Starting from a latent feature space
Such as a random variable

Starting from a **shared feature space**
Such as a shared encoding

z

Generative Adversarial Learning with Neural Networks

Generator

Task is mapping into the source domain

Starting from a latent feature space

Such as a random variable

Starting from a shared feature space

Such as a shared encoding

Starting from a **simliar feature space** as the source domain

Such as images recorded on different domains

$\boldsymbol{X}_S$

Discriminator

Task is to distinguish between generated and real samples

Generative Adversarial Learning with Neural Networks

Discriminator

Task is to distinguish between generated and real samples

Usually there are additional losses, supporting and extending the Feedback of the Discriminator to the Generator

(see in more detail within the state of the art slides)

Objective Function or the Min-Max-Game

Minimizing the loss inflicted by the generator,

While maximizing the loss of the discriminator when dealing with real data

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right].$$

For a real sample

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right].$$

For a real sample the Discriminator gets penalized in case it predicts a generated sample.

Learning how real samples look like

For a real sample

For a generated sample the Discriminator gets penalized in case it predicts a real sample.

Learning how generated samples look like

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right].$$

The Generator Objective Function

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right].$$

For a generated sample which is correctly predicted as generated by the Discriminator the Generator gets penalized.

Learning how to generate samples that would be predicted to be real by the generator.

Success over time

Generative Adversarial Nets (Goodfellow et al.)	2014
Cycle-Consistent Adversarial Domain Adaptation (Hoffman et al.)	2017
Unsupervised Image-to-Image Translation Networks (Liu et al.)	2018

Goodfellow‘s Original GAN

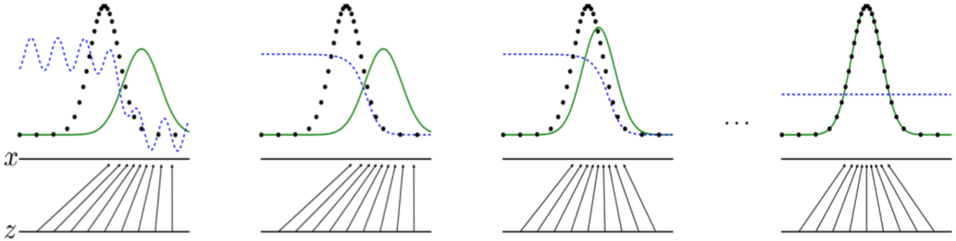
First implementation of a discriminative loss in order to move the distribution of generated samples into a source domain distribution.

Architecture is Fully Connected.

Shown on MNIST, CIFAR-10, and the TFD data sets

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7



(a) (b) (c) (d)

https://arxiv.org/pdf/1406.2661.pdf

Cycle Consistency

Cycle and Semantic consistency for domain adaptation.

Introducing additional losses during generation such as reconstruction (cycle consistency) and auxiliary task losses (semantic consistency).

Consistencies minimize label flipping

CyCADA: Cycle-Consistent Adversarial Domain Adaptation

Judy Hoffman [1] Eric Tzeng [1] Taesung Park [1] Jun-Yan Zhu [1] Phillip Isola [1 2] Kate Saenko [3] Alexei A. Efros [1] Trevor Darrell [1]

(a) GTA5 (b) GTA5 → Cityscapes (c) CityScapes (d) CityScapes → GTA5

https://people.eecs.berkeley.edu/~jhoffman/papers/2018_cycada.pdf

Generative Adversarial Learning - Overview State of the Art

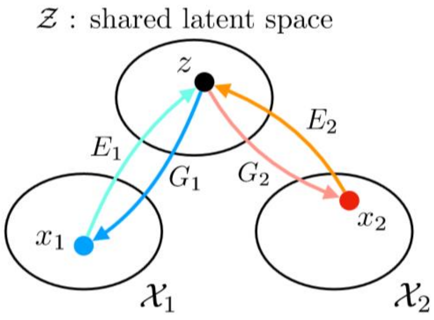
UNIT

Learning a joint distribution of images in different domains by mapping to a shared latent feature space.

Based on Coupled GANs in the from of Virtual Auto-Encoders and additional Losses such as Reconstruction Loss.

