Universität Konstanz

Department of Computer Science

Master Thesis

Towards Crime Forecasting Using Deep Learning

---

presented by:	Udo Schlegel
submitted on:	23. 07. 2018
degree course:	Computer and Information Science
in the subject of:	Machine Learning
	Data Analysis and Visualization
	Department of Computer Science
First assessor:	Prof. Dr. Daniel A. Keim
Second assessor:	PD Dr.-Ing. habil. Christian Borgelt

Konstanz, Germany
July, 2018

Abstract

Crime forecasting is one of the most wanted possible forecasts, as it could lead to fewer crimes and fewer police forces to secure threatened areas. However, predicting when and where crime will happen is challenging. Even modern predictive policing methods don't provide a reasonable or accurate approximation to forecast crimes. It has a long history of experiments and tests to reduce and to prevent crime. Deep Learning is a relatively new machine learning topic, which achieved state-of-the-art performance in many tasks and slowly but surely changes the machine learning area. For a few tasks, it is even better than the human himself.

This success leads to the questions *"Is deep learning able to forecast crime as accurate or even better as a human?"* and *"How to forecast crime using deep learning techniques?"*.

This work presents two deep learning architectures to forecast crime for the cities, Chicago, San Francisco and Los Angeles, which start to answer the second question. It presents a workflow to visual debug network architectures and to steer the architecture design for the learning process in a promising direction. Further, it compares a similar deep learning forecast architecture to the two proposed ones and shows how good or bad the two designs can handle unseen data. In the end, it concludes concerning the results and illustrates what is possible to enhance the methods.

Contents

1 Introduction

Preventing crime is one of the key aspects of a successful society. However, it is difficult to establish a general prevention mechanism for it, as crime is a dynamic process and involves much more than just a human. Thus predicting if and where a crime would be committed, is one of the most famous and prominent tasks the police and other agencies want to achieve. Such a prediction would help to make their job much more comfortable as they wouldn't have to search for possible suspects, and could catch them on the scene. Further, it could be possible to prevent the crime from even happening. The police force would be distributed much more efficient, which would lead to less money spend on and less interrogating police force in some areas of a city. The result would lead to a better feeling of security for the society with fewer police forces needed.

However, as there are no working solutions to the problem, it is evident that there are factors to it that increase the difficulty of this task. Some of these problems get slowly examined by new research and technologies, which emerge and which the police slowly starts to incorporate in their work. Primarily, the amount of data the police collects by using geographical information systems and improved reports establishes a baseline to make forecasting possible. It also enabled researchers to prove that crime is not unique and random but has patterns to it. These patterns are often not easy to identify but exist in the data as proven by recent researchers around sociological effects. Through this new technology needs to be incorporated into the workflow of the police to enhance their ability to predict where crime occurs.

Becuase of the high amount of data, which gets gathered, new technologies are needed to analyze them. As the amount of data grew in the last few years, the need for a machine supported help increased as well. Machine Learning got more and more critical for handling big data in the last few years. The tools it provides to manage and analyze data introduces new possibilities to find patterns in data, which was thought to have none.

Since a few years, artificial intelligence and so machine learning got more in the focus of computer science as it showed better and better results on tasks, which before were hard to do using a machine. Some of these tasks include machine translation, object recognition and detection. Notably, deep learning either got state-of-the-art or even improved results in these fields. Further, predicting, forecasting and generating is doable using deep learning models. This property makes it entirely fitting nearly all machine learning related task. Deep learning incorporates the idea of artificial neural networks, which try to use the brain as an example and model it by applying layers of cells to learn patterns, with a lot of data. Through better Hardware, namely new and better graphics cards, new algorithms, like adding convolutions into its structure, and more data, it got state-of-the-art or better results for many challenges and tasks. It established itself as one of the leading machine learning techniques for a broad range of tasks with various new architectures. Some of these architectures get further used later, like Autoencoders (AE), Recurrent Neural Networks (RNN) and Generative Adversarial Networks (GAN).

All this raises the question:

"How to forecast crime using deep learning techniques?"

Summary This work gives a possible direction towards using deep learning for crime forecasting. However, due to the data being only crimes itself, heatmaps get introduced to be able to abstract, if the neural network produced a reasonable result or if it just nonsense. It helps to introduce a visual analysis into deep learning and crime forecasting and brings the human back in the loop, but on a much higher level, so that he doesn't need to interpret the underlying algorithm, but to evaluate the results. It also corresponds with several crime hotspots theories developed by criminologists[1][2][3][4].

- **Chapter 3** illustrates how it is possible to abstract the data to a form with which a non-expert for the data can use it to verify the plausibility of the forecasts of the systems. Further, it helps to debug if the system produced a sense-making prediction or just gibberish. This method enables the user to get a general idea of the direction in which the system trains and adapts.

- **Chapter 4** shows the first proposed architecture to use to forecast crime. It incorporates different AE designs and uses the latent vectors produced by the AE to train a RNN. The RNN learns therefore on sequences and forecasts the next step in the series.

- **Chapter 5** examines another architecture type, namely GAN. In this case, the generator network gets trained to produce heatmaps, and the discriminator decides on the plausibility of it. The discriminator, therefore, produces another feature to judge the forecasted heatmaps.

- **Chapter 6** compares the usefulness of the two system to one state-of-the-art method to forecast crime using deep learning.

- **Chapter 7** discusses the results and the challenges and possibilities of the two proposed systems.

- **Chapter 8** concludes the work and especially the discussion.

- **Chapter 9** further illustrates a new direction for research in deep learning and crime forecasting. Also, it brings up the possible changes and research to enhance the performance of the systems.

Datasets The following three datasets are used to forecast the crime in the corresponding cities and to generate heatmaps:

- Chicago, Illinois, Crime Data[1] from 2001 till present
- San Francisco, California, Crime Data[2] from 2003 till May, 2018
- Los Angeles, California, Crime Data[3] from 2010 till present

Feature	Data Type	Summary
Date	Date	Date of the crime, e.g. 1/1/2010
Time	Time	Time or approx time of the crime, e.g. 12:01
Longitude	Float	Longitude of the crime scene (anonymized onto blocks), e.g. -87.623177
Latitude	Float	Latitude of the crime scene (anonymized onto blocks), e.g. 41.881832
Type	String	Type or category of the crime, e.g. burglary, car theft

Table 1.1: Features all three datasets share and used for forecasting

These have many different features but share the following features mainly used to generate a heatmap, namely date, time, location and type. More information can be found in Table 1.1. The datasets are not equal in size.

Name	Samples	Types	Location Boundaries
Chicago Crime Data	6504860	28	[[36.619, -91.686],[42.022, -87.524]]
San Francisco Crime Data	2176150	37	[[37.707, -122.513], [37.820, -122.364]]
Los Angeles Crime Data	1672780	115	[[33.342, -118.827], [34.790, -117.659]]

Table 1.2: Dataset statistics

The Chicago crime dataset is the largest one as can be seen in Table1.2 because the time span is much larger than all of the other ones. This much longer time span has a massive influence on the forecasting as it is a critical question of how much historical data is helpful for the task of crime forecast. The longer the used time span is, the more the city could have changed during this period. This development could lead to a significant change in the crime distribution over the city. However, it could also show much better in which area long-lasting chronic hotspots are and that there is no

[1] `https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2`
[2] `https://data.sfgov.org/Public-Safety/-Change-Notice-Police-Department-Incidents/tmnf-yvry`
[3] `https://data.lacity.org/A-Safe-City/Crime-Data-from-2010-to-Present/y8tr-7khq`

prevention for them. And that something needs to happen to solve the issues these areas have with the crime rate and to ensure further development of the city.

San Francisco stopped publishing the crime data in May 2018 to engineer a better service for the publication. However, it is still offline. Further, San Francisco publishes only a small part of all crimes of the city due to ensure the anonymity of the victims and offenders.

Los Angeles only started in 2010 to publish their data. However, they have the most features coming with the dataset. They have more crime types than the other two datasets combined, but this is due to having more possibilities to combine different crime types to form a new one.

Technologies The project is set up in Python 3.5[4] with various modules. All visualizations are created using the Matplotlib[5] module. The development and training of the neural networks are done using PyTorch[5] version 0.4.0[6] with Python on a GPU server.

[4]Python (`https://www.python.org/`) is a high-level programming language and emphasizes code readability and fast, easy development.

[5]Matplotlib (`https://matplotlib.org/`) is a python module to create line plots, heatmaps, and other visualizations.

[6]Pytorch (`https://pytorch.org/`) is a neural network library for training, testing and deploying. It uses a computational graph to calculate the massive computations of a neural network architecture.

2 Related Work

The related work splits into a Crime Forecasting, Machine Learning, and Deep Learning. Crime Forecasting introduces the sociological aspect behind crime forecasting and common crime activities theories. Machine Learning offers an overview of the topic of machine computation and crime time series forecasting. Lastly, Deep Learning presents a sketch over the history of neural networks and how they forecast spatial-temporal movement.

2.1 Crime Forecasting

Crime data gets recorded for a long time, however not to do predictive policing, but to keep a record of crimes and to justify imprisonment. The focus is not the data itself but only the history and actions of a person. For a long time, crime was described as unique and random, in a lot of cases even as an act of opportunity. And thus crime forecasting made no sense, as predicting random events is impossible. This statement means it was not possible to find a general pattern or a connection between different crimes as there is none. However, due to collecting more and more data around crime and especially storing the location made some of these problems obsolete. Recent research showed that there are patterns in the data and that crimes are not unique and random in every case.

Routine Activity Approach The "routine activity approach"[6] is one of the theories, which led to crime forecasting. Other than looking at the characteristics of the offender, the spotlight shifted to the circumstances around the crime. Not just the offender is the focus anymore, but the space and time of the motivated offenders, the targets and absence of capable guardians for the crime. The police records temporal connections between crimes quite well because they show if an individual committed not only the crime he or she was arrested for, but if there are more crimes with the same pattern as the crime of the individual. It means that there is a connection between the location of the crime and the crime itself. For example, a part of the city with a lot of wealthy

families and big houses is a better target than a poor neighborhood, in which the police is present due to drug abuse. Burglary in this area is more common than in other parts, however other crimes like illegal drug abuse are less often. Further, Cohen and Felson show that these theories have empirical evidence in the collected crime data[6]. Through this, they built the base of the plausibility of crime forecasting and show that the data contains patterns and crime is not randomly in time and space.

Hot Spots Another question raises if crime, in general, is distributed randomly only in the spatial aspect, as it is not randomly in the combination of time and space. The example from before already shows a trend that it is not, however, in most cases the research relies on the individual- and household-level. A prevailing theory is the hot spots of crime one and the proof by Sherman et al.[3]. They show that a few hot spots produce most calls to police because crime is rare and concentrated. However, while showing the hot spots, it is difficult to tell at which time, with which cause and with what confidence these hotspots emerge.

Short et al.[2] use the hot spot theory as a baseline to present a statistical model to explain crime. They further show that fine spatial and temporal scales find hotspots in crime data. There are prevailing theories like incorporating weather, daytime and seasons to enable a direction for forecasting.

Repeat Victimization The "repeat-victimization"[7] theory incorporates the fact that offenders often commit a crime against a victim and then another one after a short duration. It shows that victimization is a useful feature for further victimization. Even the more often someone is the target in the past, the higher the probability for further victimization gets. However, "repeat-victimization" often occur in a short duration and the more time passes, the lower the probability gets.

The "near-repeat-victimization"[8] theory enhances the "repeat-victimization" theory by the surroundings. If there is a crime in an area, the probability that there is another one in the same region increases massively. This effect was proven mainly by burglaries in various regions and neighborhoods. The phenomena was even that the houses on the same side as the target had a higher probability of being the target than homes on the other side of the street. However, like the "repeat-victimization" theory, the probability decreases massively after a short duration of time. In the case of burglary, it has its peak in the first 48 hours after the crime and is decreased at the average level one month later.

Hot Spot Sizes Another rather general factor for crime forecasting and sometimes forecasting is the level of aggregation. There are semantic spaces in which the aggregation is possible. In the concrete case of crime forecasting, it means how small or big the predicted geographic areas are. For the police, a quite small area is more feasible than a big one. However, small areas produce fewer crimes and through this are much more difficult to forecast. On the other side, too big areas are clear to forecast as the probability of having crime there is much higher. To find a balance between these two major factors for the size of the to be predicted area is nearly impossible. A way to tackle this problem is to use the districts of a city. However these areas are in most cases too big, but they work better than laying a grid over the whole city with a fixed size of the cells. Another possible way would be to cluster streets together to form prediction areas. This clustering enables further the possibility to enhance the forecasting by providing social-economic data of the streets.[1]

2.1.1 Predictive Policing

Predictive Policing [1] specifies the collection of ways of using different predictive and analytical techniques to predict or forecast potential criminal activity and to prevent them using police force[10]. Four categories[9] classify these methods:

1. *Methods for predicting crimes:* Forecasting the time and place of an increased crime rate area.

2. *Methods for predicting offenders:* Identify individuals with a high risk of offending

3. *Methods for predicting perpetrators identities:* Determine profiles that match with likely offenders with past crimes

4. *Methods for predicting victims of crimes:* Find individuals with a high risk becoming a target for an offender

Often, these methods consist of a combination of statistics on the crime data and sociological research on the corresponding population[11]. Perry et al.[9] also found that conventional crime analysis and investigation methods and recent predictive analysis, which statistically extend or automate processes, are nearly equal.

[1] In this work, a distinction is made between predicting and forecasting. Prediction is subjective, mostly intuitive, and non-reproducible, while forecasting is objective, scientific, and reproducible[9]. The whole work users this distinction and only changes in this chapter as like Perry et al.[9] show, the terms prediction and predictive policing are well established in the law enforcement community but describes forecasting methods.

Theories like the "routine activity approach"[6], or the "near-repeat victimization"[8] theory help to identify problematic areas. Further, hot spot theories[1] with chronic and temporary hot spots help to steer the possible prediction in a specific direction.

Due to the nature of predictive policing, it is somewhat difficult to measure the impact of possible methods. Either these methods were successful, or there was some outside factor like holidays, or due to other circumstances like neighborhood watches or because the profitable targets moved, the crime rate reduces in this evaluated area. This process makes it hard to tell how successful predictive policing methods are[12].

2.2 Machine Learning

Machine Learning together with the increasing computational ability of machines accelerated automation in the last few years. Especially, as its superset of artificial intelligence and its subset deep learning got more popular in the media, machine learning got its way in nearly every area of data and natural science. Further, due to more data being recorded and produced each day, the need for massive computations to reduce this big data to a manageable size increased even more. Machine learning tries to do this in a way that the user doesn't have to be involved in the transformation and can look at the result or the data, which is more important to him. But before this is possible, the machine has to learn what it is interesting in the data or other words what to look.

It's either a supervised task or so-called supervised learning, in which the user already specifies what to look. This task means the machine learns to classify or predict future outcomes by getting data, which a user already gave labels, classes or values to forecast.

Or it's an unsupervised task, or unsupervised learning, where the machine has to learn from data it never saw and get findings by its own. The machine has to identify cluster or groups in the data, which could help the user to either solve their task or to gain insight into the data. Another prominent case of this learning type is dimension reduction. This method enables the user to reduce the number of dimensions the data has and focus their data exploration on features, which are more critical for his task.

As there are more possibilities to use a machine to support a user and automation, there are more than just these two strict learning methods. Some other method is called semi-supervised learning in which the user supervises the machine with labeled data and uses this to process unseen data further.

In the case of crime forecasting, the data consists of labeled data, and the task is to

forecast the future from this data. These forecasts are often in the form of a regression predicting the number of crimes in a given area in the next time window.

2.2.1 Crime Time Series Forecasting

Time Series Forecasting has a long history in machine learning because for example forecasting how a stock develops can save or make millions of money in just a few seconds. Through this history, there are a lot of techniques and logarithms to do so. In the case of stock forecasting, a rather simple regression works quite reasonable in a lot of cases. However, there are more difficult time series, which need much better algorithms.

Regression The autoregressive integrated moving average (ARIMA)[13] is a model to fit and predict time series data, either to understand or to forecast developments of the series. A lot of areas use the method successfully like forecasting economic, marketing and industry production. It is especially suitable for short-term forecasting, however, needs at least 50 or more observations to forecast. The AR part stands for autoregressive and shows that the variable to forecast regresses on the historical data of it. The MA part holds the moving-average model and assumes that the error depends linearly on past and current data. The I stands for integrated and describes that the difference between current and previous values replace the data values. Each of these components helps to fit the model as well as possible onto the data.

Chen et al.[14] describe how they used the ARIMA model to forecast the amount of crime in a Chinese city for the next week from previous weeks. They take as training data the number of crimes per week for 50 weeks and predict the 51st week. Further, they compare ARIMA to two exponential smoothing methods to analyze the performance of ARIMA. Exponential smoothing is usually used to find and visualize general trends in time series to understand its development better. They conclude that ARIMA fits the time series much better than the two other methods and that it is suitable for short-term crime forecasting.

However, they only compare it to more straightforward techniques and measure alone the amount of crime over the whole city and not over districts or grid cells. Their approach applies to use for example grid cells, but needs much more historical data in general and is somewhat limited in implementing additional information.

An approach, which features combining scalable randomized Reproducing Kernel Hilbert Space (RKHS) methods for approximating Gaussian processes with autoregressive smoothing kernels[15], present Flaxman et al.[15]. They solved the National

Institute of Justice's (NIJ) Real-Time Crime Forecasting Challenge[2] with it. The underlying data to forecast was a dataset of Portland, Oregon. Abstractly, they use regression models to forecast the crimes there. Their models don't use simple regression algorithms, but instead make use of the approximation of the Gaussian process, which the RKHS provides. The Gaussian process in a regression context tries to find a finite amount of functions that model the corresponding data. Interpolation, extrapolation or smoothing use this method to achieve their task. All three parts are useful for spatial-temporal forecasting and thus enable crime forecasting.

However, due to the complexity of the model, it is difficult to tune the parameters to be fine-tuned enough to produce the best accuracy. This tuning had to be done to achieve top-notch results and brought them wins in the challenge. Specific tuning of the parameters by a human expert for every aspect is in some cases not suitable and makes this approach somewhat difficult to deploy for other cities. The two proposed systems only need training on the city dataset and are then ready to use and forecast.

Data Mining Techniques Another approach Yu et al.[16] try in using various data mining techniques to forecast crimes on a monthly bases. They use as the baseline a one Nearest Neighbor approach, and further, a location constrained decision tree, a support vector machine with radial basis function as the kernel, a two-layer neural network and a Naive Bayes classifier to compare the results. As underlying data, these methods train on a dataset from Portland. They transform the data to a grid over the city, and the grid cells correspond to the amount of crime for a specific type over a month in this area. Further, a feature vector consists of multiple crimes to forecast the amount of one crime, which is also in the feature vector. The time span for the training is a t months interval with t from one to ten and for the target is one month after the last training month. They provide reasoning for the various design choices with sociological theories like the repeat victimization[7]. Their results show that the neural network outperforms most of the other techniques and especially the baseline. Only the lastly introduced Naive Bayes has better results than the neural network.

However, they have some interesting design choices, which they don't mention any further. A classification replaces the regression problem, through this it is an easier task, as it ignores classes, which doesn't exist in the data. Through the binning they have to do to replace the problem, it is much more difficult to find areas, which develop a more significant amount of crimes over a time span, which is observable in

[2]NIJ Real-Time Crime Forecasting Challenge`https://www.nij.gov/funding/Pages/fy16-crime-forecasting-challenge.aspx` is data challenge to promote predictive models and data science in government institutes.

their results. Their classifiers perform worst on the task in which they have to predict if there is a hot spot or if the crime rate is increasing. As the two proposed systems in this work, are regression models, these binning and classification problems don't exist.

2.3 Deep Learning

Historical Background Deep Learning came a long way from the first starts in 1960 with Multi-Layer Perceptrons (MLP)[17] and the possibility of training them using backpropagation[18] and with their different name Artificial Neural Networks (ANN), to what we have today. The general idea of ANNs is to model the human brain by having different units connected with each other It struggled back then because of the performance of the machines. Further, the difficult task to understand and interpret them, made other algorithms like Support Vector Machines (SVM)[19] more interesting for the users. Algorithms like Decision Trees (DT), Random Forests (RF)[20] and tree-based methods were more widely used, because of shorter training and easier interpretability.

However, two significant events achieved a change in academic and public view.

