Personalized Automation for Air Traffic Control using Convolutional Neural Networks

S.J. van Rooijen

18 January 2019

Delft University of Technology

TU Delft

Personalized Automation for Air Traffic Control using Convolutional Neural Networks

by

S.J. van Rooijen

to obtain the degree of Master of Science
at the Faculty of Aerospace Engineering,
Department of Control & Simulation,
Delft University of Technology.

Student number:	4135091	
Project duration:	February 8, 2018 – January 18, 2019	
Thesis committee:	Prof. dr. ir. J. M. Hoekstra	TU Delft
	Dr. ir. E. van Kampen	TU Delft, supervisor
	Dr. ir. C. Borst	TU Delft, supervisor
	Dr. ir. J. Ellerbroek	TU Delft, supervisor
	Dr. ir. E. Mooij	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

"The real danger is not that computers
will begin to think like men,
but that men will begin to think like computers"

– Sydney J. Harris

Contents

List of Figures

List of Tables

Introduction

Air Traffic Controller (ATCo) workload is a major limiting factor on airspace capacity growth in the coming years (Majumdar et al., 2004). Concepts such as free flight are proposed for autonomous self-separation, but full implementation is not expected to be feasible in the near-future (Hoekstra et al., 2002). To bridge this gap, novel technologies should be introduced to assist ATCos with traffic separation to reduce controller workload (SESAR Consortium, 2007).

Specifically, Conflict Detection & Resolution (CD&R) is seen as one of the main tasks of an ATCo (EUROCONTROL, 1996) and is therefore the focus of this research. CD&R is a control problem in which the ATCo prevents conflicts between aircraft by providing flight plan adjustments in terms of altitude, heading or speed. During CD&R, automation could function as a decision aid to ultimately alleviate ATCo workload and increase safety & performance. However, as multiple alternative actions per situation are possible, proposed resolutions by automation do not always coincide with human strategy (Prevot et al., 2012), which hinders automation adoption. Therefore, *acceptance* remains an issue and is considered one of the main obstacles in the successful introduction of novel automation (Westin et al., 2016).

A proposed method to increase trust and acceptance of automation is to make the automation *strategic conformal* (Hilburn et al., 2014). Westin et al. (2016) argue that tailoring automation to an individual's strategy could increase acceptance. To create a predictive model for human actions, machine learning is an apparent means as it is able to adapt to different controller preferences without full knowledge on the underlying decision-making dynamics (Sutton et al., 1992).

A proof-of-concept by Regtuit et al. (2018) shows that machine learning techniques are able to identify and replicate simple conflict resolution strategies from artificially created data. However, when human controller data is used, two questions arise: One, are controllers sufficiently *consistent* to base individual-sensitive automation on? And two, is a group of controllers adequately *heterogeneous* in strategy for personalized automation to be beneficial? This research aims to answer these questions by creating a machine learning model able to predict ATCo commands based on a traffic situation.

As input for the learning algorithm, this research proposes the Solution Space Diagram (SSD)(Mercado-Velasco et al., 2010), also known as velocity obstacle diagram (Fiorini & Shiller, 1998). The SSD displays the solution space when detecting & resolving conflicts to support the ATCo. It is hypothesized that the SSD contains sufficient information concerning an air traffic situation to make an informed decision (Jenie et al., 2015; Mercado-Velasco et al., 2010; Van Dam et al., 2004). Learning from an image is advantageous because it eliminates the need for manually designing features based on heuristics and assumptions. A machine learning technique which is especially well-suited for this is a *convolutional neural network* (LeCun et al., 1998).

A preliminary analysis using artificially generated data was performed to test the SSD as a machine learning feature in a convolutional neural network. In addition, a human-in-the-loop experiment is devised to test whether controllers are consistent enough to base personalized automation on. An ATC simulator is used to create a datasets of corresponding SSD images and ATCo commands from a 12-participant population with two skill levels. These datasets are used to train individual-sensitive predictive models using a supervised learning algorithm. Finally, individual model performance is compared to controller consistency and a comparison with general group-based models is made to test strategy heterogeneity of the group.

The end-result provides insight in how an ATC Conflict Detection & Resolution (CD&R) task can be automated in a strategic conformal way using the SSD. By making this automation individual-sensitive, it is hypothesized that acceptance, and thus system-use, will increase. This could ultimately lead to a more reliable, safer airspace, where Air Traffic Controller (ATCo) workload is reduced.

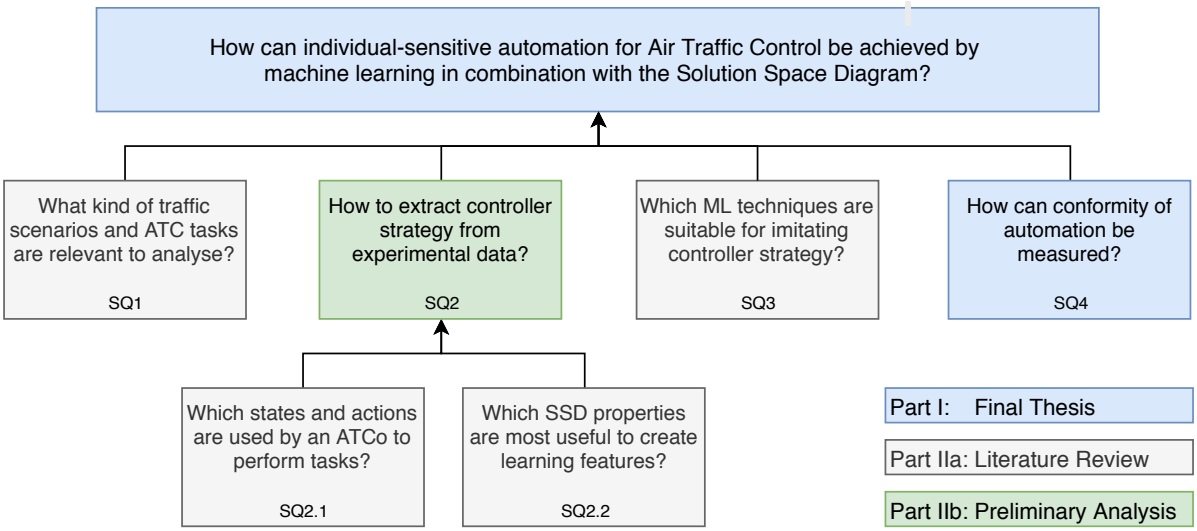


Figure 1.1: Research Question Breakdown. The Literature Review and Preliminary Analysis are both included in Part II of this document. The Final Thesis is comprised of the scientific paper in Part I and the Appendices in Part III.

1.1. Problem Statement

The objective of this thesis is to achieve strategic conformal automation for Air Traffic Control, which is linked to the research question:

> *"How can individual-sensitive automation for Air Traffic Control be achieved by using machine learning in combination with the Solution Space Diagram?"*

Four sub-questions are devised to obtain an answer on this question, as summarized in the work break-down structure in Figure 1.1. Each sub-question is detailed below:

SQ1 What kind of traffic scenarios and ATC tasks are relevant to analyze? – This considers the scope of the scenarios to be analyzed: the number of aircraft in the airspace, the number of aircraft in conflict, whether to include altitude (2D or 3D), and the complexity of the sector.

SQ2 How to extract controller strategy from experimental data? – The most relevant states of the scenario have to be identified to create 'learning features'.

(a) Which states and actions are used by an ATCo to solve tasks?

(b) Which SSD properties are most useful to create learning features from?

SQ3 Which ML techniques are suitable for imitating controller strategy? – Given the experimental data containing controller strategy, which algorithms can be trained?

SQ4 How can conformity of automation be measured? – In order to measure the effectiveness of the automation, metrics have to be defined based on e.g. accuracy, efficiency or conformity.

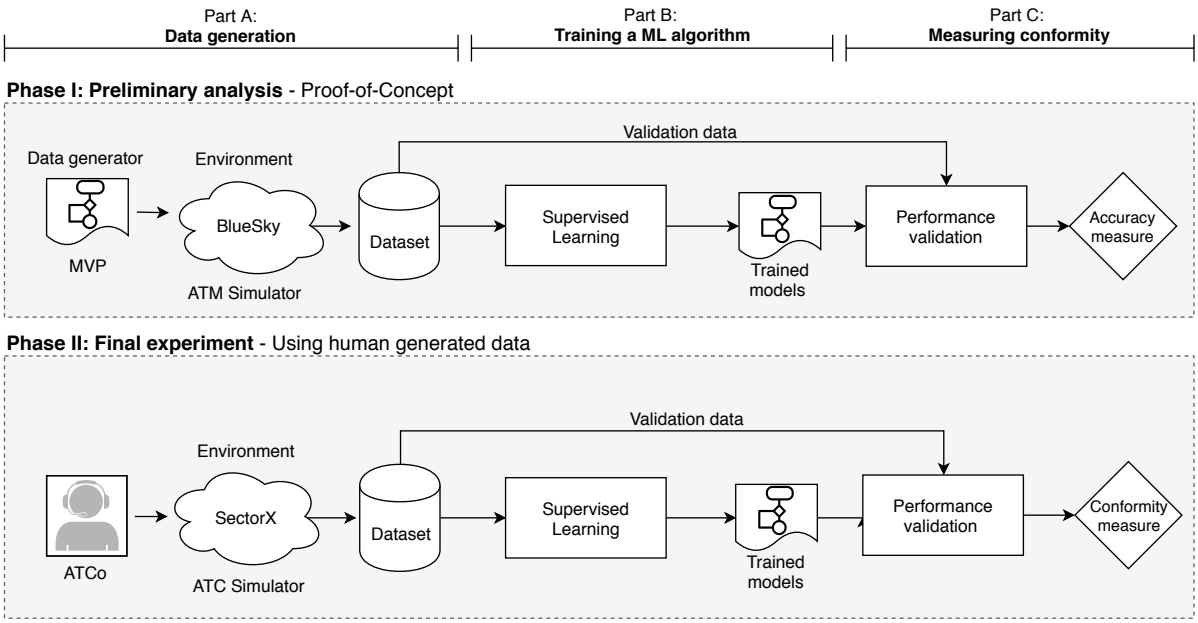


Figure 1.2: The research methodology consists of two phases. The preliminary analysis and the final experiment. Both phases consist of three similar parts.

1.2. Methodology

To answer the research questions, insights from the literature review are combined with two experiments. The literature review provides answers to sub-questions SQ1, SQ2.1, SQ2.2 & SQ3, the preliminary answers SQ2 and the final phase combines all former knowledge to answer the main research question using a human-in-the-loop experiment.

The experiments consist of two phases: Phase I – the preliminary analysis – assesses the potential of Convolutional Neural Networks in combination with the SSD to train a machine learning model. In this phase, an algorithm is used to artificially generate the dataset. Phase II – the final experiment – uses the previously obtained knowledge to determine how strategic conformal automation can be achieved using empirical data generated by human controllers. This structure is visually displayed in Figure 1.2.

1.3. Report Outline

Part I: **Scientific Paper**: The paper combines the knowledge obtained from the Literature Review, Preliminary Analysis and the Experiment to answer the main research question.

Part II: **Preliminary Thesis**: Contains a *Literature Review* regarding CD&R and machine learning techniques and a *Preliminary Analysis*, which explores the potential of using the SSD as learning feature in convolutional neural networks.

Part III: **Appendices of the Paper**: The three chapters in the appendix focus on 1) the final experimental set-up and data, 2) the machine learning training methodology and 3) validation on a participant level.

The report is concluded with a conclusion over all parts.

I

Scientific Paper

Towards Personalized Automation for Air Traffic Control using Convolutional Neural Networks

S. J. van Rooijen (MSc Student)
Supervisors: dr. ir. C. Borst, dr. ir. J. Ellerbroek, dr. ir. E. van Kampen
Section Control & Simulation, Department Control and Operations, Faculty of Aerospace Engineering,
Delft University of Technology, Delft, The Netherlands

***Abstract*—In the upcoming years, en route airspace capacity will be limited by air traffic controller workload, requiring the introduction of automation to assist controllers with conflict detection and resolution. However, acceptance is considered to be one of the main obstacles in the introduction of novel automation. Individual-sensitive automation has been proposed to increase acceptance by adapting to different controller strategies. This research evaluates how personalized automation for air traffic control can be achieved using convolutional neural networks. A human-in-the-loop experiment is devised to generate datasets consisting of conflict resolution commands with corresponding velocity obstacle images as learning feature. Results show that the trained models can reasonably predict command type, direction and directional value. Furthermore, a correlation is found between a controller consistency metric and achieved prediction performance. Finally, the individual-sensitive models performed significantly better than general group-based models, confirming the strategy heterogeneity of the population, which is a critical assumption for personalized automation.**

***Index Terms*—Strategic conformance, machine learning, learning from demonstrations, solution space diagram, velocity obstacles, consistency, decision-support**

I. Introduction

EN ROUTE airspace capacity is mainly limited by Air Traffic Controller (ATCo) availability and workload [1]. Concepts such as free flight are proposed for decentralized autonomous self-separation, but full implementation is not expected to be feasible in the near future [2]. To bridge this gap, novel technologies should be introduced to assist ATCos with traffic separation to reduce controller workload [3], [4].

Specifically, Conflict Detection & Resolution (CD&R) is seen as one of the main tasks of an ATCo [5] and is therefore the focus of this research. CD&R is a control problem in which the ATCo prevents conflicts between aircraft by providing flight plan adjustments in terms of altitude, heading or speed. During CD&R, automation could function as a decision aid to ultimately alleviate ATCo workload and increase safety and performance. However, as multiple alternative actions per situation are feasible, proposed resolutions by automation do not always coincide with human strategy [6], which hinders automation adoption. Therefore, *acceptance* remains an issue and is considered one of the main obstacles in the successful introduction of novel automation [7].

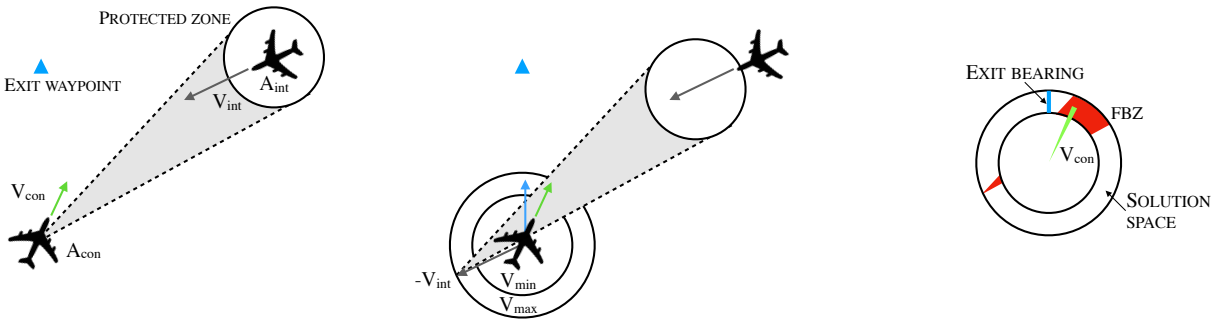
A proposed method to increase trust and acceptance of automation is to make the automation *strategic conformal* [7], [8]. Westin et al. argue that tailoring automation to an individual's strategy could increase acceptance. To create a predictive model for human actions, machine learning is an apparent means as it is able to adapt to different controller preferences without full knowledge on the underlying decision-making dynamics [9].

A proof-of-concept by Regtuit et al. shows that machine learning techniques are able to identify and replicate simple conflict resolution strategies from artificially created data [10]. However, when human controller data is used, two questions arise: One, are controllers sufficiently *consistent* to base individual-sensitive automation on? And two, is a group of controllers adequately *heterogeneous* in strategy for individual-sensitive automation to be beneficial? This research aims to answer these questions by creating a machine learning model able to predict ATCo commands based on the traffic situation.

As input for the learning algorithm, this research proposes the Solution Space Diagram (SSD) [11], also known as velocity obstacle diagram [12]. The SSD displays the solution space when detecting and resolving conflicts to support the ATCo, see Figure 1. It is hypothesized that an SSD image contains sufficient information concerning an air traffic situation to make an informed decision [11], [13], [14]. Learning from an image is advantageous because it eliminates the need for manually designing features based on heuristics and assumptions. A machine learning technique which is especially well-suited for this is a *convolutional neural network* [15].

To test controller consistency and the SSD as a machine learning feature, a human-in-the-loop experiment is devised. Data is gathered from a 12-participant population consisting of six novices with knowledge but no experience in ATC and six intermediate participants, who have undergone an extensive ATC introductory course. Traffic scenarios are simulated in an ATC simulator, which records all aircraft states and controller commands. Even though altitude changes are a preferred ATCo resolution [16], only horizontal plane CD&R was considered due to the reduction in resolution space without necessarily diminishing task difficulty [17]. The experimentally generated datasets are used to train individual-sensitive predictive models using a supervised learning algorithm. Finally, individual model performance is compared to controller consistency and a comparison with general group-based models is made to test group heterogeneity.

The fundamental concepts of this research are introduced in section II. Subsequently, experiment design is discussed in section III and training methodology in section IV. The results are provided in section V, followed by a discussion in section VI and conclusions & recommendations in section VII.



(a) (b) (c)

Figure 1: Construction of the SSD: (a) The protected zone of A_{intruder} is projected onto $A_{\text{controlled}}$ to form the Forbidden Beam Zone (FBZ), (b) The FBZ is displaced by the velocity vector of A_{intruder}, (c) The SSD is created by limiting the solution area by the minimum (V_{min}) and maximum (V_{max}) velocity of $A_{\text{controlled}}$. Adapted from Mercado et al. (2010) [11].

II. Background

A. Strategic Conformance

It seems evident that ATM is bound to move towards a more automated future [3]. However, acceptance of automation remains a major obstacle [8]. The lack of acceptance leads to reduced system use which could influence performance or safety [18]. Past automation efforts have indeed failed due to insufficient controller acceptance [19], of which one of the reasons for rejection is a mismatch in "perceived strategy" [20].

A concept to counteract this issue is *strategic conformance*, defined as "the degree to which automation's behavior and apparent underlying operations match that of the human" [7], [8]. Strategic conformal automation proposes advisories that are not necessarily algorithmically optimal (e.g. using voltage potential [21]), but rather follow heuristics defined by the controllers' individual control-strategy [16], [22].

An empirical study by Hilburn et al. confirms that controllers accept conformal resolutions more often and faster than non-conformal alternatives, especially in complex situations. However, approximately 25% of conformal advisories were rejected [8]. This gives rise to the question; are air traffic controllers consistent enough to base strategic conformal automation on?

ATCo consistency is a main assumption that enables strategic conformal automation. If controllers are not consistent over time, they are not likely to agree with individual-sensitive automation. To obtain a framework for consistency, Westin et al. analyzed resolution strategies identified by Fothergill et al. [23] and more than 500 conflict resolutions collected in real-time simulations [17]. It was concluded that ATCo consistency can be measured on different *levels of abstraction*, of which the detailed decision stages are shown in Figure 2. Given the controlled aircraft, a command in the horizontal plane is comprised by its *type* (heading, speed or direct to waypoint), *direction* (left/right, increase/decrease) and *value* (in degrees or knots).

Westin et al. subsequently used this framework in an empirical study to analyze controller consistency. Results showed that controllers were consistent, but "could not be considered homogeneous as a group" [24], confirming the findings by Hilburn et al. [8]. Additionally, Westin concluded that the expert controllers showed a slightly higher degree of consistency compared to their trainee counterparts.

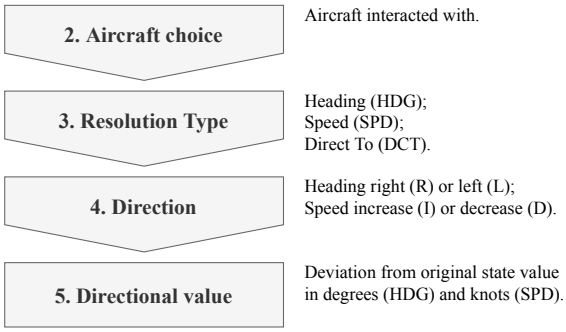


Figure 2: The detailed decision stages from Westin's consistency classifications framework can be used to measure consistency or conformity for different decision stages. Adapted from Westin (2017) [17].

Following Westin's consistency framework, one can measure controller consistency. This, however, requires repetition of specific situations during an experiment, limiting the number of different conflicts that a model can be based on. Besides, determining the similarity of situations is an ambiguous process. For these reasons, a simplified metric to indicate consistency is established in the line of Westin's framework. The following equation determines consistency in resolution *type* and *direction* by summing over all commands per participant:

$$\text{consistency} = \max\left(\frac{\sum \text{class I}}{\sum \text{class I+II}}; \frac{\sum \text{class II}}{\sum \text{class I+II}}\right) \quad (1)$$

where for example class I and II are respectively HDG and SPD for command *type* consistency. Consistency $= 1$ when a single type is used and 0.5 when they are evenly balanced. For command *values*, a different equation is used:

$$\text{consistency} = \frac{\sum \text{unique values possible}}{\sum \text{unique values used}} \quad (2)$$

where the consistency of a participant decreases when a broader range of *value* commands is used. These simplifications of consistency are only valid in constrained, asymmetrical scenarios as described in section III-A.

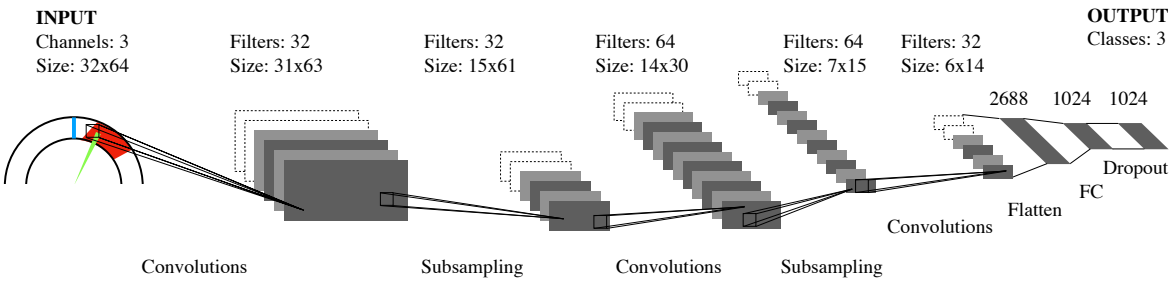


Figure 3: The Convolutional Neural Network structure used in this research. All weights in one plane are identical. Size values are given in pixels. Adapted from LeCun (1998) [15].

Although strategic conformal automation appears promising, certain drawbacks and caveats should be considered. First, the conformal solution is not always the optimal or safest solution. Secondly, when working in teams, stimulating standardized procedures might be preferred over individual problem-solving styles. Similarly, individual automation loses its value in case a group of ATCos behaves consensually according to strict procedures. Lastly, automation can never be 100% conformal, given that humans are not fully consistent.

B. Solution Space Diagram

The Solution Space Diagram (SSD) is an ecological [25] decision support tool that integrates various critical parameters of the CD&R problem. Construction of the SSD is shown in Figure 1. The SSD is composed by defining the Forbidden Beam Zone (FBZ) per intruding aircraft, which comprises all relative velocity vectors that lead to a loss of separation. When the FBZs are displaced by the intruding aircraft's velocity vector, the *no-go zones* are defined. The color of these zones is determined by the time to closest point of approach (CPA), namely red ($t_{\text{cpa}} < 60s$), orange ($60 < t_{\text{cpa}} < 120s$) or gray ($t_{\text{cpa}} > 120s$), see e.g. Figure 12.

In the SSD, the green vector indicates the controlled aircraft's heading and speed. The red geometric shapes each indicate a potentially conflicting aircraft (intruder). When the green velocity vector lies in this colored no-go zone, the aircraft is indeed in conflict and a resolution is required. The non-colored zones are thus, by definition, the solution space for the aircraft.

The concept of the SSD was first introduced as *velocity obstacles* for avoidance manoeuvres in robotics research [12]. This research hypothesizes that an SSD image as learning feature contains sufficient information concerning CD&R to make an informed decision, which is substantiated by previous findings. Van Dam et al. introduced the SSD as *state vector envelope*, aimed to assist pilots with aircraft self-separation to obtain higher pilot performance in terms of off-track distance [13]. In addition, the SSD was found to reduce ATCo workload as a decision support tool [11] and function as information basis for deconflicting maneuvers to avoid collisions between autonomous vehicles [14]. Table I provides further substantiation by evaluating the visualization of the CD&R parameters in the SSD. An additional advantage of using SSD images as learning feature is that the input size is not affected by the number of (conflicting) aircraft in the sector, making it a flexible solution which works for a variety of scenarios.

Table I: Visualization of CD&R Parameters in the SSD.

CD&R Parameter	**Visualization in the SSD**
Closet Point of Approach (CPA)	Translating FBZ
Time to CPA	Shorter time results in a wider FBZ
	Color coded (gray, orange, red)
Conflict angle	Inclination of the FBZ
Airspeed and heading	Green vector
Traffic density	Number of FBZs
Traffic complexity	Ratio of free solution space

C. Convolutional Neural Networks

Achieving conformal automation is closely related to a machine learning field called Imitation Learning (IL), which aims to perform a task based on expert demonstrations called Learning from Demonstrations (LfD). The aim is to generalize the observations to unseen situations, usually using supervised learning [26].

Artificial neural networks with a deep architecture are a powerful method for supervised learning as they automatically devise learning features, without reliance on pre-engineered parameters [27]. This research follows a similar approach, where the input images are provided by SSDs converted to pixel data. In particular, Convolutional Neural Networks (CNNs) [28] have shown their potential of training on image data by demonstrating self-driving cars how to drive [29], [30] and competing with world-class board game champions [31].

Similar to a regular neural network, a CNN consists of multiple stacked neurons. In the case of a CNN, every input is connected to a pixel-value of the original image, as shown in Figure 3. One neuron is locally connected to an area in the original image. This filter slides over the entire image to create a new output map. These overlapping filters can extract visual features such as small corners or edges from the image. The visual features are subsequently combined to form compositions while progressing through the net. A fully connected (FC) layer is used to connect all remaining

neurons to the possible output classes (e.g. *left* or *right*). The probability of outputs is determined by the output classifier function.

A softmax function [32] is used as output classifier for all abstraction levels, which transforms the fully connected layer of real values into a vector of the probability per output class $\sigma(\mathbf{z})$. The softmax function is defined by Equation 3, where all entries of σ are real, within $[0, 1]$, and add up to 1. K is the dimension of input vector $\mathbf{z}$.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e_{z_k}} \text{ for } j = 1, ..., K \tag{3}$$

A cross-entropy function is used to take these probabilities into account in the loss function [33]. The cross entropy H is calculated for M output classes by comparing the probability vector σ resulting from the softmax function to the one-hot encoded target vector y_i:

$$H(y, \sigma) = -\sum_{i}^{M} y_i \log \sigma_i \tag{4}$$

The calculated losses, averaged over a minibatch of samples are used to update the network parameters θ. Usually a gradient-based method is used, as described in section IV-C.

In between the convolutional layers, pooling layers subsample the image. By reducing the image dimensions, less trainable parameters remain and computation time is improved. Additionally, this process removes detail from the image to reduce overfitting [34].

The performance of CNNs is naturally determined by the inputs, as they should capture all relevant information for the model to make a prediction. This research utilizes the SSD as input as described in the proceeding section.

III. Experiment

An experiment is set-up to validate the assumptions and hypotheses regarding the information contained in the SSD and human controller consistency.

A. Experiment set-up

The participants are asked to control an air traffic sector with multiple incoming aircraft by giving commands following two main objectives:

- Avoid Loss-of-Separation between aircraft.
- Aim to guide the aircraft to their exit waypoint as efficiently as possible.

To reach these objectives, participants could use the following command types:

- HDG: Heading change command ([1, 360] deg).
- SPD: Speed change command ([200, 290] kts).
- DCT: Direct To exit waypoint command. Automatically changes the heading towards the assigned exit waypoint.

The experiment consists of four 20-minute scenarios, resulting in 80 minutes of data per participant, as shown in Table II.