Unsupervised Image-to-Image Translation Networks

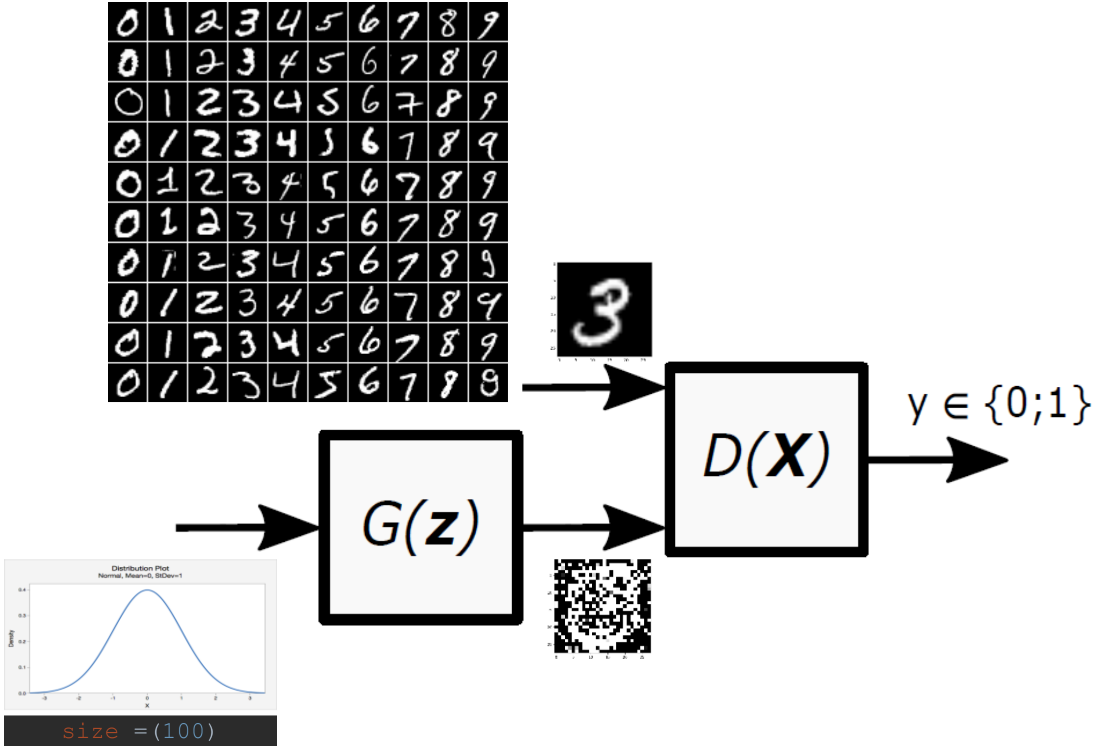
Ming-Yu Liu, Thomas Breuel, Jan Kautz
NVIDIA
`{mingyul,tbreuel,jkautz}@nvidia.com`



https://arxiv.org/pdf/1703.00848.pdf

Generative Adversarial Learning – Deep Dive MNIST GAN

Fully Connected GAN for MNIST sample generation



Fully Connected GAN for MNIST sample generation

$$G(\mathbf{z})$$

Generator

$$D(\mathbf{X})$$

Discriminator

GENERATOR

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 500)	50500
dense_2 (Dense)	(None, 500)	250500
dense_3 (Dense)	(None, 784)	392784
reshape_1 (Reshape)	(None, 28, 28)	0

DISCRIMINATOR

Layer (type)	Output Shape	Param #
input_1 (Input Layer)	(None, 28, 28)	0
flatten_1 (Flatten)	(None, 784)	0
dense_4 (Dense)	(None, 500)	392500
dense_5 (Dense)	(None, 500)	250500
dense_6 (Dense)	(None, 1)	501

Generator and Discriminator Co-Design

Complexity of both models with respect to the number of parameters leans to the generator.

Architecture is chosen similar.

Discriminator task is binary classification, which is way easier than the generation task.

GENERATOR

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 500)	50500
dense_2 (Dense)	(None, 500)	250500
dense_3 (Dense)	(None, 784)	392784
reshape_1 (Reshape)	(None, 28, 28)	0

DISCRIMINATOR

Layer (type)	Output Shape	Param #
input_1 (Input Layer)	(None, 28, 28)	0
flatten_1 (Flatten)	(None, 784)	0
dense_4 (Dense)	(None, 500)	392500
dense_5 (Dense)	(None, 500)	250500
dense_6 (Dense)	(None, 1)	501