Recurrent Neural Networks A first smaller event was the introduction of Long-Short-Term Memory (LSTM)[21] for Recurrent Neural Networks (RNN). Their approach made it possible to train them without vanishing and exploding gradient problems, which happened during the backpropagation of the error through the RNN in most cases. RNNs model repetitive or periodic data, such as sequences or events, as they employ ways to interconnect units They are often used for language or sequence tasks, e.g. language modeling[22], neural machine translation[23] or speech recognition[24].

Convolutional Neural Networks The much more significant event was the first convolution layer network LeNet[25] and afterward the ImageNet challenge[26] winner in 2012 AlexNet[27]. LeNet demonstrated how it was possible to train a neural network only on images for classification. It revealed how it was possible to integrate convolutions into neural networks and created the first Convolutional Neural Network (CNN). AlexNet then achieved an error score which was around ten percent better than the runner-up on the ImageNet challenge. It was the first time a system trained without handcrafted features had a better result than anything else. It presented that it was possible to extract the features needed for classification without any other methods

and fully automatically.

Autoencoders The next steps in the direction of deep learning were then to do dimension reduction using neural networks. Famously, Encoder-Decoder systems were built to do that. The system first encodes the input space into a latent space and then decode it back to the output space. The latent space is generally a much smaller space than the input and output space. The most common Encoder-Decoder system is the Autoencoder (AE)[28][29]. It revolves around the idea of the input space is the output space. Through this, it is possible to model the input space data distribution into a smaller space and to generate output data from the latent space by inducing latent noise into the encoder. This property makes it a good dimension reduction and a generative model.

Generative Adversarial Networks One of the more recent developments in deep learning is the Generative Adversarial Network (GAN)[30]. GANs are generative models and do for example data augmentation[30][31]. They incorporate the idea of adversarial learning[32] and let two independent networks work against each other to learn a data distribution. One trains on the real dataset called discriminator and the other one trains on a noise input called generator. The generator tries to produce data, which the discriminator says comes from the real dataset. The discriminator tries to find out if a data input originates from the real dataset or the data created from the generator. This method leads to a game of the two networks against each other until either they both convergence or they both collapse. New research presents GANs for classification tasks[33][34], Text-to-Image[35] or Pose Image Generation[36]. Due to the possibility of extending both networks with various additions, the capabilities of GANs are nearly unlimited.

Spatial-Temporal Forecasting Spatial-temporal forecasting is one of the most wanted methods to have. To forecast, where for example a traffic jam arises, where a criminal wants to break into, where the crowd runs in an emergency case, or where collectives move in general, is essential for a lot of cases. Controlling or understanding some of these movements help to work against them and to resolve issues that triggered these movements. Especially, with the new developments in deep learning, in this area is a lot of research with neural network methods. Due to forecasting crime is also a spatial-temporal movement challenge, a few highlights of approaches to this challenge present alternatives to the methods used later in this work.

The first paper to mention is the one by Ghaderi et al.[37]. Their task is to forecast wind speed to achieve a power balance for the power grid. As wind is somewhat random in some areas, the produced power by a wind turbine changes from region to region and from time to time. This phenomenon leads to a necessity to distribute the power during little wind in areas, which need more power than the wind turbines currently produce. Thus short-term forecasts need to be done. Ghaderi et al. introduce a RNN which uses a graph of the previous h hours to forecast next h_f hours. The nodes of this graph are the weather stations, which are available along the wind turbines and the value in the nodes are the wind speeds.

It is possible to use the presented approach for crime forecasting with regions or grids. However, they assume a connection between the different nodes of the graph, as they use these as a single feature vector for the hour. Assuming a connection between the nodes is in most cases not the case for crime in general. Sure, there are hotspot areas with more crimes and others with less, but it doesn't imply a connection between the one area or the other. Further, the connection of all nodes encode the neighborhood influences, and direct neighbors and far away nodes have equal weight to the forecast. But as the borders are not hard borders in most cases, there is always the possibility that there was only a small shift from one area to its neighbor and thus direct neighbors need a higher influence into the forecast of a region. The two proposed systems of this work present both influences.

Another spatial-temporal forecasting model using deep learning is the Deep Transport by Cheng et al.[38]. Deep Transport forecasts road traffic of a city to determine, which road is advisable to get from one point to another. Again a graph is composed with roads as nodes and crossings as edges. The nodes are weighted with a discrete variable to represent the traffic condition. Further, the connections are directed edges, and thus can be represented into an upstream and downstream. They calculate an upstream and downstream of first, second and third order. The order means for first all direct neighbors, second all neighbors of neighbors and third all neighbors of the neighbors of the second order. They then build a system with seven different inputs. Three inputs correspond to one component handling the upstream first, second and third order. First, the different orders are the input for a convolution layer, and afterward, an RNN combines the calculated features of the three orders. This process is done the same way for the downstream. The last seventh input is the target road. This three components get merged and predict four different time intervals, namely 15, 30, 45 and 60-min, for the target road.

This model incorporates neighborhood and borders. However, for crime forecasting, there are more than just a few connections for possible up- or downstream. Depending on the choice of area mapping, there is a possibility that there are an arbitrary amount of edges for the graph. Also, the directed edge's constraint doesn't hold anymore as crime can develop from one to the neighbor area and back, so it loses its direction. As the proposed systems don't use graphs, the problem of the direction and the amount of edges is not present in this work.

The last proposed model for spatial-temporal forecasting is the ST-ResNet by Zhang et al.[39]. The use case for this model is to predict citywide crowd flows. These are especially important to forecast emergencies, where big crowds flow into a small part of the city. This forecast could lead to utilize emergency mechanisms such as warn and evacuate people in these areas in advance. They construct two matrices, an inflow, and an outflow matrix. These get combined to a two-channel image, where the one channel is a heatmap of the inflow matrix and the other channel a heatmap of the outflow. Their system incorporates four different categories of inputs, namely external features, trends, periods and closeness. External features correspond to, e.g., weather, holidays or events in the city like concerts. Trends samples over the historical two channel images and takes these with a distant time interval to the current one. Periods do the same just for near time intervals. And closeness uses all recent images, so one of the last few time intervals. These four all have an own component in the neural network and use convolution layers to incorporate spatial features. Afterward, the four components merge into a single output, which is also a two-channel image with inflow and outflow matrices.

This model is the underlying base for a crime forecasting the next paragraph.

Crime Forecasting Wang et al.[40][41] show how it is possible to use the ST-ResNet to do crime forecasting. They first generate heatmaps for data they have on hourly bases. These heatmaps are grid heatmaps, which means a grid is put on the city and in the grid cells the crimes. The different grid cells then store the number of crimes which fall in the corresponding cell by its location. Afterward, they crop the heatmap to cover a smaller region, which still has more than 95 percent of the crimes in the global dataset. In the next step, the heatmap is interpolated to a larger width and height to smooth the data of the heatmap. Along with the weather data, these heatmaps are the training dataset for the ST-ResNet.

They incorporate neighborhood and smoothed borders with the interpolation. Through

this incorporation, they implement the hot spot theory[3]. Further, with the ST-ResNet structure, the repeat and near-repeat victimization[7][8] gets also integrated, as more recent in- and outflows are weighted more than the ones further in the past.

However, the results are shown only on a few random sampled cells of the grid and not the whole results of all cells. Also, the results consist only of a specific time span and not of a few ones showing that the model can handle different times. Further, the interpolation lowers the error of the loss function, as neural networks tend to incorporate neighborhood much more than wanted. A neural network will produce a smoothed version of the input even if it gets trained with the not interpolated or cropped heatmap as the to compare error function. Systems proposed in this work learn and focus on the whole heatmap without cropping and interpolation. Through this the results are worse in the sense of score, however, show similar results like the ones from Wang et al.[40][41] Further in Chapter 6 is a more in-depth comparison.

Another approach using ANNs use Stec et al.[42]. They forecast the crime amount for Chicago and Portland with further external data. They structure the crimes for Chicago in beats the police provides. These beats have an arbitrary size and have a high standard deviation of crimes in each beat. Through this high standard deviation, they bin each amount of crime into ten bins of different sizes. The regression problem changes to a classification one, which is simpler and faster to calculate. They further introduce a training method that they call a walk-forward training method. This walk-forward method starts at the dataset start and uses nine months to train and three months to test on. Afterward, it shifts three months in time in the dataset and does the same. This shifting is done till the end of the dataset and ensures a shifting of crime in reality modeled from the data into the model. They train different architectures, namely a fully connected network, a CNN, a RNN and a RNN + CNN one. The RNN + CNN achieves the best results. However, they don't change the architectures of the different networks. They also test if removing external data improves or worsens the results and find out that public transportation and census are both critical for the crime forecasting in their models.

However, due to changing the problem from regression to a classification, they change the challenge to an easier one as they don't have to forecast numbers, but bins. Also, other architectures would lead the fully connected network to achieve much better results, adding further dropout and other techniques would increase the accuracy of this and other systems. The shifting is quite a good idea, however, depending on the amount of data of each of these windows, the network would specialize on the ones

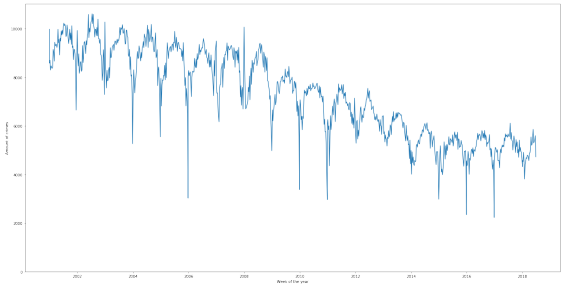
with the most data, and the data from this section would be preferred. Learning with more epochs on recent windows would improve the focus on late windows with a high probability. The systems, presented here, do a regression and are tested with a lot of options and architectures. Further, they are trained heavier on weeks before the to forecast week to provide more insight for the network to look on this recent weeks.

Further approaches from Corcoran et al.[43] describe the use case of creating clusters of hotspots and forecasting on these hotspots. They create clusters by including the chronic crime hotspot theory. First, they use a density distribution over all data points they have. If there are enough points in one of the density areas, it is a density region, and they use it for the next steps. Afterward, they calculate the centroid of a density region and store it. Next, they combine density areas which overlap each over. These then form a cluster. They further calculate the Gama Test for each cluster. "The Gamma test estimates the best mean square error that can be achieved when modeling the data using any continuous model fitting method, such as least squares regression or a neural network, for an unknown function."[43] The Gamma test enables them to estimate the MSE for every cluster and prevent the neural network from overfitting by stopping at this estimated MSE. They show that the neural network performs best on most clusters against regression and random walk. However, they identify clusters for which all there models perform poorly.

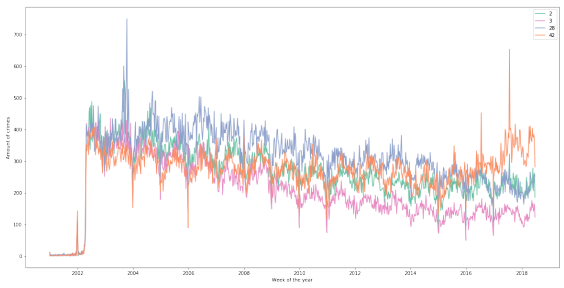
The work by Corcoran et al.[43] includes the chronic hotspot theory to support their reasoning about creating clusters. However, especially temporary hotspots are appealing to find and to forecast, as these can either develop into chronic ones or prevented by police forces and thus lowering the crime rate in general. The proposed systems incorporate all possible hotspot types beforehand as forecasting heatmaps show smaller temporary and also random hotspots.

3 Visual Debugging of Crime Forecasting Results

Crime is often not feasible just from the data for a human observant[4]. Date, time, location, type, and arrest are features in a lot of cases for crime data sets. Through this structure, it is not that easily possible for a human to grasp the underlying patterns. A vector with the data structure from above has no meaning for a pattern as it only covers one single crime, which can either contribute to one or be a coincidence and a random act of opportunity[1]. One of the steps, which could be done to aggregate these vectors, consists of combining them for hours, days or weeks and show how they develop over time. However, this shows a general increase or decrease in crime in general. Fig. 3.1a shows a time series line plot of the development of the crime data of Chicago, Illinois. A general decrease is observable, however, where it happens is not apparent. Fig. 3.1b shows the crime rate for the different wards of Chicago. At first, a high-level decrease is observable on most of the wards, but there is also the orange ward, which increases massively.

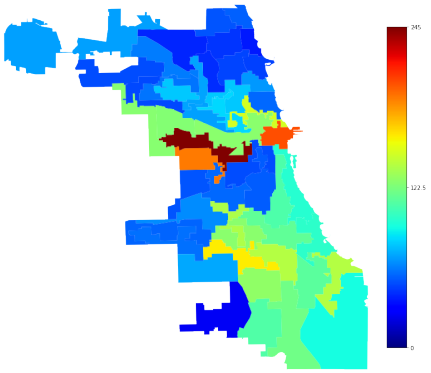


(a) Crime rate development of Chicago. A general decrease is observable. It shows the seasonal differences of crimes.

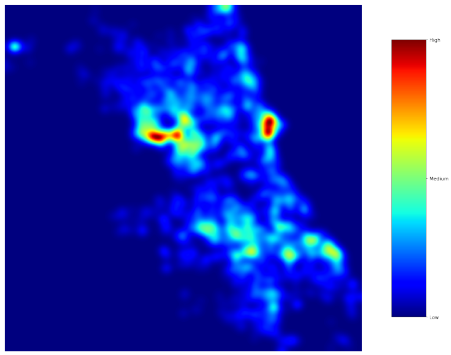


(b) Crime rate for the different wards of Chicago. An increase in one of the wards is observable, nearly all other wards decrease.

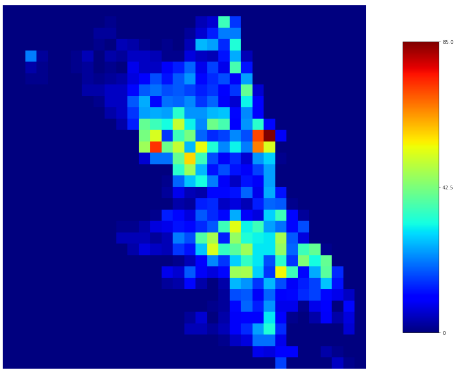
Figure 3.1: Crime rates as line plots of Chicago from 2001 till present.



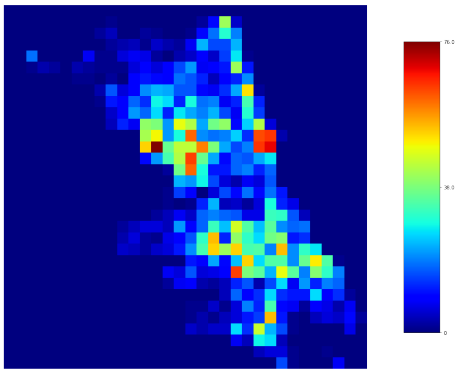
(a) Mapping of the crimes to their corresponding wards.



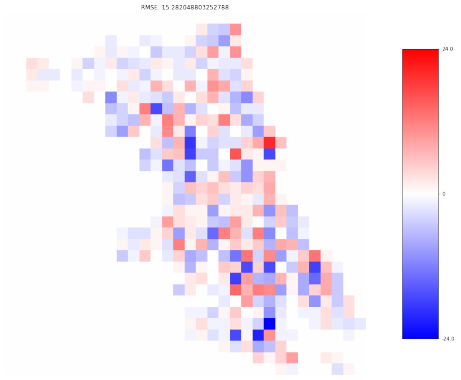
(b) Mapping of the crimes onto a map and showing the density of the crimes.



(c) Mapping of crimes into cells of a grid.



(d) Mapping of crimes into cells of a grid however one week later than the previous grid heatmap.



(e) Difference heatmap to show developments from the two grid heatmaps.

Figure 3.2: Different mapping possibilities to visualize the spatial density distribution of crime in Chicago of the first week of 2015.

Spatial Mapping To further, enhance this data, mapping to different regions or districts is possible. This spatial mapping could be used to identify areas with high crime rates and developments over time[3]. Nonetheless, the question of using regions, districts or even grid cells, raises questions. Regions can be defined as high-risk areas or as some streets, which could be in some cases rather large or in others quite small. Which region, in this case, is better fitting then others is highly debatable and hard to find.

Districts on the other hand, are in general set by the city council and have some useful social and economic knowledge behind it. Often, factors like industrial area, shopping area or urban area influence the decision of the spatial size of the district, which would help to predict certain types of crime. However, in most cases, districts are too big to patrol for the police, but would lead to a better forecast as the number of crimes is higher and so the amount of samples to learn. Fig. 3.2a shows such a mapping for Chicago. The areas shown are wards and correspond to the administrative and representative regions of the city, especially for elections.

The most flexible option is the grid one. In this case, a grid gets laid over a city. The coordinates then get translated into this grid and into the corresponding cell. However, a grid doesn't use the distribution of the rivers, railroads, and streets into account and cause of this can have a somewhat tricky route to patrol for the police. The grid size is adjustable, which helps for some of these influencing factors. A larger grid size means a better forecast accuracy, but also like a larger area to prevent crime to happen. Or more police force needed to patrol the area. A smaller grid size betters the distribution of the police force and more focused prevention. However, it also leads to worse forecasting as the data for the regions could be rather sparse.

Grid Mapping In general, the grid approach makes the most sense, as it models the crime hotspot theories[1][3]. Crime hot spots can appear in a spectrum of three categories. The first category contains crimes which occur spatially random and are somewhat non-hotspots. These are for example crimes, which happen because of opportunity. The second category consists of temporary hotspots. These are hotspots, which occur only once, infrequent or only sparsely recurring. And the last category is chronic hotspots. These occur often and need to be addressed and prevented. Chronic hotspots are the most promising crime regions to predict, as prevention of these would lead to a rapid decrease of crime. Further, they are more straightforward to predict because the amount of data for these hotspots is way higher than for the other categories. However, also temporary crime hotspots need to be addressed as burglary, for

example, happens less often than other crime types and with a different distribution. In most cases predicting temporary crime hotspots need to be addressed more due to becoming chronic in a lot of cases. It would be even better to forecast temporary hotspots before they emerge to prevent them from becoming chronic ones[1].

Heatmaps The hotspot and the grid concepts allow calculating grid heatmaps. Overlaying the city with a fixed size grid creates a grid heatmap. Each grid cell contains the crimes, which lay in the cell boundaries with its location. The minimum and the maximum amount of the contained crimes build two borders on which a color scale maps. In the case of this work, grayscale is the color scale in which it gets transformed. Grayscale means that white is the minimum and black the maximum. Fig. 3.2b shows a heatmap with the jet[1] color scale for the first week of 2015. The color scale is jet because it shows hotspots better in this case. These make use of the hotspot theorems by visualizing these with the spatial aspect[44]. Chronic hotspots can easily be identified as being the maximum color, in this case, black. Also, by taking into account the heatmaps of the next intervals, temporary hotspots can be identified. These should emerge and then disappear in a given time. However, they should be identifiable over the course of the whole dataset. The advantages shown above conclude that grid heatmaps perfectly fit the task of crime forecasting[1].

Grid Heatmaps Grid heatmaps further help to incorporate state-of-the-art deep learning into the method pool. These heatmaps are images and thus can be used with, e.g., CNNs, which are currently the most prominent and most used ANNs. Due to their success in many Computer Vision tasks and their general exceptional performance on images, a CNN used onto heatmaps helps to find interesting features, without the need to build a specific feature descriptor on them. This concept connects the input of the ANN to the information the human sees and evaluates. It enables to refine the network with the human itself. If the output produces a somewhat implausible heatmap and the loss remains stable, the human can change the architecture and steer the direction of the network. It is possible to stop the training process and adjust the different components to generate a more reasonable output even after a few training epochs. Steering this process helps to generate images, which are a good forecast and helps to find a suitable neural network architecture. Especially, since training an ANN to have a good score on the accuracy function needs a lot of training epochs, this

[1]Physics and temperature visualizations often use the jet color scale to show essential spots in the data. It is especially common in Matlab.

stopping and deciding for or against the architecture leads to a faster direction change if required. If after a few epochs the generated heatmaps still show only noise, it is evident that the network needs to change in some way. The network should show a proper direction even for a few epochs as a heatmap is quite sparse and the generative model only has to change just a few pixels in a grayscale image to get a better score.

Difference Heatmaps In the case of heatmaps, a difference image can be created using a training and test dataset split while training. A difference image shows where the original and generated heatmaps differ. An example of such a difference heatmap shows Fig. 3.2e, which highlights the discrepancies between the first and the second week of 2015. The network gets trained on the training set and predicts the test set. White shows the area, which corresponds to a correct prediction, while depending on the color scale, different colors show if the network predicted more or less than the correct value. These images can help to identify local features of the heatmap, which the network didn't learn, but offer a tremendous amount of information. Identifying these local errors can help to improve the network architecture, by for example use a smaller kernel for the convolution layers to grasp smaller local features and don't smooth as much as larger kernels. With this method, it is possible to debug the network and steer the process of improving it in a specific direction.