Table II: Time-planning of each experiment

Type	Duration [m]	Elapsed time [m]
Briefing	5	5
Training	5	10
Run 1	20	30
Break	5	35
Run 2	20	55
Break	10	65
Run 3	20	85
Break	5	90
Run 4	20	110
Debriefing	10	120

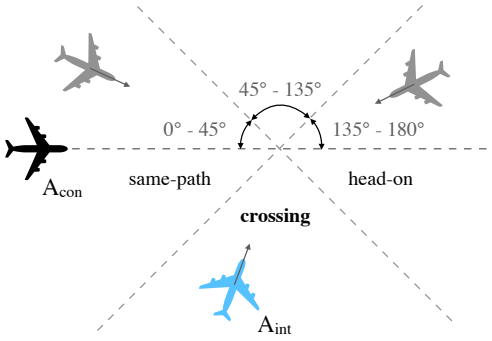


Figure 4: Conflict types as defined by the FAA. The intruding traffic stream is restricted to crossing conflicts ([45,135] deg) only. Adapted from FAA (2014) [35].

1) Scenarios: The shape of the sector – inspired by Amsterdam Sector South 1 – is shown in Figure 5. Two scenarios (S1 and S2) are used in the experiment with identical sector geometry but different conflict angles. The scenarios are restricted to the horizontal plane, i.e. all aircraft fly at the same flight level and only heading and speed changes are allowed. This restriction decreases the solution space, which enables better comparison between controllers in terms of consistency and strategy. Task difficulty is not necessarily lower because same-plane situations have a smaller solution space requiring more monitoring. In order to generate repeatable situations, conflicting aircraft are matched in two-aircraft pairs. The scenarios consist of 20 aircraft, thus 10 conflict pairs each. The main aircraft flow is directed north while the intruding aircraft flow arrives from the east with a closest point of approach (CPA) of 0nm for all conflicts. Considering the limited time per experiment, a subset of conflict angles is chosen so similar conflicts are encountered multiple times by the participant. Specifically, only *crossing* conflicts are simulated, i.e. the relative conflict angles range between 45 and 135 deg [35], see Figure 4. Every conflict angle in the set $\{45, 55, ... , 135\}$ is visited at least four times during the entire experiment. The conflicts are chronically spaced in a way to minimize interference between conflicts.

2) Participants: The population consists of 12 participants divided in two groups: 6 novices with knowledge but no experience in ATC and 6 participants with intermediate experience who have undergone an extensive ATC introductory course at

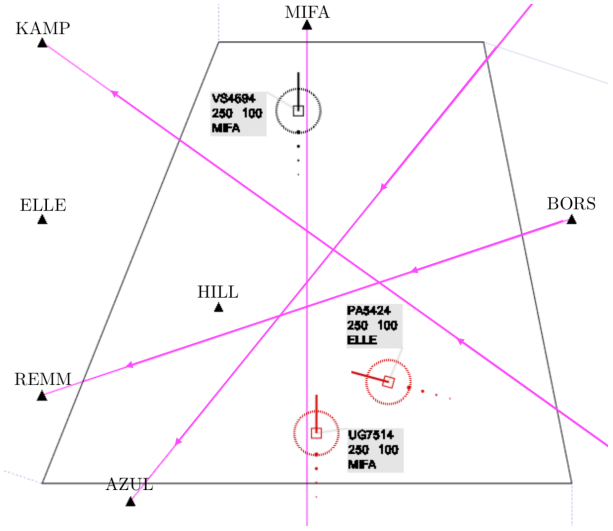


Figure 5: The 50nm × 60nm sector as displayed in ATC simulator SectorX. The magenta lines depict the main traffic flows north- and west-bound. Three aircraft are visible of which two are in conflict. The circles surrounding the aircraft indicate the protected zones ($D = 5$nm).

Table III: Experiment set-up

	Run 1		Run 2		Run 3		Run 4	
Participant	Scen.	SSD	Scen.	SSD	Scen.	SSD	Scen.	SSD
Novice								
P1-3		OFF		OFF		ON		ON
P4-6	S1	ON	S2	ON	S1	OFF	S2	OFF
Intermediate								
P7-9		OFF		OFF		ON		ON
P10-12		ON		ON		OFF		OFF

LVNL[1].

3) Conditions: As the SSD provides a clear overview of the available solution space, it may influence the decision-making process and consistency of the controller. To evaluate the effect of the SSD, each participant completed 2 scenarios with and 2 scenarios without the SSD. The order in which the SSD is available is altered to prevent confounding effects. This experiment set-up is summarized in Table III.

4) Equipment: The scenarios are simulated in *SectorX*, a Java-based, medium-fidelity ATC simulation tool developed at Delft University of Technology. It provides a means for the ATCo to interact with the environment by passing in commands using a separate command interface through clicking on buttons using a computer mouse. For example, to change an aircraft's heading west-bound to 270deg, one should use the following sequence of clicks: [Select the aircraft using the mouse] → [HDG] → [2] → [7] → [0] → [Execute].

Each time a command is given, the application saves the command together with all aircraft states at that moment plus an image of the controlled aircraft's SSD. During supervised model training, the SSD image functions as input while the given command functions as target. The SSD image is used to train on all runs, regardless of whether the SSD was available to the participant during that time.

[1]Air Traffic Control the Netherlands.

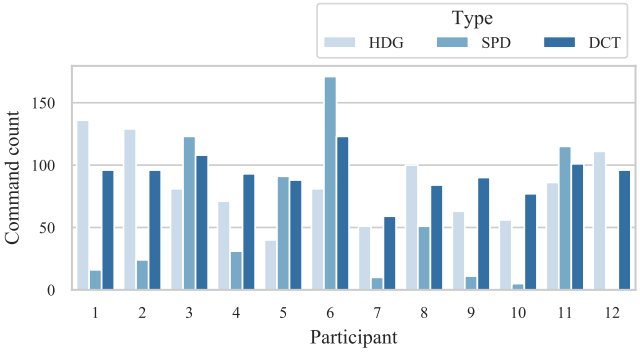


Figure 6: Command count per command *type*: Heading (HDG), Speed (SPD) and Direct To (DCT).

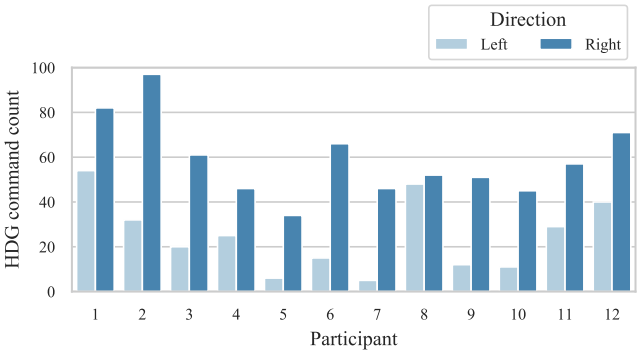


Figure 7: Heading command count per *direction*.

5) Training procedure: Before measurements, each participant performs three training runs to get acquainted with the ATC simulator. The first run involves controlling a single aircraft, the second introduces a simple conflict, and the last run introduces conflict resolution using the SSD. Each training run lasts at least 90s and can be prolonged on demand.

B. Summary of Dataset

Each command is logged and post-processed using the aircraft states, which primarily consist of all aircraft positions, the velocity vectors of all aircraft and their respective bearings to the exit waypoint. The total experiment resulted in a dataset of close to 2,800 samples, with an average of 230 samples per participant. Each sample consists of the state (i.e. the pixel-values of the SSD at the time of the command) and a corresponding target (the command). Each command is comprised of its *type* (Figure 6), *direction* (Figure 7) and *value* (Figure 8). The charts show that participants vary in number of commands and preferred command types, directions and values. Moreover, Figure 9 shows the geometric implications of these strategic differences between participants.

IV. Model Creation

The dataset as described in section III is used to create the individual models according to the procedure in Figure 10. After preprocessing the commands and SSD images, the test data (i.e. experiment run 4) is separated from the set. The

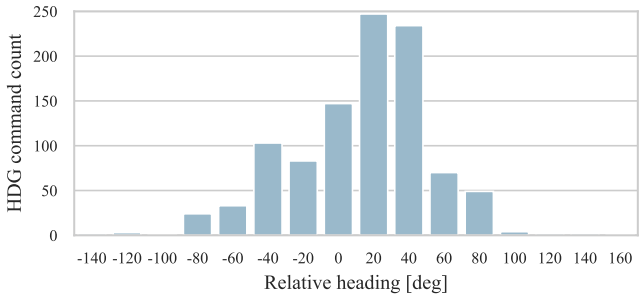


Figure 8: Heading command count per *value* of bearing relative to current bearing. Distribution is skewed due to asymmetrical scenario design.

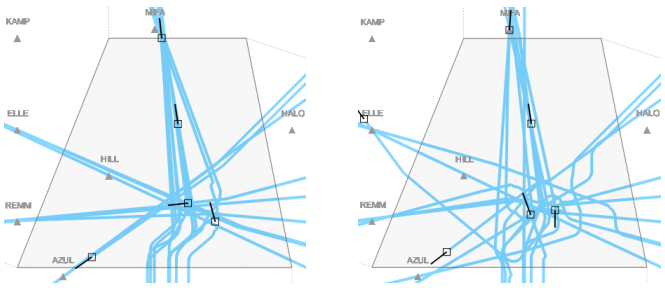


Figure 9: Aircraft track lines after experiment run 1 of P7 (a) and P11 (b) indicating geometric heterogeneity in resolution strategy and consistency.

training data is used to train three personalized models per participant, one for each abstraction level. K-fold validation is used to select the models that obtain the highest prediction performance, after which they are validated using the test set. Finally, five general models are trained on random samples of all data. The mean performance of these five models is taken and defined as the general model baseline.

A. Data preprocessing

The SSD dataset consists of 128x128 pixel RGB samples which are preprocessed for computing time improvements. The most relevant information in the SSD is located in the upper half of the image, as an aircraft is not likely to make turns larger than 90 deg. Taking this into account, the lower half of the SSD is cropped away to decrease training times (Figure 11a). Secondly, to generalize the data, the velocity vector is rotated upwards (Figure 11b. After this, two potential steps can be taken: 1) The black background is converted to white to make the features more pronounced, and 2) conflicts with a time to loss-of-separation of more than 120s (gray areas) are removed from the SSD. Although the latter two steps provided some improvements for certain participants (who for example ignored conflicts with a $t_{\text{CPA}} > 120s$), it did not considerably improve results overall. To keep data preprocessing to a minimum, background conversion and noise removal are not applied.

Table IV: Final network architecture. Sizes shown in pixels.

Layer	Input	Filter size	Stride	Num filters	Activation	Output
Conv2D	32x64x3	2x2	1	32	ReLU	31x63x32
MaxPool	31x63x32					15x31x32
Conv2D	15x31x32	2x2	1	64	ReLU	14x30x64
MaxPool	14x30x64					7x15x64
Conv2D	7x15x64	2x2	1	32	ReLU	6x14x32
Flatten	6x14x32					2688
Dense	2688				ReLU	1024
Dropout	1024					1024
Dense	1024				Softmax	3

B. Model architecture

Due to comparable input and feature characteristics, the network architecture of a deep neural network able to play Atari 2600 games [36] is selected as a starting point. In order to improve model performance, the filter size is reduced in order to capture the exact location of the relatively small features in the SSD such as the speed vector and exit waypoint vector. In contrast to the architecture from Mnih et al. [36], the input to the network is encoded in RGB instead of gray-scale. This allows the network to distinctly interpret the green velocity and blue exit waypoint vectors, as shown in Figure 12. The final network architecture as shown in Table IV is determined using A/B-testing, as computation time is too large for a complete grid search. To reduce overfitting, a dropout layer is added, which sets the weights of a fraction of the neurons to zero at each epoch during training [37]. In addition, max pooling layers [38] are used, which provided better results than large-filter convolutional layers.

C. Training

The hyperparameters and model architecture are kept constant for all participants and all target types. The network parameters are subsequently updated using the network losses using a first-order gradient-based optimization algorithm called Adam. Empirical results show that it outperforms previously popular optimizers such as AdaGrad, RMSProp and SGDNesterov [39]. In between layers, Rectified Linear Unit (ReLU) functions are used as activation functions due to their computational efficiency [40].

The remaining hyperparameters are determined through an informal search, given their relatively small effect on model performance and computational constraints due to the range of possibilities. The final parameters are shown in Table V. The CNNs are created and trained using Python-based Keras[2] with a TensorFlow [41] back-end.

D. Evaluation

Figures 6, 7 and 8 show that the number of samples per class varies considerably. Evaluation of predictions based on this imbalanced dataset solely using the accuracy metric (Equation (5) provides misleading results. For example, if a controller chooses to use heading instead of speed commands in 95% of the situations, simply always guessing 'heading' would result

[2]Keras is developed by François Chollet and others (2015, https://keras.io)

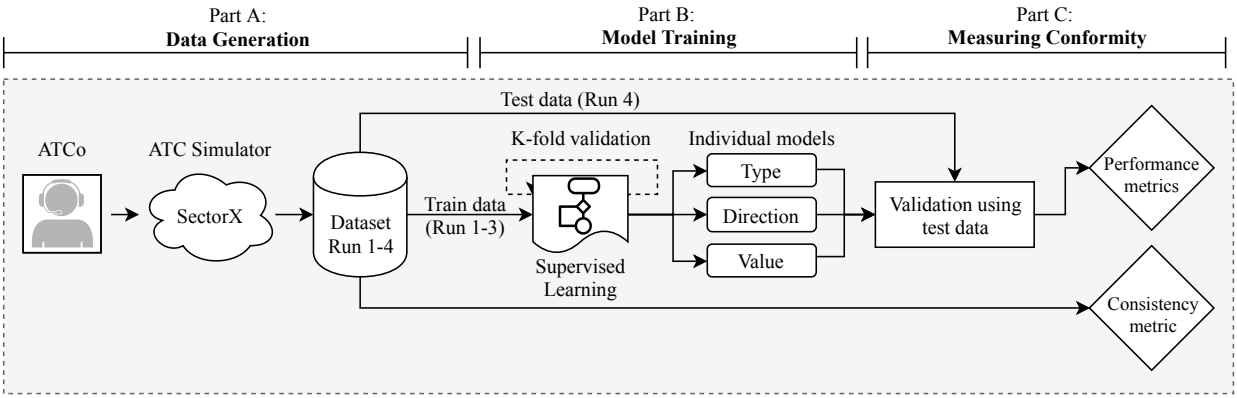


Figure 10: Data generation and training & testing of the individual models for one participant. The dataset consists of input (SSD images) and target (commands) data. The models are used to predict a command for a given SSD image. Model performance is based on prediction accuracy.

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

Figure 11: SSD preprocessing options: (a) Original, (b) rotated, (c), background converted, (d), noise removed.

(a) All (b) Red (c) Green (d) Blue

Figure 12: Color channels.

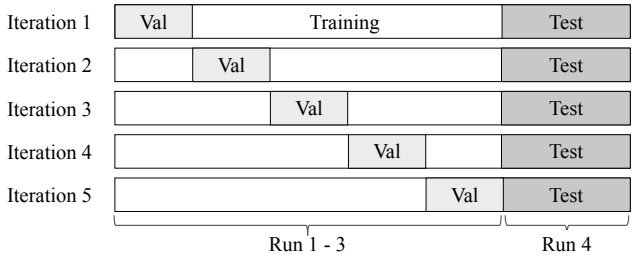


Figure 13: Stratified 5-fold cross-validation with a separate test set.

in 0.95 accuracy score. This would give the a false sense of model performance, because 0% of the speed commands are predicted. Instead, a more balanced measure is used to evaluate model performance, called the *Matthews Correlation Coefficient (MCC)* [42], which is generally considered to be less biased than accuracy [43]. Besides, it has been proven to work well for multi-class predictions [44]. The definitions of accuracy and the two-class MCC are shown in Equations (5) and (6) respectively, where TP = true positive, TN = true negative, FP = false positive, FN = false negative.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

with possible values [0, 1].

Table V: (Hyper)parameters during training.

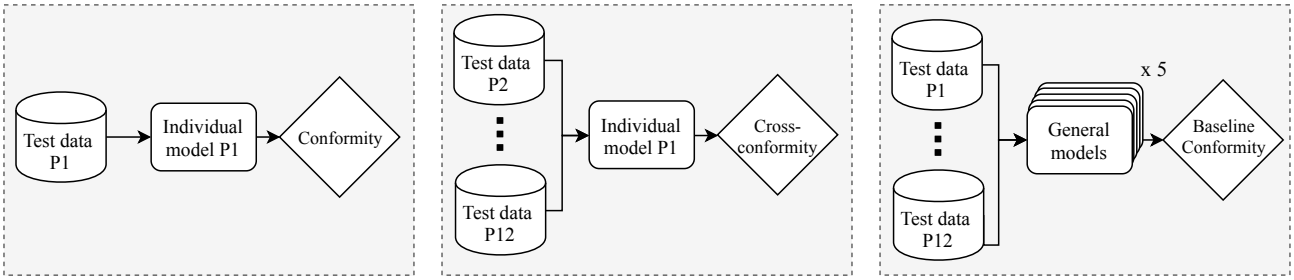
Parameters	Value	Unit
Optimization algorithm	Adam	-
Output activation	Softmax classifier	-
Loss function	Categorical entropy	-
Train/val/test ratio	60/15/25	-
K-folds	5	-
Mini batch-size	32	samples
Steps-per-epoch	2 × training samples / batch-size	samples
Epochs	30	-
Learning rate	0.01	-
Dropout rate	20	%
Input image dimensions	128x128	px

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{6}$$

with possible values [-1, 1]. As MCC is a more critical metric, MCC values are often lower compared to accuracy values for identical models.

During training, the MCC is monitored using validation data and the best performing model is saved for testing. Validation data is selected iteratively using *k-fold cross-validation*, which has proven to have lower variance and bias of performance measures compared to other methods [45]. During training $1/k^{th}$ of the data is iteratively reserved for validation, as illustrated in Figure 13. This method can be augmented using *stratified* sample selection, which ensures that the distribution of classes in the validation set is comparable to the training set [45]. Due to the limited quantity of data available, five folds are applied during training.

Each model consists of three sub-models. Each model predicts one of the detailed decision stages in Figure 2 that together comprise a command: resolution *type*, *direction* and directional *value*. The predicted *type* can be heading (HDG), speed (SPD) or Direct To exit waypoint (DCT). The direction is left or right for heading commands. The directional *value* is the given heading or speed command relative to the current state, in degrees and knots respectively. The *value* prediction divided into the classes $[0, 10]$ deg, $[10, 45]$ deg and > 45 deg. To summarize, for a given SSD, the models will provide a *type*, *direction* and *value* prediction, e.g.: HDG $\rightarrow$ LEFT $\rightarrow$ $[0, 10]$ deg.



(a) Regular (b) Inter-participant (c) Baseline

Figure 14: Three validation steps for Participant 1 (P1).

E. Validation

A validation step follows after the training all individual and general models. Each individual model is tested with the participant's corresponding test dataset (i.e. the 4^{th} run), as shown in Figure 14a. Additionally, each individual model is tested with all *other* participants' test datasets (Figure 14b). Finally, five general models are trained using random samples from all data, see Figure 14c. The number of data samples is capped to the mean number of samples available for the individual models. These general models are tested with all participants' test datasets to create a baseline.

V. Results

This section is divided into five parts: The first shows the training phase of the model. The second summarizes the performance of the individual models for each participant and abstraction level. Section V-C examines the effects of participant experience and the availability of the SSD during the experiment. Subsequently, the individual models are cross-tested using all other participants' test data to determine to which extent the models are individual-sensitive. Finally, individual model performance is compared to a baseline of general models, which are trained on all data.

Performance of the models is measured primarily using the *MCC* (see section IV-D), which ranges between -1 and 1. An MCC of 1 indicates a perfect prediction, while 0 and -1 refer to random and complete disagreement respectively. Plots in this section are scaled to $[0, 1]$ to highlight differences between categories. In certain cases *accuracy* is given as performance measure because it is more intuitive and is the measure that is directly experienced by the end-user.

A. The Training Phase

Figure 15 shows the training progress for the individual model of Participant 1. The spread around each line depicts the least and best performing fold per abstraction level during training, which lasts 25 epochs. Increasing validation MCC values during training (increasing epochs) indicate that the neural network 'learns' from the data samples. In most cases, the models reach MCC scores >0.95 during *training*, while *validation* MCC scores lack behind, as shown in Figure 15. This difference between training and validation performance indicates overfitting on the training data. The spread shows that validation MCC can differ more than 0.2 per fold, which is a relatively large amount compared to the mean value. While this figure only shows P1, other participants show comparable trend lines although final performance values differ.

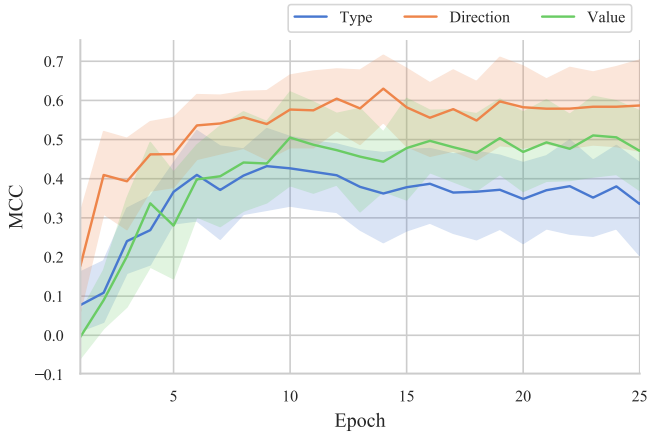


Figure 15: Validation performance during training of P1's individual model. The spread indicates the maximum and minimum performance for each fold per abstraction level.

B. Validation Performance

After training (Figure 15), the models are validated using the test dataset (Run 4), which is kept apart in all prior processes. Figure 16 shows the achieved MCC scores per participant and abstraction level. Large differences in achievable performance (MCC) occur among participants and abstraction levels indicating that the personalized predictions do not work equally well for the entire population. Specifically, the models of Participants 9 and 10 (mean MCC 0.74 and 0.75 resp.) show the highest performance, while the models of Participants 11 and 8 (mean MCC 0.43 and 0.55 resp.) show the lowest performance.

When the data per participant is aggregated and split per abstraction level, clear performance differences are observed between the abstraction levels. *Direction* prediction shows the highest MCC score (mean = 0.76, SD = 0.11), while *type* (mean = 0.52, SD = 0.21) and *value* (mean = 0.64, SD = 0.12) predictions obtain lower performances. The achieved *accuracies* are higher for *direction* (mean = 0.84, SD = 0.07), *type* (mean = 0.69, SD = 0.13) and *value* (mean = 0.76, SD = 0.08) predictions.

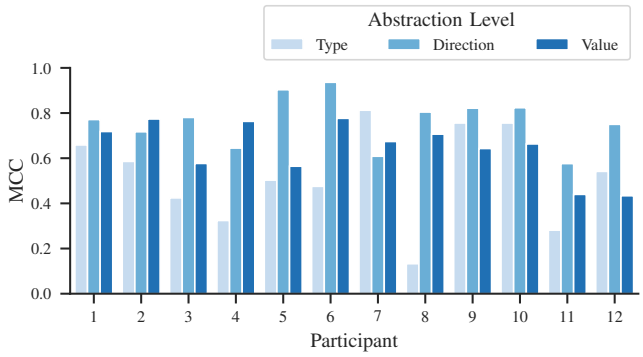


Figure 16: Model test-performance per participant and abstraction level.

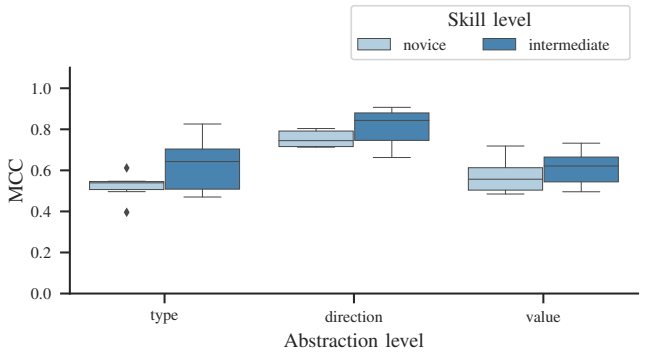


Figure 18: Model performance with the participants aggregated by skill level.

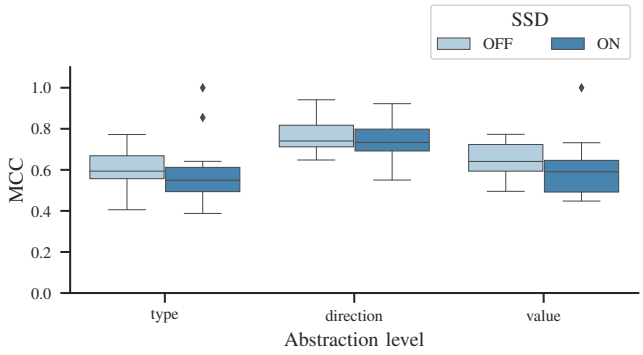


Figure 17: Effect of SSD availability on model performance.

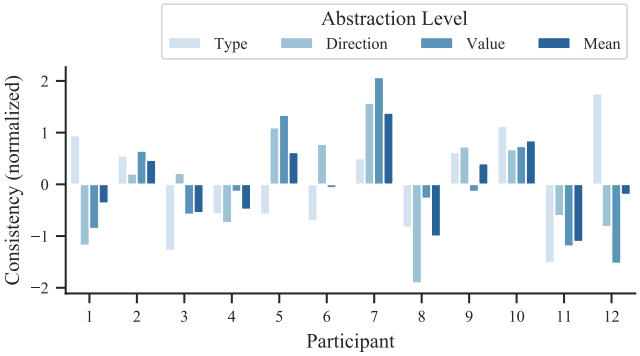


Figure 19: Consistency scores per participant split per abstraction level.

The individual models achieve a mean prediction accuracy of 76% over all abstraction levels. However, when predicting a HDG command, e.g. HDG $\rightarrow$ LEFT $\rightarrow$ [0,10] deg, the accuracies of all abstraction levels should be multiplied. This product of accuracies ranges from 22% (P11) to 58% (P10). On the other hand, when a DCT command is given, only a *type* prediction is sufficient.

C. Validation Performance per Condition

Figure 17 shows the effect of the availability of the SSD during the experiment on the obtained model performance. The models obtain a slightly lower performance with the SSD available for all abstraction levels. Contrarily, the achieved MCC increases close to 1.0 in three particular cases when the SSD is used, as depicted with the diamonds in Figure 17, indicating that the SSD has a varying effect per participant. During the experiment, participants commented on the SSD: *"It is easier with the SSD turned on because there are fewer options."* (P5), *"Without the SSD, I am paying more attention to the general picture and strategy."* (P6) and *"With the SSD, I am more focused on local conflicts instead of the overview."* (P8).

The model performances per abstraction level for both skill levels are shown in Figure 18. Better predicting models are achieved for the intermediate group (mean = 0.65) compared to the novice group (mean = 0.60). However, the population size (N=6 per skill level) is too small for meaningful statistical analysis.

D. Consistency

Figure 19 shows the normalized consistency – as defined in section II-A – per participant and abstraction level. Participants 5, 7 and 10 appear to be the most consistent compared to a lower consistency for Participants 8 and 11. A participant's consistency level varies per abstraction level, e.g. having a high *type* consistency but lacking consistency in *value*, or vice versa.

The aggregated consistencies based on skill level are shown in Figure 20. The intermediate group shows an unexpected larger spread, which indicates that the group contains both more consistent and less consistent participants compared to the novices. Figure 19 illustrates that the inconsistency in the intermediate group is predominantly caused by Participants 8 and 11.

Subsequently, the mean consistency from Figure 19 is plotted versus the average model MCC per participant (mean over all folds and abstraction levels), as shown in Figure 21. A Pearson Correlation Coefficient test finds a positive correlation ($r = 0.75$, p-value = .005) between participant consistency and individual model MCC. This shows that the personal models of more consistent participants perform better than the models of their less consistent counterparts.

E. Cross-Testing Performance

To check whether the personalized models are indeed individual-sensitive, they are tested using other participants'

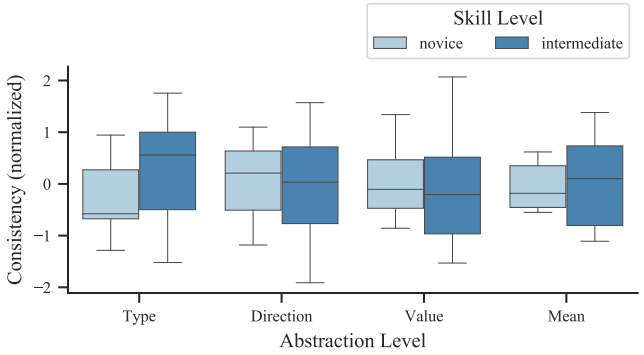


Figure 20: Consistency scores aggregated by skill level.