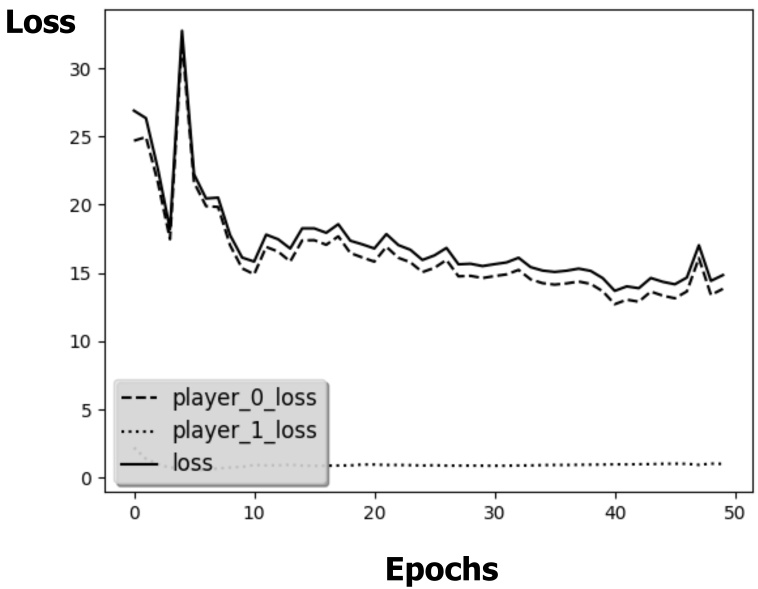
Generator and Discriminator Co-Design

Complexity of both models with respect to the number of parameters leans to the generator.

Architecture is chosen similar.

Discriminator task is binary classification, which is way easier than the generation task.

It is important to balance out the capacities of the models.



Generator and Discriminator Co-Design

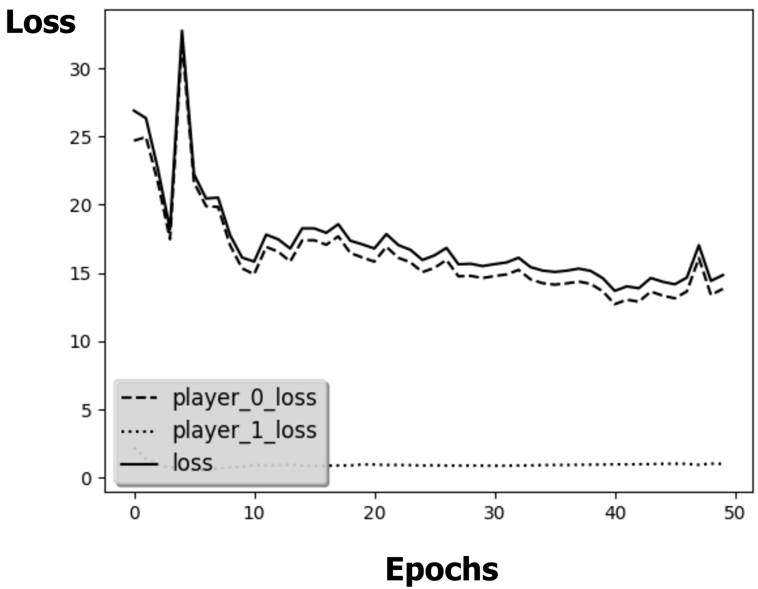
Complexity of both models with respect to the number of parameters leans to the generator.

Architecture is chosen similar.

Discriminator task is binary classification, which is way easier than the generation task.

It is important to balance out the capacities of the models.

In terms of losses we strive for a **constant low Discriminator loss** (never zero) and a continuously decreasing Generator loss.



Generator and Discriminator Co-Design

Complexity of both models with respect to the number of parameters leans to the generator.

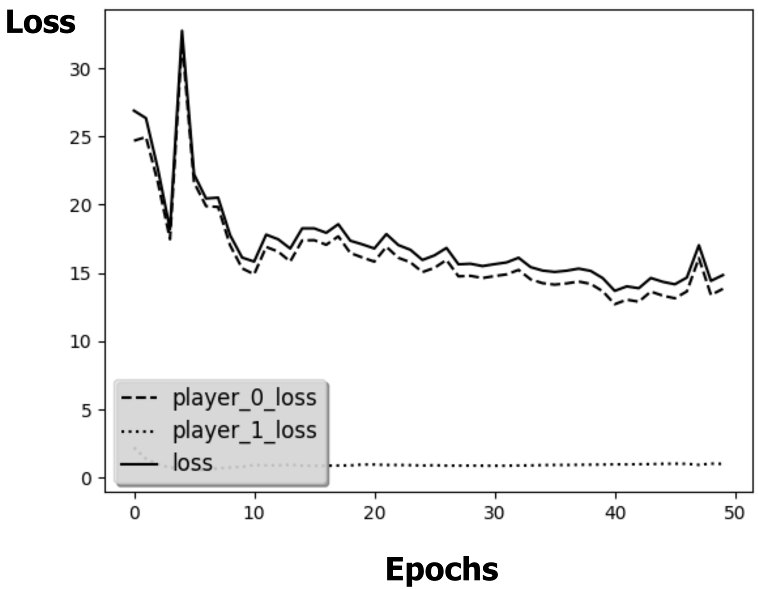
Architecture is chosen similar.