Visual debugging workflow Fig. 3.3 shows a workflow of how the process of visual debugging could be. The network trains and generates images of forecasts after some epochs. A user or expert is then able to decide if the output is plausible or if it needs improvement. The difference heatmaps show regions in which the model performs good or bad and supports the decision making of the human. If the output is valid, the network trains further, if not the expert can change parts of the architecture to increase the performance.

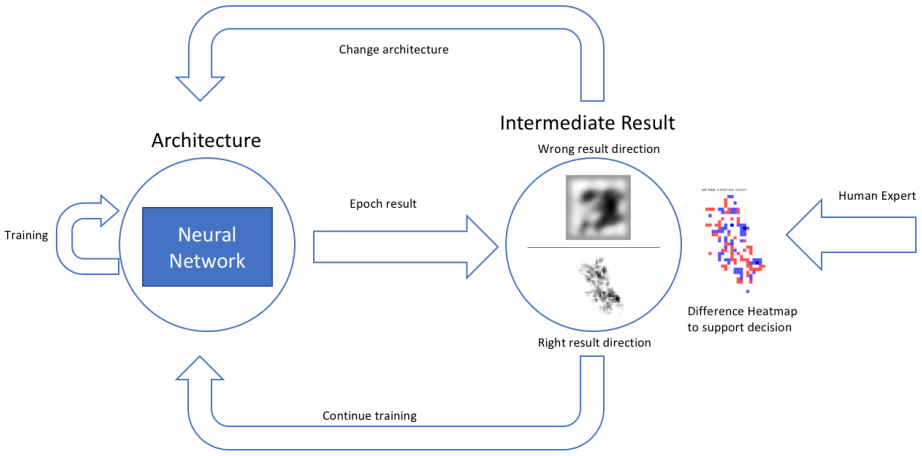


Figure 3.3: Workflow showing visual debugging with difference heatmaps.

Name	**Advantage**	**Use Case**
Spatial Mapping	Connect events with their locations	Crimes on a map, Fig. 3.2a
Grid Mapping	No need for area or city distribution analysis	Heatmaps, Fig. 3.2c
Heatmaps	Show accumulations of the events	Finding hotspots and dangerous areas in a city, Fig. 3.2b
Grid Heatmaps	Finer granularity and changes of streets	Finding finer hotspots and smaller dangerous areas in a city, Fig. 3.2c
Difference Heatmaps	Differences and similarities of two different heatmaps	Analyzing hotspot trends, comparing heatmaps, Fig. 3.2e

Table 3.1: Short summary of the possible spatial mappings

4 Sequence forecasting using Autoencoder architectures

Generative models have a long history in machine learning of being the way to create Artificial Intelligence. However, their use is very application dependent and not clear from the start. Often, they are only used to generate more examples to a given problem to enhance the dataset and thus do data augmentation or build a simulation environment with the additional samples. However, they are far more useful than just for data augmentation or simulation. They can be used to denoise medical images[45], do style transfer[46] or even colorize images[47].

A neural network architecture, which is both a generative and a discriminative model, is the Encoder-Decoder system. The encoder, in this case, transforms the input space into another space called latent space. Afterward, the decoder uses this latent space to transform it back into the output space. Latent space is in most cases smaller than the input space. However, it can also be larger. In the case of smaller latent spaces, encoder-decoder systems function as a dimension reduction mechanisms by using the latent representation as the new representation of the data.[48] In the other one, they implement the sparsity constraint and improve classification by implementing somewhat specialized neurons for the classification.[49] They are used for machine translation, by encoding the input language into a smaller latent space, from which it gets decoded into the output language[22][23][50].

4.1 Autoencoder

Special case of an Encoder-Decoder system is the Autoencoder (AE)[28]. AE uses the input as output space and learns to recreate the input space with its latent space. So, in theory, the encoder part models the input into the latent space and the decoder part recreates the input data from the latent representation. This works as a direct dimension reduction algorithm, as it allows to map a much larger input space to a smaller latent space. Most common use cases include denoising of images[45], as,

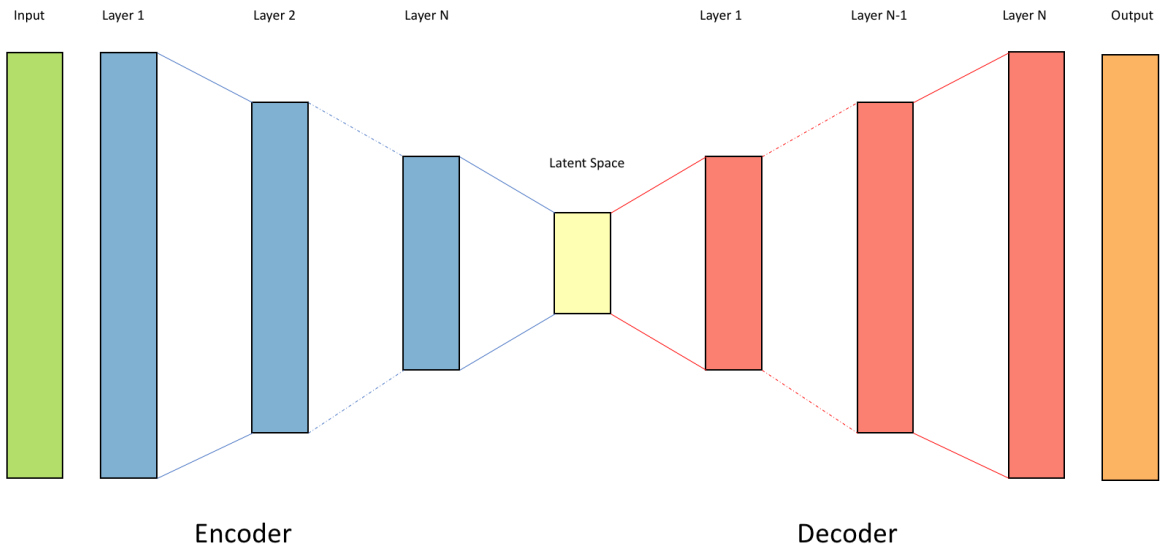


Figure 4.1: Encoder-Decoder system with latent space.

through the dimension reduction, the AE has to learn to distinguish between features and noise. At the reconstruction step, it ignores the noise and only uses the needed features to recreate the input.

Mathematical Foundations The mathematical explanation of the concept follows the case of two functions e (Encoder), d (Decoder):

$$
\begin{aligned}
e &: X \to L \\
d &: L \to X \\
AE &= \underset{e,d}{argmin} ||X - (e \circ d)(X)||^2
\end{aligned}
$$

With $X \subseteq \mathbb{R}^d$ being the input space and $L \subseteq \mathbb{R}^{ls}$ being the latent space. In most cases, d is much larger than ls, and a dimension reduction results. If $d \leq ls$ the AE could potentially learn the identity function and would be rather useless for most tasks. However, as shown by Makhzani and Frey[49], the AE could still learn valuable features, and the latent feature vector could have a more discriminative power as other architectures by learning specific features and specializing on them. The AE trains on the data to minimize the reconstruction error and uses an arbitrary error function for it. In the case above, the error function is the squared error; however, others are also

possible like the root mean square error.

In the case of a single hidden layer, the Encoder e maps $x \in X$ onto $z \in L$ with:

$$e(x) = \sigma_e(W_e x + b_e)$$
$$z = \sigma_e(W_e x + b_e)$$

z represents the latent feature vector for the input x. σ_e is a activation function such as sigmoid or ReLU[51]. W_e is the weight matrix with which x gets multiplied and corresponds to the connection from input to the layer. b_e is the bias vector, which adds the zero activation to the weight and input multiplication to steer the activation in a specific direction.

The Decoder d then maps $z \in L$ onto $x' \in X$ with:

$$d(z) = \sigma_d(W_d z + b_d)$$
$$x' = \sigma_d(W_d z + b_d)$$

With the reconstruction x' of x and the corresponding activation function σ_d, weight matrix W_d and bias vector b_d.

This single hidden layer AE can then be trained like above on the squared error and the loss can be calculated with:

$$L(x, x') = ||x - x'||^2 = ||x - \sigma_d(W_d \sigma_e(W_e x + b_e) + b_d)||^2 \tag{4.1}$$

This enables to train a AE with the loss function L and data from X.

Training Up until this point, AE was only used to recreate the input space in the out space, primarily as an application for images. For the use with crime data, the AE gets trained on the heatmaps. So, it has to map the heatmaps into the latent space and then the latent representation back to the corresponding heatmap. The challenge, in this case, is the fact that the heatmaps are somewhat sparse and contain less information than the general image use case of AE. The heatmaps, in this case, are only grayscale images and through this incorporate only a single color channel and not three like RGB images. The grayscale makes it more difficult for convolutions to extract additional features as the source consists not of three channels with information but only one. It is not possible for them to combine the three channels and form new features and channels. They have to work with only one channel to extract features. Further, as crime has a temporal aspect and the AE has neither the date nor the general temporal

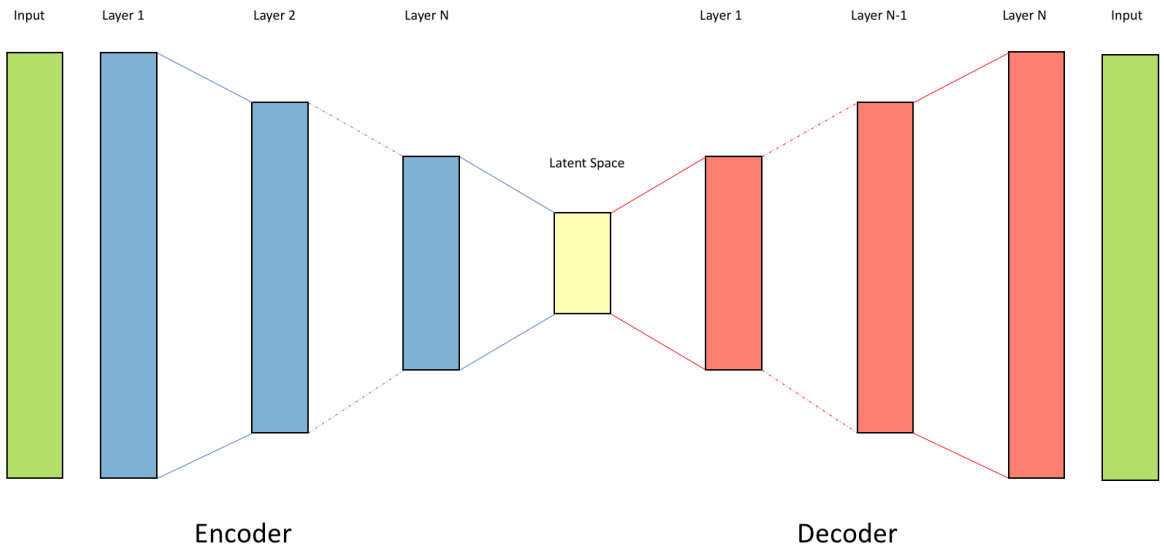


Figure 4.2: Autoencoder system with latent space.

dimension encoded in itself, the forecasting system gets another component. This other component is a Recurrent Neural Network (RNN), which trains on the Encoder output. In the case of this work, the RNN consists of Long-Short-Term-Memory (LSTM)[21] layers. Latent vectors the AE produced after training are then used to train this RNN system. First, the AE converts heatmaps of the data into its latent vector representations. Then the data creates a sequence of a fixed length sorted by its date of the heatmaps. For example, four weeks will build up a sequence, with the earliest week as the starting point and the last one the week, which is three away from the first one. The RNN will use these weeks as the training set. It gets the first three weeks as the base and has to predict the last week. The definition of the loss function for the RNN has two different possibilities. It either can calculate the error of the produced latent vector to the one the encoder generated or of the corresponding heatmap to the one created with the latent vector of the RNN component. The approach which incorporates a human-like workflow is using the heatmaps, which the generated and the learned vector represent. On these two images, an error function calculates the loss for the backpropagation through the network.

Forecasting The weeks before the current week build the data for the forecasting of the current week. The encoder encodes these into their latent vectors, and the RNN component uses these representations to forecast and to generate the vector for the

current week. The forecast can then put back into the decoder of the AE. An expert can then inspect the result of the RNN component and the decoder. If the result is not plausible the number of weeks before the week to predict can be increased to incorporate more historical data.

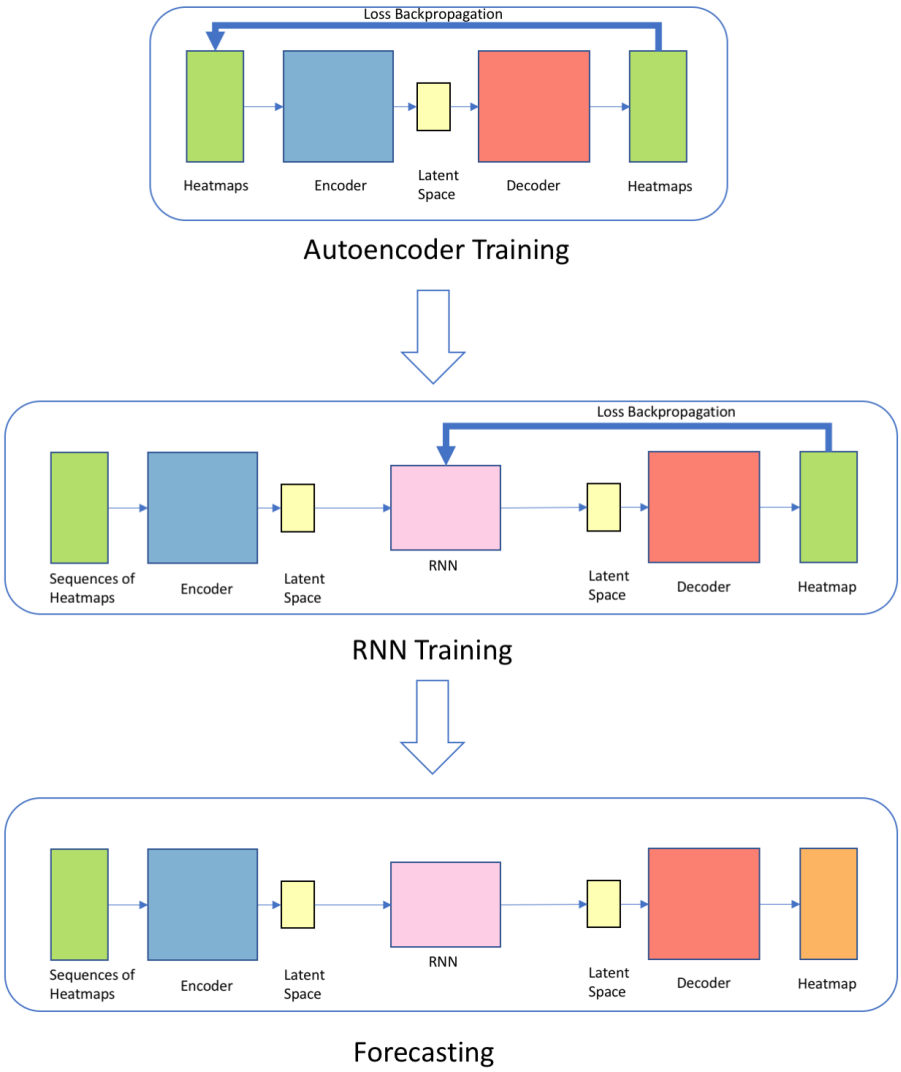


Figure 4.3: Workflow for the training of the AE and the RNN and the forecasting using both.

The general workflow for this is shown in Fig. 4.3.

The exact architecture can be found in Table 4.1.

Layer (type)	Input Shape	Output Shape	Param #
Linear	[-1, 1024]	[-1, 1024]	1,049,600
LeakyReLU	[-1, 1024]	[-1, 1024]	0
Linear	[-1, 1024]	[-1, 512]	524,800
LeakyReLU	[-1, 512]	[-1, 512]	0
Linear	[-1, 512]	[-1, 256]	131,328
LeakyReLU	[-1, 256]	[-1, 256]	0
Linear	[-1, 256]	[-1, 100]	25,700
ReLU	[-1, 100]	[-1, 100]	0
Encoder	[-1, 1024]	[-1, 100]	0
Linear	[-1, 100]	[-1, 256]	25,856
LeakyReLU	[-1, 256]	[-1, 256]	0
Linear	[-1, 256]	[-1, 512]	131,584
LeakyReLU	[-1, 512]	[-1, 512]	0
Linear	[-1, 512]	[-1, 1024]	525,312
LeakyReLU	[-1, 1024]	[-1, 1024]	0
Linear	[-1, 1024]	[-1, 1024]	1,049,600
ReLU	[-1, 1024]	[-1, 1024]	0
Decoder	[-1, 100]	[-1, 1024]	0

Total params: 6,927,560

(a) Autoencoder architecture. Top showing the layers for the encoder, bottom showing the layers for the decoder.

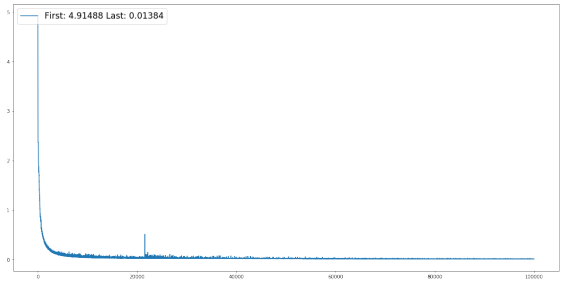
Layer (type)	Input Shape	Output Shape	Param #
LSTM	[-1, 100]	[-1, 100, 100]	1,010,000
LSTM	[-1, 100, 100]	[-1, 100, 200]	101,020,000
Linear	[-1, 200]	[-1, 100]	20,000

(b) RNN design for the Autoencoder RNN system.

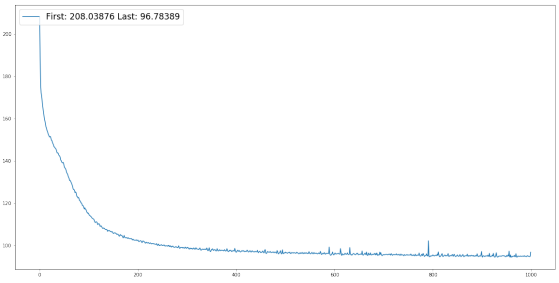
Table 4.1: Exact architecture with layers and neurons for the AE RNN system.

4.2 Variational Autoencoder

An extension of AE is the variational AE (VAE)[29]. They extend the AE design by using a variational approach to learning the latent space. The encoder adds a constraint to itself, which forces generated latent vectors to follow a Gaussian distribution roughly. This addition results in an extra loss component and another optimizer algorithm for the training called Stochastic Gradient Variational Bayes (SGVB)[29]. VAE assume that the training data is generated by some random process, involving an unobserved random variable z, which corresponds to the latent representation. Further, it is assumed that z is generated from some prior distribution $p_d(z)$ and x is generated from some conditional distribution $p_d(x|z)$. They introduce a probabilistic encoder $q_e(z|x)$



(a) Example loss of the AE training on the Chicago crime dataset.



(b) Example loss of the RNN training on the latent space of the AE of the Chicago crime dataset.

Figure 4.4: Example losses of the AE and RNN training on the Chicago crime dataset.

to approximate the posterior inference of the latent variable z given an observed x for some d and to have an approximation to the true posterior $q_d(z|x)$. Additionally, to this encoder, a probabilistic decoder $p_d(x|z)$ produces a distribution over possible corresponding x to z.

The loss function has now an additional loss component, the latent loss:

$$L(e, d, x) = D_{KL}(q_e(z|x)||p_d(z)) - \mathbb{E}_{q_e(z|x)}(\log p_d(x|z)) \tag{4.2}$$

With D_{KL} as the Kullback-Leibler divergence[52], measuring the divergence of one probability distribution from a another expected probability distribution.

See [29] for further information.

In most use cases, the reconstruction error function adds this latent loss function, eq. (4.2), to produce the complete loss, which is needed to backpropagate through the network. And again, the reconstruction error function usually uses the mean squared error. However, to be able to use the latent loss function the encoder has to be changed. The encoder changes to calculate a vector of means and a vector of standard deviations from a vector of variables. The Kullback-Leibler divergence needs these two to calculate a result and produce the latent loss. The mean vector and the standard deviation vector are further used to sample the latent space and produce the latent vector for the decoder to train.