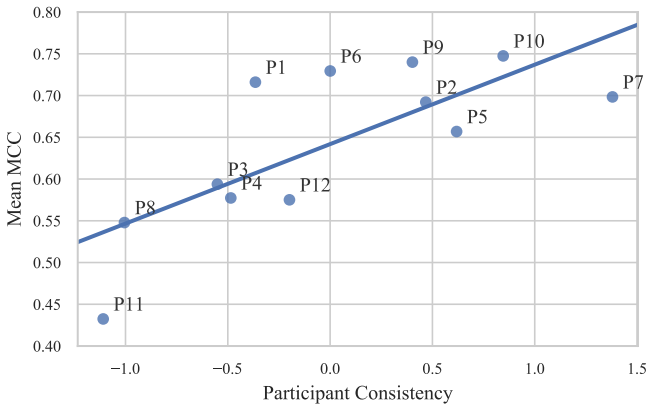


Figure 21: Participant consistency vs individual model performance. $R^2 = 0.56$.

test datasets and compared to the performance of five general models, in accordance with Figure 14b and 14c.

Figure 22 shows one of the twelve resulting spider charts featuring the model of P1. The achievable performance is distinctly higher with P1's own test data (mean MCC = 0.72) compared to testing with other participants' data (mean MCC = 0.37). All test datasets are only used during final model validation and not during model design or training. The peak at P1 indicates that Participant 1 makes different decisions in similar situations compared to the rest of the population. Other participants' models that show a distinct peak in mean model performance for their respective participant's test data are P1, P6, P7, P9 and P10. The remaining models obtain more uniform MCC scores, regardless of which test set is used.

F. Comparison Between Individual and General Models

As a third validation step, the individual models are compared to a baseline of general models. The average obtained performance per participant is shown in Figure 23 for the individual models and baseline of general models. The chart shows that most individual models outperform the mean of the general models, but some cases show near equal or even worse (P4 and P8) performance, possibly caused by a strategy change in the final run.

A paired t-test shows that the individual models perform significantly better (t(11) = 2.9, p-value = 0.02) than the

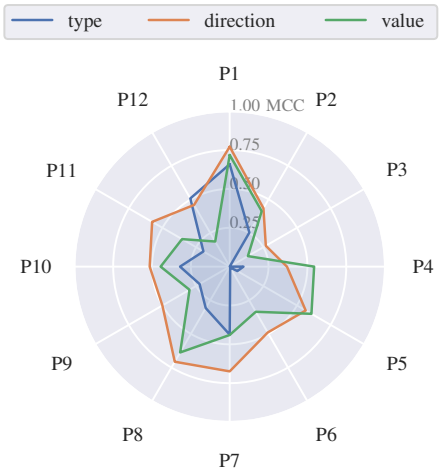


Figure 22: Performance (in MCC) of P1's individual model tested on the test datasets of all other participants.

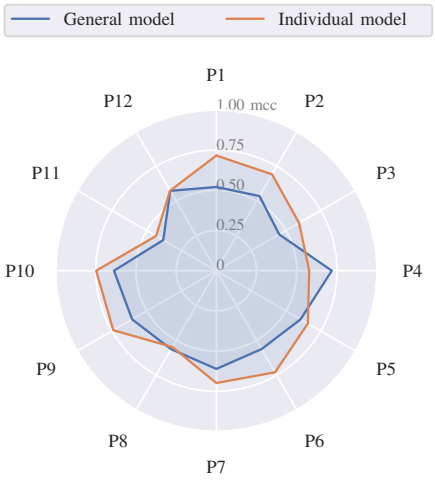


Figure 23: Performance (in MCC) of each participant's individual model compared to the mean performance of five general models, averaged over all abstraction levels.

general models in terms of MCC, see Figure 24. The individual models provide a mean 0.08 (SD = 0.10) MCC improvement over the general models. The personalized approach is most effective for P1, whose individual models score 0.20 MCC higher than the baseline. Furthermore, the right-hand side of Figure 24 shows the general and individual model performances measured in accuracy. A paired t-test shows a significant accuracy improvement (t(11) = 3.2, p-value = 0.01) of individual models compared to the general models, with a mean accuracy improvement from 71% to 76% (5.3%, SD = 5.8%). The least and most accurate individual models achieve 61.4% and 83.3% test accuracy respectively.

VI. Discussion

Discussion of results is divided in the main elements of the research: experiment design, machine learning using the SSD, controller consistency and group heterogeneity.

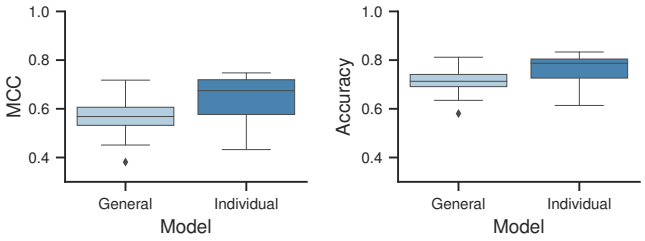


Figure 24: Comparison of model performance between general and individual models.

A. Experiment Design

The population consists of non-professional ATCos, divided into novices, with knowledge but no experience in ATC, and intermediate controllers, who followed the ATC introductory course. However, consistency and model performance metrics do not provide demonstrable differences between both groups, which implies that the ATC course has a limited effect on average controller consistency in these simplified scenarios. It is expected that professional ATCos are more consistent than novices [24], but might also be more homogeneous as a group due to trained procedures [22]. Although this non-professional population provided a proof-of-concept, professional ATCos are required to validate real-world feasibility. Besides, part of the presented results use statistical analysis to substantiate the findings while the population ($N = 12$) is fairly small. Future experiments could include more participants to obtain higher statistical power.

Scenario design is constrained to encounter similar conflicts multiple times. Specifically, intruding traffic always originates from the east, only converging conflicts are encountered and altitude is not taken into account. These constraints limit the number of conflict types and thus the resolution space. Removing them could have a detrimental effect on prediction performance and require more training data. The quantity of training data is a common limitation in machine learning. Results show that model performance per validation fold can fluctuate up to 0.2 MCC, which shows sensitivity to random state-action pair sampling from the training set. This variance is expected to decrease with exposure to more training data, which can be obtained with longer experiments. To verify whether data quantity per participant influences model performance in this research, the number of available training samples is plotted versus the achieved model performance in Figure 25. These figures show no indication that the difference in available samples per participant is a major influence on model performance.

Surprisingly, P4's individual model performs worse than the mean general models. As can be expected of a novice, P4 may have changed strategy between runs causing the final experiment run to coincide more with the strategy of other participants. Due to the limited data quantity in this research, all data was used. With prolonged experiments, data tainted with training effects will be eligible for removal.

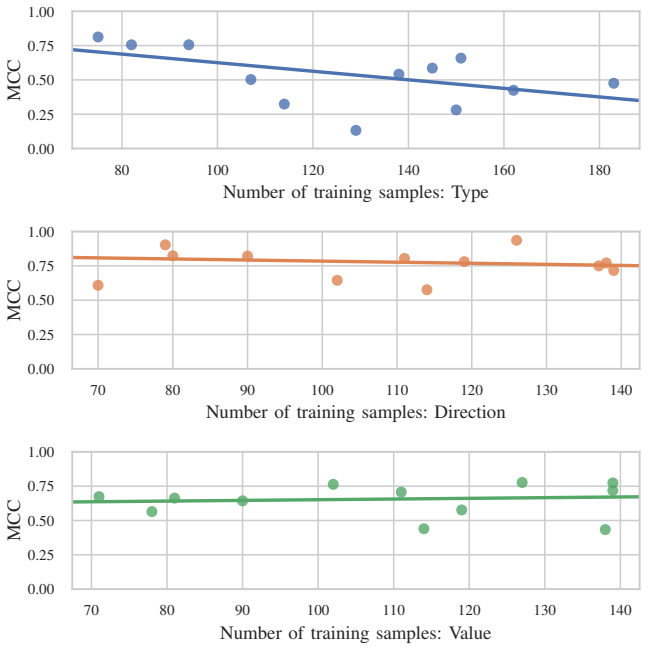


Figure 25: Model performance per abstraction level vs the number of training samples used per model. Each data point represents the summation of 4 runs per participant.

B. Machine Learning using the SSD

Evidence that convolutional neural networks are able to interpret the SSD is shown by accuracy increase during training. However, this might indicate that the network overfits on the training data at pixel-level without generalization to new samples. The ability to generalize is deemed likely because the validation performance during training follows the same upward trend. Validation results using the separate test set further confirm this induction. On the other hand, overfitting on the training samples does occur to an extent, indicated by the performance difference between the training and test sets.

It is expected that performance improvements can be achieved by a formal grid search to optimize hyperparameters. However, one training iteration generates 360 models (12 participants $\times$ 3 abstraction levels $\times$ 2 SSD conditions $\times$ 5 cross-validation folds) thus iterating over parameters is computationally expensive and meticulous. The network architecture used in this research is kept constant for all abstraction levels, i.e. it can predict command *type*, *direction* and *value* by only altering the last fully connected layer. Designing a network architecture per abstraction level could consequently improve performance.

Command *value* predictions are obtained using classification to more easily compare results to *type* and *direction* predictions. As command values are continuous, this caused the accuracy of *value* predictions to be reliant on the classification, i.e. the design of the discretization. Therefore, more precise predictions are expected to be achieved using regression.

Data quantity is often a limiting factor in machine learning experiments with human data. A method to increase

effectiveness of the available data is *reinforcement learning*. One option is to start with supervised learning – incorporating human strategy – and to improve the models with more experience using reinforcement learning [31]. Another is to use *inverse* reinforcement learning, which could learn a personalized reward function that is subsequently used to train a model through interaction with a simulated environment.

Nonetheless, the achieved MCC scores using the current methods indicate a considerably better than random prediction, even for the general models (up to 81% accuracy, averaged over all three abstraction levels). This illustrates that the SSD indeed captures sufficient information concerning CD&R scenarios to base prediction on – given the simplified scenarios used in this research – confirming the hypothesis. However, apart from the CD&R parameters included in the SSD, research on control heuristics indicate that traffic flows, sector geometry and controller goals also play an important role in ATCo strategy [22], [23]. Including these sector characteristics in the visual network input could further increase prediction conformity. Moreover, model predictions could be taken to a higher-level decision stage such as aircraft selection and resolution geometry (e.g. 'behind' or 'in front'), as proposed in Westin et al.'s consistency framework [17]. Furthermore, humans base their decision on a dynamic situation, while the SSD is a static input. Using multiple consecutive SSD frames could incorporate dynamics into the model.

Besides, given the black-box operation of CNNs, it remains difficult to fully comprehend the decision-making process. Since this might be unacceptable in a safety-critical domain such as ATC, a comparison between parametric and non-parametric algorithms could clarify the benefits and drawbacks of using image data (the SSD) over engineered learning features (conflict parameters).

C. Controller Consistency and Population Heterogeneity

A driver of model performance is hypothesized to be controller consistency. Indeed, the results confirm a correlation between the consistency metric and model performance. This is in line with the expectation that more consistent controllers are better suited to base strategic conformal automation on. Note that this is a correlation and not necessarily a causal relation, as there might be underlying effects that cause both measures to increase. Besides, the validity of the metric is limited to asymmetric scenarios and constrained conflict angles as used in this research. A general purpose consistency metric should evaluate all situations in a case-by-case fashion to determine the true consistency of a controller.

The availability of the SSD during the experiment was expected to influence controller consistency and thus model performance. Contrarily, the results show a slight decrease in mean model performance when the SSD is available to the controller. Based on remarks during the experiment, the SSD might emphasize local conflicts which influences the decision-making process. The large spread in data indicates that the effect of the SSD depends on the participant, which might be mitigated with more training. Regardless, the availability of the SSD during the experiment is not required when creating strategic conformal automation, as it does not improve model performance.

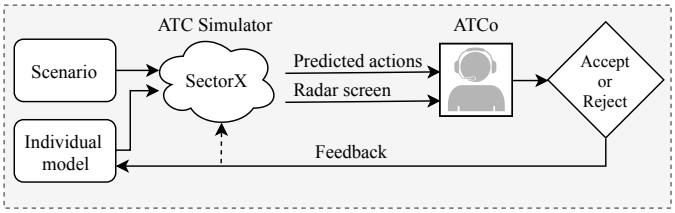


Figure 26: A future human-in-the-loop experiment should evaluate ATCo acceptance of proposed resolutions.

Two findings illustrate the strategy heterogeneity of the participant group: One, part of the models predict distinctly more accurately when predicting their participant's test data. And two, individual-sensitive models significantly outperform the general models. Although the mean accuracy and MCC improvements are not large for all participants, for half of the population MCC scores increased by more than 0.10 (up to 0.20). Nevertheless, the difference between the most accurate models (mean = 83% accuracy) and the least accurate models (mean = 61% accuracy) is considerable. Evidently, strategic conformal automation is not applicable to all controllers in the population. Nonetheless, the results on controller consistency and a heterogeneous controller population are in accordance with previous empirical research where professional ATCos were used [8], [24].

D. Acceptance and Implementation

To place the achieved model accuracies into context, recall that 25% of proposed conformal resolutions was rejected in prior empirical research [8]. The cause for rejection is not necessarily non-consistency, as i.a. advisory source and interface representation are also found to influence acceptance [24]. To close the acceptance feedback loop, a second human-in-the-loop experiment is required in which controllers rate the resolutions as proposed by their personalized models. Furthermore, this feedback could subsequently be used to iteratively update the models, see Figure 26.

Finally, proper implementation methodology is key to obtain high acceptance rates. A critical element in this is Level of Automation (LoA) design, which should ensure controller performance, workload and situational awareness [46] by determining the conformal automation's degree of authority, such as management-by-consent or management-by-exception [47]. Besides, strategic conformal automation might propagate human errors and propose unsafe or non-optimal resolutions. To counteract this, additional algorithms could support the conformal automation to find a balance between conformity, safety and optimality.

VII. Conclusions and Recommendations

This research evaluates how strategic conformal automation for air traffic control can be achieved using convolutional neural networks through a human-in-the-loop experiment. A 12-participant experiment is devised to generate training data consisting of solution space diagram (SSD) images and

conflict resolutions. Achieved model performances show that the SSD contains sufficient information to make accurate predictions on command *type*, command *direction* and command *value* given by controllers in 2D CD&R scenarios.

Results show a correlation between the controller consistency metric and achieved model performance, confirming the hypothesis that consistent controllers are more suited for strategic conformal automation. Regardless, the majority of controllers in the population is sufficiently consistent to base the conformal automation on. Personalized models obtain significantly higher prediction accuracies than general models, indicating that controllers in this experiment exhibit differentiating strategies, i.e. are not homogeneous as a group. This is a critical assumption for strategic conformal automation. However, the performance improvement due to individual modeling substantially differs per controller, ranging from a deterioration to improvements of 0.20 MCC (Matthews Correlation Coefficient) and 12% accuracy. Nonetheless, convolutional neural networks appear to be a feasible method to achieve strategic conformal automation.

Recommendations focus on three elements: learning methodology, experiment design and implementation. More sector information could be added to the model input, incorporating important sector characteristics such as traffic flows and sector geometry. This could add higher abstraction level decisions such as resolution geometry and aircraft selection to the model predictions. The benefits and drawbacks of using image data (the SSD) over engineered learning features (conflict parameters) can be clarified by a comparison between parametric and non-parametric algorithms. Finally, as data quantity is limited in human-in-the-loop experiments, inverse reinforcement learning could provide a means to use the available data more effectively by learning a personalized reward function and gaining experience through simulated scenarios.

Secondly, a future experiment could improve on population and scenarios. This research shows that achievable prediction accuracies are expected to be higher for more consistent controllers. Using a professional air traffic controller population could confirm this finding and provide insight in real-world applicability. Furthermore, as the scenarios in this research proved to be predictable, the next step is to create higher fidelity scenarios, without constraints on conflicts angles or altitude.

Finally, research on implementation methodology is vital to achieve strategical conformal automation that is safe, efficient and accepted by controllers. The acceptance feedback loop should be closed through a human-in-the-loop experiment to assess if strategic conformal automation using individual-sensitive predictions indeed increases trust and acceptance. After all, machines are becoming more intelligent every day, but as the incredible cognitive models of humans prove difficult to match, the interaction between human and automation is more relevant than ever.

REFERENCES

[1] Arnab Majumdar, Washington Y Ochieng, Gérard McAuley, Jean Michel Lenzi, and Catalin Lepadatu. The Factors Affecting Airspace Capacity in Europe: A Cross-Sectional Time-Series Analysis Using Simulated Controller Workload Data. *Journal of Navigation*, 57(3):385–405, 2004.

[2] J. M. Hoekstra, R. N. H. W. Van Gent, and R. C. J. Ruigrok. Designing for safety: the free flight air traffic management concept. *Reliability Engineering & System Safety*, 75(2):215–232, 2002.

[3] SESAR Consortium. The Concept of Operations at a glance. *Single European Sky*, 2007.

[4] Joint Planning and Development Office (JPDO) and Next Generation Air Transportation System (NextGen). Concept of operations for the next generation air transportation system. Technical report, 2011.

[5] EUROCONTROL. Model for Task and Job Descriptions of Air Traffic Controllers. *European Air Traffic Control Harmonisation and Integration Programme*, 1996.

[6] Thomas Prevot, Jeffrey R Homola, Lynne H Martin, Joey S Mercer, and Christopher D Cabrall. Toward automated air traffic control: investigating a fundamental paradigm shift in human/systems interaction. *International Journal of Human-Computer Interaction*, 28(2):77–98, 2012.

[7] Carl Westin, Clark Borst, and Brian Hilburn. Strategic Conformance: Overcoming Acceptance Issues of Decision Aiding Automation? *IEEE Transactions on Human-Machine Systems*, 46(1):41–52, 2016.

[8] Brian Hilburn, Carl Westin, and Clark Borst. Will Controllers Accept a Machine That Thinks Like They Think? The Role of Strategic Conformance in Decision Aiding Automation. *Air Traffic Control Quarterly*, 22(2):115–136, 2014.

[9] Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement Learning is Direct Adaptive Optimal Control. *IEEE Control Systems*, 12(2):19–22, 1992.

[10] Robert M. Regtuit, Clark Borst, Erik-Jan Van Kampen, and M. (René) M. van Paassen. Building Strategic Conformal Automation for Air Traffic Control Using Machine Learning. In *AIAA SciTech Forum*, Kissimmee, Florida, 2018. AIAA Information Systems.

[11] Gustavo Mercado-Velasco, Max Mulder, and M.M. Van Paassen. Analysis of Air Traffic Controller Workload Reduction Based on the Solution Space for the Merging Task. *AIAA Guidance, Navigation, and Control Conference*, AIAA 2010-(August):1–18, 2010.

[12] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[13] Stijn B. J. Van Dam, An L.M. Abeloos, Max Mulder, and M.M. Van Paassen. Functional presentation of travel opportunities in flexible use airspace: An EID of an airborne conflict support tool. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 1(January):802–808, 2004.

[14] Yazdi I Jenie, Erik-Jan van Kampen, Coen C de Visser, Joost Ellerbroek, and Jacco M Hoekstra. Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 38(6):1140–1146, 2015.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Esa M. Rantanen and Ashley Nunes. Hierarchical Conflict Detection in Air Traffic Control. *The International Journal of Aviation Psychology*, 15(4):339–362, 2005.

[17] Carl Westin. *Strategic Conformance: Exploring Acceptance of Individual-Sensitive Automation for Air Traffic Control*. PhD Thesis. Delft University of Technology, Netherlands, 2017.

[18] Brian G Hilburn. Evaluating human interaction with advanced air traffic management automation. Technical report, NATO: Brussels, 2003.

[19] Marek Bekier, Brett R.C. Molesworth, and Ann Williamson. Tipping point: The narrow path between automation acceptance and rejection in air traffic management. *Safety Science*, 50(2):259–265, 2012.

[20] R Flicker and M Fricke. Improvement on the acceptance of a conflict resolution system by air traffic controllers. In *6th USA/Eur. ATM R&D Seminar, Baltimore, MD, USA*, 2005.

[21] Martin S Eby. A Self-Organizational Approach for Resolving Air Traffic Conflicts. *Lincoln Laboratory Journal*, 1994.

[22] B. Kirwan and M. Flynn. Investigating air traffic controller conflict resolution strategies. Technical report, EUROCONTROL, Brussels, Belgium, 2002.

[23] Selina Fothergill and Andrew Neal. Conflict-resolution heuristics for en route air traffic management. *Proceedings of the Human Factors and Ergonomics Society*, pages 71–75, 2013.

[24] Carl Westin, Clark Borst, and Brian Hilburn. An empirical investigation into three underlying factors affecting automation acceptance. *Proceedings of the Fifth SESAR Innovation Days*, pages 1–9, 2015.

[25] Kim J. Vicente and Jens Rasmussen. Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man and Cybernetics*, 22(4):589–606, 1992.

[26] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems*, 28(8):1814–1826, 2017.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[28] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation*, 1(4):541–551, 1989.

[29] Dean a Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems 1*, pages 305–313, 1989.

[30] Zhilu Chen and Xinming Huang. End-To-end learning for lane keeping of self-driving cars. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 1856–1860, 2017.

[31] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[32] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[34] Yann LeCun, R Pfeifer, Z. Schreter, F. Fogelman, and L. Steels. Generalization and network design strategies. Technical report, Elsevier, Zurich, Switserland, 1989.

[35] Federal Aviation Administration (FAA). Air Traffic Organization Policy (1-2-2). Technical report, Doc. No. JO 7110.65V, 2014.

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

[37] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[38] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *International Conference on Learning Representations*, Freiburg, Germany, 2015.

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[40] Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[41] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[42] Brian W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[43] David Martin Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[44] Giuseppe Jurman, Samantha Riccadonna, and Cesare Furlanello. A comparison of MCC and CEN error measures in multi-class prediction. *PloS one*, 7(8):e41882, 2012.

[45] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[46] Mica R Endsley. Level of automation forms a key aspect of autonomy design. *Journal of Cognitive Engineering and Decision Making*, 12(1):29–34, 2018.

[47] C.E. Billings. *Aviation automation: The search for a human-centered approach*. Lawrence Erlbaum Associates, Publishers, Mahway, New Jersey, 1997.

II

Preliminary Thesis

1

An overview of automation in Air Traffic Control

This chapter elaborates on the current state of Air Traffic Control (ATC) in combination with automation. Initially, only ATC is considered, specifically focusing on the current structure and tasks. Secondly, automation in ATC is introduced to see how the human controller can be assisted by machine systems. Finally, past and currents efforts of automation are discussed to define the main challenges and state-of-the-art.

1.1. Introduction to Air Traffic Control and CD&R

Air Traffic Management (ATM) is a term encompassing all systems that assist aircraft in safely and efficiently departing, transiting airspaces, and arriving at aerodromes. Among others, it includes Air Traffic Services (ATS), which in turn incorporates ATC. A lot has changed in ATC since the first ATCo was appointed in 1929. Spanning almost a century, the world has moved away from the checkered flag towards integrated Communications, Navigation & Surveillance (CNS) and ATM systems. This last development, which started in the 1990s, is considered to be *fourth generation* of ATM incorporating Future Air Navigation System (FANS). FANS allows for improvements concerning communication (e.g. ACARS[1] and CPDLC[2]) and surveillance. However, even though the interaction between aircraft and ATC is transitioning to digital, resolving conflicts still requires human intervention.

ATCos are tasked with maintaining a safe, orderly, and expeditious flow of air traffic in the global aviation system (ICAO, 2016). Larger aerodromes facilitate more than 10 different ATCo positions, which can be divided into three departments: Area Control Center (ACC), Approach Control (APP), and Aerodrome Control (TWR). The controllers in ACC are mostly concerned with en-route traffic, the APP controllers mainly focus on arrivals and departures, and the TWR primarily handles ground traffic (ICAO, 2016). This research focuses on ACC, as it is the area with least external constraints and automation in ATC is still in an early phase. An en-route controller has to perform 13 primary tasks (Seamster, Redding, Cannon, Ryder, & Purcell, 1993), of which the CD&R task is the most important for ATCos, according to Bekier, Molesworth, and Williamson (2012), which consequently makes it the main focus of this research.

The Conflict Detection & Resolution Task
An empirical study by Seamster et al. (1993) has identified four primary goals to avoid during CD&R, in descending order of priority:

1. Violation of minimum separation standards

[1] ACARS: Aircraft Communications Addressing and Reporting System
[2] CPDLC: Controller–pilot data link communications

2. Deviations from standard operating procedures
3. Disorder that may result in cognitive work overload
4. Making unnecessary requests of the pilot.

Following up on the primary goal, ICAO (2016) dictates a vertical and horizontal separation minimum of 1000 ft and 5 nm respectively under FL 410 in designated airspace, called the protected zone. In case aircraft are on a trajectory where these minimums will be violated, the aircraft are said to be in *conflict*. When the minimums are actually violated, Loss of Separation (LoS) occurs. To ensure this is never the case, ATCos use extensive strategies and internal rules for prevention, detection and resolution of conflicts. An elaboration on ATCo strategies is given in section 3.1.2.

1.2. The need for automation

One of the major challenges in aviation today is to maintain efficiency, safety and reliability of ever increasing air traffic globally (Agogino & Tumer, 2012). According to IATA's 20-year Passenger Forecast, air traffic is expected to double to 7.2 billion passengers annually within two decades[3]. While an advanced scheduling system is in place, small deviations in aircraft trajectories due to weather or airport conditions are difficult to accommodate and consequently cause large overall delays. For example, in 2017, only 80% of U.S. flights arrived on time[4]. The combination of slow responses to changes and fast increasing air traffic overall causes the ATM system to be at maximum capacity. As infrastructure (e.g. number of airports) will not increase significantly and hardware (e.g. computing power) only fixes part of the problem (Agogino & Tumer, 2012). This is caused by the fact that the primary constraint that limits the capacity of an airspace sector is the human controller, specifically his workload concerning conflict detection and resolution (Erzberger, 2004). Because a controller can only monitor around 15 aircraft at a time, previous strategies to reduce workload involved subdividing airspace sectors. However, due to physical constraints (maneuverability of aircraft) and an increased inter-sector coordination workload, this is not a profound solution (Erzberger, 2004).

Taking into account these capacity issues and the fact that the system is built upon 50-year old principles, it becomes clear that the ATM infrastructure could fundamental changes. (Erzberger, 2004) Specifically, a more decentralized approach seems to be a promising domain to explore, following multiple proposed solutions such as (hybrid) multi-agent systems and the 'free flight' concept. (Agogino & Tumer, 2012; Hoekstra et al., 2002; Nguyen-Duc, Briot, & Drogoul, 2003; Tomlin, Pappas, & Sastry, 1998) Although free flight appears to be promising, "the distributed nature and the infinite number of conflict geometries make it very hard to estimate the actual safety level compared to a centralized system." (Hoekstra et al., 2002). Due to strict certification criteria in the aviation sector, this proof is a necessity before the concepts can be implemented in everyday operations. In the mean time, it seems that more pragmatic solutions that lie closer to the current procedures need to be implemented. In any case, ATM has reached a level of technology where only the most complex elements of the human controller are left to be modeled, namely decision making and strategic control, making it one of the most ambitious attempts of automation (Hilburn, 2002).

1.3. Drawbacks and hurdles of automation

Although automation is inevitable in the future of ATM, important disadvantages and hurdles have to be overcome. They can be divided in two categories: acceptance issues and difficulties that are inherent to automation. Both are discussed below.

According to a literature review performed by Westin et al. (2016), the following conclusions can be drawn regarding trust and automation: "a) trust in automation develops over time as a result of prolonged experience, b) acceptance and operator performance decrease when the authority and autonomy of automation increase, and c) acceptance and operator performance benefit from automation actively involving the operator in the control and decision-making loop." (Westin et al., 2016).