Discriminator task is binary classification, which is way easier than the generation task.

It is important to balance out the capacities of the models.

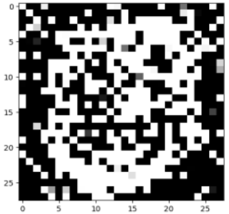
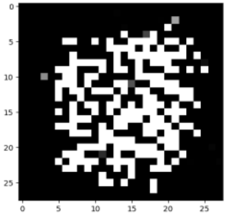
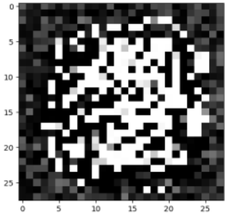
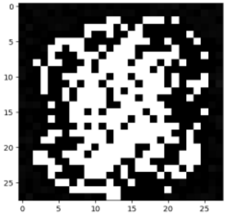
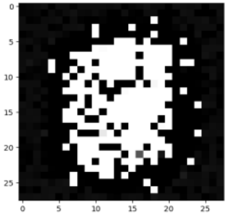
In terms of losses we strive for a constant low Discriminator loss (never zero) and a **continuously decreasing Generator loss**.

Adam Optimizer – 128 Batch

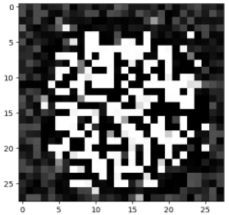
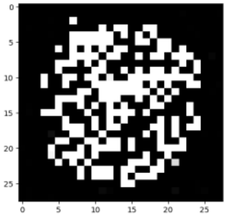
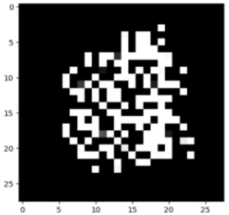
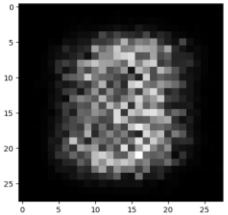
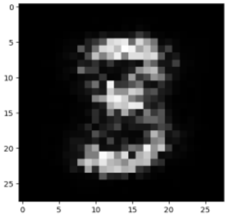


Generative Adversarial Learning – Deep Dive MNIST GAN

Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5

Epoch 6	Epoch 7	Epoch 8	Epoch 9	Epoch 10

Nash Equilibrium

Informally, a strategy profile is a Nash equilibrium if no player can do better by unilaterally changing his or her strategy.

Nash Equilibrium

Informally, a strategy profile is a Nash equilibrium if no player can do better by unilaterally changing his or her strategy.

If your opponent does not change his strategy but you would, could you be better of?

Nash Equilibrium

Informally, a strategy profile is a Nash equilibrium if no player can do better by unilaterally changing his or her strategy.

If your opponent does not change his strategy but you would, could you be better of?

If the answer is yes, for either player, there is no Nash equilibrium.

Nash Equilibrium

For GANs, reaching Nash equilibrium is not trivial. Thus convergence of GANs is never really a possibility.

Nash Equilibrium

For GANs, reaching Nash equilibrium is not trivial. Thus convergence of GANs is never really a possibility.

This is due to the **training on mini-batches**.

And the **complexity of the source domain distribution** that might be very complex.

Nash Equilibrium

Mode Collapse

While training a specific subdomain of the source domain might be covered earlier and thus *easier* than other subdomains.

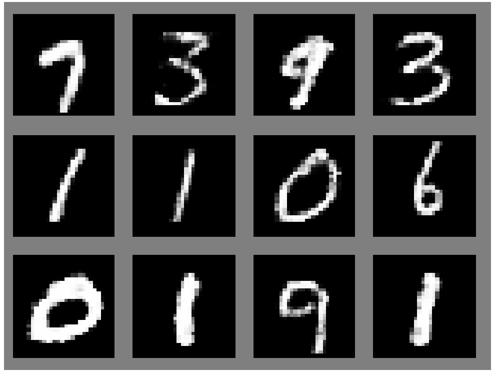
Nash Equilibrium

Mode Collapse

While training a specific subdomain of the source domain might be covered earlier and thus *easier* than other subdomains.