Training The training follows the same way as for the AE. At first, the AE will be trained to model the input to the latent space and then the latent to the output representation. However, due to added constraint, it is slower than a standard AE approach.

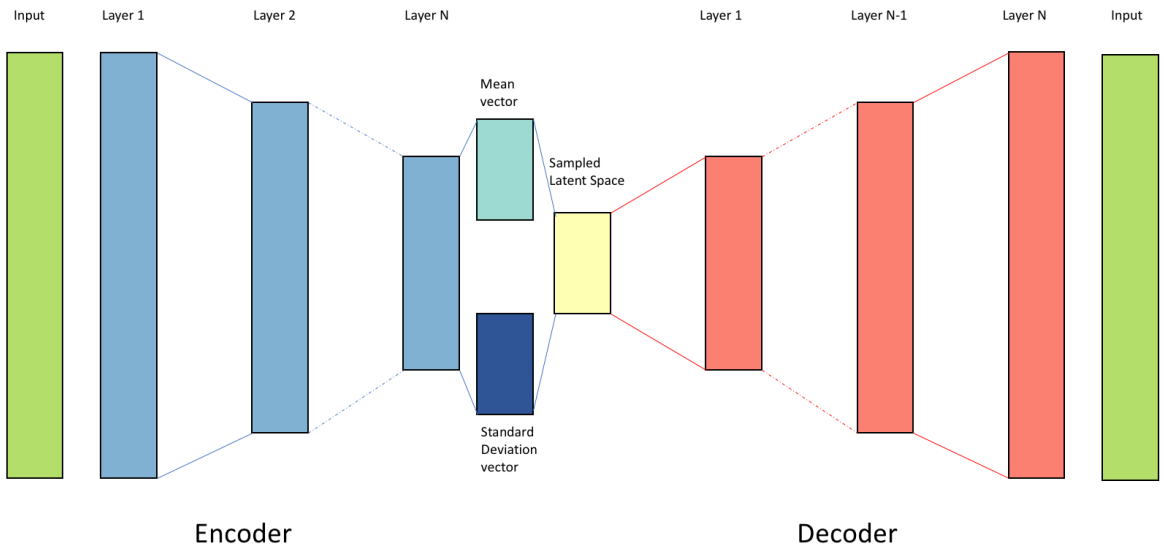
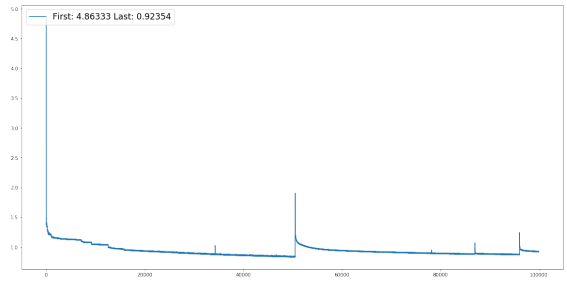


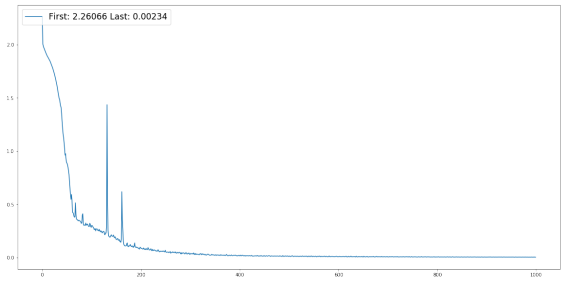
Figure 4.5: Variational Autoencoder system with sampled latent space.

It needs more epochs to train in some cases, as there are a few more computations for the mean, standard deviation and the Kullback-Leibler divergence required for each training step. The training still stays the same with the RNN as it doesn't care for the underlying AE architecture and works with every dimension reduction. It learns the latent vector and gets its backpropagated loss with the heatmaps of the underlying AE.

Forecasting The forecasting also stays the same. However, it is now possible to change small parts on the latent vector the RNN produces to shift the heatmap in



(a) Example loss of the VAE training on the Chicago crime dataset.



(b) Example loss of the RNN training on the latent space of the VAE of the Chicago crime dataset.

Figure 4.6: Example losses of the VAE and RNN training on the Chicago crime dataset.

a much smaller way than with the previous architecture. The proof for that is the Gaussian distribution of latent space. Changing only small parts of the vector should still let them be near the unchanged vector due to this constraint.

The general workflow for this is shown in Fig. 4.3.

The exact architecture is found in Table 4.1a as the layers are entirely identical.

4.3 Outcome

As there are nearly unlimited options to configure the architecture of the AE and the RNN, a short explanation is given to provide inside into the work process. Due to the various additions and possibilities of combining layers and activation functions, a few methods get further mentioned and introduced.

Activation Functions The activation functions, which are important for the architectures, are sigmoid, tanh, softmax, ReLU[53] and LeakyReLU[51].

Sigmoid or also sometimes called logistic function is an S-shaped function, which has its zero in 0.5 and a minimum at 0 and maximum at 1. It is a smoothed version of the threshold function, which was the thought baseline for neurons in the brain. It doesn't have a hard step from 0 to 1, but a rather smooth gradient.

Tanh is the tangens hyperbolicus and another activation function, which changes the borders of the sigmoid function to be at -1 for the minimum and 1 for the maximum. The zero shifts to 0. This shifting and border changing have the advantage of getting larger gradients. This method helps to avoid the vanishing gradient problem.

The vanishing gradient happens if the gradients of the activation functions get too small. Backpropagation through a deep network with a lot of layers needs to backpropagate the error through the network. It needs, therefore, to calculate the gradient of every layer to decide in which direction the weights and biases have to be adopted. If the gradient is small, it gets even smaller in the next layer and through this early layer don't change. So, they stop learning any additional information. Tanh improves, therefore, the gradients better than the sigmoid function.

The softmax function has another property, which extends the activation methods. It enables multi-class prediction as it is in a way a multi-class sigmoid. Output layers often use softmax activations to output a class distribution. Through its multi-class applicability, it is somewhat difficult to compute efficiently and needs more time than sigmoid and tanh.

The rectifier linear unit, on the other hand, is a younger activation function and

especially for deep learning a good option. Rectified linear units (ReLU)[53] are easy and fast to compute and remove the vanishing gradient problem. They are constant zero up to the zero point and then follow the linear function, $f(x) = max(x, 0)$. However, due to being zero up to zero, there emerges a new problem called dead neurons. These dead neurons happen if it has a massive negative bias and the gradient will be consistently zero. Backpropagation through the network could not change the neuron as the gradient to change it, is zero. In most cases, a few dead neurons don't matter for a deep and wide neuronal network, nevertheless, in some rare cases, this can happen.

Instead, ReLUs changed to leaky ReLUs[51]. These change the ReLU function to incorporating another parameter. This parameter sometimes learns along with the other neural network parameters. In most cases, the parameter is 0.01 or starts at this value. The ReLU function transforms into being x if x is larger than zero and x times the parameter while it is smaller or equal to zero. This addition makes it more complicated to calculate the activation, but still not dramatically, and it solves the dead neurons problem as there is still always a gradient which can backpropagate through the network.

There are some other more recent activation functions, but these were not tested, due to ReLU and leaky ReLU being the top activation functions for most use cases.

Layers Further, different layers are essential for the neural networks. These are fully connected, convolution and transposed convolution[54] layers and LSTMs[21].

Fully connected layers are neural network layers, which connect from every neuron of the previous layer to every neuron of this layer. These are the most common layers and the general concept for neural networks. They implement a linear transformation and an activation function to become, depending on the activation function, non-linear. Often the last layer of a neural network consists of a fully connected layer, as due to be able to implement every activation function and to produce nearly every possible output with the correct activation.

Further, convolution layers are layers, which incorporate the computer vision convolution arithmetic. In the case of a neural network, it is a cross-correlation and not a convolution anymore, but this has only a small influence on the procedure and changes not much of the workflow of the neural network. The method, abstractly, slides a kernel over an input matrix, in most cases, an image, and calculates the convolution of the sliding window, or so-called receptive field, with the kernel. Thus does a many-to-one relationship between the receptive field, the kernel and the result of the kernel in the

result matrix. The result is a new matrix containing features the kernel extracts from the previous. Depending on the pattern the kernel learned, these features are edges, circles or arbitrary objects. Often, an activation function uses the resulting matrix as input to calculate the data for the next layer. A fully connected layer is also able to learn the same as a convolution layer. However, it needs much more memory, because of the need to have much more neurons to calculate the same.

Transpose convolution layers[54] is a particular kind of layer, because it does a convolution but transposes it to generate a larger version of the input matrix. It does an up-sampling without the need to previously specify the interpolation technique to use as it learns the best way to do the task by itself. As it is the opposite of a convolution layer, it is a one-to-many relationship. The transpose convolution layer takes the former smaller matrix and creates with a learned kernel a larger version of it. Further, an activation function learns how to handle the result of the transposed convolution and generate the needed output.

Long-Short-Term-Memory[21] layers are the most common RNN layers. A big problem with RNNs is the vanishing gradient through the recurrent layers as they need to backpropagate the error through a lot of layers. LSTM solves this issue by building a highway that can backpropagate the error more efficiently through the RNN.

Other techniques Other techniques include Batch Normalization[55] and Dropout[56]. Both can be used to prevent overfitting and to let the neural network generalize much better. They both try to tackle the problem of overfitting from different directions. Batch Normalization does it on the data, while Dropout tries to prevent it using the neural network and its components.

Batch Normalization[55] uses the same technique a data scientist uses before the training to adjust and scale the input. It is not done before the input layers, so direct on the input data, but on every layer which uses the Batch Normalization. The layers use it on values the hidden layer produces. The Batch Normalization normalizes the activation values by subtracting the mean and dividing the result by the standard deviation of the batch. It helps to decrease the values and to increase through this the training speed of the whole neural network. Because activations can't get too high or too low and so it is possible to increase the learning rate to find a minimum faster. Further, due to normalizing values, it also functions as a slight regularization through the subtraction and the division.

Dropout[56] is a technique, which got famous and widely used very fast. It transforms a part of the brain neuron theories into the neural networks. It forces the network

to learn features with more than just one neuron. The idea is to deactivate neurons during the training randomly. This deactivation means they don't change their bias or weights while backpropagating the error through the network. On testing or predicting, all neurons are active and produce output. This process leads to redundant learning of concepts by different neurons and a better result.

Lessons Learned The AE RNN works good and is fast to train by first training the AE part and later the RNN on the AE converted data. The forecasting is then able to take a sequence of previous weeks to forecast the current one.

Both the AE and VAE show a larger starting loss with convolution and transposed convolution layers. However, both develop a smaller loss quite fast.

The VAE works better as the variational approach works and makes it easier for the RNN to learn to forecast with the latent vectors. These results also show that the forced distribution on the latent space proofs the point of the original paper[29] with the fact that changing the values in a small way does only change the output of the decoder in a small way.

In general, the VAE loss is lower in fewer epochs but needs more time to train one. The AE can get to the same loss in a few more epochs. Both stay a long time at a particular loss and oscillate around them in an epsilon region. The loss of the VAE looks much smoother compared to the loss of the AE, which is interesting as it shows that the mapping work of the different spaces between each other work better at the beginning with the VAE. However, this finding fits the variational property precisely and shows a better mapping of the VAE to the different spaces. The AE in most cases achieves to break the plateau and decreases the loss further, which is curious as it somehow manages to find another better minimum.

The convolution and transpose convolution layers surprisingly help only the AE and are worse with the VAE regarding the loss. It even shows a more substantial loss decrease than the fully connected only AE, while the VAE variant is worse and doesn't decrease slowly like the fully connected one.

For the training of the RNN, the assumption is to have a nearly correctly trained AE. Else, the results the RNN produces aren't useful as they train with an erroneous data. For this work, the assumption nearly holds, as the loss for example from the AE is close to zero. It is also arguable if a fitting AE produces, plausible novel heatmaps.

Guidelines There are some general ideas and advises, which are useful to train AEs on heatmaps. Also, training and using the latent vectors with the RNN has some

difficulties and generate some guidelines for it. These show the configuration for the proposed system to recreate the experiment and distill some of the experiences of the work.

1. VAEs converge much slower than AEs. However, the results of VAE with RNN are better.
2. The AE and the RNN learn with the mean squared error as loss function.
3. The AE and the RNN both use the Adam optimizer[57] with the recommended parameters.
4. The start learning rate for the Adam optimizer is set to 0.0002.
5. The latent space of the AEs is 100 and the same as the GAN noise vector.

To summarize, this chapter shows how it is possible first to train an AE on the data and then use the latent space of the AE to train a RNN. The RNN trains on the sequence of the previous weeks to forecast the upcoming one. So, the AE produces for every data sample of the sequence the latent representation and provides it for the RNN. It illustrates further how the system then can create forecast heatmaps.

5 Heatmap forecasting using Generative Adversarial Network architectures

Generative Adversarial Networks (GAN) have become one of the most prominent systems to generate new data, do data augmentation and model data distributions. Since their introduction in 2014 by Ian Goodfellow et al.[30], the research, use cases and application grew to a new height. Notably, the possibilities of the proposed system attracted a lot of researchers to it. For example, Text to Image[35] synthesis with impressing results or generating person images in arbitrary poses[36].

Adversarial learning is not a new technique. The first basic idea was developed in 1992 by Jürgen Schmidhuber[58], but it lacked implementation and practical use cases. It was later brought up again by Li et al.[32] for behavioral inference and then further developed and published as a working system by Ian Goodfellow et al.[30]. The idea behind adversarial learning is that machine learning gets improved by introducing an adversary. This adversary tries to either manipulate the input data distribution through new unknown and sometimes invalid examples or the output data distribution by the same techniques. Most famous in this category are adversarial attacks, which lead an image classifier to produce a wrong classification, by only changing a few pixels in an image.

Generative Adversarial Networks GANs work by a similar idea. They introduce two competing artificial neural networks. One is called the Generator G, and the other is called Discriminator D.

The following is explaining the concept with images as the underlying data, as it is the most common case and as it is more descriptive than talking about raw data. The discriminator is trained on the input data and has the task to identify images, which the input data doesn't contain. It, therefore, learns the input data distribution and has to decide on the correctness of the data corresponds to the input data. The

generator produces images, which could be in the input data distribution, but without ever seeing the input data. This training is done by showing the generated images to the discriminator and using the result of the discriminator to train the generator. However, the discriminator gets also further trained on the generated images to be able to identify them more efficiently and to work a bit better against the generator. It is an adversarial workflow in which both networks try to beat the other one and result in the convergence of both in the working case.

Mathematical Foundations For a more general and mathematical explanation of the concept, G models the data distribution, and D estimates the probability of a sample being generated by G or contained in the training data. G learns a generator distribution p_g over the training data x. A mapping function for G is constructed from a prior input noise distribution $p_z(z)$ to the training data space as $G(z, \theta_g)$, where θ_g corresponds to the parameters of the ANN of G. D models the function $D(x, \theta_d)$, which outputs a single scalar. θ_d corresponds to the parameters of the ANN of D. $D(x)$ serves as a representation of the probability that x came from the training data rather than from p_g.

The task of D is to maximize the probability of assigning the correct label to training data and generated samples. Thus D tries to maximize $log(D(x))$. Simultaneously G tries to minimize $log(1 - D(G(z)))$ to beat D. This simultaneous training leads D and G to play a two-player minimax game with the function $V(D, G)$ for G and D.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{z \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{5.1}$$

See [30] for more information.

Noise Space It is possible even to use the noise space to identify visual concepts.[59] Radford et al. collect the noise vectors of generated images of a specific image class and average them to produce a mean vector for this class. Through vector arithmetic, it is possible to combine classes and generate examples of new ones. E.g., a class man without glasses subtracts from a man with glasses. Then the results add woman without glasses. This addition produces small changes to the resulting vector, samples of a woman with glasses. They show that it generates much better results than doing it in the pixel space.

This abstraction further shows that the noise vector is usable for more than just producing images and that the generator maps the input data distribution onto the noise space.

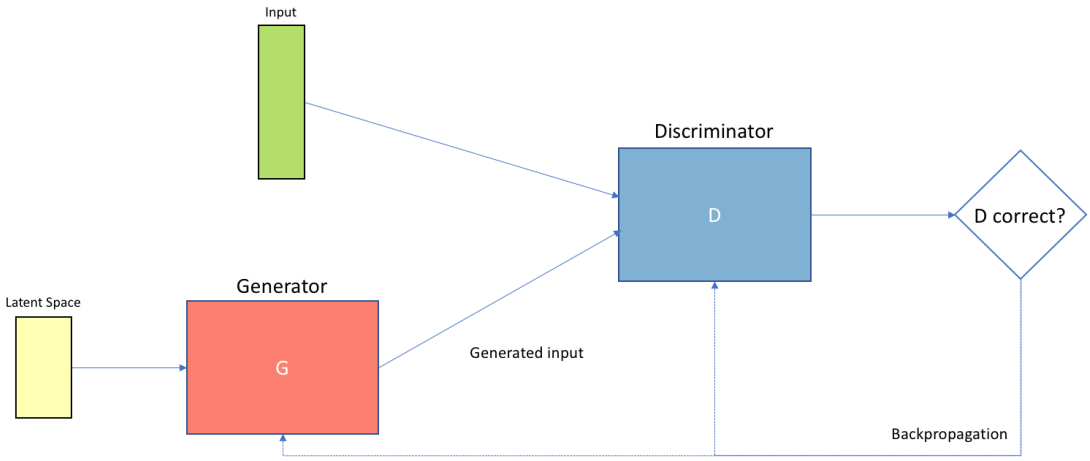


Figure 5.1: General GAN structure.

Mode Collapse Unfortunately, in some cases, a so-called mode collapse can happen, or there is just no convergence in the training time.

A mode collapse can happen if the distribution of the input data has multiple different peaks with significant gaps in between. The generator learns that data from just one of the peaks is correct and that if it generates data from it, the discriminator will say its right. However, the discriminator learns that these are wrong and that another peak, which the generator didn't produce, is the correct one. The generator then models the other peak and starts the whole process again. This oscillation between peaks leads to no convergence and an endless cat-and-mouse game. The challenge is this occurs randomly, and it is quite difficult to tell beforehand if and when a mode collapse happens.

This unstable behavior while training is one of the drawbacks of using GANs. There are some methods to work against it and make them more stable[59], like using multiple GANs with just different subsets or changing batches or mini-batches to limit the possibility of this problem as much as possible, but they work only in some cases. However, their results, the avoidance of many difficult probabilistic computations and the

enhancements, which got published and currently researched, make them a method, which should test various open research questions.

Heatmaps, like in Chapter 3 already shown, are a beneficial method to see changes in the forecast and to further validate the results of the prediction. In this use case, the heatmaps get generated by overlaying the city for which a forecast is wanted with a grid. In the next step, all crimes of a specific type get mapped into the grid cell, which corresponds to the location of the crime. Then a grayscale gets used to color the grid cells. This method produces images which can be used to train the GAN.

5.1 Conditional Generative Adversarial Network

An enhancement to GANs is Conditional Generative Adversarial Networks (C-GAN). It enhances the idea of GANs by adding more conditions to both, the generator and the discriminator. These added conditions can be extra information y like labels of a class or other data and can be directly attached to the generator and discriminator as further input layers.

Mathematical Foundations The input of the generator gets then composed of the prior input noise $p_z(z)$ and y. These were then combined in a joint hidden representation by for example multiplying them with each other. The two-player minimax game function has to adopt the condition y.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}|\boldsymbol{y}))] + \mathbb{E}_{z\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))] \quad (5.2)$$

See [31] for further information.

Conditions enable to incorporate needed information into the GAN further. Further, class conditions also improve the results of the generated samples[60]. In the case of crime predicting with heatmaps, the date is another condition, which needs to be added. The date represents a feature and through this another information the generator needs to be able to forecast. Without it, it would map the noise input space into the data space, and it would be nearly impossible to forecast for a specific date. The noise space needs to be able to interpolate or having a sequential mapping, which is both somewhat difficult to train and validate. These possibilities make conditions a

needed extension for GANs and crime forecasting.

5.2 Auxiliary Classifier Generative Adversarial Network

Another further adaption of GANs is the Auxiliary Classifier Generative Adversarial Networks (AC-GAN). It enhances a GAN architecture with conditions and the discriminator with a class probability. This enhancement means the generator gets like in C-GANs another condition or input, namely the class, and the discriminator receives another output.