[3] Source: *IATA Forecasts Passenger Demand to Double Over 20 Years* - IATA (Press release 59, 2016)
[4] Source: *On-Time Arrival Performance U.S.* - Bureau of Transportation (2017)

This, however, does not take into account initial acceptance and trust in a system. Paradoxically, “an operator might only develop trust after using a system, but might also be unwilling to trust a system he/she has not used." (Westin et al., 2016). This stipulates the importance of a concept called *compatibility*, which is defined as the perceived fit of technology within the context in which it is used, driven by the user's values, experiences, and needs (Rogers, 1983). This can directly be translated back to the multi-actor ATM solutions. As they are less compatible than intermediate supporting centralized automation, they are less likely to be accepted by the user, i.e. the ATCo. Concluding, one would ideally automate following the reasoning of the end user, a concept that is introduced in chapter 2.

The second category of drawbacks and hurdles consist of the inherent difficulties of automation. These ‘ironies of automation’ are well-described by Bainbridge (1983). The effects on human performance can be summarized as: complacency (over-reliance), vigilance problems (reduced alertness), and skill degradation & transient workload peaks. These difficulties can be partly counteracted by consciously applying Levels of Automation (LoA) frameworks. Many variants have been proposed, such as Sheridan and Verplank (1978) or Endsley and Kiris (1995), which can be used both statically as well as in an adaptive or adaptable manner (Kidwell, Calhoun, Ruff, & Parasuraman, 2012). When designing novel automation, these LoA models should be considered. Specifically for this project, regarding the maturity of machine learning technology and the impact of failures, a the lowest form of automation – decision support (level 2) – would be advised initially.

1.4. Past automation efforts

Because the current method of controlling air traffic is not scalable to the extend of the expected growth in traffic (Erzberger, 2004), automation has been a major domain of interest in ATM. Several articles have been written on a vision of the role of automation in future ATM, going beyond the propositions of SESAR[5] and NextGen[6] (Sáez Nieto, 2016). More concretely, multiple projects were undertaken to automate ATCo workflow, such as ARC2000, and CORA.

ARC2000 (Dean, Fron, Miller, & Nicolaon, 1995) was an elaborate project which aimed to determine the feasibility of automating air traffic control. A main challenge in building the simulator was CD&R, as there were no reliable algorithms available. In the process of creating these, a concept called ‘forbidden areas’ was introduced, which translated the trajectory of an intruding aircraft to a region that should be avoided, similar to the SSD (see section 3.2).

CORA (Controller Resolution Assistant) was developed within EUROCONTROL (Kirwan & Flynn, 2002) and is based on decision-making heuristics which are found to be widely used by ATCos Fothergill and Neal (2013). Although CORA was based on human controllers preferences, it could only match the decision-making style of a subset of them. This was due to the fact that ATCos have different strategies of solving similar conflicts (Westin, 2017, p. 24).

More recently in the U.S., Data Comm is being implemented for en-route traffic in 2019 providing a digital communication link between pilots and controllers. “Controllers will be able to reroute, hand off aircraft to the next center, and send messages to change altitude."[7] This digital communication link is a critical step towards ATC automation as the human voice element is taken out of the equation.

In the end, the main question is not whether, but in which manner to automate. A large variety of automation methods have been proposed, including multi-agent self-separation (Agogino & Tumer, 2012; Hoekstra et al., 2002; Nguyen-Duc et al., 2003; Tomlin et al., 1998), fuzzy logic (Pineau, 2018), adaptive automation (Hilburn, Jorna, Byrne, & Parasuraman, 1997) and Reinforcement Learning (Cruciol, Weigang, De Barros, & Koendjbiharie, 2014; Weigang, de Souza, Crespo, & Alves, 2008) solutions.

A recent publication by Regtuit et al. (2018) aims to automate CD&R in a strategic conformal way (chapter 2) using machine learning (chapter 4). In this study, Regtuit et al. consider two-dimensional horizontal conflicts with one conflict pair (i.e. two aircraft). Data is generated artificially by purposely using distinctive strategies,

[5] SESAR: Single European Sky ATM Research (EU, EUROCONTROL)
[6] NextGen: Next Generation Air Transportation System (FAA)
[7] Source: *Air Traffic Controller Workforce Plan* - Federal Aviation Administration (FAA) (2018)

such as always steering the controlled aircraft in front or behind the intruding aircraft. These strategies are extracted from the entire dataset using a clustering algorithm and are subsequently used to train a reinforcement learning algorithm with three states.

Regtuit et al.'s exploratory study offers a first proof-of-concept of using machine learning to achieve strategic conformal automation for ATC. Yet, opportunity for improvement and further research is evident. Firstly, more advanced scenarios including more aircraft can be considered. Secondly, the data should be generated in a natural, bias-free manner using professional ATCos. And lastly, more than three states should be considered, as the CD&R problem is characterized by more than three parameters.

The aim of this thesis is to work towards strategic conformal automation by building upon the insights of previous authors and improving on the opportunities that arise. The concept of strategic conformance is introduced in the next chapter.

2

Strategic conformance: individual-sensitive automation

As discussed in section 1.3, one of the major hurdles in automation advances is human acceptance. One concept that aims to improve this is *strategic conformance* (Westin, Hilburn, & Borst, 2011). This chapter will focus first on the theoretical background, advantages and disadvantages of strategic conformance in section 2.1. Subsequently, the practical implications for determining the performance of conformal automation are discussed in section 2.2.

2.1. Introduction to strategic conformance

Strategic conformance was introduced as a high-level form of *compatibility*, specifically where the problem-solving style of the automation is designed to match the human controller's. This section elaborates on strategic conformance from both a theoretical and an empirical standpoint.

2.1.1. Compatibility

When discussing strategic conformance as a means to increase automation acceptance, it is key that a commonly accepted view of this concept is presented. Although many models exist, the Technology Acceptance Model (TAM) (Venkatesh, Morris, Davis, & Davis, 2003) appears to be the accepted standard. This model was later augmented with knowledge of automation by Ghazizadeh, Lee, and Boyle (2012) to create the Automation Acceptance Model (AAM) (Figure 2.1). This model implies that the main factors that influence acceptance, and thus actual system use, are compatibility and trust. The diagram shows that acceptance is not steered by a linear relation but is rather incorporated in a reinforcing loop; by using the system, trust and compatibility may increase over time. It must be recognized that conformal automation does not infer anything about the efficiency or optimality of the solution. Combining these two statements; strategic conformal automation could be used to increase acceptance of a novel conflict resolution system while it gradually transitions to a more efficient (e.g. less additional track miles) solution. (Westin, 2017, p. 30-35)

Human-machine compatibility appears to be the underlying framework to achieve automation acceptance. It can be evaluated at multiple levels, as synthesized by (Westin, 2017, p. 29). Figure 2.2 shows various categories of tasks with an increasing cognitive demand. To the right of the categories, descriptions are given that assist the human controller in achieving these tasks. At the highest, most abstract level, strategic conformance is offered as a means to achieve compatibility between man and machine on a decision-making level.

This higher level of compatibility mainly consists of two elements where strategic conformance exceeds traditional compatibility. Firstly, strategic conformance not only mimics obvious communication forms or

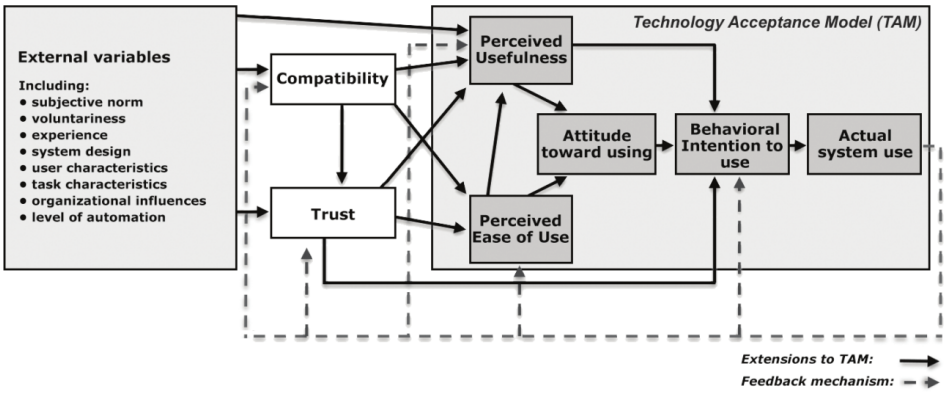


Figure 2.1: The Automation Acceptance Model. Adapted from Ghazizadeh et al. (2012).

appearance of humans, but aims to imitate the underlying decision process. Secondly, strategic conformance acknowledges that humans have different problem-solving styles, which makes it unrealistic to view humans as a homogeneous group (under certain circumstances). (Westin, 2017, p. 30–31)

The underlying hypothesis is that when a decision aid proposes a solution that is conformal with the intended resolution of the controller, it will appear to have followed a similar rationale. This could take away the cognitive workload that is caused by the process of understanding why the automation is behaving in a particular way. (Westin, 2017, p. 30)

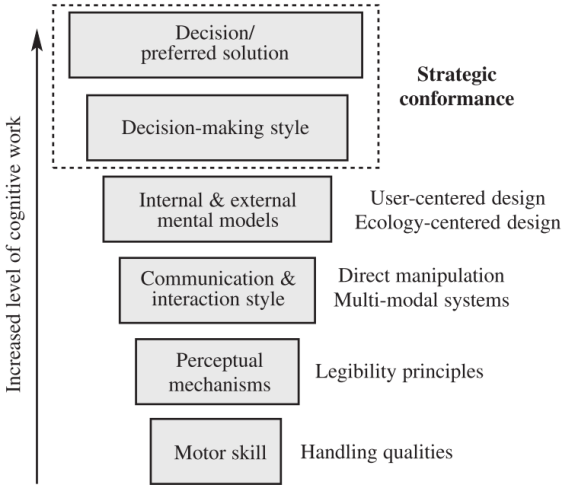


Figure 2.2: Levels of human-machine compatibility. The inverse pyramid shows tasks with increasing cognitive demand and the descriptions on the right offer methods to assist the human in achieving these tasks. Adapted from (Westin, 2017, p. 30).

Individual preferences

The foundation of strategic conformance lies in the diversity of solving conflicts among controllers. As an example to show multiple solutions to the same conflict, a right angle conflict with a Closest Point of Approach (CPA) of zero nautical miles is shown in Figure 2.3. A common strategy is to steer one aircraft behind the other, as it requires less monitoring (Fothergill & Neal, 2013). Another common strategy involves a heading change of both aircraft, to share the track deviations (Westin, 2017). One can imagine that the space of possibilities only increases when conflicts become more complex, including multiple aircraft and a third dimension.

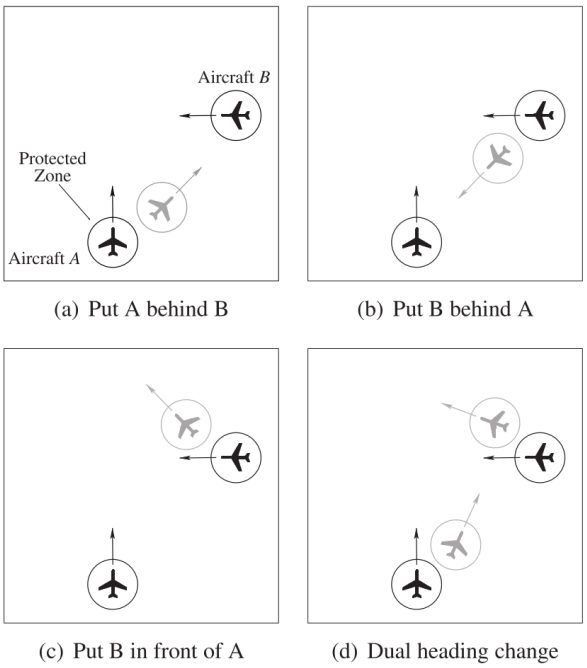


Figure 2.3: Possible conflict solutions to a right angle problem. Adapted from (Westin, 2017, p. 32).

2.1.2. Empirical evidence

To prove whether strategic conformance is more than a theoretical framework, multiple empirical studies were conducted (Hilburn et al., 2014; Westin et al., 2016).

Westin and his co-authors are the first to perform empirical work “specifically focused on differences in decision aid problem solving styles and its effect on individual operator acceptance." (Westin, 2017, p. 20). In acknowledging the benefits of strategic conformal automation, two main assumptions are made. One, ATCos accept resolutions significantly more often when they are conformal to their own problem-solving style and two, ATCos are relatively consistent in their actions.

The first assumption was tested by Hilburn et al. (2014). Sixteen subjects used the SSD (section 3.2) to resolve conflicts in multiple scenarios without considering altitude. Two weeks later, the subjects performed an identical task, only now supported by an advisory system that replayed their own decisions in the same situations. It was found that “conformal advisories (exact replays of a given controller’s previous solution) were accepted more often, rated higher, and responded to faster than were non-conformal advisories (replays of a colleague’s different solution)." (Hilburn et al., 2014). Specifically, the conformal advisories were accepted in 76% of the cases whereas non-conformal advisories were only accepted 57% of the time. This indicates the striking result that even conformal advisories are rejected 24% of the time. To understand this result, three follow-up experiments were performed, testing the effects of problem-solving consistency, source bias, and interface representation. However, analyzing the simulation data with relation to source bias and interface representation did not result in significant results. (Westin, Borst, & Hilburn, 2015) Results regarding consistency are discussed in subsection 2.2.1.

2.1.3. Drawbacks

Strategic conformance aspires to increase compatibility and trust, and thus actual system use to ultimately reduce workload. However, in certain situations, individual problem-solving styles may not be desired. Besides, strategic conformance requires prerequisites that may not be valid in all cases.

One situation in which a homogeneous decision-making process is advantageous is when ATCos work in teams (Westin, 2017, p. 34). This might be the case when there is high traffic flux between two adjacent

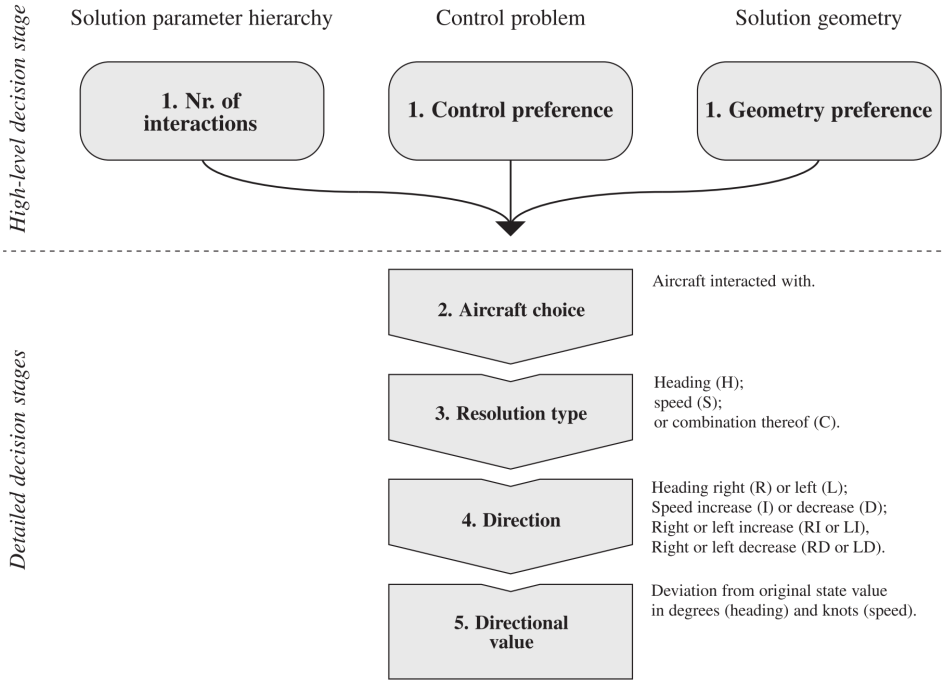


Figure 2.4: The consistency classifications framework can be used to measure consistency or conformity for different decision stages. Adapted from (Westin, 2017, p. 121).

sectors, when a team of a tactical and planning controller work together, or when a controller hands off his work to a colleague after a shift.

The prerequisite of achieving strategic conformance is that individual controllers are consistent, while *consensus* is not required. Consistency is defined as individuals taking the same action under the same circumstances. If this is not the case, an automation can never be 100% conformal. Consensus regards the entire group of controllers. Traditional automation usually devises a single strategy for all users. When considering strategic conformance, it is assumed that there is no mutual consensus among all controllers to solve a problem in a certain way, otherwise individual-sensitivity loses its value and a homogeneous algorithm is sufficient.

2.2. Quantifying strategic conformance

In order to ultimately rate the performance of the automation, some definitions require to be quantified. Specifically, this section aims to make the concepts *consistency* and *conformity* more concrete. Controller consistency is used to evaluate whether a controller is consistent enough to create individual-sensitive automation, and finally conformity is used to score the performance of the trained algorithm, as it is the goal of this thesis to create strategic conformal automation.

2.2.1. Consistency

ATCo consistency is one of the major assumptions that enables the relevance of strategic conformal automation. If controllers are not consistent over time, they will likely not agree with individual-sensitive automation. To obtain a framework for consistency, Westin analyzed resolution strategies identified by Fothergill and Neal (2013) and more than 500 conflict resolutions collected in real-time simulations (Westin, 2017, p. 121). He concluded that ATCo consistency can be measured on different levels of abstraction, as summarized by Figure 2.4.

Westin used this framework in an empirical study to analyze controller consistency (Westin, Hilburn, & Borst,

2015). The aim of the experiment was to measure the degree of consistency for repeated conflicts over time (intra-rater variability), and the degree of agreement within the group on resolutions (inter-rater variability). Results showed that “controllers were consistent, but disagreed on how to solve conflicts". Consistency was especially prevalent in the higher-level decision stages of Figure 2.4 (levels 1, 2 and 3) and decreased in the detailed decision stages (levels 4 and 5). Creating conformal automation based only on low-level decisions might therefore not be the optimal approach. To clarify, the *control preference* is defined as how the controlled aircraft will pass the intruding aircraft (behind, in front, under or above). The *geometry preference* is the spatial relationship, irregardless of what is the controlled aircraft (e.g. the aircraft are behind each other or above each other).

Additionally, Westin concluded that expert controllers were slightly more consistent compared to their trainee counterparts. This indicates that it would be beneficial to use expert controllers to create conformal automation, while trainees are not necessarily unusable.

The next section applies Westin's framework for consistency to determine the degree of strategic conformance of an automation.

2.2.2. Conformity

As seen in the previous section, consistency can be measured on multiple levels. When this line of thought is extended, *conformity* can be measured at similar degrees of abstraction. A devision in levels can be made between high-level and detailed decision stages, as shown in Figure 2.4. In measuring conformity, this line has a very practical implication; the detailed decision stages consist of one action, while the high-level stage aggregate multiple actions.

Measuring detailed decisions is relatively straightforward. Given the current state, a controller selects an aircraft, a resolution type, a direction and a directional value and subsequently performs this action. This enables the state to be linked to an action, creating a labeled dataset. To measure if a trained algorithm is conformal with the original controller, this dataset is used to validate the policy. The dependent variable *accuracy* is measured by whether the algorithm and the controller perform the same action in the same situation. This approach assumes that an ATCo bases his or her decision for an action only on the current state (independent of previous states and actions). This enables us to treat the problem as an Markov Decision Process (see subsection 4.2.1). A requirement for this assumption is consistency. Referring back to the previous section, ATCos are not always consistent in comparable situations. However, consistency seems to increase with increasing experience level (Westin, 2017, p. 121). Empirical insight is required to prove that ATCos are sufficiently consistent to base automation on (see Part I).

Measuring high-level decisions is slightly more elaborate, as it involves more than just state-action pairs. The high-level decision process is divided into three categories: the number of interactions, the control preference and the geometry preference. Conformity can determined for all three categories by having the automation and an ATCo perform the same (short) scenarios: the first category is measured by summing all interactions with aircraft, the second will measure how aircraft pass each other in conflicts (behind, in front, under or above), and the third compares spatial relationships.

Spatial relationship can be defined in numerous ways. Comparing relations between e.g. conflict angle and CPA work well for pair-wise conflicts (Regtuit et al., 2018), but do not translate well to more complex situations. One solution is to define geometry by statistical parameters (mean, median, variance) of the heading, altitude and speed of all aircraft. For example: if aircraft fly in an orderly flow, the variance in heading and speed is low. If a controller prefers to separate aircraft on different levels, the variance in altitude is high. Another spatial measure could be the amount of sector space that a controller uses. Orderly flows will direct aircraft in similar paths, while in an unorganized sector, aircraft can be all over the place. A third geometric preference is the safety margin that ATCos accept. One could measure the CPAs throughout a scenario to obtain the acceptable proximity of aircraft to each other. Conversely, on could also measure the additional track miles with respect to the optimal path.

To conclude, it is assumed that when automation achieves comparable results in all five consistency categories of Figure 2.4, an ATCo will experience the automation to be strategically conformal.

3

Extracting strategy from data

To automate the CD&R task in a conformal manner, the algorithm requires features that incorporate the strategy of the human controller. Preferably, the algorithm and the ATCo would use the same features or parameters to base their strategy on. This will enable the human and the automation to work from a shared mental model, which could improve conformity. This chapter examines the factors that influence a human controller during detection and resolution as well as keeping an orderly flow. Additionally, section 3.2 proposes a visual tool to assist with these tasks – the Solution Space Diagram (SSD) – that is hypothesized to incorporate the aforementioned factors into one visualization.

3.1. Sector management

Managing a sector consists of two elements; CD&R and keeping an orderly flow towards the exit point. This section initially discusses sector and conflict detection parameters after which the resolution strategies a controller uses are elaborated on.

3.1.1. Sector and conflict parameters

A literature review by Regtuit et al. (2018) analyses the factors that impact conflict detection by ATCos during CD&R. Although ATCos tend to solve conflicts in a pair-wise approach, their strategy is also determined by a global approach through assessing the impact on remaining conflict pairs (Kirwan & Flynn, 2002). For this reason, Regtuit et al.'s list of parameters was extended with parameters that affect the entire sector, as synthesized in Table 3.1. To provide an intuitive sense of the top three parameters that determine a pair-wise conflict, a visualization is shown in Figure 3.1.

Literature showed that ATCos do not only solve conflict after conflict, but actively direct traffic in a flow to lower its complexity and thus the controller's workload. A clear representation of the difference between traffic density and complexity is shown in Figure 3.2; Both scenarios show an equal number of aircraft in identical locations. However, in the right-hand case, some aircraft headings are slightly changed, creating a less organized sector which is evidently more difficult to monitor. (Van Gent, Hoekstra, & Ruigrok, 1997)

A successful strategic conformal automation algorithm should take at least these factors into account in attempt to imitate human detection strategy.

3.1.2. Resolution strategies

Detection of conflicts is currently assisted by automation. Nonetheless, controllers need to assess priority and thus precedence per conflict themselves. Resolution of conflicts is performed following three mechanisms.

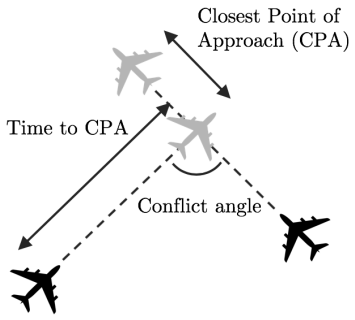


Figure 3.1: A visualization of three CD&R parameters, CPA, t_{CPA} and conflict angle.

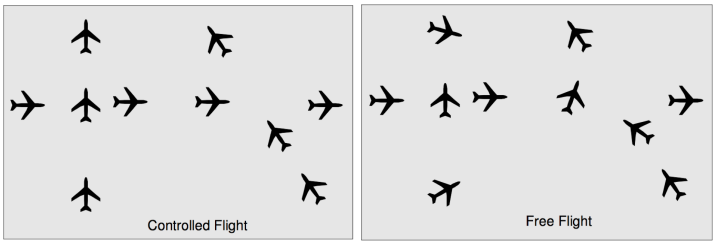


Figure 3.2: Two sectors with an equal number of aircraft. One with route structure (left) and the other without (right). Density is equal, while traffic complexity is not. Adapted from Van Gent et al. (1997).

Table 3.1: CD&R and sector parameters with their effect on ATCo strategy. The top four parameters are applied to conflicts, i.e. single conflict pairs. The bottom parameters are applicable to the entire sector. Adapted and extended from Regtuit et al. (2018).

Parameter	**Description**	**Reference**
CPA	Closest Point of Approach; the minimal distance at any point in time between to aircraft. Controllers use this to asses whether aircraft are in conflict at all and if they have to interfere.	Xu et al. (2016)
Time to CPA	The time until CPA. A smaller t_{CPA} results in higher a detection rate and naturally a higher priority.	Rantanen et al. (2004); Remington et al. (2000)
Conflict angle	The heading angle between the controlled aircraft and the conflicting aircraft. Increasing conflict angles lead to higher difficulty of detection.	Kimball (1970); Remington et al. (2000)
Relative velocity	The relative speed vector of conflicting aircraft. Speed differences between objects are more resource limited than relative distances.	Law et al. (1993)
Traffic density	The situational awareness, and thus detection rate of conflicts, decreases rapidly when the number of aircraft exceeds a certain number.	Endsley and Rodgers (1998)
Traffic complexity	Comparable to traffic density, traffic complexity is a multiplier of traffic density and decreases detection rate (Figure 3.2)	Galster et al. (2001)
Exit waypoint	Vectoring the aircraft towards their exit waypoint (or hand-off point) is a main ATCo goal which influences strategy and sector organization.	Endsley and Rodgers (1994), Appendix A

According to Rantanen and Nunes (2005) the hierarchical sequence that controllers use to solve a conflict is:

1. An altitude change
2. A heading change
3. A speed change

The preferred resolution is an altitude change because it requires the least monitoring thus reducing workload (Rantanen & Nunes, 2005). On the other hand, speed changes are less effective, due to the small speed envelope of commercial aircraft at high altitudes (Kirwan & Flynn, 2002).

Within these high level options, more elaborate strategies arise. To assess the decision-making style of an ATCo, their main goal needs to be evaluated, which is to safely expedite and maintain an orderly flow of air traffic (ICAO, 2016, p. 1-3). In order to achieve this, specific strategies and principles are followed which are listed by Kirwan and Flynn (2002). A selection of the most important strategies is shown in Table 3.2, accompanied by a metric to quantify the principle. These quantifications can be used to compare ATCos to automation in order to measure conformity. Additionally, these metrics could be used to assign numerical values to an ATCo's performance to subsequently define a reward function for a RL algorithm (section 4.3).

Measuring total performance or efficiency can be an ambiguous task because the main goal consists several trade-offs. For example, the number of additional track miles should be minimized, while at the same time safety margins should be as high as possible. To solve this trade-off, the priorities defined by Seamster et al. (1993) are used, as shown in section 1.1.

Table 3.2: A selection of ATCo strategies and principles as listed by Kirwan and Flynn (2002) with a metric to quantify these aims.

Strategy or principle	**Metric**
“The bottom line is safety" Need more than 5nm to be safe	Closest Point of Approach (CPA)
“Reduce the complexity (eliminate problems)"	% free SSD area
“Minimise the number of aircraft to move"	Number of different aircraft that are controlled
“Look for one key action that will resolve the situation"	Number of resolutions in total
“Minimize additional track miles flown"	Deviation from original path in nautical miles

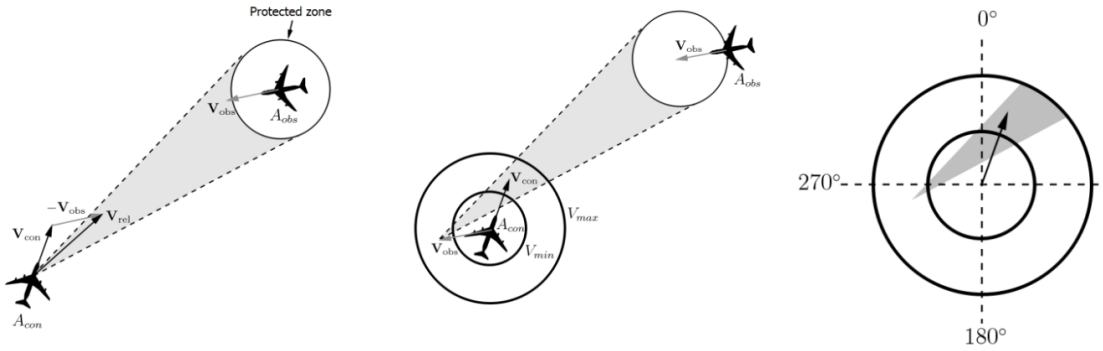
3.2. The Solution Space Diagram

This section introduces the Solution Space Diagram (SSD). It has been conceived following Ecological Interface Design (EID) principles which implies certain benefits that are discussed in subsection 3.2.1. After this, the SSD is introduced and linked to the strategic parameters as discussed above.