When generating MNIST samples it can be observed, that:

Class 1 samples are generated more often, whereas class 8 samples are generated quite seldom.



https://arxiv.org/pdf/1406.2661.pdf

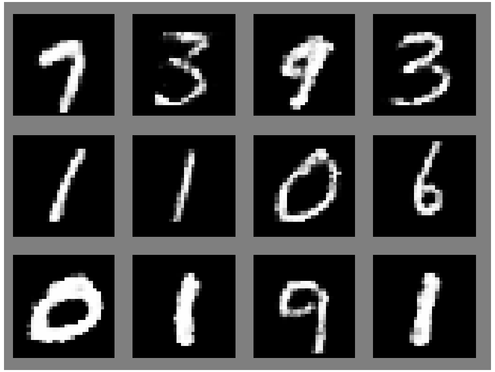
Generative Adversarial Learning – Why GANs do not work yet

Nash Equilibrium

Mode Collapse

While training a specific subdomain of the source domain might be covered earlier and thus *easier* than other subdomains.

This happens since the discriminator only classifies with respect to a single sample and not the whole batch of generated samples.



https://arxiv.org/pdf/1406.2661.pdf

Nash Equilibrium

Mode Collapse

Hard to Evaluate

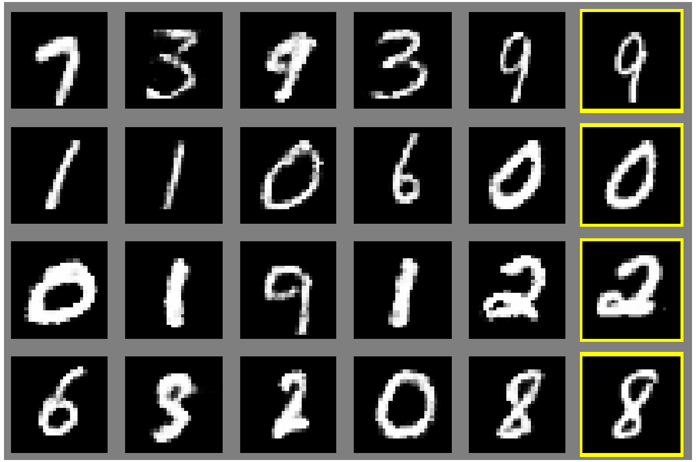
Then once generated it is also hard and not clear how the generated samples are to be evaluated

Both in quantitative, as well as qualitative aspects.

Look and Feel

The first and obvious step is usually a qualitative comparison between:

Individual generated samples
and real samples (in yellow frame).

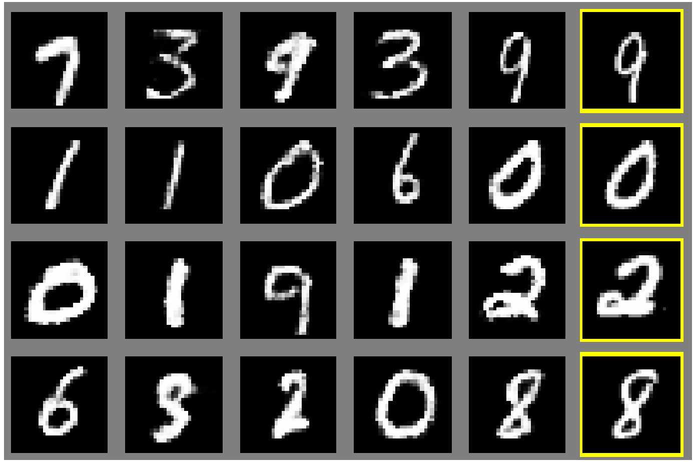


https://arxiv.org/pdf/1406.2661.pdf

Look and Feel

The first and obvious step is usually a qualitative comparison between individual generated samples to real samples.

This is comprehensible and makes for great visualization. And a first intuition whether the generation does something meaningful to begin with.



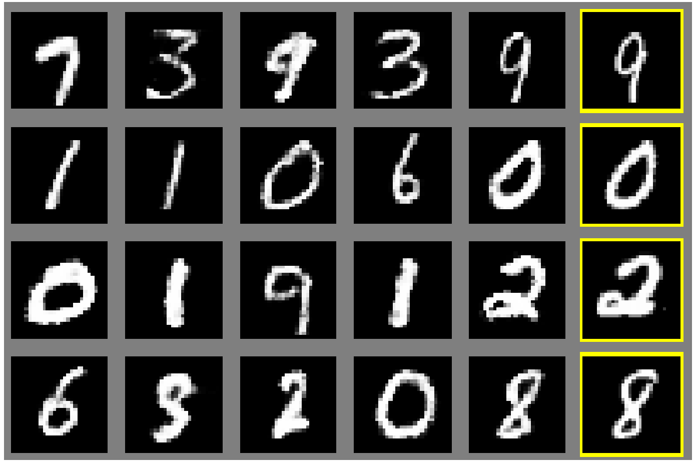
https://arxiv.org/pdf/1406.2661.pdf

Look and Feel

The first and obvious step is usually a qualitative comparison between individual generated samples to real samples.