Mathematical Foundations To introduce another notation for GANs, D can be written as $P(S \mid X) = D(X)$ and G as $X_{fake} = G(z)$. And the training of D is to maximize the following log-likelihood equation[34]:

$$L = \mathbb{E}[\log(P(S = real \mid X_{real}))] + \mathbb{E}[\log(P(S = fake \mid X_{fake}))] \tag{5.3}$$

With the addition of the corresponding correct class, G gets enhanced to $G(c, z, \theta_g)$ with $c \sim p_c$ and produces $X_{fake} = G(c, z)$. D gets further enhanced to $P(S \mid X), P(C \mid X) = D(X)$ and the equation from above gets split into two parts. The log-likelihood of the correct source L_S and the log-likelihood of the right class L_C[34].

$$L_S = \mathbb{E}[\log(P(S = real \mid X_{real}))] + \mathbb{E}[\log(P(S = fake \mid X_{fake}))] \tag{5.4}$$
$$L_C = \mathbb{E}[\log(P(C = c \mid X_{real}))] + \mathbb{E}[\log(P(C = c \mid X_{fake}))] \tag{5.5}$$

The discriminator D trains to maximize $L_S + L_C$ while the generator G has to maximize $L_C - L_S$.

This condition enables to include the date as a class into the GAN directly. In the case of forecasting a particular week, only the week of the year will be used to enhance the data of the class. Else there would only be one example for every week of the years, and it would be nearly impossible to learn patterns from it.

Training The training is done, using the heatmaps of the weeks and the corresponding date. The generator gets as input the date and a noise vector. The noise vector introduces a likelihood to generate various heatmaps for the same date. This noise vector breaks the deterministic property. However, it introduces a possibility to have

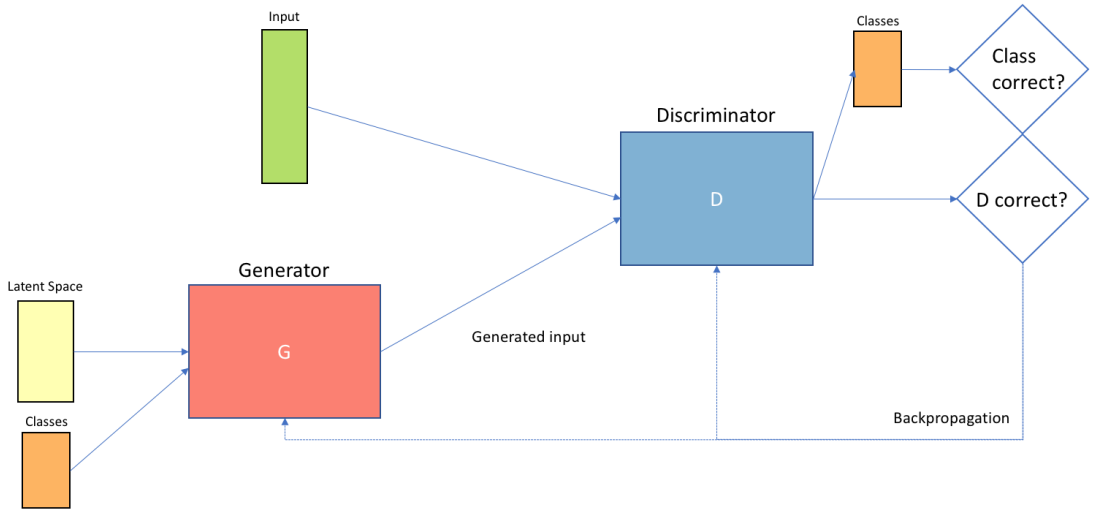


Figure 5.2: General AC-GAN structure.

more than one heatmap for the given date. The discriminator gets trained on the heatmaps of the dataset with their date and the property that they are correct. However, it gets also trained on the generated heatmaps with the corresponding dates and the property that they are fake. The generator gets trained on the decisions of the discriminator if the generated input heatmaps are real or fake and do they correspond to the correct date, and backpropagates this error through itself. This process leads to the two-player minimax game of a GAN and enables the discriminator also to be a confidence estimator.

Forecasting While forecasting, the generator gets the date to forecast on and a random noise vector as input. The discriminator is used to calculate confidence for the prediction of the generator. It has two methods for this. On the one hand, the likelihood of the generated heatmap being a real heatmap or not is one way to decide the plausibility of the forecasting. On the other hand, the probabilistic class output of the discriminator predicts in this case, how likely, the generated heatmap of the generator is the wanted date of the user. So, if the maximum of the probabilistic class output corresponds to the wanted date, the confidence of the discriminator is the highest that the generator is correct. Further, if the maximum is one of the neighbors of the wanted date, it still holds high confidence, as the crimes could have been shifted by one week due to features outside of the model, like weather influences. This methodology intro-

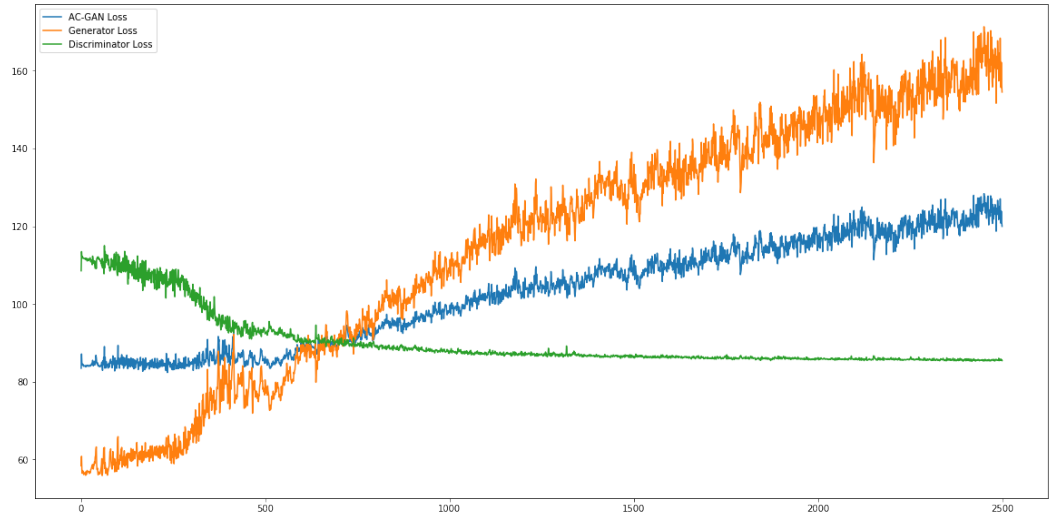


Figure 5.3: Example loss of the AC-GAN, the Generator and the Discriminator showing the general idea of GANs to get to a Nash equilibrium, but eventually overshooting due to the data and architectures.

duces two ways to filter and verify the plausibility of the heatmap before giving it to a human expert to decide if it fits the crime for this given date or not.

Through the random noise vector, it is possible to use the generator to generate various heatmaps and show the human expert only the heatmap with the highest probability of the discriminator. And even if the expert is not satisfied with the result, he can explore already produced heatmap and has the opportunity to find one, which the discriminator decides against, but he likes more. Generated heatmaps will be stored for further inspection.

Fig. 5.4 shows the training and forecasting of the AC-GAN.

Table 5.1 presents the exact architecture.

5.3 Outcome

As like for the AE and the RNN in the previous chapter, there are an unlimited amount of architectures and configurations of GANs. Especially, as GANs are currently one of the most researched topics in this area and as many extensions and enhancements are developed, there are many options, which way to follow. Explanations of used activation functions and layers are in the Outcome Summary of the previous chapter. Due to training a lot of different approaches and architectures, a lot of material to establish lessons learned and guidelines for generating heatmaps got collected.

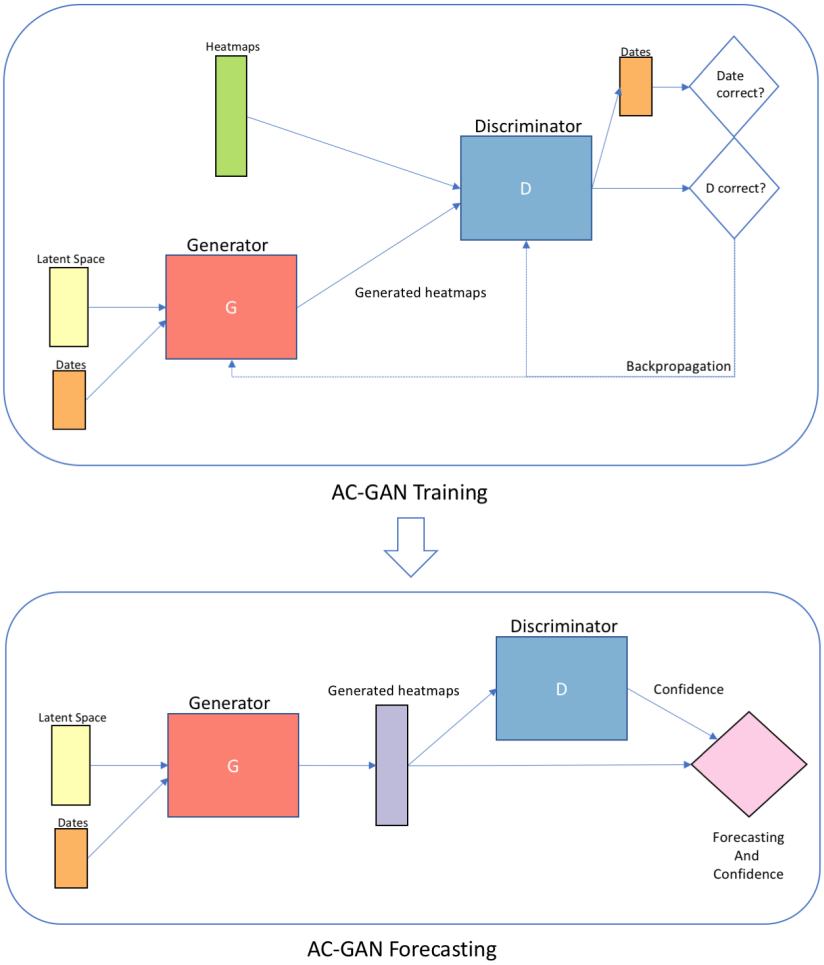


Figure 5.4: Workflow for the AC-GAN training and forecasting process.

Lessons Learned The first lesson of doing regression with a GAN is to use a suitable activation function at the output of the generator. The activation function, which the literature in most cases uses, the tanh function is not suitable to use, as the method has its maximum at 1 and thus can't predict higher scores for the grid cells. This result also lets the network only learn how to generate an image with values between -1 and 1 with distribution to minimize the loss function to an image with values from 0 to x. The generator, in this case, generates fascinating images, which look like noise in a lot of cases. If someone wants to use tanh as the output activation function, he has to normalize the input data into the same scale as tanh. This normalization removes a constraint from the forecast, as the possible maximum is the maximum of the input data. Because if he reverses the normalization for the output data, the scale still stays the same from the input data. So, in general, to create a heatmap with a generator the output layer has an activation function like a ReLU. Especially, ReLU is useful for

this task, as the minimum amount of crimes for an area is zero and the maximum is infinite (in theory), which perfectly fits the ReLU definition.

Output layers smooth the result of a heatmap. This smoothing is one of the most interesting points, as it is entirely unexpected. Neighbors of peaks also have an activation larger than zero, even if the input data doesn't have one. It doesn't have a solid border. This smooth border means if there is a high value at one of the positions, the neighbors also have lower ones. The heatmaps get smooth with a somewhat Gaussian filter. If as preprocessing a Gaussian filter smooths the input data, the GANs have a lower error in general. However, this preprocessing makes the forecast less convincing and enhances the space the police force has to cover to prevent crime.

Guidelines To conclude this chapter a few guidelines show useful tips for generating heatmaps. There are also some guidelines from for example[59] in it to incorporate as much information as possible.

1. ReLU and Leaky ReLU activations produce the best result with reasonable training time.
2. ReLU works best as last activation layer and the output layer for heatmap generation.
3. In the case of grayscale heatmaps, fully connected layers perform best at generating heatmaps.
4. Batch Normalization reduces training time and increases the speed in general.
5. AC-GANs work best for the task, as they introduce a possibility to use dates.
6. The discriminator and the generator both use the Adam optimizer[57] with the recommended parameters.
7. The start learning rate for the Adam optimizer is set to 0.0002.
8. The noise vector has a length of 100 and so the same length as the latent space of the AEs.

To summarize, this chapter shows how it is possible to use GANs and a date to generate heatmaps and to forecast crimes using these heatmaps. It illustrates the concept of using GANs to forecast heatmaps with just the date. Further, it shows how the discriminator can help to filter plausible and implausible heatmaps.

Layer (type)	Input Shape	Output Shape	Param #
Linear-1	[-1, 153]	[-1, 256]	39,424
LeakyReLU-2	[-1, 256]	[-1, 256]	0
Dropout-3	[-1, 256]	[-1, 256]	0
Linear-4	[-1, 256]	[-1, 512]	131,584
LeakyReLU-5	[-1, 512]	[-1, 512]	0
Dropout-6	[-1, 512]	[-1, 512]	0
Linear-7	[-1, 512]	[-1, 1024]	525,312
LeakyReLU-8	[-1, 1024]	[-1, 1024]	0
Dropout-9	[-1, 1024]	[-1, 1024]	0
Linear-10	[-1, 1024]	[-1, 2048]	2,099,200
LeakyReLU-11	[-1, 2048]	[-1, 2048]	0
Dropout-12	[-1, 2048]	[-1, 2048]	0
Linear-13	[-1, 2048]	[-1, 1024]	2,098,176
LeakyReLU-14	[-1, 1024]	[-1, 1024]	0
Dropout-15	[-1, 1024]	[-1, 1024]	0
Linear-16	[-1, 1024]	[-1, 1024]	1,049,600
ReLU-17	[-1, 1024]	[-1, 1024]	0
Total params: 5,943,296			

(a) Architecture of the generator of the AC-GAN.

Layer (type)	Input Shape	Output Shape	Param #
Conv2d-1	[-1, 1, 32, 32]	[-1, 16, 30, 30]	160
LeakyReLU-2	[-1, 16, 30, 30]	[-1, 16, 30, 30]	0
Dropout-3	[-1, 16, 30, 30]	[-1, 16, 30, 30]	0
Conv2d-4	[-1, 16, 30, 30]	[-1, 32, 27, 27]	8,224
LeakyReLU-5	[-1, 32, 27, 27]	[-1, 32, 27, 27]	0
Dropout-6	[-1, 32, 27, 27]	[-1, 32, 27, 27]	0
Conv2d-7	[-1, 32, 27, 27]	[-1, 64, 24, 24]	32,832
LeakyReLU-8	[-1, 64, 24, 24]	[-1, 64, 24, 24]	0
Dropout-9	[-1, 64, 24, 24]	[-1, 64, 24, 24]	0
Conv2d-10	[-1, 64, 24, 24]	[-1, 128, 21, 21]	131,200
LeakyReLU-11	[-1, 128, 21, 21]	[-1, 128, 21, 21]	0
Dropout-12	[-1, 128, 21, 21]	[-1, 128, 21, 21]	0
Linear-13	[-1, 56448]	[-1, 53]	2,991,797
Softmax-14	[-1, 53]	[-1, 53]	0
Linear-15	[-1, 56448]	[-1, 1]	56,449
Sigmoid-16	[-1, 1]	[-1, 1]	0
Total params: 3,220,662			

(b) Architecture of the discriminator of the AC-GAN.

Table 5.1: Exact architecture with layers and neurons for the AC-GAN system.

6 State of the art comparison

As the approach by Wang et al.[40][41] is one of the most recent ones and it uses a heatmap as the underlying data, the results from this work compare to the results from the two proposed systems. The comparison divides into two perspectives:

- RMSE comparison
- Heatmap comparison

Assumptions Wang et al.[40] make different assumptions about the data input for the neural network, they use. First, they exchange the original crime intensity by a diurnal periodic integral mapping. This method smooths the original time series and replaces it with a much clearer version. Peaks are much better, and general patterns are apparent to see. However, outliers are also smooth away. Notably, time intervals in which it has a high peak in the same range before and later, smooth to a maximum like the other ones around. This method helps the task of forecasting as it removes some outliers and normalizes the time series a bit more. Next, they accumulate hours of the smoothed integral data and create a grid heatmap of the hourly data. Then the heatmaps crop down to a focused version. These focused versions consist of 95 percent of the total crimes and are much smaller than the original ones. In the next step, they use bilinear interpolation to enhance the focused versions again to a larger heatmap. This method smooths the whole data again and changes the location of the real crimes by a bit. The created data is the input for the ST-ResNet[39]; they use for forecasting.

RMSE For their RMSE, they use a modified version:

$$\text{RMSE} = \sqrt{\frac{1}{T * N} \sum_{i,t} (I_{it} - {I^p}_{it})^2} \tag{6.1}$$

The modified version incorporates the time periods, they introduce. These periods are 24 hours. This addition normalizes to the period mean of the RMSE for every

interval.

With Convolution Layers	
Training RMSE	Test RMSE
Spatial Temporal Super-resolution	
0.189	0.231
Spatial Super-resolution Only	
0.293	0.361
No Super-resolution	
1.92	1.83
Without Convolution Layers	
Training RMSE	Test RMSE
Spatial Temporal Super-resolution	
0.366	0.385
Spatial Super-resolution Only	
0.413	0.426
No Super-resolution	
0.413	0.425

Table 6.1: Results of Wang et al.[41]. Only taken for 32×32.

Table 6.1 shows the results of Wang et al. for different usage of the smoothing methods. *Spatial Temporal Super-resolution* corresponds to using both ways from above. *Spatial Super-resolution Only* is using the focused versions of the heatmaps. And *No Super-resolution* uses none of both methods. The training and test RMSE are the results of the training and test dataset. The general better scores of the training set show that the method fits right on the training data and that the test data also has samples, which the network doesn't know and just guesses. The result of Wang et al. is tremendously good. However, as they use different assumptions, which are at least controversial, like using all crime data and don't split between for example burglary and auto theft or smoothing data.

As the AE RNN showed the best results in early tests, it trained on the hourly heatmaps of Los Angeles with all crimes using nearly the same information the ST-ResNet got. However, the training set for the AE RNN doesn't have any of the assumptions of the work by Wang et al. Table 6.2 shows the results for the AE RNN with the standard RMSE and the ACC. The next chapter introduces the RMSE and ACC equations. The results of the AE RNN show that it is possible to achieve state-of-the-art results without the need for different smoothing assumptions. It is not as good as using the assumption of *Spatial Temporal Super-resolution*, but it is comparable to the *Spatial Super-resolution Only* and can generalize a lot better than the ST-ResNet,

AE RNN	
Training RMSE	Test RMSE
0.294	0.235
Training ACC	Test ACC
0.967	0.968
AC-GAN	
Training RMSE	Test RMSE
0.659	0.638
Training ACC	Test ACC
0.941	0.942

Table 6.2: Results of the AE RNN.

which states the better RMSE on the test than on the training set.

Heatmaps Fig. 6.1 provides the heatmaps of Wang et al.[41] and consists of the focused versions of the heatmaps they use for forecasting. They depict the forecasting with and without convolution layers. A critical remark is that the heatmaps of Wang et al. need to be flipped to show the real spatial distribution of the city. This flaw is a small error when parsing the longitude and latitude into a matrix style.

The heatmaps the AE RNN don't correspond to the heatmaps of Wang et al. as it was not possible to recreate their heatmaps. The results of the AE RNN show the same days and times two years later and the last hour of the training data. Through this, it was possible to divide the dataset into a 90% training and 10% test split.

The last heatmaps, Fig. 6.2c or the ones from the previous training sample show how good the system can be.

Summary As seen forecasting doesn't need some assumptions and with the right architecture, it is possible to achieve similar results to the ones using smoothing assumptions. The two proposed systems achieve good results, which are comparable to the ones of an ST-ResNet, with just common layers. Further, the RMSE is in some cases not a good indicator to show how accurate the forecast is. In most cases, the number of crimes is not high enough. Further, often is the difference between the forecast and the ground truth in even tricky regions one and zero in the other cases. If the number of cells divides this small sum, the outcoming error is minimal and it is difficult to learn from these. However, the ACC as another example struggles with the same faults of the function. The comparison needs a more general equation, which incorporates the non crime cells and focuses only on the crime scenes.