3.2.1. Ecological Interface Design

This section introduces the concept of EID as a paradigm to increase situational awareness and better problem-solving performance. In the next section, an application of EID, the Solution Space Diagram (SSD), is introduced.

The EID principles have been introduced in the field of Process Control but are applicable to a large variety of domains. (Vicente & Rasmussen, 1992) EID “addresses the cognitive interaction between humans and complex sociotechnical systems." (Van Dam, Mulder, & van Paassen, 2008). During the design of new automation or instrumentation, the imposed constraints of the actual work environment (“ecology") of the user are taken in account. Research has shown that this improves worker adaptation and consequently results in better problem-solving performance compared to other principles (Vicente, 2002). EID is built-up from a combination of *structure* and *form*. The structure, or content, entails a deep understanding of the end-user's work domain, including relationships, degrees of freedom, constraints and hence the solution space. This analysis provides a basis for the possible actions which are goal-directed (Van Dam et al., 2008). The second element of EID, the form, focuses on how to show this solution space to the user. Through meaningful



(a) The Forbidden Beam Zone (FBZ) definition.

(b) Displacement of the FBZ by intruding aircraft's velocity.

(c) Projection of FBZ constrained by the aircraft's velocity limits.

Figure 3.3: Basic solution space construction. Adapted from Mercado-Velasco et al. (2010) and Westin et al. (2011)

and functional visualization, the user is able to intuitively[1]see the solution space and take the appropriate action. Practically, this involves making normally hidden system dynamics more visible to the user to increase situational awareness (Van Dam et al., 2008).

3.2.2. Introduction of the SSD

The SSD is an ecological decision support tool that integrates various critical parameters of the CD&R problem. The concept was first introduced in an aviation context by Van Dam et al. (2004), who used the SSD as a self-separation tool.[2] A representation of the SSD is given in Figure 3.3. The inner and outer circle denote the minimum and maximum velocity of the aircraft respectively. The arrow indicates the *controlled* aircraft's heading and speed. The colored geometric shapes (triangles) each indicate a potentially conflicting aircraft (*intruder*). When the velocity vector lies in this colored 'no-go zone', the aircraft is indeed in conflict and a resolution is required. The non-colored zones are thus, by definition, the solution space for the aircraft.

The SSD is constructed by defining the Forbidden Beam Zone (FBZ) per observed aircraft, which comprises all relative velocity vectors that lead to a loss of separation. When the FBZs are displaced by the velocity vectors of these observed aircraft, the SSD is obtained. A more detailed explanation of the construction of the SSD can be found in Mercado-Velasco et al. (2010), who was one of the first to introduce the SSD in combination with Air Traffic Control.

In past research, the SSD has been applied for two main objectives: creating sector metrics and improving human performance. It has been used to measure sector complexity and as predictor of performance and workload (Rahman, Borst, Mulder, & Van Paassen, 2010). Research has shown high correlations between solution-space properties, self-reported task difficulty (Hermes et al., 2009) and subjective workload (D'Engelbronner, 2010). Secondly, the SSD improves situational awareness and system understanding according to human-in-the-loop experiments (Borst et al., 2015; Van Dam et al., 2008). In addition, an experiment which involved merging aircraft into a single route, has shown significant effects of the SSD on the reduction of controller workload (Mercado-Velasco et al., 2010).

It is hypothesized that the SSD allows the human and machine to work in a shared work domain (ecology). Automation based upon the same SSD parameters a human uses (through visual input), may be more conformal and more human-like. The SSD also offers a more 'transparent'[3] decision process than ordinary automation might. It should be noted, however, that higher transparency does not necessarily lead to higher acceptance (Göritzlehner et al., 2014) and that too much transparency results in a higher workload and thus lower Situation Awareness (SA) (Duggan, Banbury, Howes, Patrick, & Waldron, 2004).

[1] Although the aviation domain features 'intuitive' EID displays, it is a common misconception to interpret 'ecological' as 'intuitive'. The systems are designed to be used by experts and may require extensive training. (Borst, Flach, & Ellerbroek, 2015)

[2] Before Van Dam et al.'s aviation interpretation, the concept of the SSD, or 'state vector envelope', was first introduced as *velocity obstacles* in robotics research (Fiorini & Shiller, 1998).

[3] Mark and Kobsa (2005) define transparency as understanding of the reasoning and behavior of automation by the human operator.

Table 3.3: Parameters incorporated in the SSD; with CD&R (top) and sector (bottom) parameters.

Parameter	**Visualization in the SSD**
CPA	The CPA is reflected by a translating FBZ.
Time to CPA	A smaller t_{CPA} results in a wider FBZ. Additionally, it can be indicated with color coding.
Conflict angle	The conflict angle is reflected by the inclination of the FBZ.
Relative velocity	Relative speed is included by the rate of increasing width of the FBZ.
Traffic density	Each aircraft is represented by a separate FBZ in the SSD.
Traffic complexity	With higher complexity, the FBZs of separate aircraft overlap less and will occupy more area of the SSD.
Exit waypoint	The exit waypoint is visualized using a strikingly colored heading vector within the speed limits.

3.2.3. Parameters incorporated in the SSD

Table 3.3 shows the same parameters as introduced in Table 3.1, only now with their visualization in the SSD. It seems that CPA, time to CPA, conflict angle, relative velocity, traffic density and traffic complexity are all visually represented in the SSD. On the other hand, the important exit waypoint is not included in the SSD and should be passed in manually, either as visual addition or as separate numerical value.

To conclude, this chapter evaluated the decision-affecting parameters from literature and related these to the SSD visualization. Based on these relations, it can be hypothesized that the SSD incorporates sufficient information to contain the basis of decision-making strategy. However, two limitations must be remarked. First, the exit point is a strategy-determining feature that should be included in the algorithm. And second, although some sector information is reflected in the SSD, the complete picture is not fully observable. To include more sector information, a sector overview image or a vector with all aircraft states could be passed into a learning algorithm.

4

Machine Learning techniques

Artificial Intelligence is the simulation and demonstration of intelligent behavior in machines. One of the major goals in this field is to create autonomous agents that can learn from their environments thus learning optimal behavior, frequently through trial and error. (Arulkumaran, Deisenroth, Brundage, & Bharath, 2017). A sub-category of Artificial Intelligence (AI) is Machine Learning (ML), which enables computer systems to improve their performance on a task without being explicitly programmed.

According to Sutton and Barto (1998), ML can be subdivided into multiple paradigms, consisting of at least Supervised Learning (SL), Unsupervised Learning (UL) and Reinforcement Learning (RL). Figure 4.1 shows these three types of learning, including subcategories and examples of algorithms. It is difficult to devise a one-size-fits-all taxonomy for ML algorithms as learning type, method, and purpose are not logically ordered. Moreover, certain methods – such as neural networks – find their use in all learning types.

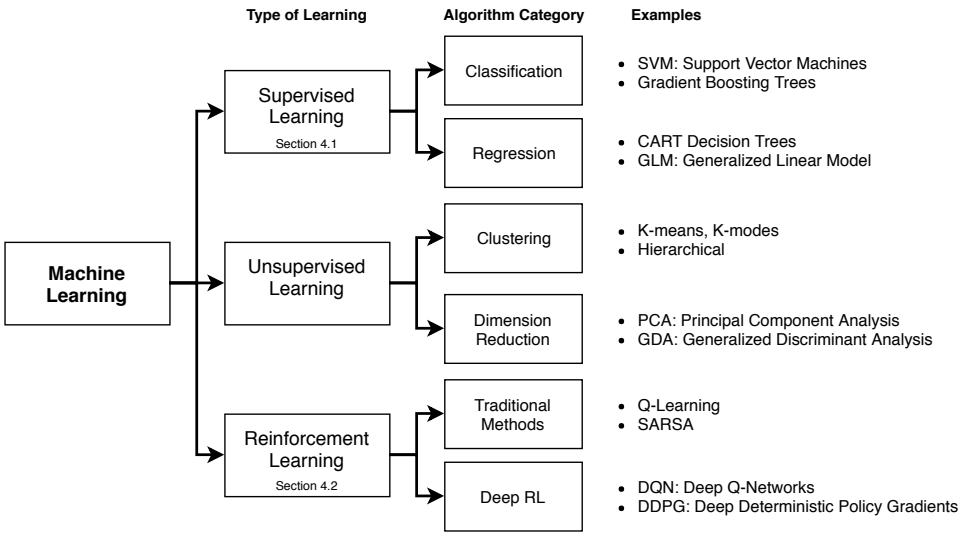


Figure 4.1: A taxonomy of Machine Learning types, subcategories, and examples of algorithms. Both supervised learning and reinforcement learning are elaborated on in this chapter.

Supervised learning requires a set of labeled data with linked in and outputs. Through extrapolation or generalization, these algorithms can provide useful results to data that is not present in the training set. However, this limits use-cases to situations where prior knowledge of the system exists, even though learning would be most beneficial in situations where agents do not have specific information and have to interact to learn. (Sutton & Barto, 1998) On the other hand, unsupervised learning can learn without any prior

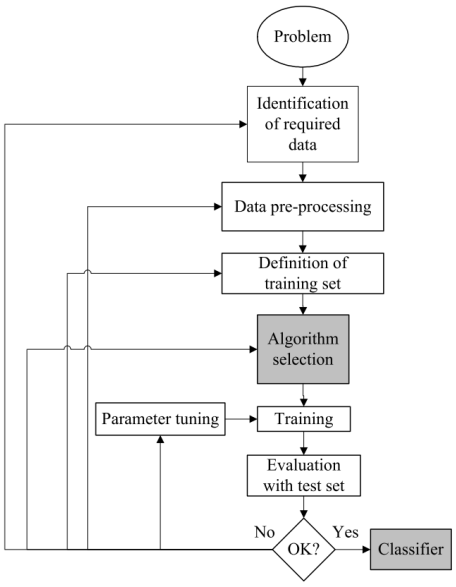


Figure 4.2: The problem-solving process for supervised learning. (Kotsiantis, 2007)

knowledge of the system. Usually, this terminology is used to define classification algorithms, i.e. finding clusters or patterns in unlabeled data. RL does not fit in any of these two categories and is thus categorized as its own paradigm. RL learns and improves by interacting with the environment and is reward driven, whereas the other methods are not. Unsupervised learning is used for both dimension reduction and clustering. Dimension reduction is a useful technique when reducing the complexity of input data to enhance algorithm performance. In the case of this research the data consists only of a few parameters or an SSD image (Section 3.2), which renders it less relevant for this application. On the other hand, clustering is the task of grouping data together that is more similar compared to other data points. This technique can be used to identify and group controller strategies, as was shown by Regtuit et al. (2018). This is, however, not the aim of this research.

Please note that the objective of this research is not to improve on existing machine learning methods, but rather to apply the best suited one to achieve strategic conformal automation in ATC. The goal of this research is imitating expert behavior or preference, for which SL seems best suited as the algorithm is trained based on expert data (section 4.1). On the other hand, recent successes were made in imitation learning using various methods of RL, which is therefore covered in section (4.2).

4.1. Supervised Learning

SL aims to classify or regress data given an input and target database. This labeled data is used to train a model to subsequently predict values that are not in the original data set. Figure 4.2 shows the typical supervised learning process where the training set contains input and target data. In context of conformal resolution advisories, the input would be the state space of the current conflict situation and the target data would consist of the resolutions (or actions) taken by controllers in those states. The assumption is that the model will be able to predict conformal resolutions for new situations even when state space conditions vary from the original input data.

The review paper by Kotsiantis (2007) defines four principal categories of supervised learning: logic based, perceptron based, statistical learning, and a Support Vector Machine (SVM). Six techniques that fall into these categories are shown in Table 4.1, including their main advantages and disadvantages. Although Decision Trees and Rule-Learners provide transparency, the overall performance of these algorithms is relatively low. Based on a preference for accuracy, neural networks and SVMs seem promising. Caruana and Niculescu-Mizil (2006) confirm this result and conclude that neural nets, SVMs and bagged trees yielded the best performance.

Table 4.1: Overview of traditional supervised learning methods with their main advantages and disadvantages. Summarized from (Kotsiantis, 2007)

Method	Advantage	Disadvantage
Decision Trees	Transparency	Accuracy and tolerance to interdependent or redundant attributes
Neural Networks	Accuracy	Transparency, overfitting, tolerance to irrelevant data
Naive Bayes	Speed of learning, transparency, incremental learning	Accuracy, tolerance to interdependent or redundant attributes
k-Nearest Neighbor	Speed of learning and incremental learning	Speed of classification, tolerance to missing values and interdependent attributes
Support Vector Machine (SVM)	Accuracy, tolerance to irrelevant attributes	Speed of learning, transparency
Rule-Learners	Transparency	Overall low performance

A main advantage of NNs is that they are very flexible in the types of data they accept. They accept a broad range of data structures without the specific need to manually pre-engineer features, although it does improve computational performance. This makes them adaptable to a broader range of scenarios, e.g. a varying number of aircraft per conflict. Contrarily, a major advantage of SVMs is that this method requires less hyperparameters to be tuned, resulting in a workable model without the use of extensive grid-searches. Additionally, SVMs are less likely to overfit and they theoretically converge to the global optimum due to quadratic programming (Cortes & Vapnik, 1995).

On the other hand, both methods endure the drawbacks of being black-box models, making it difficult to verify and validate them. Especially in safety-critical domains – a term applicable to this research – this is currently considered unacceptable (Jacobs, 2015). Despite this drawback, neural networks have risen in popularity due to their numerous applications across domains. Due to the large body of recent research and practical achievements, neural networks are selected for in-depth review in the next section (section 4.1.1). Of particular interest are *deep neural networks* that allow image processing (section 4.1.2) or are combined with reinforcement learning (section 4.2.2) to create more adaptable solutions.

4.1.1. Neural Networks

The basic building block of Artificial Neural Networks (ANN) is the neuron. The neuron was formally known as the *perceptron* as introduced by Rosenblatt (1958) (Figure 4.3a). The output y of a neuron follows from the inputs x_i following Equation 4.1:

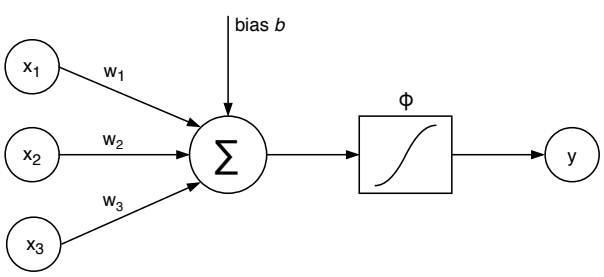
$$y = \phi\left(\sum_i w_i x_i + b\right) \tag{4.1}$$

The neuron $y(x_i)$ can approximate functions by adapting its weights w_i and bias b. ϕ is called the *activation function*, which determines the neuron's final output. 'Learning' is achieved through *back-propagation*. Using known input-output pairs, knowledge is back-propagated through the neuron to update the weights and biases. To approximate more complex functions, multiple neurons are stacked to form a neural network (Figure 4.3b).

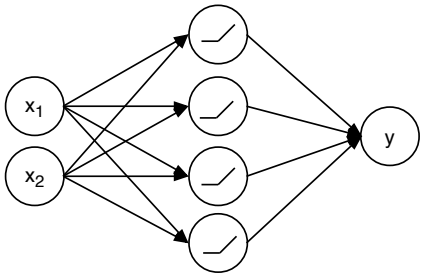
Activation function

Activation functions determine the relationship between a neuron's input $\mathbf{x}$ and output $\phi(\mathbf{x})$. By using non-linear activation functions, the network is able to approximate non-linear functions. Traditional functions for neural networks are $\phi(x) = \tanh(x)$ or $\phi(x) = (1 + e^{-x})^{-1}$. However, these functions result in longer training times compared to non-saturating non-linear functions such as Equation 4.2. (Krizhevsky, Sutskever, & Hinton, 2012)

$$\phi(x) = \max(0, x) \tag{4.2}$$

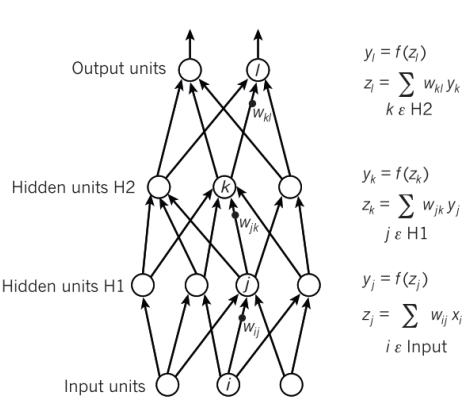


(a) Symbolic representation of a perceptron with inputs x_i and output y. Inspired by (Rosenblatt, 1958)

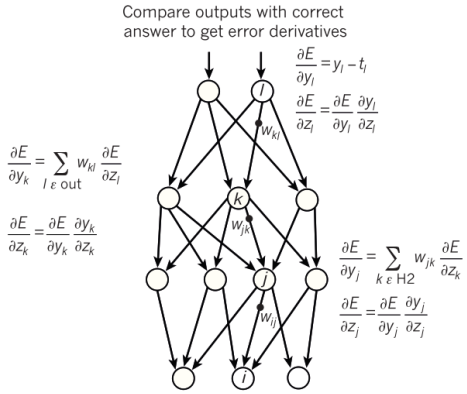


(b) Stacked neurons form an ANN with one hidden layer.

Figure 4.3: Structure of an Artificial Neural Network (ANN).



(a) Forward pass



(b) Backward pass

Figure 4.4: Propagation through a NN with two hidden layers and one output layer. The difference between the input and the target, the error E, is back-propagated through the network to update network weights. Please note that biases are not included for simplicity. Adapted from LeCun, Muller, Ben, Cosatto, and Flepp (2005)

Neurons with these functions are called Rectified Linear Units (ReLUs) (Nair & Hinton, 2010), which have gained popularity due to their computational efficiency. Since then, many improvements have been proposed (J. Gu et al., 2017).

Weight updates through back-propagation

Although adjusting weights for a single layer is straightforward, it becomes more complicated with multiple layers. Which weight specifically contributed to a correct or false output? In order to determine the performance of the current network, a total error E should be defined, for example by Equation 4.3. (Rumelhart, Hinton, & Williams, 1986)

$$E(w) = \frac{1}{2} \sum_c \sum_j (y_{j,c}(w) - t_{j,c})^2 \tag{4.3}$$

Where w are the parameters/weights that determine output y, c is an index of the number of samples, j is the number of outputs and t is the desired output, or target. By applying the *chain rule* multiple times backwards through the network, the partial derivative of the total error with respect to the weights $\delta E / \delta w$ can be obtained (Werbos, 1974). An example of back-propagation in a NN with two hidden layers and one output layer is shown in Figure 4.4. Here, Figure 4.4a shows the forward pass to calculate the outputs y_i from inputs x_i. After this, the error E is calculated (e.g. using equation 4.3). This error is subsequently used to update network weights through the error derivatives (Figure 4.4b).

The simplest form of a first order gradient descent is to increment the weights proportionally to this derivative. In order to minimize total error E, *standard gradient descent* evaluates the entire dataset at once to

update the weights according to Equation 4.4, where ϵ is called the learning rate (Rumelhart et al., 1986). The gradient of the error ∇E can be interpreted as $\delta E / \delta \mathbf{w}$.

$$\Delta \mathbf{w} = \epsilon \sum_{i=1}^{n} \nabla E_i(\mathbf{w}) / n \tag{4.4}$$

In restricted cases, the sum and sum gradient can indeed be computed over the entire dataset. However, in modern applications where network complexities and data sets grow, this is not feasible anymore. For this reason, *stochastic gradient descent* is introduced, where the sum and gradient are evaluated per sample $E_i(w)$ (Bottou, 1998).

$$\Delta \mathbf{w} = \epsilon \nabla E_i(\mathbf{w}) \tag{4.5}$$

A hybrid method that uses more than one sample per iteration, called a *minibatch*, has shown smoother gradients (Mnih et al., 2015). Efficiency is gained because gradients and sums can be calculated in parallel due to vector operations. Two main methods to speed up minibatch learning exist: using momentum and using separate adaptive learning rates for each weight (Tieleman & Hinton, 2012).

Implementing momentum not only considers the 'position' of the weight but also incorporates its 'velocity'. The velocity is defined as the weight change: $\mathbf{v}(t) = \Delta \mathbf{w}(t)$. The weight update is then defined to be the previous velocity multiplied with a decay factor α minus the current gradient multiplied with the learning rate ϵ:

$$\Delta \mathbf{w}(t) = \alpha \Delta \mathbf{w}(t-1) - \epsilon \frac{\delta E}{\delta \mathbf{w}}(t) \tag{4.6}$$

The magnitudes of gradients in a network can vary a lot, especially with small initial weights. One learning rate throughout the network is not the most efficient manner to approach this, which is why adaptive learning rates per weight were introduced: RMSProp (Root-mean-square propagation) keeps a moving average of the squared gradient for each weight (Tieleman & Hinton, 2012).

The inputs x_i are frequently called *features*. Oftentimes these are manually selected based on knowledge of the underlying system. In addition, they can be pre-engineered for better network performance, e.g. having an 'area' feature instead of only length and width. For well-understood systems such as games, this is indeed an effective solution. However, when the system is less defined or discrete – for example in visual environments – a trainable feature extractor is required. This is the idea behind deep learning, which has achieved major feats using Convolutional Neural Networks (CNNs), which are discussed in the next section.

4.1.2. Introduction to Convolutional Neural Networks

Networks with a deep architecture can automatically create learning features without reliance on "hand-crafted" features, which is why they have risen in popularity over the past years (Sutton & Barto, 2018, p. 225). The first application of Convolutional Neural Networks (CNNs) was by LeCun, Boser, et al. (1989) to recognize hand-written zip-codes in 1989. Since then, CNNs have been subject to a tremendous amount of research. A notable milestone is the work of CireşAn, Meier, Masci, and Schmidhuber (2012), who achieved human-competitive image recognition for the first time. Advancements have primarily focused on new network architectures, network functions and computational efficiency. However, the underlying concepts remain the same.

A Convolutional Neural Network (CNN) is a type of feed-forward neural network that is commonly applied to image data. Similar to a regular NN, a CNN consists of multiple stacked neurons (Figure 4.3a). In the case of a CNN, however, every input is connected to a (normalized) pixel-value of the original image, see Figure 4.5. One neuron is locally connected to an area in the original image. For example, a 4 pixel squared area results in 16 inputs plus one bias. This area, or *filter*, slides over the entire image to create a new output map.

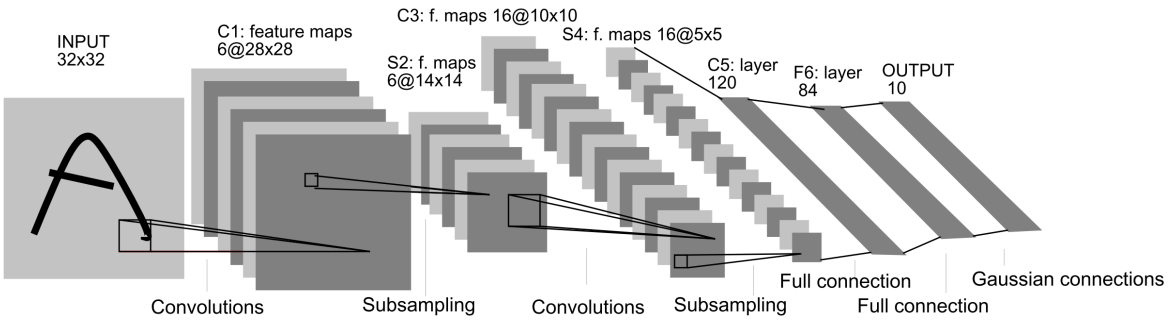


Figure 4.5: A Convolutional Neural Network structure (LeNet-5) for character recognition. The weights of each plane are kept equal for computational efficiency. Adapted from LeCun et al. (1998)

These overlapping filters can extract visual features such as small corners or edges from the image. The visual features are subsequently combined to form compositions while progressing through the net.

The network architecture as seen in Figure 4.5 contains six different convolutional kernels (or convolution matrices) resulting in six feature maps (C1). Per kernel, the weights are kept constant to reduce the number of trainable parameters. It is not uncommon for modern architectures to feature 32 or 64 kernels per convolutional layer (e.g. Silver et al. (2017)).

In between the convolutional layers, pooling layers are used to subsample the image. By reducing the image dimensions, less trainable parameters remain and computation time is improved. Additionally, by removing detail from the image, overfitting is reduced (LeCun, Pfeifer, Schreter, Fogelman, & Steels, 1989). The pooling layer in Figure 4.5 uses a kernel that takes the average of a 2x2 pixel area $\mathbf{x}$ to obtain a 1x1 pixel value (LeCun et al., 1998). Nowadays, a max-operator max($\mathbf{x}$) is more common (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2015). With no overlapping kernels, the image size is thus reduced by 75%.

Near the output of the network, a fully connected layer is used to connect all remaining neurons to the possible output classes. These fully connected layers can be interpreted as a standard NN or as a convolutional layer with 1x1 pixel kernels. The final probability of outputs is determined by the output classifier function.

Output classifier function
The concluding softmax function (Sutton & Barto, 1998) transforms the remaining vector of real values into a vector of probabilities per output class. This provides the model not only with an output value, but also with a certainty factor. The softmax function is defined by Equation 4.7. Where all entries of σ are real, within [0,1], and add up to 1. K is the dimension of input vector $\mathbf{z}$.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e_{z_k}} \text{ for } j = 1, ..., K \tag{4.7}$$

Loss function with probability
When probabilities are included in the model output, basic cost functions such as Equation 4.3 are not compatible anymore. To take the probability vector into account, a cross-entropy cost function (De Boer, Kroese, Mannor, & Rubinstein, 2005) can be used to calculate the network loss, which in turn is used to update the network weights. It fits discrete output problems well and the gradient increases with increasing error (as opposed to quadratic loss functions), consequently speeding up the learning process. The cross entropy H is calculated for M output classes by comparing the probability vector σ resulting from the softmax function to the one-hot target vector y_i:

$$H(y, p) = -\sum_{i}^{M} y_i \log \sigma_i \tag{4.8}$$

4.1.3. Advances in Convolutional Neural Networks

Initially, convolutional networks were applied to domains where large amounts of data are available such as character recognition (LeCun, Boser, et al., 1989) and traffic sign classification(CireşAn et al., 2012). In the following years, more dynamic domains were used in applications. In these years, LeCun, Bengio, and Hinton (2015) trained a self-driving RC car using imitation learning based on human data. Training data consisted of raw camera images (input) with steering data (target) to train end-to-end self-driving behavior using a 6-layer CNN. This concept was later improved by Hadsell et al. (2009) by using a stereo camera to enable self-supervised learning.

It took three more years for a broader adoption of CNNs by research communities after successful application to a million image dataset called ImageNet (LeCun et al., 2015). Krizhevsky et al. (2012) devised GPU[1] optimized training algorithms and non-saturating neurons (ReLUs, see Equation 4.2) to reduce the error rate on ImageNet from 26% to 15%.

To reduce overfitting, Krizhevsky et al. (2012) used two techniques; dropout layers and data augmentation. A dropout layer randomly sets the weights of a fraction of the neurons to zero at each epoch during training. This will lower the network's reliance on these neurons and will thus prevent overfitting (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). The second technique is data augmentation, where the existing training data set is altered by techniques such as flipping horizontally & vertically, rotating, scaling, cropping, translating, adding Gaussian noise and adjusting brightness & contrast.

To increase simplicity of network architectures, Springenberg et al. (2015) analyzed all components of the pipeline of a CNN. It appeared that max-pooling layers can be replaced by increasing the kernel size of the convolutional layers without performance loss on image databases. Additionally, it retains the original locations of features better which is intuitively required for analyzing SSDs images.

Further advances in CNNs are primarily focused on the design of convolutional and pooling layers, loss functions, activation functions, optimizers and computation time (J. Gu et al., 2017) but these incremental improvements are beyond the scope of this literature review.