This is comprehensible and makes for great visualization. And a first intuition whether the generation does something meaningful to begin with.

Yet the drawbacks such as subjectivity and the prevention of comparability between methods are obvious.



https://arxiv.org/pdf/1406.2661.pdf

Look and Feel

Feature Space Mappings

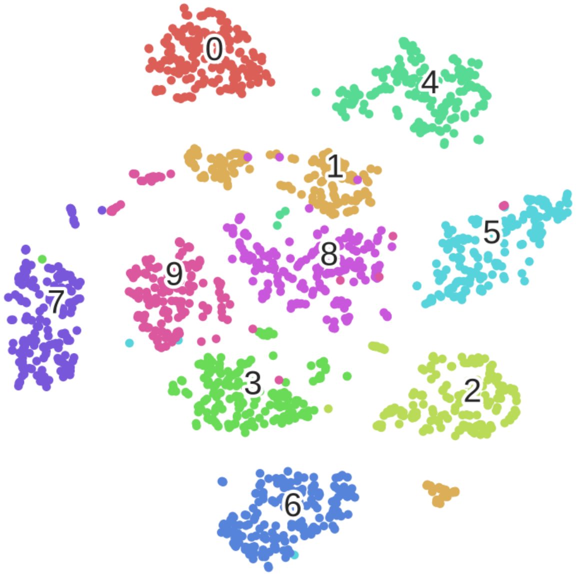
Embedding generated and real samples allows to quantize the domain gap and interdomain distances.

Look and Feel

Feature Space Mappings

Embedding generated and real samples allows to quantize the domain gap and interdomain distances.

For MNIST the subdomains can be seen very nicely separated and clustered once mapped to a 2D feature space.



https://scholar.google.de/scholar?oi=bibs&cluster=8767406368118438995&btnI=1&hl=en

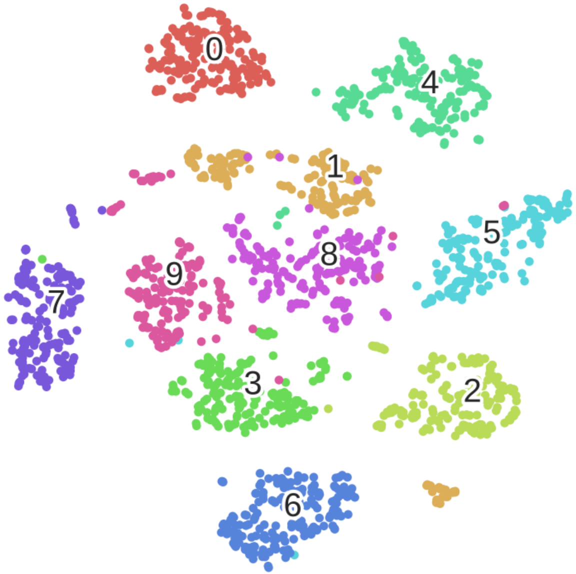
Look and Feel

Feature Space Mappings

Embedding generated and real samples allows to quantize the domain gap and interdomain distances.

For MNIST the subdomains can be seen very nicely separated and clustered once mapped to a 2D feature space.

This can be achieved by iterative mapping algorithms such as t-SNE



https://scholar.google.de/scholar?oi=bibs&cluster=8767406368118438995&btnI=1&hl=en

Look and Feel

Feature Space Mappings

Auxiliary Task Metrics

Once GANs reach application the central question becomes, do those samples help?

Look and Feel

Feature Space Mappings

Auxiliary Task Metrics

Once GANs reach application the central question becomes, do those samples help?

Thus the quality of the generated samples becomes measurable by their influence on a given application task.

Look and Feel

Feature Space Mappings

Auxiliary Task Metrics

Once GANs reach application the central question becomes, do those samples help?

Thus the quality of the generated samples becomes measurable by their influence on a given application task.

Such as an object detector, but can really be anything.