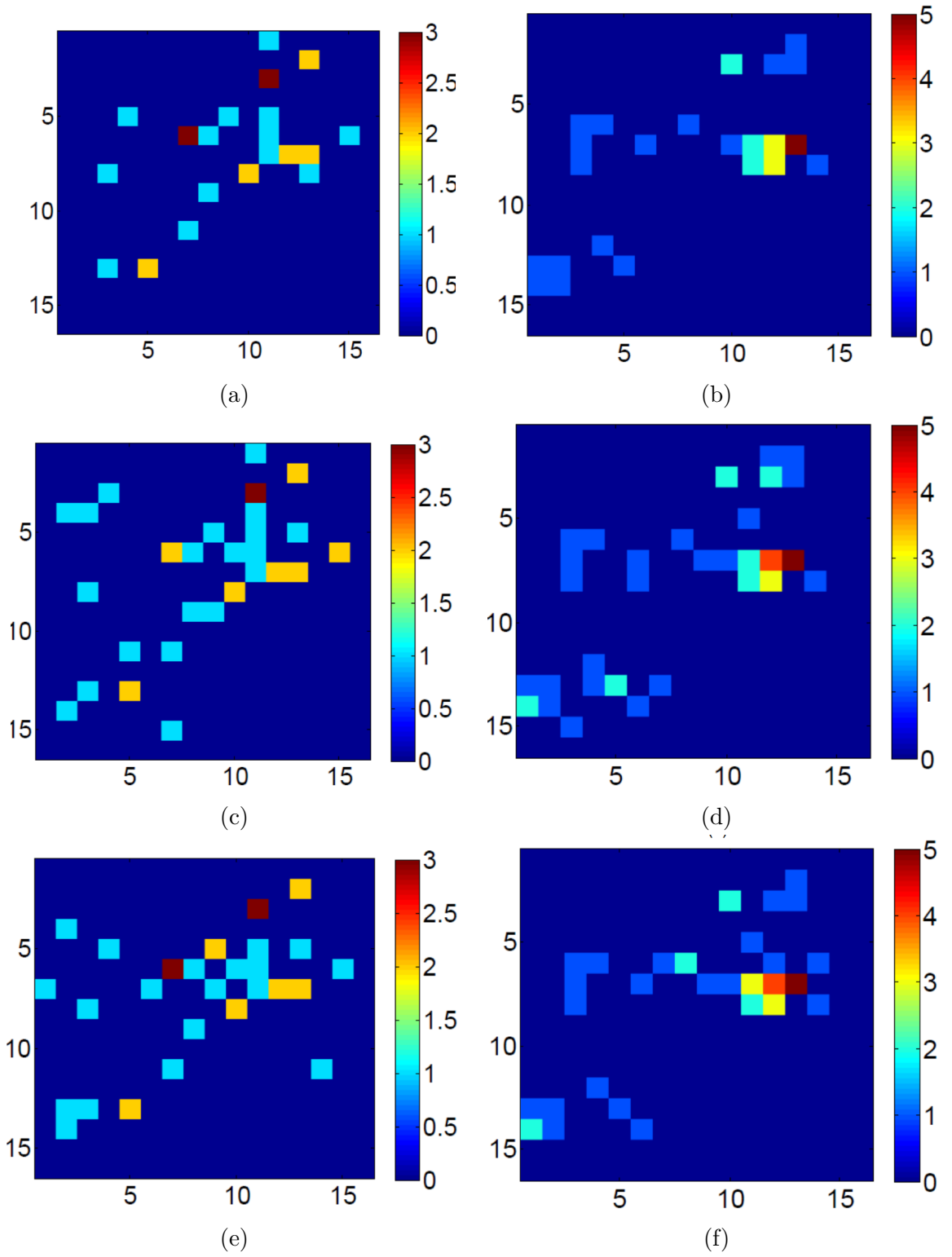
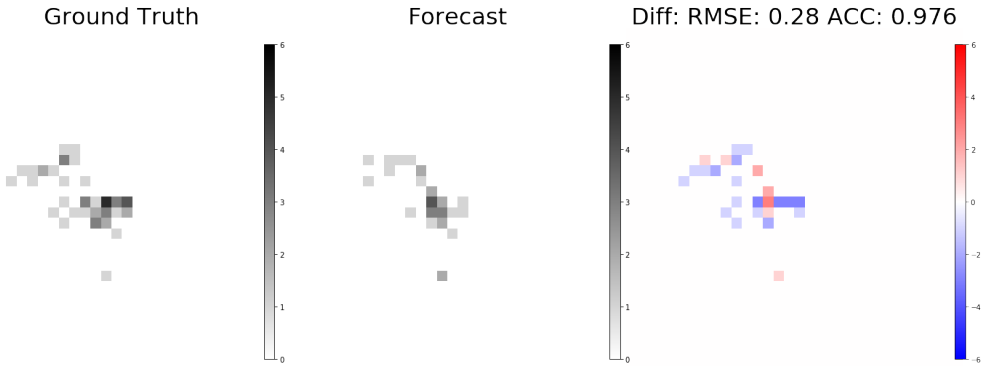
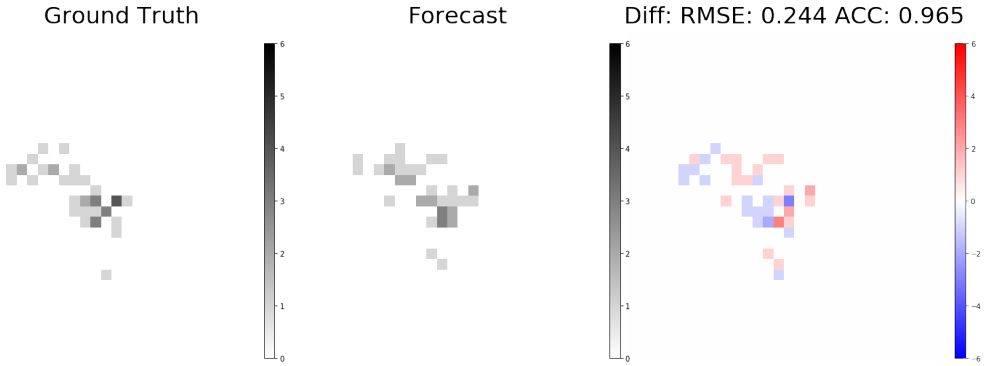


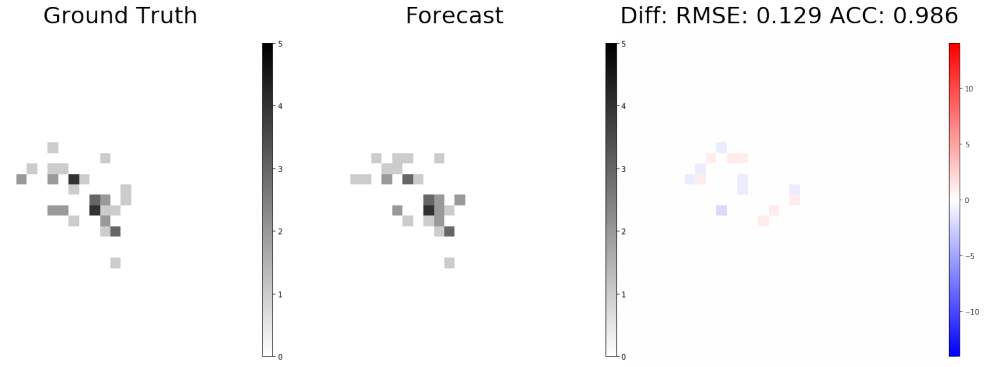
Figure 6.1: Predicted vs. exact crime spatial distribution. Panels (a), (b) plot the crime spatial distribution at 1 p.m. on Dec 19, 27, 2015, respectively. Panels (c), (d) are the predicted results without convolution layers. (e), (f) are the predicted results with convolution layers[41]. Important remark: the location is flipped.



(a) Heatmaps showing all crimes and all forecasted crimes in Los Angeles on the 19 December 2017 at 1 p.m. The AE RNN found the major crime hotspot, but still missed some of the surrounding crime neighborhoods.



(b) Heatmaps showing all crimes and all forecasted crimes in Los Angeles on the 27 December 2017 at 1 p.m. Again the AE RNN found the major hotspots, however it is much better than above as it modeled it much better.



(c) Heatmaps showing all crimes and all forecasted crimes in Los Angeles on the 7 April 2017 at 10 p.m. The AE RNN nearly forecasts the crime scenes in a perfect way. Forecast and ground truth are nearly the same.

Figure 6.2: Results of the AE RNN for the hourly Los Angeles dataset using all crimes showing the ground truth, the forecasted and the difference heatmaps.

7 Discussion

There are many possible ways to evaluate the resulting heatmaps of the different proposed systems. The most used method is the root mean squared error for two images. The loss function for the training of neural networks on images is often a mean squared error and calculates this error for the backpropagation through the network. Further, Wang et al.[40] use it as their metric comparing the generated results to the ground truth. The definition for the root mean squared error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} \tag{7.1}$$

The RMSE shows how similar the two heatmaps are. In the ideal case, the result would be zero; then the two heatmaps would be the same.

Another metric gets introduced to calculate the accuracy (ACC) of the heatmaps:

$$\text{CrimeDiff}(x, y, \epsilon = 0) = \begin{cases} 1, & \text{if } |x - y| \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{7.2}$$

$$\text{ACC} = \frac{\sum_{n=1}^{N} \text{CrimeDiff}(\hat{y}_n, y_n)}{N} \tag{7.3}$$

If the difference of the grid cells is lower or equal than a threshold, in most cases zero, the system forecasted the grid cell correctly, and the function returns one, else zero. The number of all cells then divides the sum of this function on all cells. The final result then shows how correct the whole system is on the grid heatmap and has values from zero to one with one being two identical heatmaps.

However, to bring the user into the evaluation, it is also possible to calculate difference heatmaps. These show in which areas the heatmaps correspond to the ground truth, in which they overshoot and in which they have way too less crimes predicted.

In the end, a combination of the RMSE, the ACC, and the difference heatmaps is an excellent method to compare the ground truth to the forecasts. This combination also enables to incorporate a user or an expert into the decision process. In this work, the

RSME and the ACC present the results in a brief overview. The difference heatmaps enhance the RMSE and the ACC results to show another impression of how correct or incorrect the forecasts are. In combination, the three methods show a result for every system and some other possible enhancements. It is possible to find the advantages and disadvantages of the two proposed systems and also changes, which could help to improve the results.

The training dataset consists of 90% and the test dataset, which the result uses, consists of 10% of the datasets. Further, the dataset consists only of burglary, because it is one of the most significant categories and it is one of the crimes, for which a lot of research exists and is the most wanted misconduct by the population to prevent. As the dataset sorts by age, the test dataset is the last few weeks of the different city datasets. This method corresponds to the forecasting, which the police wants to use in reality. So, to say to forecast from historical data the development of the current crime rate.

7.1 Results

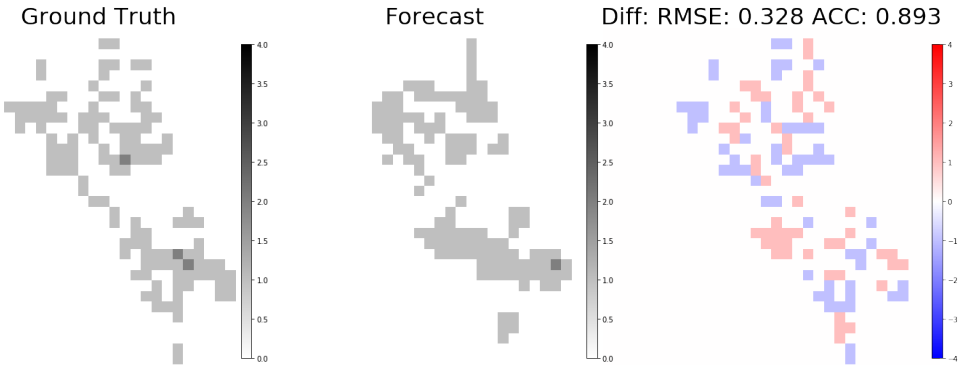
The results section is divided into three parts, containing the RNN AE, the RNN VAE, and the GAN results. The difference heatmaps, the RMSE and the ACC, present the results of the systems. The ACC directly presents how good or bad a system was on the forecasting task. The same does the RMSE, but with presenting a worse intuition than the ACC. The difference heatmaps show the trend of the forecasts if the results of the systems are too low or too high in general. These three different result evaluations provide information and description for Chicago, San Francisco, and Los Angeles. Only the RNN AE and the AC-GAN have results for all three cities, as there are some constraints and heavy training times for the RNN VAE architecture, which don't work for San Francisco.

Table 7.1 states the exact RMSE and ACC of all tested variants with the corresponding datasets. The table shortly summarizes the scores. However, the heatmaps and especially the difference heatmap present the results in a more precise way.

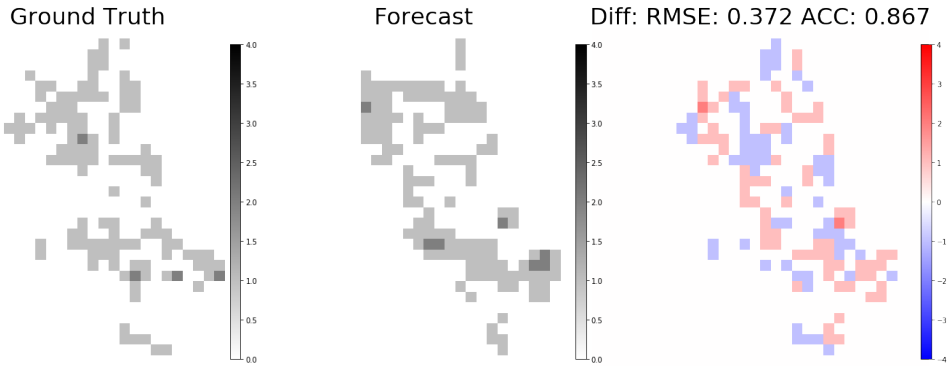
RNN AE Results

The AE RNN results show the most promising RMSE and ACC. Further, the difference heatmaps look more like they should than by the other approaches. In most cases, it has equal or better results than the other two architectures.

Chicago The Fig. 7.1 presents the results of the AE RNN for the fifth and eighth week of 2018. It provides inside into the systems and shows that it learned multiple more massive hotspots over the city. Overall, the heatmaps show a proper direction in which the forecast is developing, but are still not perfect and need further improvements. These improvements could lead to a more precise forecast and could potentially make the larger gray regions smaller and more focused.



(a) Heatmaps showing the burglary in Chicago in week 5 of 2018. It grasps quite good the general distribution, but has some minor errors everywhere.

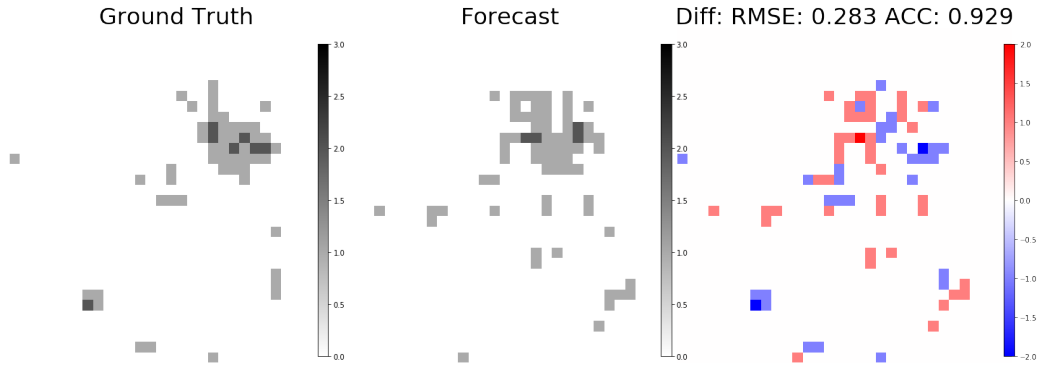


(b) Heatmaps showing the burglary in Chicago in week 8 of 2018. It further shows that regions near some hotspots are covered, however also wrong forecasted hotspots.

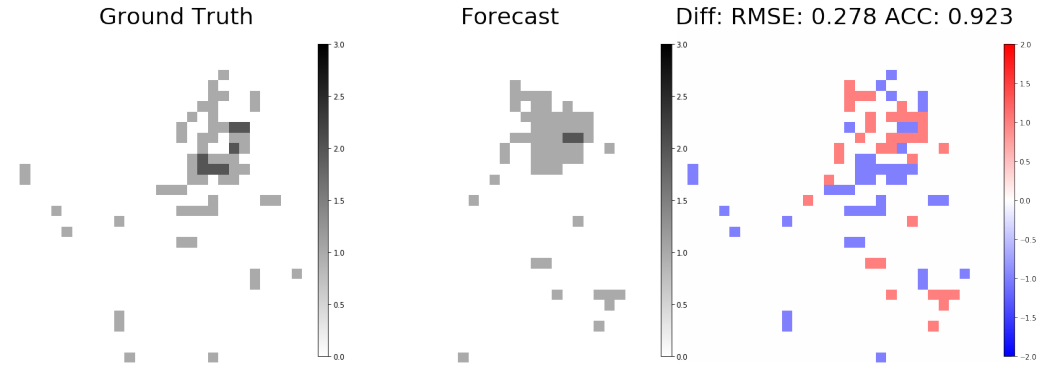
Figure 7.1: Results of the AE RNN on the Chicago crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

San Francisco San Francisco with its quite short dataset is also very sparse. Due to the city being quite large and the police being in charge of the whole area, crimes distribute over nearly the entire region. There is no city center recognizable like in Chicago or Los Angeles. However, the AE RNN has again quite good scores and identifies the area with the highest crime rate. But it is not able to identify small and

short emerging crime scenes.



(a) Heatmaps showing the burglary in San Francisco in week 9 of 2018. New hotspots in the bottom left are left out. It found old hotspots more or less.



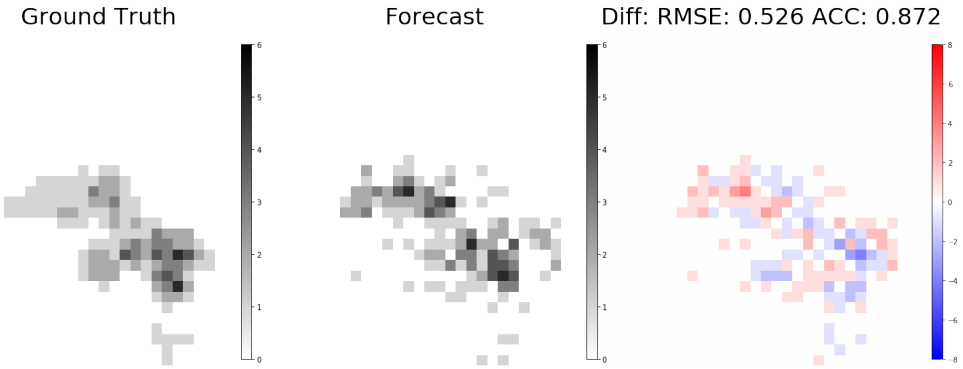
(b) Heatmaps showing the burglary in San Francisco in week 16 of 2018. Somehow manages to grasp the major area, but still too small. Bottom beneath forecast has some hotspot in the ground truth.

Figure 7.2: Results of the AE RNN on the San Francisco crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

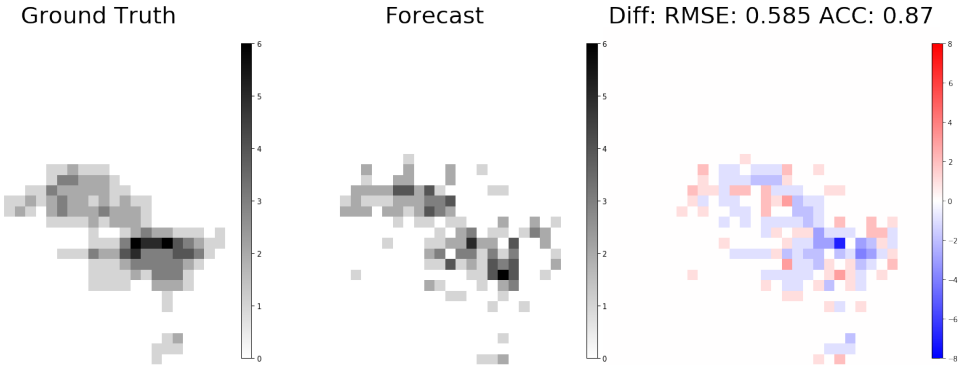
Los Angeles Los Angeles is the dataset with the most burglaries in a week and thus a particular case to forecast. It has way more crimes in the major city than the other two cities. The results further show that the forecasting is often under the reality. Especially Fig. 7.3b shows this. It even forecasted a hotspot but missed a few others around the hotspot.