A main drawback of supervised learning using CNNs is the vast amount of data required to achieve acceptable accuracy. A subfield of machine learning that aims to solve this is Deep Reinforcement Learning, which is covered in the next section.

4.2. Deep Reinforcement Learning

This section ultimately introduces the concept of Deep Reinforcement Learning (DRL). In order to do this, a short introduction on RL is given which will focus on the Q-Learning algorithm and the current challenges in RL. Section 4.2.2 will elaborate on Q-Learning with Deep Q-Networks. Subsequently, recent work on Policy Gradients is discussed (section 4.2.3). Finally, all techniques will be combined to analyse a short case study on AlphaGo in section 4.3.3

4.2.1. Introduction to Reinforcement Learning

The essence of RL is learning through interaction. An agent develops itself based on feedback it receives from its environment and alters its behavior based on rewards it receives (Sutton & Barto, 1998).

A key principle that forms the basis for RL are Markov Decision Processes (MDPs), as introduced by Bellman (1957). The underlying principle – the *Markov Property* – is that the next state is fully defined by the current state and the current action. This means that given the current state s_t and action a_t, they are conditionally independent of previous states and actions. By interaction with its environment, the agent receives potential rewards r_t per action a_t it takes. These rewards are summed and discounted (γ) over an epoch to obtain a final *return* R. The agent's goal is to develop a strategy, a policy π, that results in the highest return. When an optimal policy π^* is found, it dictates – for a given set of states – which actions will ultimately result

[1] Graphics processing unit

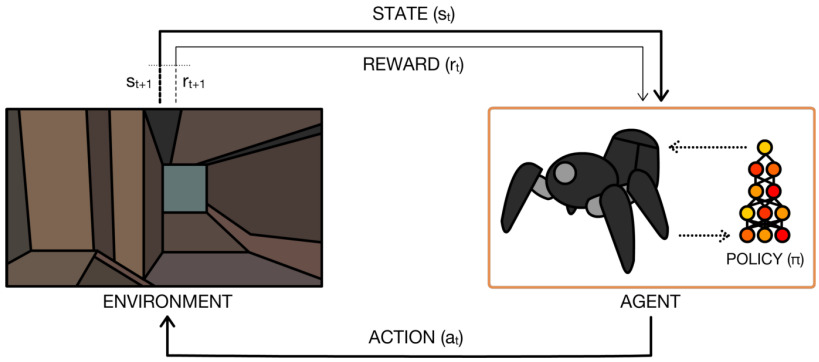


Figure 4.6: The reinforcement learning loop: Based on policy π, the agent determines an action a_t based on the current state s_t. After interaction with the environment, this action results in a new state s_{t+1} and potentially a reward r_t. Indirectly, the rewards incrementally alter the policy to improve it. Adapted from Arulkumaran et al. (2017).

in the highest return. The ideology behind RL is that the optimal policy is found through trial and error, by constantly updating the agent's knowledge. This loop is shown visually in Figure 4.6. A mathematical representation of this goal is given in Equation 4.9 (Sutton & Barto, 1998).

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[R|\pi] \tag{4.9}$$

where the reward R is defined as:

$$R = \sum_{t=1}^{T} \gamma^t r(s_t, a_t, s_{t+1}) \tag{4.10}$$

in which γ is the *discount factor* $0 \leq \gamma \leq 1$ (Sutton & Barto, 1998).

The next step is to maximize this reward. One of the first and most popular methods for this is called *Q-Learning* (Watkins, 1989). The concept of Q-Learning is to find a function $Q(s,a)$ that reflects the predicted future reward for action a in state s. When this Q-function is subsequently followed, the maximum reward should be obtained. The Q-function is updated iteratively using the current reward and the estimate of the optimal future value, as shown in Equation 4.11.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{4.11}$$

where α is the *learning rate* and γ is the *discount factor*. A learning rate of 0 will force the agent to solely use the most recent information, whereas a discount factor of 0 will make the agent only regard the current reward as opposed to future rewards.

The Q-values are updated using the Q-value of the next state with a *greedy* ($\max_a$) action a, regardless of the actual action that the agent takes, making Q-learning an off-policy learner (Sutton & Barto, 1998). An alternate form of learning is to directly learn the policy in on-policy learning, which is discussed in section 4.2.3.

RL is a technique that offers adaptability, global optimization and original solutions for complex problems. However, there are still challenges that make applications not always easy and straightforward (Sutton & Barto, 1998):

- Credit Assignment Problem: The consequences of an action could be experienced many time steps later which makes it difficult to link certain rewards to the action that led to it.

- Curse of dimensionality: When problems require a large state-action space, it becomes infeasible for an algorithm to visit all states sufficient times to converge to the optimal policy.
- The exploration – exploitation trade-off: The agent must converge towards actions that yield a high reward, but to find these actions, it has to try new ones. A balance should be found for optimal results which varies per problem.
- Reward functions have to be designed carefully, which can be a difficult process as their design influences learning in sometimes unintended ways.
- A policy might get stuck in a bad local optimum if hyperparameters are not tuned properly, which can be a tedious task.

The next sections introduce more recent advances in reinforcement learning which aim to tackle the above-mentioned challenges in various ways.

4.2.2. Deep Q-Networks

As determined in section 4.1.2, one of the advantages of deep learning is that a feature extractor is trained in the process, rendering hand-crafted feature engineering redundant. Nonetheless, RL algorithms based on manual features usually outperform generic 'raw-input' algorithms. This is proven by the difference between TD-Gammon 0.0 – using a raw representation of the Backgammon board – and the much better performing TD-Gammon 1.0, which included Backgammon specific features (Sutton & Barto, 2018; Tesauro, 1995).

Although this remains true for well-understood systems such as games, automatic feature extraction clearly adds value in highly complex, poorly-understood domains such as human strategy. An important advancement was made when deep neural networks were combined with RL by Mnih et al. (2013), showing that high performance can be achieved with a generic algorithm that was applied to a multitude of environments. Mnih et al. introduced the Deep Q-Network (DQN) concept; a deep neural network combined with Q-Learning. It was used to play 49 different Atari 2600 games without any changes of hyperparameters per game (Mnih et al., 2015). The algorithm was trained on the same input for all games, being four[2] 64x64 pixel screen captures per state. The reward signal is tied directly to the game score and no further feedback was provided to the agent.

DQN can be interpreted as a standard Q-learning algorithm with a deep neural network as Q-function. Additionally, Mnih et al. (2015) made two major improvements to Q-Learning to increase stability and convergence: experience replay and using a target network, which are discussed below. Q-Learning was chosen because it is model-free – making it easier to implement and more generic – and off-policy, which is a requirement for experience replay.

Experience replay
Experience replay (Lin, 1993) is a technique that stores all agent experience in a dataset $D = e_1, ..., e_t$. One experience is defined by $e_t = (s_t, a_t, r_t, s_{t+1})$ or in words: the state, action, reward and the next state per a timestep. Every time the weights are updated, a random batch of samples is drawn from dataset D. Experience replay has three advantages: One, it improves data efficiency because one experience is used in multiple weight updates. Two, consecutive frames in a game are highly correlated making training inefficient. And three, with an on-policy strategy, a preference for a certain action generates more samples with that action, possibly creating an unwanted feedback loop which could lead to a local minimum or divergence. (Mnih et al., 2015)

A separate network for target generation
The second improvement to Q-Learning is using a different neural network to generate the targets y_i. This target network $\hat{Q}$ is "cloned" from the original network Q every C steps. This adds a delay between adjusting the weights and the time the targets are affected. Without this temporal separation, oscillations could occur due to the inter dependability. (Mnih et al., 2015)

[2] Four subsequent screen shots were contained in one state to capture velocity and acceleration.

Network architecture and backpropagation The network architecture consists of three convolutional layers, followed by a fully connected layer and the output layer. Sub-sampling was achieved without pooling layers by making the convolutional kernels larger (ranging from 20x20 pixels to 7x7 pixels per kernel). This conserves the location of the features (e.g. the ball), which is vital for most games. 32 or 64 feature maps per layer were used and the activation functions are ReLUs (Equation 4.2). The network concludes with 18 output neurons which correspond to the maximum of 18 possible actions in an Atari game. Actions are selected using an ϵ-greedy strategy, with a linearly decreasing exploration–exploitation ratio ϵ (Mnih et al., 2015).

The Q-network is trained by minimizing loss functions $L_i(\mathbf{w}_i)$ that change every iteration i when the weights of the target network are updated:

$$L_i(\mathbf{w}_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}_i^-) - Q(s, a; \mathbf{w}_i)^2\right)\right] \tag{4.12}$$

where $\mathbf{w}_t$ is the weight vector, γ is the discount factor, $\hat{Q}$ represents the function approximating neural network, s is the state, a is the action with the highest expected reward and $U(D)$ displays a uniform, random sample pull from the database D. Please note that the first term between the brackets represents the target y_i for iteration i.

When this loss function is differentiated with respect to the weights, the following gradient is obtained:

$$\nabla_{\mathbf{w}_i} L_i(\mathbf{w}_i) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}_i^-) - Q(s, a; \mathbf{w}_i)\right) \nabla_{\mathbf{w}_i} Q(s, a; \mathbf{w}_i)\right] \tag{4.13}$$

For computational efficiency, instead of computing the full expectations of the gradients, the weights are updated using stochastic gradient descent (Sutton & Barto, 2018):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\left[R_{t+1} + \gamma \max_{a} \hat{Q}(s_{t+1}, a; \mathbf{w}_i^-) - \hat{Q}(s_t, a_t; \mathbf{w}_t)\right] \nabla \hat{Q}(s_t, a_t, \mathbf{w}_t) \tag{4.14}$$

where α is the learning rate and R is the cumulative reward. The learning rate α (also: step size parameter), is adapted per weight specifically based on a running average using RMSProp (as seen in section 4.1.1).

To summarize, the DQN algorithm is given by Algorithm 1.

Algorithm 1 deep Q-learning with experience replay by Mnih et al. (2015)

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode = 1, M **do**
  Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \text{argmax}_a\, Q(\phi(s_1), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_t, a_t, r_t, \phi_{t+1})$ from D
    if episode terminates at step $j+1$ **then**
      $y_j = r_j$
    else
      $y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to
    the network parameters θ
    Every C steps reset $\hat{Q} = Q$

Since their first publication, DQNs have undergone multiple improvements to increase the algorithm's performance, of which a summary of recent advances is given in Table 4.2 Although not all techniques will be discussed in detail, DQN from Demonstrations (DQfD) is relevant to this research and is elaborated on in section 4.3.

Table 4.2: Overview of recent advances regarding Deep Q-Networks (DQNs).

Improvement	Description	Reference
Double Q-Learning (DDQN)	Reduces the overestimation of the expected return by using a second Q-Network	Van Hasselt et al. (2016)
Duelling DQN	Uses the state-value function V^{π} as baseline and advantage function A^{π} for adjustment which is easier to learn.	Wang et al. (2016)
Prioritised experience replay	Samples important experience more frequently compared to the original uniform sampling to increase learning efficiency.	Schaul et al. (2016)
Normalized Advantage Function (NAF)	Uses an advantage layer to enable it to be applied to continuous functions for continuous control problems.	S. Gu et al. (2016)
DQN from Demonstrations (DQfD)	Accelerates learning by using demonstration data (section 4.3).	Hester et al. (2017)
Categorical DQN	Learns the reward distribution instead of only the expected reward to obtain more knowledge about the reward model.	Bellemare et al. (2017)

4.2.3. Policy Gradients

Although DQN has shown very promising initial results, the authors of its initial publication concluded that Policy Gradients (PG) proved to be a more effective method in some cases (Mnih et al., 2016). PG is an end-to-end RL algorithm, where the policy – instead of the value function – is directly optimized to obtain the highest reward. Explicitly learning the policy enables stochastic action decisions, something that could resemble a human better than the deterministic argmax function in traditional DQN.

PG find their origin in the work of Williams (1992), who researched a class of algorithms that could update weights in the direction of a gradient, without actually calculating gradient estimates. These algorithms, called REINFORCE algorithms, learned much slower than traditional RL algorithms. Nonetheless, Sutton, McAllester, Singh, and Mansour (2000) followed up on Williams' work and proved for the first time that PG methods converge to a "locally optimal policy".

New interest was sparked when Silver et al. (2014) formalized *deterministic* policy gradient algorithms. These deterministic gradients can be estimated more efficiently than the original stochastic policy gradient because it reflects the "expected gradient of the action-value function". Silver et al. (2014) have shown that deterministic policy gradients "significantly" outperform stochastic ones.

Another breakthrough was made by Schulman, Levine, Abbeel, Jordan, and Moritz (2015), who devised a method for "optimizing control policies with guaranteed monotonic improvement". Their method is called Trust Region Policy Optimization (TRPO) and is particularly well-suited for optimizing large nonlinear policies, which are for example seen in neural networks. TRPO was benchmarked against earlier results on Atari games (Guo, Singh, Lee, Lewis, & Wang, 2014; Mnih et al., 2013) and outperformed them. Additionally, implementing TRPO appeared to be less complex than incumbent methods.

A final leap was made by Mnih et al. (2016), who improved on their original 2015 paper using PG. The original paper used experience replay (section 4.2.2) as one of its key features, which made on-policy training impossible. To replace the experience replay functionality, they "asynchronously execute multiple agents in parallel, on multiple instances of the environment" Mnih et al. (2016). Due to this parallel computing, the input states become uncorrelated and learning is stationary again. By opening up the door for on-policy algorithms, it was found that an asynchronous variant of the actor-critic algorithm outperforms the current

state-of-the-art with only half of the training time.

Both Policy Gradients and DQNs found their way in applications regarding imitation learning, which is the focus of this research and will be discussed in the next section.

4.3. Imitation Learning

Achieving conformal automation is closely related to a machine learning field called *imitation learning*. Imitation learning aims to perform a task based on expert demonstrations, without reinforcement signal. The main two methods are *Inverse Reinforcement Learning (IRL)* and *behavioral cloning*. This report defines a related field called *transfer learning* as using stored knowledge of a previous problem to solve a comparable, but different problem. Because the automation solves the same problem as the ATCo, transfer learning is not reviewed in this section.

4.3.1. Inverse Reinforcement Learning

IRL is the process of determining the reward function from observations by an expert Ng and Russell (2000). To assess the practicality of this, two examples are shortly discussed.

The first notable publication aimed to predict hourly Ground Delay Program decisions for real-world data from two international airports (Bloem & Bambos, 2015). Both actual and scheduled air traffic data was used to train multiple IRL and random forest models. It was however found that the random forest behavioral cloning models outperformed IRL.

The second publication is again applied to self-driving cars Wulfmeier, Rao, Wang, Ondruska, and Posner (2017). The aim was to extract the “underlying reward or cost structure from demonstrations of human behavior" (Ziebart, Maas, Bagnell, & Dey, 2008). The input data consisted of LIDAR[3] images and driving variables of more than 25,000 demonstration trajectories (120 km of urban driving). By using a Maximum Entropy Deep Inverse Reinforcement Learning (Wulfmeier, Ondruska, & Posner, 2015) algorithm, Wulfmeier et al. were successful in developing a scalable approach that was proven against multiple benchmarks.

4.3.2. Behavioral cloning

Behavioral cloning is also referenced to as *apprenticeship learning* or *learning from demonstrations*. It usually implies using supervised learning to define a policy from state-action data that was obtained from expert trajectories. A straightforward example of this is training self-driving cars to keep their lane based on front-camera data and human steering angles (Chen & Huang, 2017).

More elaborate cases use a combination of RL and supervised learning. Despite the successes of Deep RL, it requires large amounts of (simulation) data to reach acceptable performance. Although not an issue in simulation environments, this data requirement limits the number of use-cases in practical applications. Hester et al. (2017) aim to mitigate this weakness by using demonstrations with a novel method called *Deep Q-learning from Demonstrations (DQfD)*. Initially, DQfD only trains on the demonstration data set to obtain a policy (which should be close to the demonstrator’s policy) and a value-function. This value-function is used for the next phase, where the agent is improved using Q-learning. In the second phase, the agent receives a combination of demonstration data and self-generated RL samples. The method uses a regularized loss to make sure the algorithm does not overfit on the demonstration data set. It was shown that DQfD indeed shows better initial performance compared to state-of-the-art DQN implementations (Prioritized Dueling Double DQN, combining three methods from Table 4.2).

One high-profile case of supervised learning to achieve behavioral cloning was presented by Silver et al. (2016) to mimic experts playing the game of Go. Their solution combines multiple techniques that were introduced in this chapter into one machine learning pipeline. To analyze the synergies and connections between these methods, a short case study is performed in section 4.3.3.

[3] LIght Detection And Ranging of Laser Imaging Detection And Ranging: A type of radar mounted on the vehicle

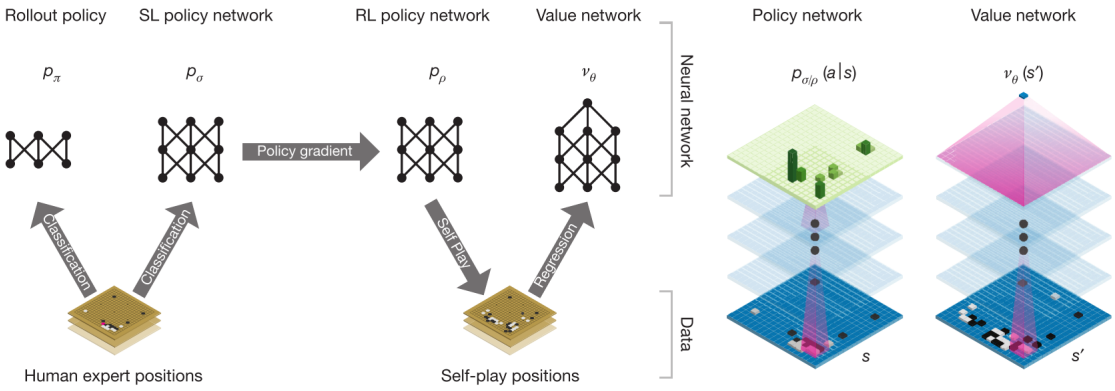


Figure 4.7: The AlphaGo neural network training pipeline. Adapted from Silver et al. (2016)

4.3.3. Case study: AlphaGo

One of the most well-known recent papers on AI advancement was published by Silver et al. (2016) in Nature. It combines supervised learning, policy gradients, value functions and Monte-Carlo Tree Search (MCTS) to achieve super-human performance in the game of Go[4]. This paper is particularly interesting to this research because it uses SL to train on expert-moves after which the policy is improved using RL, supported by neural networks. It is not difficult to see the analogy with an ATCo, i.e. the expert, whose moves are used to train an algorithm to obtain conformal automation.

The pipeline of the AlphaGo machine learning implementation is shown in Figure 4.7. Based on human expert moves, two neural networks are trained. A rollout policy network p_π to support quick MCTS (Coulom, 2006) and a supervised learning (SL) policy network p_σ to predict expert moves given the current state of the board. The SL policy network, trained on more than 30 million moves, achieved an accuracy of 57.0 % by using stochastic gradient ascent through maximizing the likelihood of conformity in Equation 4.15.

$$\Delta\sigma \propto \frac{\delta \log p_\sigma(a|s)}{\delta\sigma} \tag{4.15}$$

where σ represents the weights of the NN and p_σ the probability of the human action a in state s. Subsequently, the weights of the SL policy network were cloned to an identically structured RL policy network p_ρ ($\rho = \sigma$). This policy network improved itself through self-play against a randomly sampled older version of itself. Random sampling of 'opponents' prevented the policy from overfitting to a local maximum. The weights are updated proportionally to the function that maximizes the likelihood of a positive outcome for AlphaGo:

$$\Delta\rho \propto \frac{\delta \log p_\rho(a_t|s_t)}{\delta\rho} z_t \tag{4.16}$$

where the predicted outcome of the game z_t is determined by winning (+1) or losing (-1) as seen from the current time step t. This prediction is performed by a combination of the outcome z_L of a random rollout of the MCTS algorithm using the fast policy network p_π and the outcome of value network $v_\theta(s')$. The two methods are combined using mixing parameter λ to calculate the "leaf evaluation $V(s_L)$", which can be interpreted as the strength of a particular move.

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L \tag{4.17}$$

It must be noted that the input states used by AlphaGo were constructed using knowledge about the environment (i.e. game rules). The input consists of $19 \times 19 \times 48$ image stacks, which are the dimensions of the

[4]Go is one of the oldest and most complex games around (en.wikipedia.org/wiki/Go_(game))

Go board times the number of features that were constructed. A major improvement on AlphaGo – called AlphaGo Zero (Silver et al., 2017) – removed the requirement for handcrafted features and initial supervised learning using expert moves, taking another step towards generic AI.

4.4. Conclusion

This chapter focused on machine learning (ML) techniques to achieve conformal automation in ATC. Supervised learning (SL) has been analyzed and literature showed that it has been applied to successfully mimic expert actions, albeit in driving or playing (board) games. In these applications, neural networks, and especially Convolutional Neural Networks (CNNs) were well-represented; numerous applications used raw image data as only feature to train the learning algorithm because it eliminates the need for manual feature selection and engineering, thus not biasing the learning based on prior assumptions. The approach of using image data will be used throughout the rest of this thesis project.

Although standalone SL proved successful, many of the evaluated publications used a combination of techniques, preferably SL with Reinforcement Learning (RL). RL techniques that proved successful consisted of off-policy (DQN) and on-policy (PG) algorithms. The combination of SL and RL to obtain well-performing automation is especially useful in situations were little data is available. This might indeed be the case in this research, where human data is generated through low-scale experiments.

5

Preliminary analysis: The visual value of the SSD using deep neural networks

In order to test the feasibility of modeling control strategies using the SSD, a preliminary analysis is performed. This chapter will introduce the goal and methodology of the analysis in section 5.1, elaborate on data generation in section 5.2, discuss the algorithm training in 5.3 and show the results in section 5.4.

5.1. Research goal and methodology

Firstly, one of the major assumptions of the main research is the ability to model ATC control strategy from the SSD. The SSD is assumed to contain almost all features relevant for a controller to make a decision in a CD&R task. This preliminary analysis aims to examine whether the decision for a certain resolution is indeed captured in the SSD, and how precise this prediction might be.

Secondly, in chapter 4, several machine learning methods have been proposed and examined. Supervised learning using a Convolutional Neural Network (CNN) seemed promising, since all SSD features are visually captured without having to pre-select them based on prior knowledge or assumptions. However, as seen in subsection 4.1.2, the wide range of tunable hyperparameters results in an infinite number of network structures and setups. This preliminary analysis is designed to give insight into the effect of these hyperparameters and the precision that can be achieved when training a CNN on labeled SSD data. Specifically, the first three sub-questions address these queries.

The goal of the final research is to find out whether machine learning is suitable to model ATCo strategy. One important element of this is training on an empirical dataset created by human controllers. In order to make this feasible, model training should be feasible with limited quantities of data only. The fourth sub-question of this research addresses this by examining the correlation between model accuracy and the quantity of input data. Moreover, the quality of the dataset can fluctuate due to controller inconsistency. Sub-question five assesses the impact of this on model performance. Finally, data can be augmented to potentially increase the effectivity per sample. This possibility will be examined by the last sub-question.

Summarizing, this results in the following research questions:

Main question
How can an ATC resolution strategy be modeled using a Convolutional Neural Network (CNN) with labeled SSDs as input?

Sub-questions

1. How does model accuracy vary with action resolution?

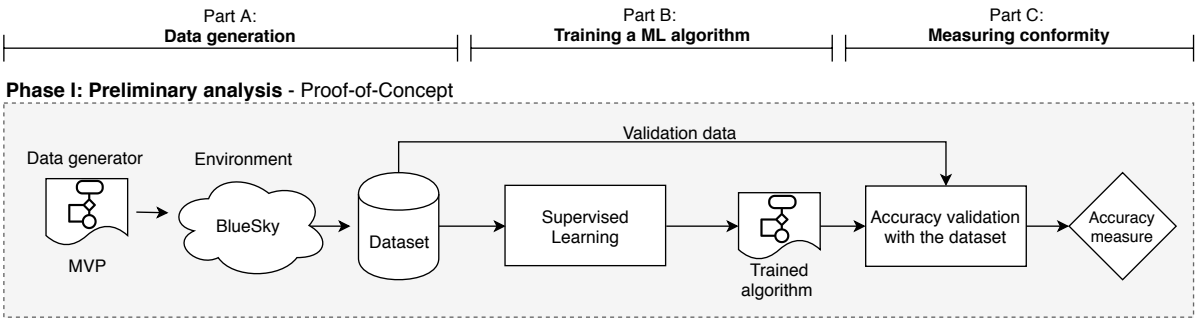


Figure 5.1: Methodology of the preliminary analysis

2. What is the effect of network architecture on model accuracy?
3. How does model accuracy vary with the dimensions of input data?
4. How does model accuracy vary with the quantity of input data?
5. What is the impact of controller inconsistency on model accuracy?
6. How can existing data be augmented to improve model accuracy?

The experiment aims to measure validation and test accuracy of the network as a result of varying parameters; First, the labeled resolution dataset is generated, which is kept constant for all results shown in this analysis. Secondly, the dataset is used to train the CNN with varying hyperparameters and finally, the accuracy of the algorithm is measured using a separate test set. A summary of the experiment pipeline is shown in Figure 5.1.

5.2. Part A: Data generation

The performance of a supervised learning algorithm is closely related to the quality of the input data used during training. For this reason, the generation of the dataset deserves emphasis in this analysis. Labeled data can be generated using either a human controller or a computer algorithm. This analysis makes use of an algorithmic controller because of higher data consistency and the ability of generating large batches of data relatively quickly, as opposed to a human controller.

The dataset was generated using BlueSky, an open data and open-source ATM simulator developed at the TU Delft (Hoekstra & Ellerbroek, 2016) (Figure 5.2). The simulator was extended with a plug-in to generate resolution-labeled SSDs by simulating many conflict scenarios with varying parameters. To create the dataset a built-in resolution algorithm was used, namely Modified Voltage Potential (MVP) (Eby, 1994; Hoekstra et al., 2002). The MVP algorithm was originally designed as a decentralized CD&R solution. To use it for ATC purposes, three changes are made. Firstly, the update interval of the resolution algorithm is increased from 1 s to 10 s to act more 'human-like'. Secondly, only one aircraft receives resolutions, i.e. the controlled aircraft. And thirdly, a safety margin is added to the resolutions. This decreases efficiency but reduces workload for the controller, which resembles the actions of a human ATCo (Fothergill & Neal, 2013).

Due to the exploratory nature of this preliminary analysis, only 2 aircraft per scenario are considered in a two-dimensional space (excluding altitude). Furthermore, resolutions consist of heading changes only. The protected zone around the aircraft follow conventions (ICAO, 2016). All controlled variables regarding data generation are summarized in Table 5.1.

Training, validation and test data
To create a complete set of input samples, the conflict angle, CPA and time to CPA per conflict pair are varied over a range of values. The used ranges result in a total training dataset with 5901 scenarios. This training set is subsequently divided in a training and validation set, which is used to measure validation accuracy during training. Additionally, a second batch of scenarios is simulated using different values for the above-mentioned parameters to generate a test set with scenarios that the neural network has not seen before. This

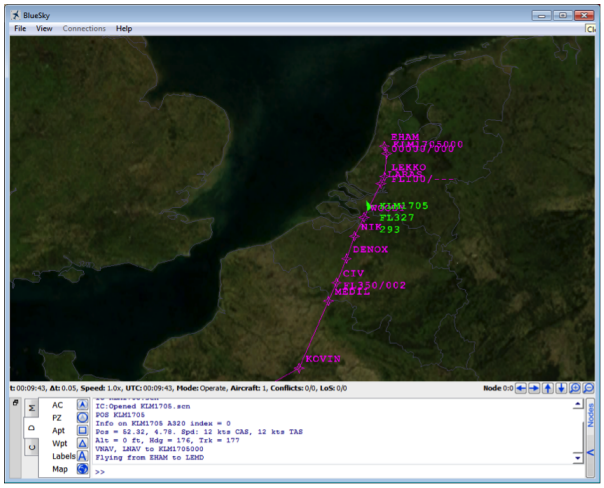


Figure 5.2: A screen capture of ATM simulator BlueSky.