Look and Feel

Feature Space Mappings

Auxiliary Task Metrics

Once GANs reach application the central question becomes, do those samples help?

Thus the quality of the generated samples becomes measurable by their influence on a given application task.

Think, do the samples improve my object detector **performance** after they have been **translated** to the target domain? Usually does require additional training.

Look and Feel
Feature Space Mappings

Auxiliary Task Metrics

Once GANs reach application the central question becomes, do those samples help?

Thus the quality of the generated samples becomes measurable by their influence on a given application task.

Think, does my **object detector work** on the **generated samples**? Thus the semantic can be assumed to be still the same. Usually requires additional labeling.

General Remark

Intrinsic Ground Truth

As GANs usually consist of two independent feedback loops and intrinsic objective functions, ground truth or labels are not necessary for this type of architecture.

General Remark

Intrinsic Ground Truth

As GANs usually consist of two independent feedback loops and intrinsic objective functions, ground truth or labels are not necessary for this type of architecture.

Benchmarking not clear yet

Further it is not yet clear how to evaluate GAN architectures in general, this also does away with the possibility for targeted benchmarking.

Evaluating Local Features for Day-Night Matching

Description

Day-night image matching is important for many computer vision applications. Matched samples at different points in time are also used to support GAN training.

http://users.umiacs.umd.edu/~hzhou/dnim.html

Evaluating Local Features for Day-Night Matching

Description

Day-night image matching is important for many computer vision applications. Matched samples at different points in time are also used to support GAN training.

Targeted Problem

Detectors are affected by day-night illumination changes to a large extent.

There is great potential for improving both detectors and descriptors for night images.

http://users.umiacs.umd.edu/~hzhou/dnim.html

Generative Adversarial Learning - Take it from here

Paper

Generation of spatio-temporal samples to improve LSTM training and prediction capabilities

Tutorial

at-Automatisierungstechnik 2018

Method
Research Article

Mark Schutera*, Stefan Elser, Jochen Abhau, Ralf Mikut, and Markus Reischl

Strategies for supplementing recurrent neural network training for spatio-temporal prediction

Abstract: In autonomous driving, prediction tasks address complex spatio-temporal data. This article describes the examination of Recurrent Neural Networks (RNNs) for object trajectory prediction in the image space. The proposed methods enhance the performance and spatio-temporal prediction capabilities of Recurrent Neural Networks. Two different data augmentation strategies and a hyperparameter search are implemented for this purpose. A conventional data augmentation strategy and a Generative Adversarial Network (GAN) based strategy are analyzed with respect to their ability to close the generalization gap of Recurrent Neural Networks. The results are then discussed using single-object tracklets provided by the KITTI Tracking Dataset. This work demonstrates the benefits of augmenting spatio-temporal data with GANs.

Keywords: Generative Adversarial Networks, data augmentation, Recurrent Neural Networks, generalization, trajectory prediction