VAE RNN Results

As the training time of the VAE RNN is quite long, it trains and forecasts only the Chicago crime dataset. Also, the results are worse than the AE RNN variant, so it only presents the Chicago forecasted heatmaps.



(a) Heatmaps showing the burglary in Los Angeles in week 2 of 2018. Manages to forecast hotspots quite good in the right bottom, but above is way off.



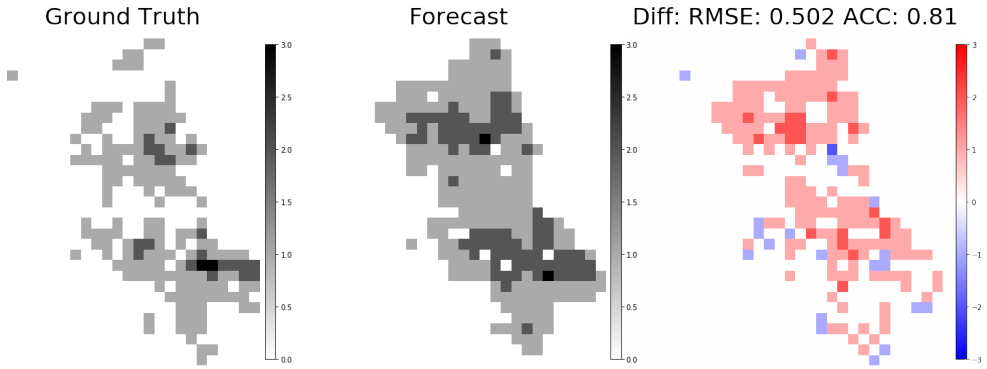
(b) Heatmaps showing the burglary in Los Angeles in week 45 of 2017. Underestimates the crime rates in a lot of cells. Couldn't grasp hotspot (blue dot).

Figure 7.3: Results of the AE RNN on the Los Angeles crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

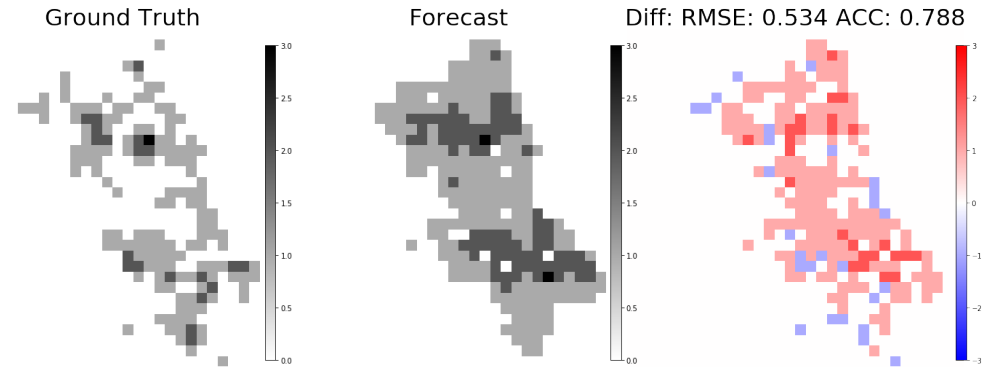
Chicago Fig. 7.4 shows the results of the VAE RNN. The results are quite impressive in correspondence to the distribution of crime in Chicago. The VAE somehow started to assume that there is an area, in which every week is a crime and this area is quite vast. The gray color depicts this area in the image. It managed to identify the central part of the city somewhat but created the assumption that there is a crime in the whole city area. The difference heatmaps show that it forecasted way more crime than there was (more red).

AC-GAN Results

The unique enhancement of the AC-GAN is the possibility to get a confidence score for the differently generated heatmaps. However, a general threshold isn't possible as the datasets differ quite heavy. For example, the threshold for Chicago and San Francisco are entirely different. This difference is also evident by looking at the results from the



(a) Heatmaps showing the burglary in Chicago in week 51 of 2017. It further shows that regions near some hotspots are covered, however also wrong forecasted hotspots.



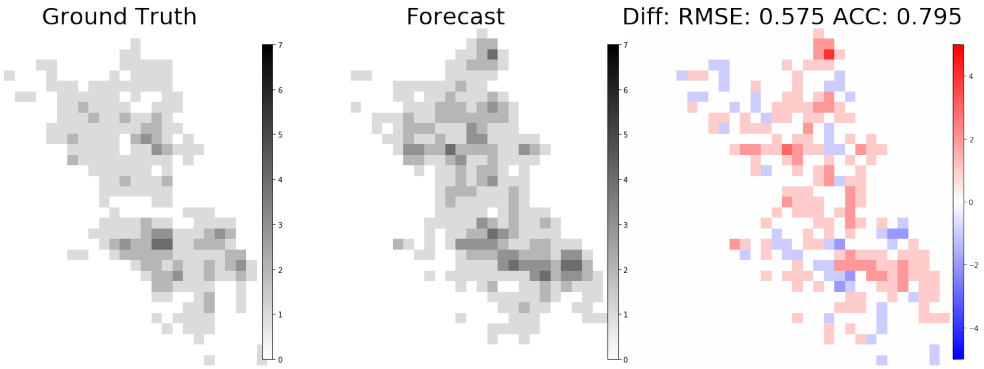
(b) Heatmaps showing the burglary in Chicago in week 2 of 2018. Same problem as in the result above.

Figure 7.4: Results of the VAE RNN on the Chicago crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

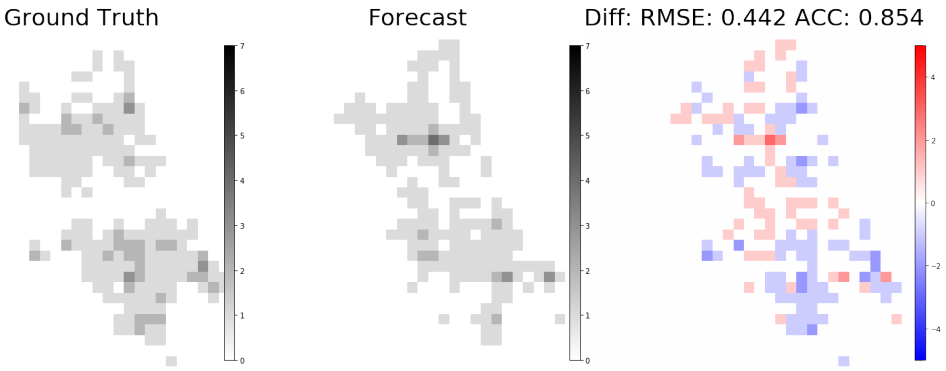
two cities.

Chicago The forecasting for Chicago achieves good results. The maximum difference on the whole test dataset is 5, which is rather small. However, it still shows improvements as the maximum amount of crimes in the test set is seven. So, there are still grid cells which are entirely wrong. Fig. 7.5 shows the results from the weeks 29 and 31 of 2017 exemplary for patterns how the AC-GAN works on this city. The difference heatmaps show only the dissimilarities between the ground truth and the forecast.

San Francisco The dataset of San Francisco is quite sparse. As there are not many burglaries and because the time span is much shorter than the other two. However, it is perfect dataset to show the performance of the systems on very sparse heatmaps with a low crime rate. Fig. 7.6 shows that the RMSE and ACC are both quite good,



(a) Heatmaps showing the burglary in Chicago in week 29 of 2017. It further shows in the difference heatmap that the forecast has more crimes (more red) than in reality happened.



(b) Heatmaps showing the burglary in Chicago in week 31 of 2017. It further shows in the difference heatmap the common forecast quality. A few too high (red) and a few too low (blue) forecasts.

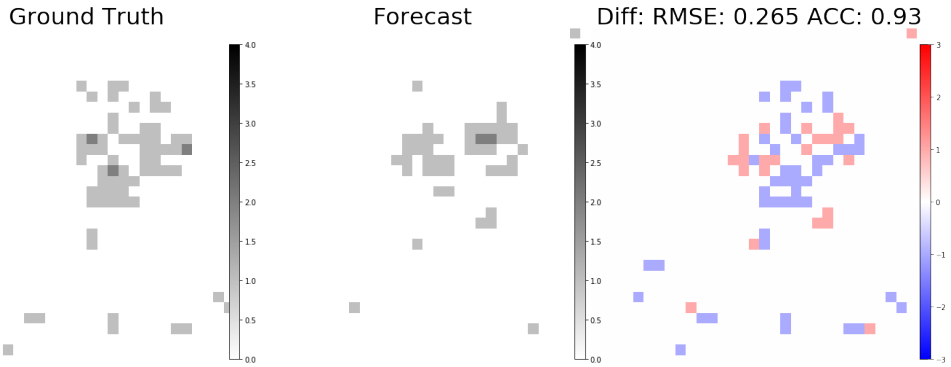
Figure 7.5: Results of the AC-GAN on the Chicago crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

but the heatmaps and especially the difference heatmap show that the forecast is quite wrong.

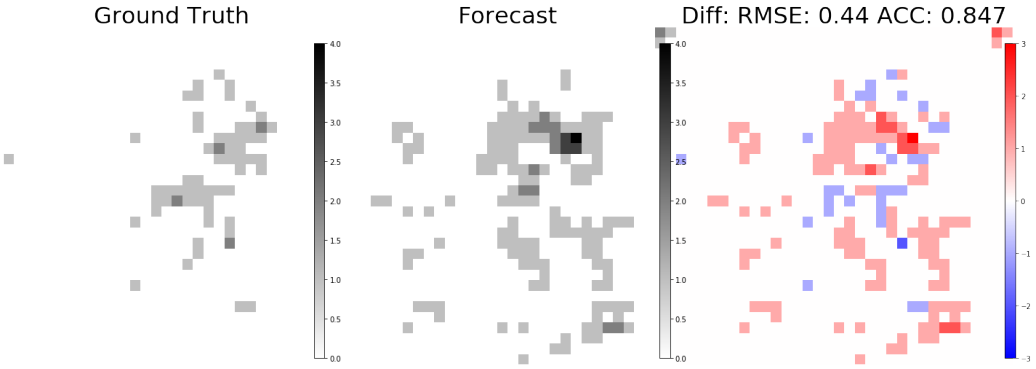
Los Angeles Los Angeles is a special case in the number of crimes committed there. Primarily, the number of burglaries is higher than for the other datasets in general. There are apparent hotspots shown in the data. The heatmaps of Fig.7.7 show it quite clearly. Also, the AC-GAN somewhat grasps the patterns of these hotspots.

7.2 Advantages

Both systems are quite easy and fast to train. Further, both of them have many possibilities to extend them as they currently incorporate only basic architectures.



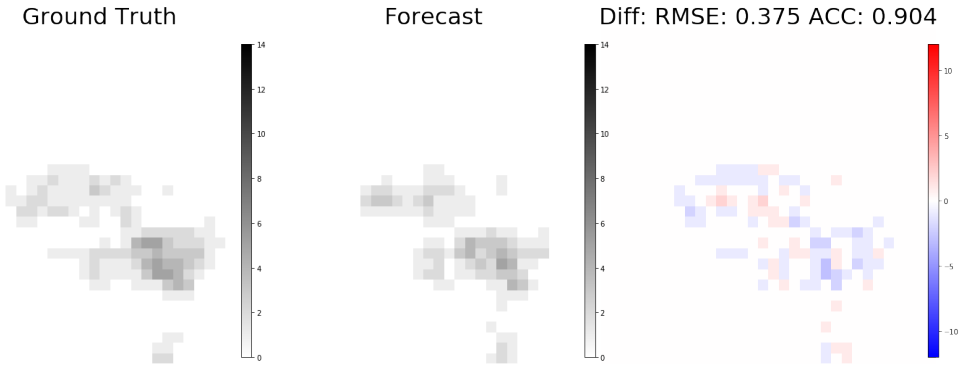
(a) Heatmaps showing the burglary in San Francisco in week 8 of 2017. The RMSE and ACC show that the forecast is very close, however, looking at the grid heatmaps this is not the case.



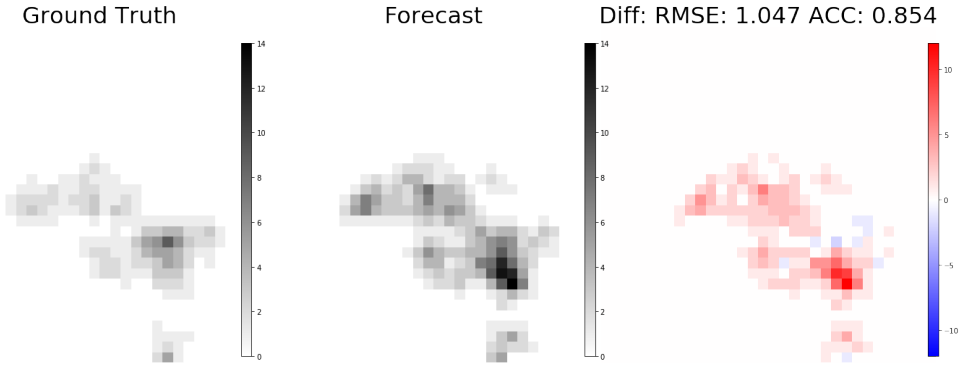
(b) Heatmaps showing the burglary in San Francisco in week 11 of 2017. The difference heatmap perfetcly shows that the forecast is quite off, but the RMSE and the ACC are quite good.

Figure 7.6: Results of the AC-GAN on the San Francisco crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

The training time of the AE RNN variant is higher due to the RNN, but it is still quite fast for not so deep networks. Besides being the most basic architecture of the three shown variants, the AE RNN outperforms the VAE RNN and the AC-GAN in Chicago and San Francisco but struggles a bit in Los Angeles. The possibilities to enhance and extend the AE RNN is nearly infinite. Every part of it is exchangeable and easy to further build on. The visual debugging helps to steer the architecture design into a proper direction.



(a) Heatmaps showing the burglary in Los Angeles in week 26 of 2017. The forecast is quite good for most regions.



(b) Heatmaps showing the burglary in Los Angeles in week 27 of 2017. The forecasting is nearly everywhere too high (red in the difference heatmap).

Figure 7.7: Results of the AC-GAN on the Los Angeles crime dataset. Left: Ground Truth, Mid: Forecast, Right: Difference Heatmap.

7.3 Disadvantages

Disadvantages of the RNN AE systems are that the RNN needs a good AE to train and learn. A bad AE doesn't reduce the dimensionality good enough to separate the input data and to be able to interpolate in the latent space to produce new data. In the best case, the AE needs to model the input data distribution correctly. However as crime data is not complete and somewhat sparse, this modeling is nearly impossible. Due to this rather lousy estimation of the input data, the latent space distribution is not entirely suitable to the need of the RNN. In this case, the VAE should improve it but somehow is not able to adjust the distribution of the latent space to a fitting distribution. It instead produces heatmaps with crime rates over the city center. This problem could emerge due to an overfitting process when the VAE still wants to decrease the general loss. But is only able to do this by building a prototype image

that fits the whole dataset and through this decreases its overall loss. Nonetheless, this prototype heatmap is neither a good description of the input data nor useful for the forecast.

The AC-GAN is challenging to train and to get it to a stable state. In most cases, it overshoots the Nash-Equilibrium and divergences afterward. It is quite tricky to find the moment in which the generator provides good heatmaps, and the discriminator has reasonable confidence in these heatmaps. Often the generator loses its ability to create good heatmaps and starts to optimize in a wrong direction.

The largest drawback is the nearly infinite possibilities of architectures. Both techniques have two different networks, which

	Chicago		San Francisco		Los Angeles	
	RMSE	ACC	RMSE	ACC	RMSE	ACC
AC-GAN	0.505	0.804	0.258	0.913	0.572	0.883
AE RNN	0.307	0.856	0.242	0.919	0.611	0.867
VAE RNN	0.519	0.770	-	-	-	-
AE RNN CNN	0.428	0.804	-	-	-	-
VAE RNN CNN	0.536	0.816	-	-	-	-

Table 7.1: Results in RMSE and ACC for the AC-GAN, the AE RNN CNN, the VAE RNN CNN, the AE RNN and the VAE RNN.

8 Conclusion

This work answers the question *"How to forecast crime using deep learning techniques?"* using heatmaps in two different ways. Using the date as a method to get a forecast and applying a sequence of the last few weeks to forecast the next week.

Visual Debugging The visual debugging shows that taking the visible difference between the forecast output and the ground truth into account improves the steering in a direction in which the neural network produces satisfactory results. It enables a more natural visual comparison of the result and what it should be. This debugging process helps to introduce more capacity to the network to learn more and don't generalize too much or do the exact opposite because the neural network overfitted. If a network shows even after the first epochs a massive difference in the visual debugging, it is highly advisable to change the architecture and rethink some of the decisions made for the architecture. This method shows that a visual component supports the process of creating and training neural networks and that involving the user a bit more into the training process leads to better results.

Date as input The first system, the proposed AC-GAN, has as seen in the discussion in most cases a higher RMSE and a lower ACC than the other method. It somewhat generalizes the problem of crime forecasting to a classification problem. The images of the week of the year build classes. This collection process means that the first week of the year 2001 for example for Chicago is in the same class as the one from 2017. In some cases, this is not a wanted variant as it assumes that these two things are not connected. However, the things are connected in space and in time. The time is later but still, for example, the first week of the year and has a relationship from the one data sample to the other one. A lot of things change during time, especially hotspots differ and move, but there could be still some hotspots, which emerge periodically during this time. These hotspots lead the forecasting system to find and learn patterns for this time and to use these patterns as a forecast for the next year. So, this concludes with the fact that the AC-GAN system learns representatives for the different classes and

varies them with the noise input vector. That's also why the scores for this method are lower than for the other one. Small changing representatives don't model the actual situation as good as a sequence, which incorporates the previous weeks.

Sequences as input The second system doesn't use the date as an input but implicitly does this by getting the time series of heatmaps as input. This technique enhances the data by adding a temporal component, which connects each of the heatmaps. It further incorporates a development indirectly over time as using the data as a single time series shows an advancement. Learning this change over time with a sliding window approach should teach the neural network, that there is a connection between the last few weeks and the upcoming one. It should implicitly implement the "near-repeat victimization"[8] as long it is observable in the data. If not it should learn patterns that are also important for the crime forecasting, which are maybe not even discovered by humans, because the abstraction is not something a human would find at first sight. However, it could also learn patterns, which don't correspond to theory and are in some cases just random patterns, which occur one time, but improve the result enough to learn them. In the end, a sequential input is far better for the forecasting mechanism as it implicitly uses different crime forecasting theories. Further, an AE can reduce the data enough to train and use a RNN faster and more efficient. And it presents a method to not only use on heatmaps over time but on time series in general. This approach is then not only able to forecast but also to reproduce the trained time series as the modeling process should learn the whole time series.

The two proposed systems Using sequences to describe and forecast the data is a better way than to sort the different heatmaps into classes. This technique leads to a better result and a more stable neural network. The determinism is in the case of the sequence model another factor, which is useful to have. It enables to have a static output, which is reproducible. However, it also limits the forecasting to be the same every time for the forecasted week.

In general All in all, crime forecasting is another area, where deep learning techniques can produce top-notch results and state-of-the-art forecasts. Nearly all possible models and types lead to exciting results and increase the pool of possibilities.

Wang et al.[40] show that it is possible with ST-ResNets incorporating CNNs and fully-connected layers. This architecture can forecast with outstanding results.

Other experiments from for example Stec et al.[42] show promising results with

standard CNN and RNN structures. These results present further that the idea of using deep learning for crime forecasting is an excellent way to go.

The proposed AC-GAN shows that forecasting is possible using GAN systems and using the generator of it as a forecast model. The generator impresses with the ability to learn how to generate plausible heatmaps. Also, the usage of the discriminator to verify the generated heatmaps is another useful option this architecture type presents.

The proposed AE RNN architecture impressively presents whats possible with simple methods put together. The AE compresses the data as much as possible to let the RNN learn only the most essential features. It, therefore, helps as a preprocessing before the RNN and thus reduce the data onto a dimensionality which the RNN can learn quicker.

As a last drawback, which is critical especially for the task of crime forecasting, is the fact that the neural network and thus deep learning is difficult to interpret and explain. Both interpretability and explainability are lacking in neural networks, but predictive policing needs reasoning behind the decision. This challenge is enormous as it lowers the usability of deep learning in this and many other areas. But there is also a big community, which now focuses on the explainability of neural networks and slowly but surely has first findings and methods to do so.

9 Future Work and Research Opportunities

The proposed systems build a baseline for crime forecasting with deep learning methods and heatmap generation. Especially, as forecasting in most cases is not done using unsupervised and semi-supervised generative models, the systems show what is possible using a form of artificial intelligence to forecast crime. However, there are many possibilities to improve the systems and methods.