Table 5.1: Controlled variables of the dataset generator in BlueSky

Parameter	Value	Unit
Resolution algorithm	MVP (augmented)	-
Number of A/C in conflict	2	-
Resolution types	Heading changes only	-
Update interval	10	s
Look-ahead time	180	s
Protected zone radius	5	nm
Protected zone height	1000	ft
Controlled aircraft heading	40	deg

test set will be used to ultimately validate the models. The parameters used to generate the scenarios are displayed in Table 5.2.

Per scenario, the SSD of the controlled aircraft is generated and saved together with the proposed conflict resolution by the MVP algorithm during the time that an aircraft pair is in conflict (with an interval of 10 s). This data set is then filtered to remove scenarios where the algorithm took too long to solve the conflict or when LoS occurred. The histogram of proposed resolutions by MVP for the dataset is shown in Figure 5.3.

Ten samples of the dataset are shown in Figure 5.4. One triangle corresponds to one conflicting aircraft. The green speed vector indicates the heading of the controlled aircraft. When the arrow head lies in the red area, the aircraft is in conflict and a resolution is required.

5.3. Part B: Training the neural network

A deep Convolutional Neural Network (CNN) was created to be trained on the SSD dataset. This section elaborates on the design choices that were made. The parameters are either controlled or varied, based on

Table 5.2: Parameters of the training, validation and test scenarios

Parameter	Train and validation values	Test values	Unit
Conflict angle	[40 .. 320]	[40 .. 320]	deg
CPA	[-3, -2, -1, 0, 1, 2, 3]	[-2.5, -1.5, 0, 1.5, 2.5]	nm
Time to CPA	[240, 300, 360]	[180, 420]	s
Total scenarios	5901	1410	samples

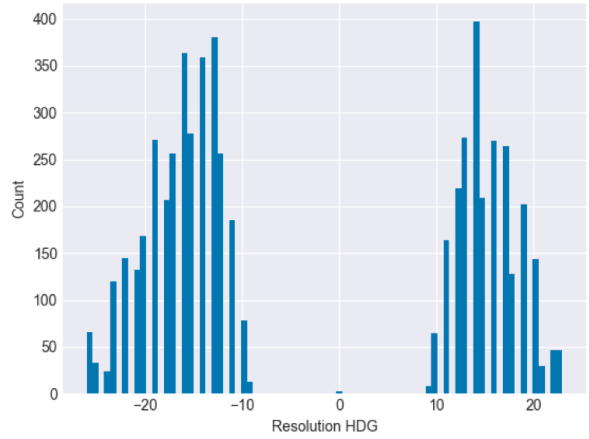


Figure 5.3: Histogram of proposed resolutions by MVP for the entire train/val-dataset.

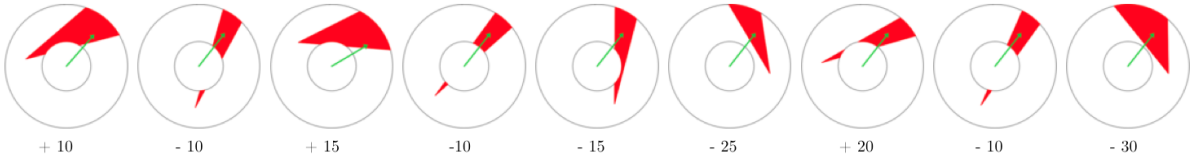


Figure 5.4: 10 samples from the SSD dataset. The numbers underneath the SSDs indicate the relative heading change that the MVP algorithm recommended in this situation.

whether they were used to answer the research questions posed in section 5.1. All parameters used to create and train the CNN are summarized in Table 5.3.

Optimizer algorithm
Section 4.1.1 introduced two new improvements for updating network weights compared to traditional SGD, namely momentum and adaptive learning rates per parameter. An optimizer called Adam (Kingma & Ba, 2015) implements both techniques in one algorithm. Empirical results show that Adam outperforms previously popular optimizers such as AdaGrad, RMSProp and SGDNesterov, as can be seen in Figure 5.5.

Convolutional and pooling layers
The network has a similar architecture as LeNet (LeCun et al., 1998), consisting of combinations of convolutional and max-pooling layers and a fully-connected layer followed by a softmax layer. A representation of one network is shown in Figure 5.6. All evaluated network structures are displayed in Table 5.6.

The number of convolutional layers determines the number of features that can be detected by the network. The downsides of adding more layers are overfitting, an increased computation time and decaying gradient values when back-propagating through too many layers, i.e. the vanishing gradient problem (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). Due to the relatively low-detail input in this analysis, a range of 1 to 4 layers has been examined. The pooling layers serve the purpose of decreasing the number of trainable parameters to decrease training time. Additionally, the chance of of overfitting decreases.

Number of output classes
The number of output classes determines how precise the conflict resolution advisory will be. The maximum heading deviation is set to be 30 deg, left or right. For example, in case of four classes, the output will be -30, -15, $+15$ or $+30$ degrees relative to the current heading. Table 5.4 shows the relation between the number of output classes and the corresponding heading increment resolution in degrees.

Table 5.3: Controlled and varying hyperparameters of the CNN.

Controlled parameters	**Value**	**Unit**
Optimization algorithm	Adam	-
Loss function	Categorical entropy	-
Activation functions	ReLU	-
Output activation	Softmax classifier	-
Train/val ratio	80/20	-
Train/val/test ratio	65/15/20	-
Mini batch-size	128	samples
Epochs	25	-
Dropout	None	-
Independent parameters		
Number of output classes	[2, 4, 6, 8, 12]	-
Model architectures	[1 .. 7] see Table 5.6	-
Input image dimensions	[(16x16), (32x32), (64x64), (96x96), (120x120)]	px
Number of input samples	[150 .. 6000]	samples
Data rotation (augmentation)	[0, 1, 2, 3, 4, 5, 10, 20, 30]	deg
Sample randomization	[0 .. 100]	%

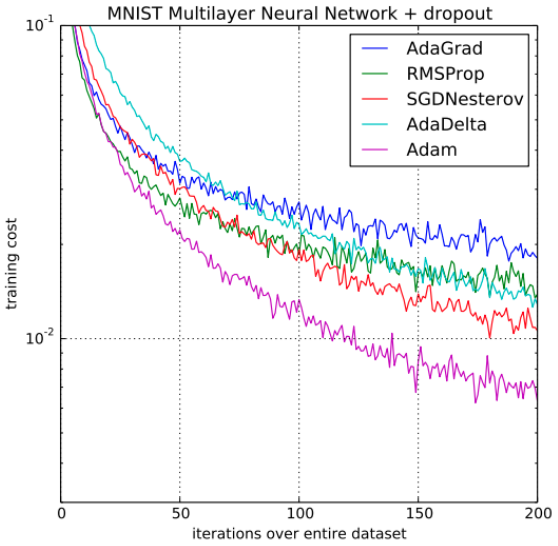


Figure 5.5: Adam optimizer outperforms similar algorithms on an image dataset (MNIST). Adapted from Kingma and Ba (2015).

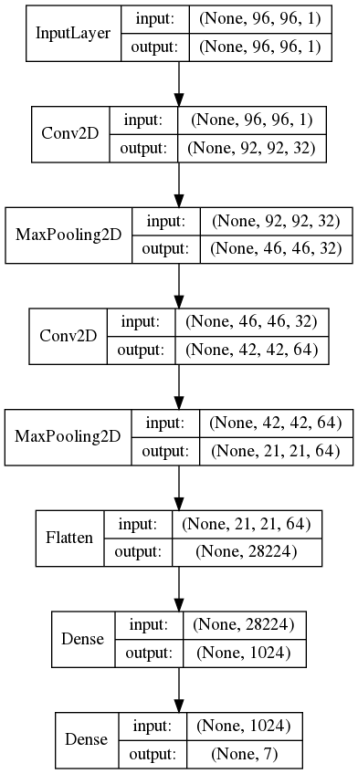


Figure 5.6: Baseline network architecture. 96x96 indicates the input dimensions and 7 in the final layer results in 6 resolution classes and the redundant 'no resolution' class.

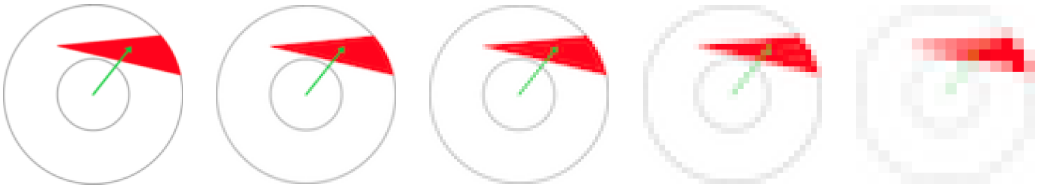
Table 5.4: Relation between number of output classes and the heading increment resolution.

Output classes [-]	**Heading resolution**[deg]
2	30.0
4	15.0
6	10.0
8	7.5
10	6.0
12	5.0

Data augmentation
Because the format of the input samples is well-known and does not fluctuate, most data augmentation techniques (subsection 4.1.3) are not applicable to this analysis. The only methods that produce valid SSDs are flipping and rotating. However, although conflict resolution algorithms are insensitive to the orientation, humans are not. For example, a human might have a preference for steering left instead of right in certain scenarios, regardless of conflict orientation. Therefore, only slight rotations are considered for data augmentation.

Input dimensions
To test the impact of input sample size (in pixels), all SSDs were scaled before entering the network, ranging from 16 pixels squared to 120 pixels squared. The same SSD with different dimensions is shown in Figure 5.7. It appears that certain features may be barely recognizable by a human controller in the 16 px case. Nonetheless, this sample could still provide a neural network enough information to extract learning features. Please note that the final images fed into the networks are converted to gray-scale for computational efficiency.

Sample randomization
A human controller is expected to be less consistent than the MVP algorithm, choosing a different resolution in two identical situations. It is assumed that a human has one main strategy, which results in the majority of



(a) 120 px (b) 96 px (c) 64 px (d) 32 px (e) 16 px

Figure 5.7: An example of an SSD with dimensions used to train the networks. The baseline dimensions are 64x64 px

Table 5.5: Controlled parameters during model training.

Parameter	Value	Unit
Number of output classes	6	-
Convolutional layers	2	-
Filter depth	32 and 64	-
Convolutional stride	(1, 1)	px
Max-pooling layers	2	-
Pooling-size	(2, 2)	px
Input image dimensions	(96x96)	px
Randomized samples	0	%
Data rotation (augmentation)	0	deg

resolutions. The percentage of other resolutions is considered to be random. To simulate this effect, and to assess the impact on model accuracy, a varying percentage of samples was randomized.

Additional training parameters
This section shortly discusses the additional training parameters. The division of the dataset into training and validation data is 70% and 30% respectively, unless mentioned otherwise. The initial learning rate has been set to 0.01. Generally 25 epochs were used for training, which proves to be enough to obtain a stable accuracy. The mini-batch size is dependent on the hardware used and has been set at 128 samples per batch after experimentation.

Dropout layers (subsection 4.1.3) are not included in the network architecture.

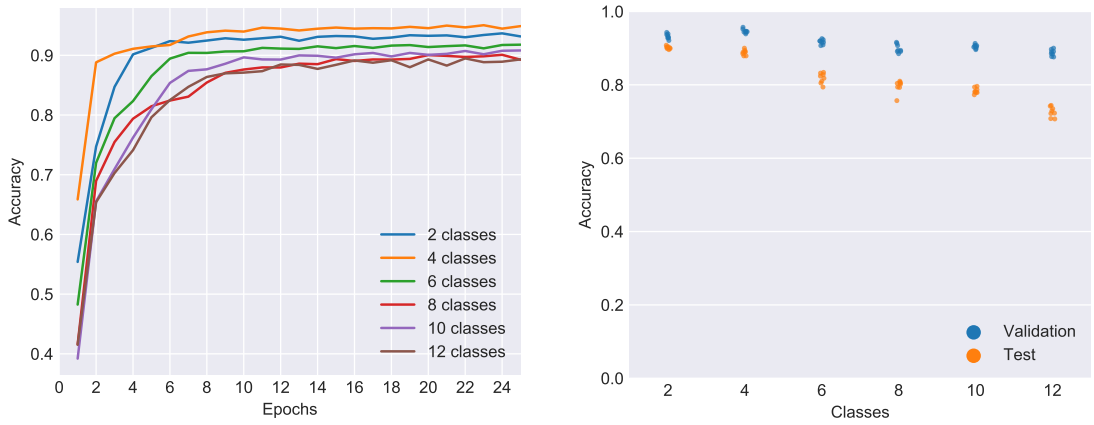
5.4. Part C: Measuring conformity

This section shows and discusses the results of the preliminary analysis. The effects on model validation and test accuracy are examined by varying the following independent parameters: action resolution, network architecture, input data dimensions, number of input samples, data rotation and sample randomization. If not mentioned otherwise, all parameters as shown in Table 5.5 are kept constant.

This section contains one subsection per independent variable. In every subsection, the dependent parameters – training, validation and test accuracy – are graphically shown. Accuracy is defined by $\frac{N_{\text{correct}}}{N_{\text{total}}} \cdot 100\%$. Training accuracy is displayed on the left-hand side, where the training progress over time can be evaluated. The validation and test accuracy are plotted jointly in the right-hand plot. The *validation set* includes similar scenarios as the model was trained on, although it has never seen these samples before. The *test set* includes completely new scenarios, that were generated using different settings in BlueSky. The test set resembles a real-world application the most closely.

5.4.1. Action resolution

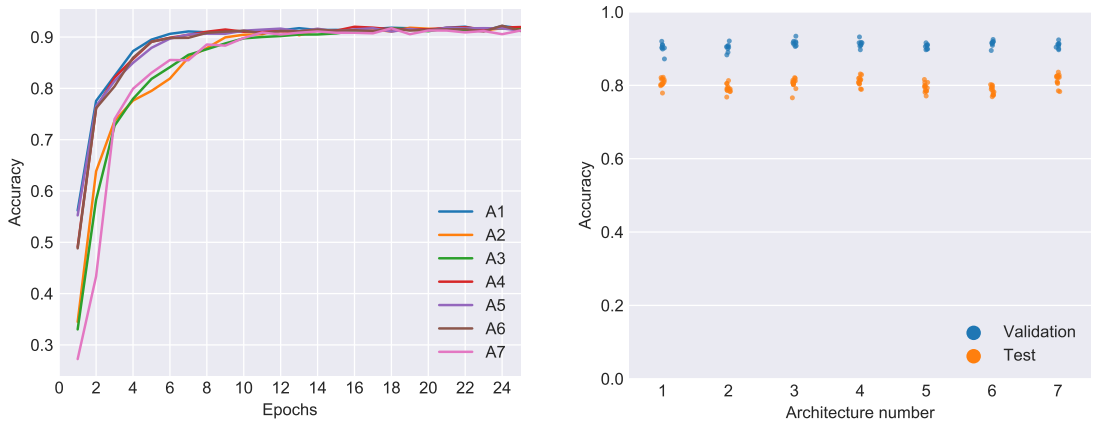
Figure 5.8a shows accuracy during training for a varying number of output classes. By visual inspection, most sets stabilize after 10 epochs. However, it can be seen that instability increases with an increasing number of output classes. Figure 5.8b shows that – intuitively – both validation and test accuracy decrease with an increasing number of output classes. Besides, the difference between validation and test accuracy respectively



(a) Training accuracy

(b) Validation and test accuracy

Figure 5.8: Model accuracy with a varying number of output classes.



(a) Training accuracy

(b) Validation and test accuracy

Figure 5.9: Accuracy for different network architectures

also seems to increase, indicating more overfitting when training with more output classes. Please note that the accuracy is measured in a binary manner, which means that an error is weighed equally regardless of how far off the prediction was.

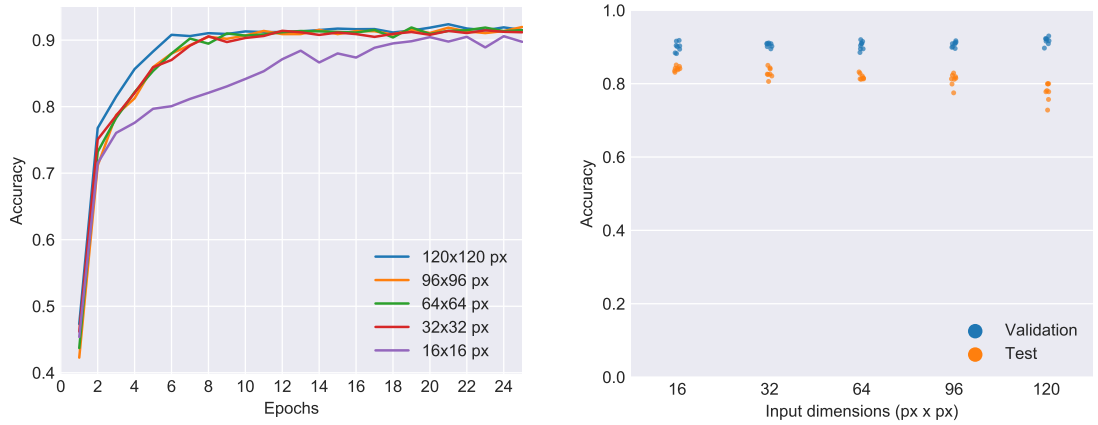
5.4.2. Network architectures

The network architectures that are evaluated are shown in Table 5.6. The baseline architecture is inspired by LeNet by LeCun et al. (1998). LeNet was used for character recognition, which entails similar difficulty to interpreting basic SSDs. Small alterations are made to this baseline to evaluate the effects of pooling layer dimensions, convolutional stride, and number of hidden layers.

Figure 5.9 shows that the considered network architectures do not have a large impact on model accuracy, varying by a maximum of two percentage points. However, network composition did have an impact on training time, as shown in Table 5.6. The network architecture that results in the lowest number of trainable parameters provides the most efficient training. Another benefit of a simple architecture is that it may mitigate overfitting. The degree of overfitting can be observed in the figure as the difference between the validation and test set, given the model architecture.

Table 5.6: Network Architectures used for evaluation in Figure 5.9. The number behind a Pooling layer indicates the pool size in pixels squared. (sX) indicates the stride of a convolutional filter in pixels. All convolutional layers have a filter size of 5x5 pixels.

Architecture #	**Composition**	**Training Time** [s]
baseline	CONV (s1) → POOL (4) → CONV (s1) → POOL (2)	540
2	CONV (s1) → POOL (2) → CONV (s1) → POOL (2)	1410
3	CONV (s2) → POOL (4) → CONV (s1) → POOL (2)	110
4	CONV (s1) → POOL (4) → CONV (s2) → POOL (2)	380
5	1x (CONV (s1) → POOL (2))	920
6	3x (CONV (s1) → POOL (2))	1030
7	4x (CONV (s1) → POOL2)	980



(a) Training accuracy

(b) Validation and test accuracy

Figure 5.10: Accuracy versus input sample dimensions.

5.4.3. Input data dimensions

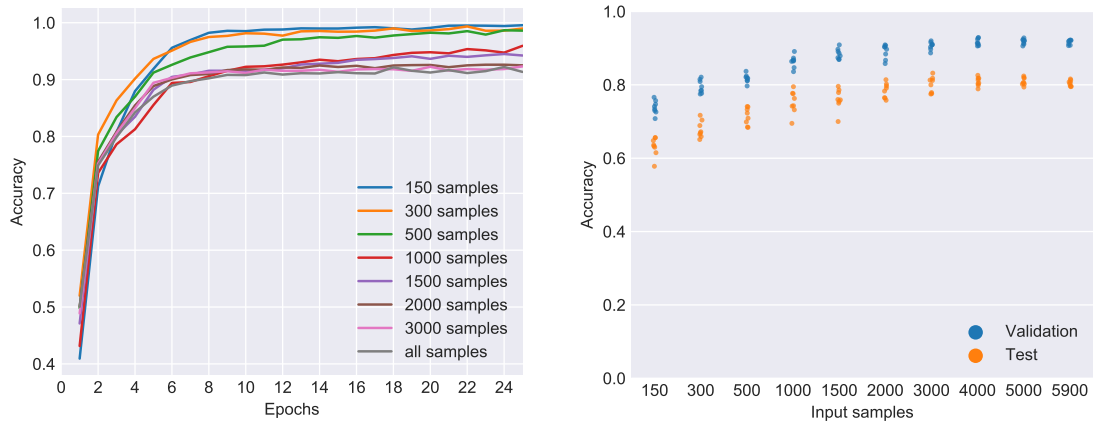
The effects of input sample sizes are displayed in Figure 5.10. Smaller input sizes cause the network to require more epochs to train to the same level of accuracy. A remarkable result is observed in Figure 5.10b; while validation accuracy increases with increasing input dimensions, test accuracy decreases. The highest obtained accuracy increased from 80.0% to 85.1% by decreasing input dimensions. As the difference between the validation and test set increases, this is a clear sign of overfitting. With larger input dimensions, the network 'memorizes' the entire SSD, instead of generalizing and looking for features. Overfitting can be reduced by using dropout layers, using more pooling layers or using larger convolutional filters.

5.4.4. Number of input samples

Figure 5.11 shows a clear relation between the number of samples (the size of the input dataset) and model performance. Although a smaller dataset inevitably converges faster to its stable accuracy, accuracy itself lacks behind. Furthermore, increasing the dataset up to 1000 samples considerably impacts accuracy, while increasing it even further only results in marginal performance increase. A reasonable test accuracy of more than 70% is achieved using 300 samples. Note that during training, networks have learned from the same numbers of samples in total. The networks that trained on smaller datasets trained on the same samples more often.

5.4.5. Data augmentation

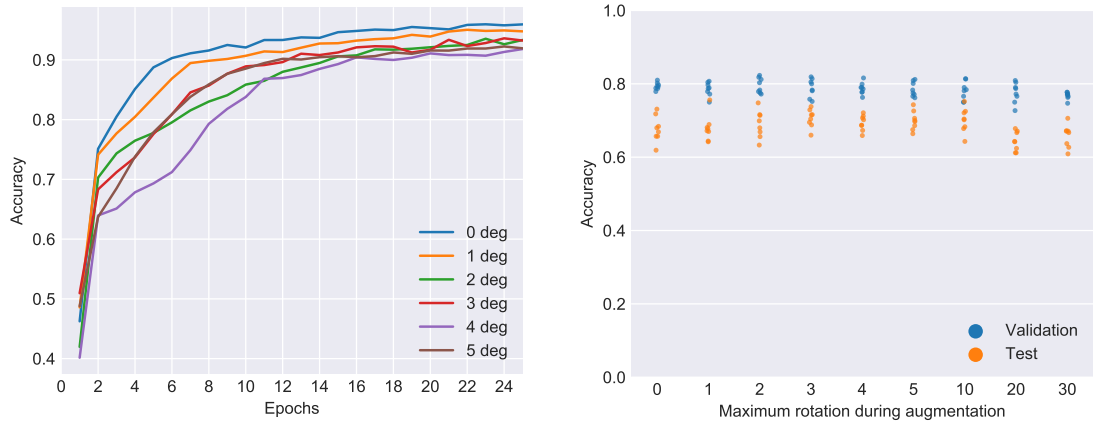
Judging from Figure 5.12, data augmentation by random rotation seems to have a marginal impact on accuracy. The best model that was obtained using up to 1 degree random data rotation performed 2.5% better compared to the baseline (75.6% versus 73.1%). Moreover, it appears to have a negative influence when the



(a) Training accuracy

(b) Validation and test accuracy

Figure 5.11: Model accuracy versus number of input samples.



(a) Training accuracy

(b) Validation and test accuracy

Figure 5.12: Accuracy versus the maximum allowable data augmentation range in degrees.

rotations are large. The benefits of data augmentation might be mitigated by the way of anti-aliasing the rotated images. More experimentation with data augmentation should be performed to evaluate whether it could be a way of improving model accuracy with a limited number of samples.

5.4.6. Sample randomization

To simulate human inconsistency, a varying percentage of samples was randomized, as can be seen in Figure 5.13. Recall that the number of output classes for the baseline architecture is 6. This corresponds exactly to the expected $100/6 \approx 17\%$ accuracy when 100% of the samples in randomized which can be observed in Figure 5.13a. In the right-hand figure, an almost linear relation appears between validation accuracy and % of randomized samples. Validation accuracy decreases because both the training and the validation set are randomized. More importantly, the figure shows that the test accuracy remains relatively constant with increasing randomness. This shows that the model is quite robust and is not over-sensitive to noise due to controller inconsistency, which is a promising result.

(a) Training accuracy

(b) Validation and test accuracy

Figure 5.13: Accuracy versus the percentage of randomized samples.

5.5. Conclusions and discussion

This section summarizes the most important insights from the results and points out any limitations and caveats that should be taken into account. Furthermore, the next steps for further research are discussed.

The following analysis insights reflect on the sub-questions as introduced in section 5.1.

1. Action resolution – Model accuracy decreases considerably with an increasing number of output classes. The network should be improved to account of this loss in accuracy.

2. Network architectures – The network architectures that were analyzed do not have a large impact on model accuracy. However, network architectures did have a large influence on training time.

3. Input dimensions – Input dimensions have a relatively large impact on model accuracy. Given that larger dimensions result in lower accuracy, special attention should be given to overfitting.

4. Number of input samples – Model accuracy increases steeply up to 1000 input samples. After that, accuracy stabilizes and added samples only marginally add to test accuracy. To achieve a reasonable 70% accuracy, 300 samples are sufficient.

5. Sample randomization – Controller inconsistency up to 30% did not notably influence test accuracy, which is a promising result when considering using human generated data.

6. Data augmentation – Randomly rotating the input samples did not result in demonstrable improvements on validation or test accuracy.

Besides these insights, the following caveats and opportunities for improvement are pointed out:

First, the traffic scenarios that were used in this proof-of-concept analysis are considered quite simplistic. Although controllers solve conflicts in a pairwise manner, further research should evaluate more advanced scenarios with multiple aircraft having potentially different speeds, altitudes and headings in one sector. These increasingly complex scenarios will resemble a more real-life application. Adding to this, a more advanced interpretation should also include speed and altitude changes as possible resolutions, instead of heading changes only. A last addition to the scenario data is to include more sector information, such as traffic flow, constraints and exit waypoints.

Secondly, a considerable limitation to this preliminary analysis is that all data is generated using a deterministic resolution algorithm. It is not evident that human controllers behave in a consistent, trainable manner when solving conflicts using the SSD. A next step is therefore to perform an empirical experiment where

human resolution data is gathered. Even though it was shown that the CNN is robust up to a certain degree of randomness, usability in practice still has to be examined.

And lastly, the algorithm experienced quite some overfitting and overall accuracy could be improved. This can be done by using 1) a different loss function, 2) dropout layers and 3) better data augmentation. A continuous loss function will expectedly train the network more accurately than the categorical discrete classes function that was used in this analysis, as magnitude of the heading error will be taken into account, rather than a binary right or wrong. Two, differences between validation and test accuracy are quite large. Implementing dropout layers could help to reduce overfitting of the model, which seems necessary as Another option would be to use an 'all convolutional network', as discussed in section 4.1.3. Three, when an empirically generated dataset is used, every sample becomes valuable and should be used as effectively as possible. This can be done using data augmentation, which did not show promising results so far. Additional methods of augmentation should be examined to effectively train the network with a low number of samples.

These three improvements are the aim of the experiment as performed in Part I.

III

Appendices

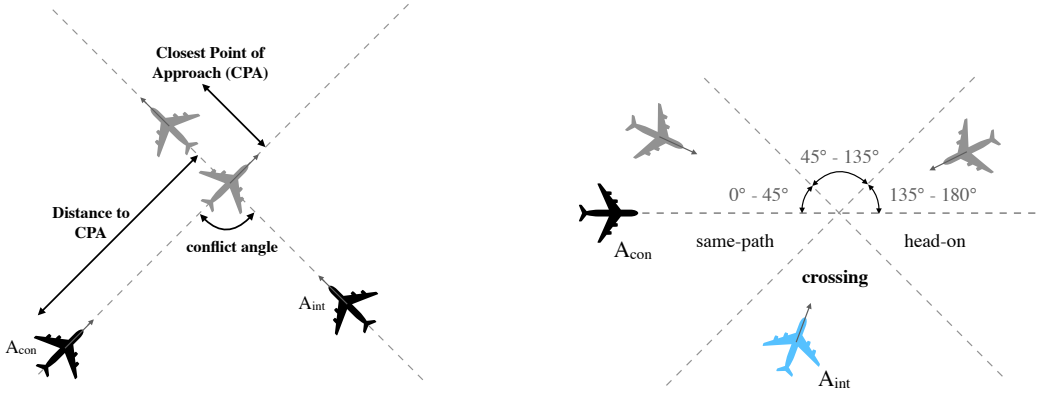
A

Experiment details

This chapter details the experiment design and obtained dataset. Initially, the scenario design parameters and the experiment briefing are provided. Subsequently, the participants' remarks during the experiment are provided. Finally, the dataset is analyzed on a temporal scale and on a higher abstraction level concerning aircraft selection, geometric resolution preference and losses of separation.