1 Motivation

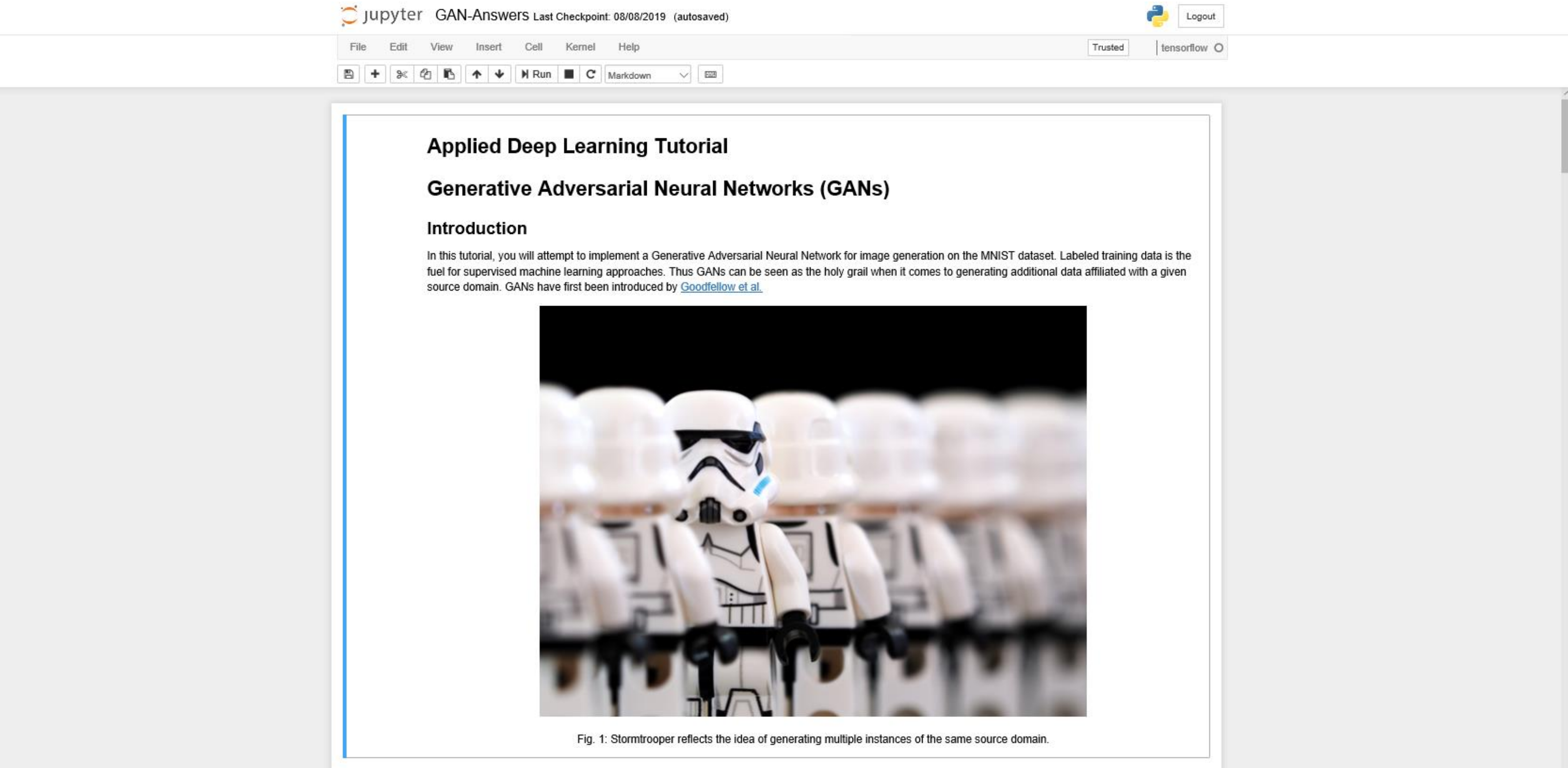
For autonomous driving to succeed, it is essential to track other traffic participants [1, 17, 22]. Applications range from collision avoidance systems and vehicle behavior recognition to path planning algorithms, with the autonomous car needing to process spatio-temporal information. One-step ahead predictions are generally sufficient, since subsequent steps of the trajectory are known, either implicitly or explicitly, the resulting knowledge can be utilized. The information gathered is therefore used to predict future states of the scene or to refine and improve understanding of the current state by spatio-temporal information updates that would have been lost without tracking. The state of traffic participants can be understood in terms of pose and velocity or high-level characteristics such as lane-change maneuvers, evasive movements and the process of turning.

Markov models, such as Kalman filter approaches, are well-established and considered an understood method for capturing sequential interdependencies by modeling state transition probabilities [3, 33]. Despite being applied successfully, however, Markov models are restricted: States of the model are bound to a discrete state space, state transitions depend on the direct predecessor state and hence long-term dependencies are only modeled implicitly. Further deviations from the ideal model are induced by inaccuracies in the assumed probabilistic model [23]. In other approaches, kinematic models are developed explicitly to solve the prediction and tracking task [36]. In autonomous driving and road traffic, vehicles and pedestrians are the subject of extremely complex models. Recurrent Neural Network (RNNs) can be understood in terms of a classical, dynamic system and likewise share the ability to capture sequential interdependencies: Cycles inside the computational graph of the neural network lead to dynamic temporal behavior. In addition, Long Short-Term Memory (LSTM) units have memory cells, enabling the network to model long-term dependen-

Generative Adversarial Learning - Take it from here

Paper

Tutorial



jupyter GAN-Answers Last Checkpoint: 08/08/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted

tensorflow

Run Markdown

Applied Deep Learning Tutorial

Generative Adversarial Neural Networks (GANs)

Introduction

In this tutorial, you will attempt to implement a Generative Adversarial Neural Network for image generation on the MNIST dataset. Labeled training data is the fuel for supervised machine learning approaches. Thus GANs can be seen as the holy grail when it comes to generating additional data affiliated with a given source domain. GANs have first been introduced by Goodfellow et al.

Fig. 1: Stormtrooper reflects the idea of generating multiple instances of the same source domain.

Thanks for your time
Questions?

Contact
mark.schutera@kit.edu