Improve Architecture

One way to improve the systems would be to revise the architecture itself further and add further extensions and layers to it. It is always possible to change the general layers, by adding or removing fully connected or convolution layers. Further, changing the parameters of the different layers like using smaller or larger kernels for convolution layers or more or fewer neurons for a fully connected layer. However, there are also some other improvements, which are possible to implement, to increase the accuracy and decrease the error.

Inception Layers A possible enhancement is using inception layers[61]. These employ parallel convolution layers with different kernel sizes. In the recommended case, a one on one, a three on three, a five on five kernel convolution layer and a pooling layer. The network calculates these four computations in parallel, and afterward, collects and concatenates the results of these operations. The basic idea behind these layers is to let the neural network learn the best weights or better most possible weights and automatically select the most useful features from these possibilities. Further, due to some tricks, a one on one convolution layer can be added before the larger kernels to do a dimension reduction and reduce the channels for the next layer. This method makes it possible to increase the number of neurons and layers in the network. However, the parallelization increases the computational cost of this layer and the network in

general, because instead of one component the system now needs, like in the case above, four times the number of computations. This technique decreased the error rate in the ImageNet challenge[26] and won the authors the challenge. It is one of the advisable methods while using images as input data, as in most cases it improves the accuracy.

Residual Layers Another way to increase the accuracy and decrease the error could be to add residual blocks[62]. These use a highway identity function around a block of layers and connect with the output of this block with an activation function. The layers in the block can be any neural network layer. In most cases, they consist of convolution, pooling, and fully connected layers. However, others are also possible. This method helps to create deeper architectures and thus deeper networks. It somewhat shifts the problem of the vanishing gradient in deep networks further to the end as the identity function always introduces a higher value and thus the gradient in most cases is still above zero, and the system is still able to learn. However, it is not a solution to completely remove the vanishing gradient problem and the problem of not learning layers. Their approach was also one of the winners of the ImageNet challenge[26] and established itself as a reliable method to create and train deep networks. For example, ST-ResNet[39] uses this approach to improve and construct their network. As the ST-ResNet is the base for the crime forecasting approach from Wang et al.[40], residual layers improve the results of movement and crime forecasting. Also, through deeper networks, the training time increases as there are far more computations to calculate in a deeper one.

Conditions Another possible option is to add further conditions to the AC-GAN or C-GAN. Instead of only providing the week of the year, the previous heatmaps could be further conditions put into the GAN. This extension would increase the information the GAN has to learn. It would even be possible to map RNNs and sequences into it. The GAN and especially the generator could learn to use the sequence as a previous knowledge to forecast the future.

Improve Parameters

Another possibility to decrease the error is the question of setting different parameters, as there are more parameters around the bias and weight parameters of the neural network. The latent space size, for example, is one of them. In most cases, the

literature decides the value of these parameters, however, to get the best possible result, these variables need a lot of tweaking.

Latent Space The latent space, in this case the projected space of the AEs, can have an arbitrary dimensional size. It is difficult to find a suitable size as everything from one to the size of the last layer minus one is possible. Primarily, due to the training time, either a heuristic to find the right size needs to be used or taking the proposed dimensions of the literature is the way to find a proper size. In the case of this work, the training uses different sizes. However, these all come from the corresponding AE papers. But it should be possible to find a size in which the construction error is at a minimum and where it is possible to model different interpolations into the space to see new heatmaps. This smaller space would help to make easier forecasts for the AE as the possible space for the estimates would also decrease.

Noise Space The latent space corresponds to the AE; the noise space corresponds to the GAN and models the input data distribution. However, the size of this space is essential as larger or smaller ones decide the mapping of the distribution. A larger one enables to fit more data into, but it also makes it harder to generate plausible heatmaps. A smaller one maps the space in a smaller, however, could also lead to shrinking the generation distribution. Small tests with changing the noise space to ten, didn't show any improvements, but also didn't decrease the accuracy.

Sequence Length Another parameter, which needs to further tests, is the sequence length the RNN learns in the RNN AE system. Currently, the sequence length is four data samples long, and the system forecasts forth data sample in the sequence. This length shifts a month through the data and incorporates the time, for example, the near-repeat victimization mentions, which it takes to move the crime probability back at the normal state. However, it is possible to limit the sequence to two or three to include fewer weeks and to limit the amount of information the RNN gets to train and later needs to forecast. It is also possible to enlarge the sequence to use more historical data and to weight this data heavier into a forecast.

Transfer Learning

A further possibility to implement is to use transfer learning. This method describes the use of an already trained model for the task. The trained model is then just fine-tuned onto the new data. Transfer learning showed a lot of success in different areas

and a lot of applications use fine-tuned models to accomplish their tasks. Notably, the VGG architecture trained on the ImageNet challenge is prominent for a lot of use cases as it is easily changeable to the needed output. It is also one of the winners of the ImageNet challenge. A pre-trained AE model could improve and reduce the training time.

External Data

One of the most promising additions, all systems could benefit from is the usage of external data. This external data means to incorporate weather, census or traffic data. Works like Stec et al.[42] show that external information increases the accuracy and decreases the error. However, finding suitable data for this task is quite difficult. Data like census data is either year old or only sold for an enormous price. Also, in most cases, this data is an aggregation of data, which would be openly available, but somewhat difficult to collect and to process. A bias could as well be introduced with the census data, as regions with low incomes have a high probability of having crimes. A more massive police force in regions like that would lead to an increase in crimes there because the police would find crimes even if they are only minor ones.

Other data like the weather is more straightforward to collect and to incorporate. However, the improvements and the correlation between crime and weather is much more difficult to explain and to observe in a model. It is unclear if the weather improves the model or if the randomness of the weather integrates a random factor into the model, which prevents it from overfitting on specific patterns and therefore enhances the results.

Extensions to the RNN AE

The RNN AE is easily one of the most promising architectures to use on many different tasks. The dimension reduction with the sequence learning can help a lot of sequence prediction tasks and many more. It further enables the RNN to learn much faster as the input is decreased in a much better fashion because it is easily possible to recreate the dimension reduced data back to its previous state with the AE. Also using one-dimensional convolution layers before RNN layers could generate better features for them. Other connections between the RNN cells could make them even more robust to unseen circumstances.

Even an exchange of the RNN with a GAN would be possible. This exchange could increase the generative power of the system much more and could lead to a generator,

which can find even more impressive images. A drawback against this component would be the loss of the sequence property and the even more considerable training time. However, due to be able to implement further conditions into the GAN, it could potentially lead to a way to fully automate the input of a label and get a forecast by this label.

A way to train the AE and the RNN simultaneously would decrease the training time by a lot and could potentially steer the RNN into a faster convergence. Because the RNN would learn with further errors and thus would have more ways to know how the forecasting results for erroneous input looks. A somewhat noisy input leads to better training due to more information to learn from in general.

Explainability

Last, the explainability of the decision making is a crucial factor for crime forecasting, but also one of the most difficult tasks for neural networks to do. This challenge is an area, which needs more focus and research. There are many tasks a neural network could do, however, due to the limitation of the explainability is not used. Such topics include the medical domain and the public domain in general, in which decisions of any kind have to be explained and understood. The visual debugging shows a first starting point for neural networks as it combines the results with the ground truth and enables steering of the architecture in a prosperous direction. However, this method only introduces a data-driven approach. A combination with another model-based approach would undoubtedly produce better network architectures in a faster time. A model approach would further introduce a way to use other data and be able to steer the architecture into a successful and accurate one for this data. It would make a data generalization more easily. Especially, visualizing errors in the network would help in implementing this approach. If it is possible to trace back the error of the network to one of the layers or even to one or a few neurons, it would be an excellent way and starting point to change. Visualization would help this process a lot and is not done that much currently. A good starting point is the survey from Hohman et al.[63]. However, it also shows the gap in visual analytics for this topic. This lack of incorporating visualizations into the explainability and interpretability displays a research gap, which gets bigger and bigger with every new deep learning advancement.

Appendix

List of Figures

List of Tables

Bibliography

[1] Yong Jei Lee and Wilpen Gorr. "Chronic and Temporary Crime Hot Spots". In: November (2017) (cit. on pp. 3, 9, 10, 19, 21, 22).

[2] M. B. Short, M. R. D'Orsogna, V. B. Pasour, G. E. Tita, P. J. Brantingham, A. L. Bertozzi, and L. B. Chayes. "A Statistical Model Of Criminal Behavior". In: *Math. Model. Methods Appl. Sci.* 18.supp01 (Aug. 2008), pp. 1249–1267. DOI: 10.1142/S0218202508003029 (cit. on pp. 3, 8).

[3] Lawrence W. Sherman, Patrick R. Gartin, and Michael E. Buerger. "Hot Spots Of Predatory Crime: Routine Activities And The Criminology Of Place". In: *Criminology* 27.1 (Feb. 1989), pp. 27–56. DOI: 10.1111/j.1745-9125.1989.tb00862.x (cit. on pp. 3, 8, 17, 21).

[4] Wilpen Gorr and Richard Harries. "Introduction to crime forecasting". In: *Int. J. Forecast.* 19.4 (2003), pp. 551–555. DOI: 10.1016/S0169-2070(03)00089-X (cit. on pp. 3, 19).

[5] Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. "Automatic differentiation in PyTorch". In: *Adv. Neural Inf. Process. Syst. 30* Nips (2017), pp. 1–4 (cit. on p. 5).

[6] Lawrence E. Cohen and Marcus Felson. "Social Change and Crime Rate Trends: A Routine Activity Approach". In: *Am. Sociol. Rev.* 44.4 (Aug. 1979), p. 588. DOI: 10.2307/2094589 (cit. on pp. 7, 8, 10).

[7] Farrell G. and K Pease. *Once bitten, twice bitten: Repeat victimisation and its implications for crime prevention*. 46. 1993, p. 38 (cit. on pp. 8, 12, 17).

[8] Ken Pease. *Repeat Victimisation: Taking Stock*. 1998, p. 48 (cit. on pp. 8, 10, 17, 66).

[9] Walter L Perry, Brian McInnes, Carter C Price, Susan C Smith, and John S Hollywood. *Predictive Policing: The Role of Crime Forecasting in Law Enforcement Operations*. 2013, p. 189. DOI: 10.1214/07-EJS057. arXiv: 0706.1401 (cit. on p. 9).

[10] Manfred Rolfes. “Predictive Policing : Beobachtungen Und Reflexionen Zur Einführung Und Etablierung Einer Vorhersagenden Polizeiarbeit”. In: (2017), pp. 52–76 (cit. on p. 9).

[11] Daniella Pollich and Felix Bode. *Predictive Policing - Zur Notwendigkeit eines sozialwissenschaftlich basierten Vorgehens* (cit. on p. 9).

[12] Felix Bode, Florian Stoffel, and Daniel Keim. “Variabilität und Validität von Qualitätsmetriken im Bereich von Predictive Policing”. In: (2017), pp. 1–14 (cit. on p. 10).

[13] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1970 (cit. on p. 11).

[14] Peng Chen, Hongyong Yuan, and Xueming Shu. “Forecasting crime using the ARIMA model”. In: *Proc. - 5th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2008* 5.November 2008 (2008), pp. 627–630. DOI: 10.1109/FSKD.2008.222 (cit. on p. 11).

[15] Seth Flaxman, Michael Chirico, Pau Pereira, and Charles Loeffler. “Scalable high-resolution forecasting of sparse spatiotemporal events with kernel methods: a winning solution to the NIJ "Real-Time Crime Forecasting Challenge"”. In: (2018), pp. 1–35. arXiv: 1801.02858 (cit. on p. 11).

[16] Chung-Hsien Yu, Max W. Ward, Melissa Morabito, and Wei Ding. “Crime Forecasting Using Data Mining Techniques”. In: *2011 IEEE 11th Int. Conf. Data Min. Work.* (2011), pp. 779–786. DOI: 10.1109/ICDMW.2011.56 (cit. on p. 12).

[17] F. Rosenblatt. *The Perceptron - A Perceiving and Recognizing Automaton*. 1957. DOI: 85-460-1 (cit. on p. 13).

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. arXiv: arXiv:1011.1669v3 (cit. on p. 13).

[19] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. DOI: 10.1007/BF00994018. arXiv: arXiv:1011.1669v3 (cit. on p. 13).

[20] Leo Breiman. “Random forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324 (cit. on p. 13).

[21] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. arXiv: 1206.2944 (cit. on pp. 13, 28, 34, 35).

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: (2014), pp. 1–9. DOI: 10.1007/s10107-014-0839-0. arXiv: 1409.3215 (cit. on pp. 13, 25).

[23] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (2014). DOI: 10.3115/v1/D14-1179. arXiv: 1406.1078 (cit. on pp. 13, 25).

[24] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech Recognition with Deep Recurrent Neural Networks". In: 3 (2013). DOI: 10.1109/ICASSP.2013.6638947. arXiv: 1303.5778 (cit. on p. 13).

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (Feb. 1998), pp. 2278–2324. DOI: 10.1109/5.726791. arXiv: 1102.0183 (cit. on p. 13).

[26] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conf. Comput. Vis. Pattern Recognit.* IEEE, June 2009, pp. 248–255. DOI: 10.1109/CVPRW.2009.5206848 (cit. on pp. 13, 70).

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Adv. Neural Inf. Process. Syst.* (2012), pp. 1–9 (cit. on p. 13).

[28] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol". In: *J. Mach. Learn. Res.* 11 (2010), pp. 3371–3408. DOI: 10.1111/1467-8535.00290. arXiv: 0-387-31073-8 (cit. on pp. 14, 25).

[29] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: Ml (2013), pp. 1–14. DOI: 10.1051/0004-6361/201527329. arXiv: 1312.6114 (cit. on pp. 14, 30, 31, 36).

[30] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Networks". In: (2014), pp. 1–9. DOI: 10.1001/jamainternmed.2016.8245. arXiv: 1406.2661 (cit. on pp. 14, 39, 40).

[31] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: (2014), pp. 1–7. arXiv: 1411.1784 (cit. on pp. 14, 42).

[32] Wei Li, Melvin Gauci, and Roderich Gross. "A coevolutionary approach to learn animal behavior through controlled interaction". In: *Proceeding fifteenth Annu. Conf. Genet. Evol. Comput. Conf. - GECCO '13* (2013), p. 223. DOI: 10.1145/2463372.2465801 (cit. on pp. 14, 39).

[33] Jost Tobias Springenberg. "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks". In: 2009 (2015), pp. 1–20. arXiv: 1511.06390 (cit. on p. 14).

[34] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis With Auxiliary Classifier GANs". In: (2016). arXiv: 1610.09585 (cit. on pp. 14, 43).

[35] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks". In: *Proc. IEEE Int. Conf. Comput. Vis.* 2017-Octob (2017), pp. 5908–5916. DOI: 10.1109/ICCV.2017.629. arXiv: 1612.03242 (cit. on pp. 14, 39).

[36] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. "Pose Guided Person Image Generation". In: Nips (2017), pp. 1–11. arXiv: 1705.09368 (cit. on pp. 14, 39).

[37] Amir Ghaderi, Borhan M. Sanandaji, and Faezeh Ghaderi. "Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting". In: ii (2017). DOI: 10.1049/el.2018.0336. arXiv: 1707.08110 (cit. on p. 15).

[38] Xingyi Cheng, Ruiqing Zhang, Jie Zhou, and Wei Xu. "DeepTransport: Learning Spatial-Temporal Dependency for Traffic Condition Forecasting". In: (2017), arXiv preprint arXiv:1709.09585. arXiv: 1709.09585 (cit. on p. 15).

[39] Junbo Zhang, Yu Zheng, and Dekang Qi. "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction". In: (2016). arXiv: 1610.00081 (cit. on pp. 16, 49, 70).

[40] Bao Wang, Duo Zhang, Duanhao Zhang, P. Jeffery Brantingham, and Andrea L. Bertozzi. "Deep Learning for Real Time Crime Forecasting". In: (2017), pp. 33–36. arXiv: 1707.03340 (cit. on pp. 16, 17, 49, 55, 66, 70).

[41] Bao Wang, Penghang Yin, Andrea L. Bertozzi, P. Jeffrey Brantingham, Stanley J. Osher, and Jack Xin. "Deep Learning for Real-Time Crime Forecasting and its Ternarization". In: (2017), pp. 1–14. arXiv: 1711.08833 (cit. on pp. 16, 17, 49–52).

[42] Alexander Stec and Diego Klabjan. "Forecasting Crime with Deep Learning". In: (June 2018). arXiv: 1806.01486 (cit. on pp. 17, 66, 72).

[43] Jonathan J. Corcoran, Ian D. Wilson, and J. Andrew Ware. "Predicting the geo-temporal variations of crime and disorder". In: *Int. J. Forecast.* 19.4 (2003), pp. 623–634. DOI: 10.1016/S0169-2070(03)00095-5 (cit. on p. 18).

[44] R Maciejewski, S Rudolph, R Hafen, A. Abusalah, M Yakout, M Ouzzani, W.S. Cleveland, S.J. Grannis, and D.S. Ebert. "A Visual Analytics Approach to Understanding Spatiotemporal Hotspots". In: *IEEE Trans. Vis. Comput. Graph.* 16.2 (Mar. 2010), pp. 205–220. DOI: 10.1109/TVCG.2009.100 (cit. on p. 22).

[45] Lovedeep Gondara. "Medical image denoising using convolutional denoising autoencoders". In: (2016). DOI: 10.1109/ICDMW.2016.0041. arXiv: 1608.04667 (cit. on p. 25).

[46] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. "StyleBank: An explicit representation for neural image style transfer". In: *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 2770–2779. DOI: 10.1109/CVPR.2017.296. arXiv: 1703.09210 (cit. on p. 25).

[47] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: (2015), pp. 3–7. DOI: 10.1167/16.12.326. arXiv: 1508.06576 (cit. on p. 25).

[48] Cheng Yuan Liou, Wei Chen Cheng, Jiun Wei Liou, and Daw Ran Liou. "Autoencoder for words". In: *Neurocomputing* 139 (2014), pp. 84–96. DOI: 10.1016/j.neucom.2013.09.055 (cit. on p. 25).

[49] Alireza Makhzani and Brendan Frey. "k-Sparse Autoencoders". In: (2013). arXiv: 1312.5663 (cit. on pp. 25, 26).

[50] Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. "Semi-Supervised Learning for Neural Machine Translation". In: (2016). arXiv: 1606.04596 (cit. on p. 25).

[51] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *Proc. 30 th Int. Conf. Mach. Learn.* 28 (2013), p. 6 (cit. on pp. 27, 33, 34).

[52] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Ann. Math. Stat.* 22.1 (Mar. 1951), pp. 79–86. DOI: 10.1214/aoms/1177729694 (cit. on p. 31).

[53] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proc. 27th Int. Conf. Mach. Learn.* 3 (2010), pp. 807–814. DOI: 10.1.1.165.6419. arXiv: 1111.6189v1 (cit. on pp. 33, 34).

[54] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. "Deconvolutional networks". In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (2010), pp. 2528–2535. DOI: 10.1109/CVPR.2010.5539957. arXiv: 1302.1700 (cit. on pp. 34, 35).

[55] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Int. Conf. Mach. Learn.* Lille, France, July 2015, pp. 448–456. arXiv: 1502.03167 (cit. on p. 35).

[56] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15 (2014). DOI: 10.1214/12-AOS1000. arXiv: 1102.4807 (cit. on p. 35).

[57] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014), pp. 1–15. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. arXiv: 1412.6980 (cit. on pp. 37, 47).

[58] Jürgen Schmidhuber. "Learning Factorial Codes by Predictability Minimization". In: *Neural Comput.* 4.6 (Nov. 1992), pp. 863–879. DOI: 10.1162/neco.1992.4.6.863 (cit. on p. 39).

[59] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: (2015), pp. 1–16. DOI: 10.1051/0004-6361/201527329. arXiv: 1511.06434 (cit. on pp. 40, 41, 47).

[60] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. "Conditional Image Generation with PixelCNN Decoders". In: (2016). arXiv: 1606.05328 (cit. on p. 42).

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 07-12-June (2015), pp. 1–9. DOI: 10.1109/CVPR.2015.7298594. arXiv: 1409.4842 (cit. on p. 69).

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *arXiv Prepr. arXiv1512.03385v1* 7.3 (Dec. 2015). DOI: 10.3389/fpsyg.2013.00124. arXiv: 1512.03385 (cit. on p. 70).

[63] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers". In: (2018), pp. 1–20. arXiv: 1801.06889 (cit. on p. 73).