A.1. Scenario design

The scenarios are designed to be repeatable with clearly distinguishable two-aircraft conflicts. Each Scenario (S) consists of 20 aircraft, thus 10 predesigned conflicts. The conflicts are designed according to the three parameters in Figure A.1a. In the final experiment, only the conflict angles are varied to obtain the set of conflicts. To make the learning more robust, the supervised learning algorithm should see similar conflicts multiple times. Due to the limited experiment duration, the designed conflicts are restricted to have 0nm CPA. The conflict angles are constrained to *crossing* conflicts, i.e. the interval [45, 55, ... , 135] deg, as shown in Figure A.1b. Each angle occurs twice over the two scenarios, see Table A.1. The spacing between consecutive conflicts is at least 20nm to minimize interference, while maintaining a reasonable workload to keep the controller engaged.



(a) CD&R parameters

(b) Conflict Types

Figure A.1: Naming conventions for conflicts.

Table A.1: Conflict parameters of the designed conflicts.

(a) Scenario 1

Conflict	**Conflict Angle [deg]**	**CPA [nm]**
C1	85	0
C2	95	0
C3	65	0
C4	135	0
C5	75	0
C6	85	0
C7	75	0
C8	125	0
C9	135	0
C10	105	0

(b) Scenario 2

Conflict	**Conflict Angle [deg]**	**CPA [nm]**
C1	105	0
C2	45	0
C3	95	0
C4	45	0
C5	115	0
C6	65	0
C7	55	0
C8	125	0
C9	55	0
C10	115	0

A.2. Experiment briefing

The following pages show the Experiment Briefing as provided to and discussed with all participants.

Experiment Briefing

Strategic Conformal Automation using Deep Learning

September 27, 2018

Abstract

The goal of the experiment is to safely manage an airspace by providing aircraft with directional commands. This will be done in Air Traffic Control simulator SectorX. The experiment consists of 4 runs that take 20 minutes each. All actions will be logged to create a conflict resolution algorithm using machine learning.

1 The objective

The main objective is to safely guide all aircraft to their respective exit way-points in an efficient manner. More specifically:

- Loss-of-Separation should be avoided. It occurs when the radii surrounding the aircraft make contact.
- Aircraft should be controlled *within* the sector.
- Aim to guide the aircraft to their exit waypoint as efficiently as possible.
- Whenever the aircraft near the end of the sector, a transfer of control (TOC) should be given.

  All aircraft fly on the same flight level. Altitude changes are not allowed. Without instructions, the aircraft will proceed in a straight path.

2 The display and the sector

The experiment will be conducted with an air traffic control simulator (SectorX), of which an image is shown in Figure 1. The sector as displayed on screen is loosely based on Sector 3 above The Netherlands. Characteristics of the sector traffic are discussed below.

- Traffic arrives from the south or east.
- There is one major traffic flow originating from the south running towards waypoint MIFA. Aircraft with different origins have other target waypoints.
- All aircraft enter the sector at equal speeds (250 kts) and equal altitude (FL100).

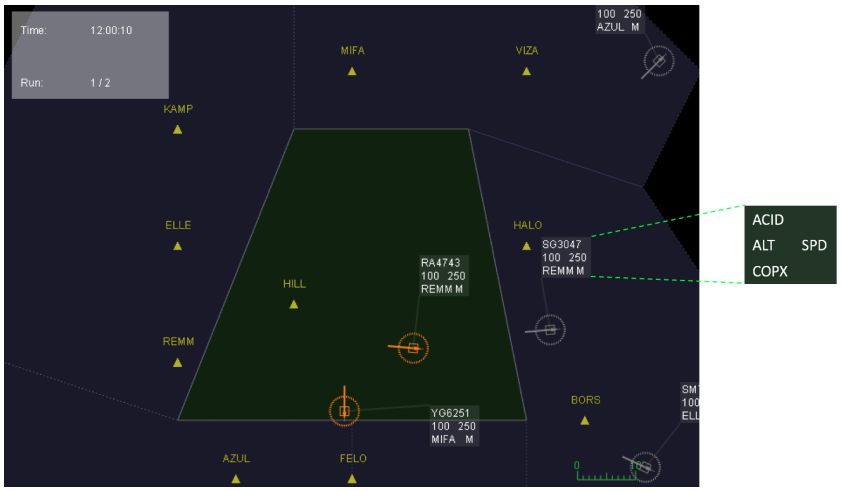


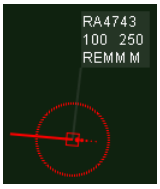
Figure 1: The SectorX display. The dark green shaded trapezoid indicates the sector area. The yellow triangles denote the waypoints. The aircraft are depicted in grey and orange.

The aircraft label – as shown in Figure 1 – denotes the following information:

- **ACID**: Aircraft ID
- **ALT**: Altitude in thousands of feet (FL)
- **SPD**: Indicated Airspeed
- **COPx**: The required exit waypoint

When aircraft are in conflict, their color will change from grey to orange or red, depending on the Time to Loss-of-Separation limits given in Figure 2.

(a) Caution: Time to Loss-of-Separation less than 120 seconds.



(b) Warning: Time to Loss-of-Separation less than 60 seconds.

Figure 2: The color of the aircraft symbol indicates if the aircraft is in conflict. When the symbol changes to *green*, the aircraft is directed at its exit waypoint.

3 The Solution Space Diagram (SSD)

The Solution Space Diagram (SSD) is used in half of the scenarios. Whenever an aircraft is selected, the SSD is displayed. The SSD incorporates the Time-To-Contact through color-coding, as displayed in Figure 3.

Interpretation of the SSD
All possible combinations of speeds and headings lie within the two green circles, where the inner and outer circles represent the minimum and maximum speed respectively. The purple dashed circle shows the indicated airspeed, the pink line indicates the aircraft's speed vector, and the purple line indicates the bearing towards the exit waypoint (COPx). Within the green circles, black space indicates 'solution space', whereas grey,

orange, and red areas indicate future Loss-of-Separation whenever the tip of the speed vector is pointed in these areas.

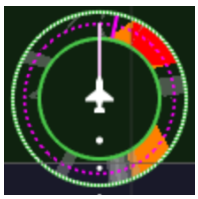


Figure 3: The Solution Space Diagram (SSD). Time to Loss-of-Separation denoted by: grey (> 120s), orange (< 120s), and red (< 60s).

4 Command display

Commands are passed through the command display as depicted in Figure 4.

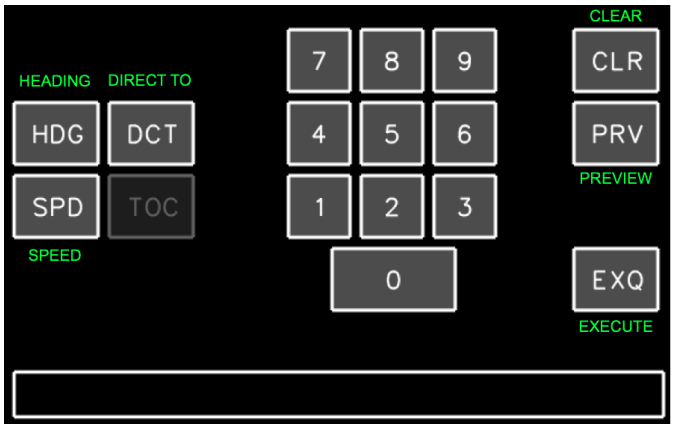


Figure 4: The command display

- HDG: Give a heading change command [1 - 360 deg].
- SPD: Give a speed change command [200 - 290 kts].
- DCT: Give a Direct To exit waypoint command. Automatically changes the heading towards the assigned exit waypoint.
- CLR: Clear the command line.
- PRV: Preview the selected commands.
- EXQ: Execute the string of commands.

For example, to change an aircraft's heading west (270 deg), one should use the following sequence of clicks: [Select the aircraft using the mouse] → [HDG] → [2] → [7] → [0] → [EXQ].

5 The aircraft

Only one type of aircraft is encountered in this sector, with performance comparable to a Boeing 737-400. The airspeed limits are shown in Table 1.

Table 1: Airspeed limits of the aircraft

B737-400	**IAS [kts]**
IAS min	200
IAS initial	250
IAS max	290

6 Planning

A time-schedule of the experiment is shown in Table 2. Breaks can be shortened or extended on request.

Table 2: Time-planning of the experiment

Type	**Duration [m]**	**Elapsed time [m]**
briefing	5	5
training	5	10
run 1	20	30
break	5	35
run 2	20	55
break	10	65
run 3	20	85
break	5	90
run 4	20	110
debriefing	10	120

7 Training runs

Training run 1: Controlling one aircraft

1. Direct the aircraft towards the west
2. Slow the aircraft down to 200 kts
3. Direct the aircraft north
4. Increase speed to 290 kts and direct it towards its exit waypoint

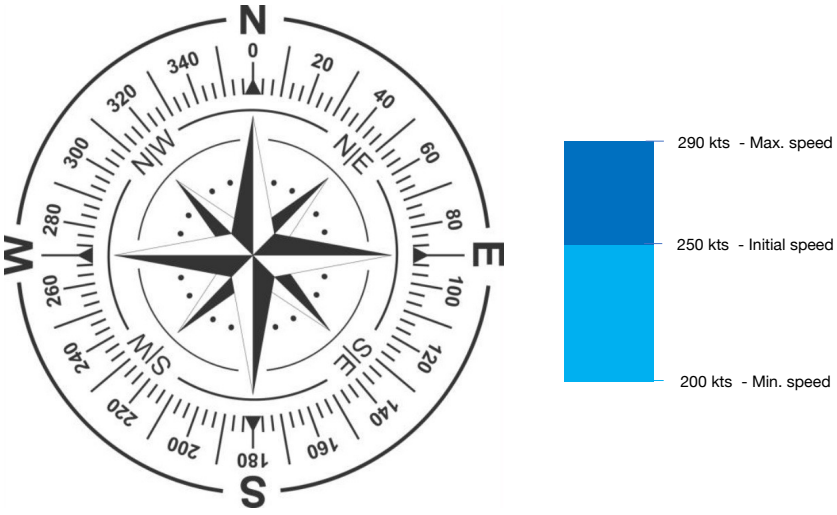
Training run 2: A simple conflict

1. Observe the screen and note how the color of the aircraft changes to orange
2. Resolve the conflict

Training run 3: Introduction of the SSD

1. Select both aircraft and observe how the SSD changes over time
2. Resolve the conflict
3. Observe the SSD
4. Direct both aircraft to their respective exit waypoints

8 Cheat Sheet



If you have any questions left unanswered, please don't hesitate to ask!

A.3. Participant remarks during experiment

Table A.2 shows the noted participant remarks during the experiment. All remarks were translated from Dutch to English for this report. The remarks give insight in some of the applied strategies and usages of the SSD. Additionally, certain comments indicate that the briefing and training runs were not fully sufficient mitigate all training effects.

Table A.2: Logged comments by the participants during the experiment. Based on these comments, some commands have been removed from the dataset.

Participant	Run	Time (min)	Comment
P1	1	18	This move was too extreme
P1	2	8	I thought HDG changes were kind of cheating
P1	2	18	The north of the sector is easy
P1	3	18	I use SPD as backup command
P1	4	18	Command on OM3185 was a mistake
P2	1	2	It takes a while before the A/C indicator becomes green
P2	2	18	Would be nice to have a visual indication of heading change
P2	3	5	I like the SSD, you can solve conflicts way ahead of time
P3	1	1	I thought heading commands were relative. First command incorrect
P3	1	2	I dislike the update rate
P3	1	7	Accidental TOC of VS4694
P3	1		Accidental TOC of SM7071
P3	1		Accidental TOC of PG4310
P3	3	4	SSD causes higher workload because you have more options, it is less gambling
P3	3	8	Very difficult
P4	3	15	Low workload
P4	3	16	I am trying to visualize the SSD now that it is not there anymore
P4	4	2	Lost focus, causing Loss of Separation
P5	1	1	In the SSD I see information of aircraft that are not in the sector yet.
P5	1	5	I steer aircraft into the grey SSD area to solve the conflict later
P5	3	12	it is easier with the SSD turned on because there are fewer options
P6	1	5	Accidental TOC of SM7071
P6	3		I would not steer so close to other aircraft without the SSD
P6	3		Without the SSD, I am paying more attention to the general picture strategy
P6	3		SSD is an information overload, I would like a more basic SSD
P7	1	6	Accidental Loss of Separation. Thought that the circles could touch.
P7	1	14	LoS: Tried to steer HDG 0 but error message did not clear automatically.
P8	1	0	Accidental Loss of Separation due to familiarization with the system
P8	1	12	I am trying to steer away from the entry points
P8	1		Felt like a training run, I am still building my internal model.
P8	1		I miss-clicked a couple of times, I cannot focus on the screen while giving commands
P8	1		Clicking does not work perfectly all the time
P8	1		I am trying to model the traffic streams in my head
P8	2		I place the main aircraft vector perpendicular to the intruding aircraft vector
P8	4		With the SSD: Lower SA, I am more focused on local conflicts instead of the overview
P9	1	3	I use the label vector to steer the around around each other
P10	2	2	Erroneous command given
P10	3	14	This was messy, not how I would normally do it
P11	1	1	I just completed the course
P11	1	11	Accidental heading command
P11	3	1	I got used to having the SSD
P11	3	10	I aim the speed vector to the middle point of the intruding aircraft
P12	1	6	LoS due to premature DCT command
P12	2	19	Very unusual sequence of commands with large relative values
P12	3	7	LoS due to premature DCT command
P12	4	9	Accidental 360 degree turn of aircraft

A.4. Commands over time

This section shows all commands per Participant (P) over time. Figures A.2, A.3 and A.4 show the heading HDG, speed SPD and direct to DCT commands respectively.

Figure A.2: Heading commands over time per participant

Figure A.2 shows clear differences in the number of heading commands given per participant, ranging from approximately 40 to 140 HDG commands for P1 and P5 respectively. This is a first indicator of strategy heterogeneity in the population. In addition, the chart shows that command density in time can differ

from run to run for one participant. The scenarios in Runs 1 & 3 and 2 & 4 are identical. Therefore, if the participants were 100% consistent, the runs would look identical as well, which is not the case. P4, for example, gives less heading commands in Run 4 compared to Runs 1-3. This change in strategy may be the cause of the low achieved accuracies of P4's individual model compared to the general models.

This temporal view also indicates that the moment at which commands are given varies per participant and per run. This stresses again that besides command *type*, *direction* and *value*, command time is also parameter in ATCo heterogeneity and potentially strategy.

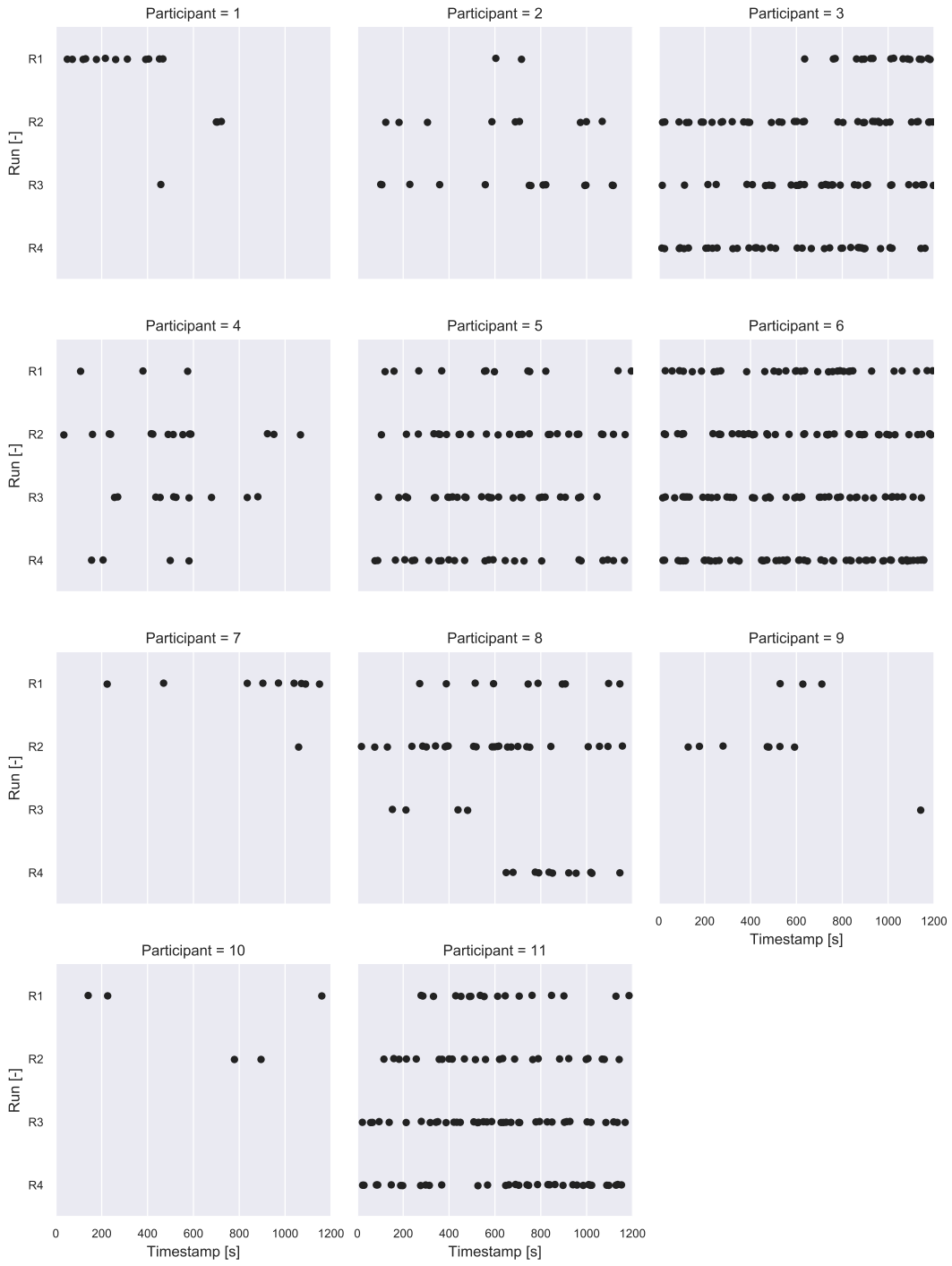


Figure A.3: Speed commands over time per participant. Participant 12 did not give speed commands.

Figure A.3 shows the timeline of speed commands per participant. Substantial differences in strategy can be seen both inter and intra-participant. Participants 1 and 3 clearly changed their command *type* strategy during the first run, after which it remains reasonably constant. This change in strategy implies that Run 1 (and in some cases Run 2) still includes training effects. Ideally, Run 1 should be removed from the training set. However, since Run 4 is separated for validation, this would leave too few samples in the training set. Besides, certain participants (P7, P9, P10) only used SPD commands in emergency situations. A finding that is confirmed by the participants' remarks, see Table A.2.

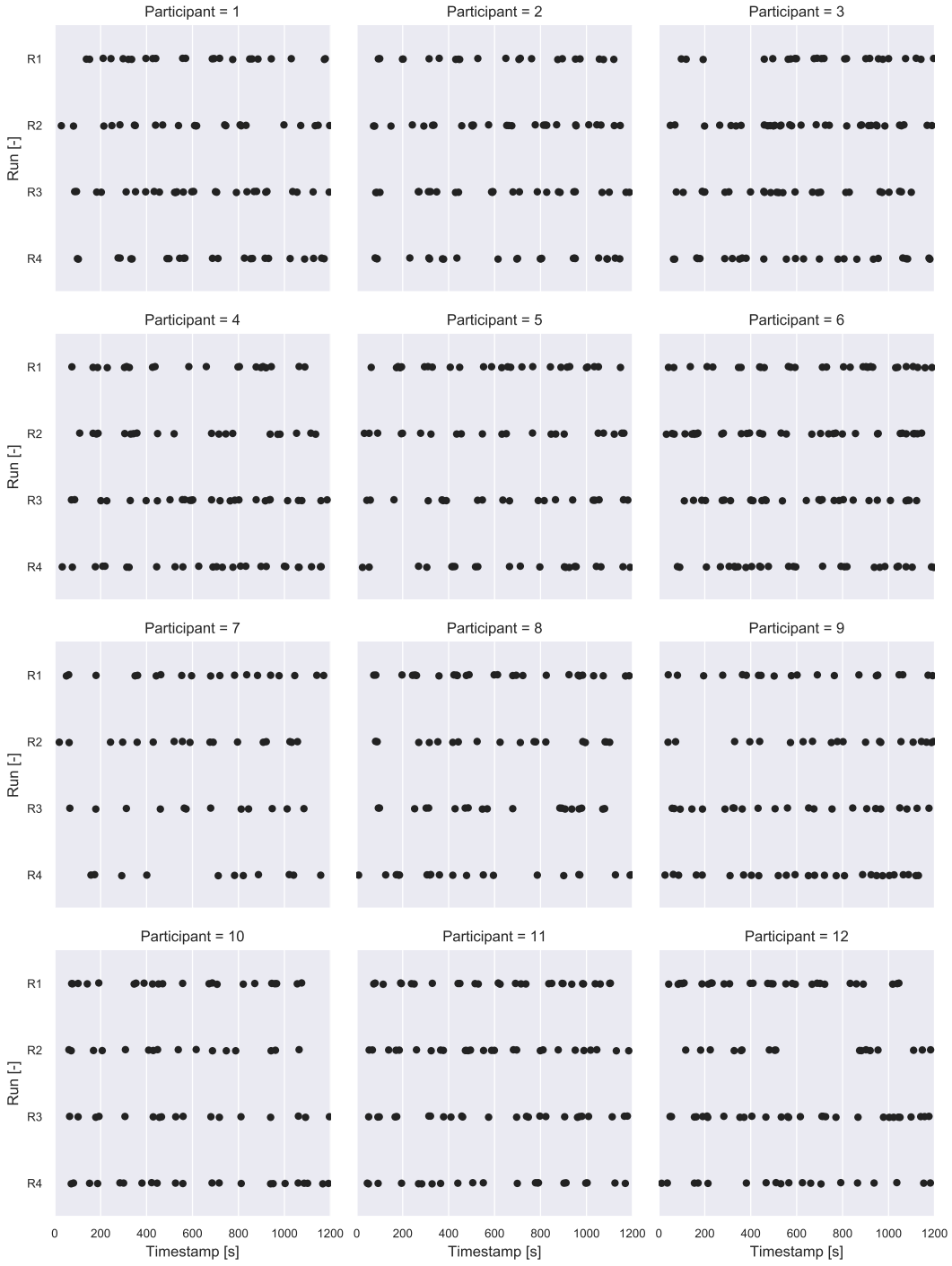


Figure A.4: Direct To commands over time per participant

Figure A.4 shows the ‘Direct to exit waypoint’ commands for all participants. The differences inter and intra-participant are not substantial enough to draw differentiating conclusions.

A.5. Preferred aircraft flow

Figure A.5 shows the preference for either the main or the intruding traffic flow per participant. The total number of commands per flow is well-balanced with 1738 and 1607 commands for the main and intruding flow respectively. Although the preference is balanced on average, specific participants show different strategies. For example, P5, P7 and P9 show a clear preference for the main flow, while P1, P4, and P6 provided more commands to the intruding flow. Besides, certain participants show a clear preference for one flow (P9), while others are impartial (P8). These differences confirm yet another abstraction level of consistency, i.e. aircraft choice, as included in Westin et al.'s consistency framework (Figure 2.4). As Figure A.5 shows heterogeneity in the population, it seems evident that aircraft choice is part of controller strategy and should therefore be included in consistency and conformity metrics. Moreover, aircraft preference consistency seems to coincide with the participant consistencies as determined in the paper.

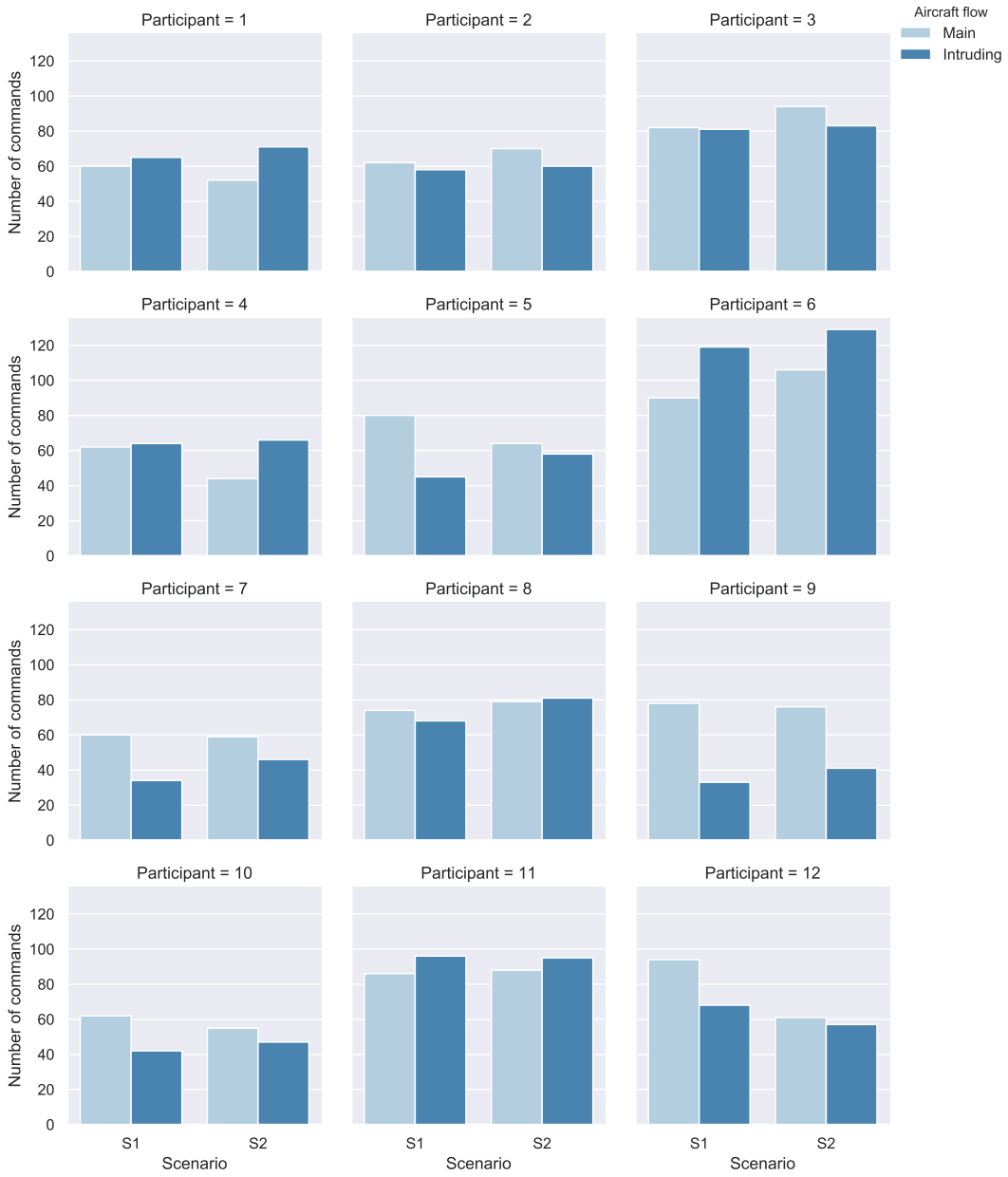


Figure A.5: Number of commands given to either the aircraft in the main or the intruding traffic flow.

A.6. Preferred geometric resolution

Figure A.6 shows the number of commands that resulted in the controlled aircraft going *in front* or *behind* the conflicting aircraft (see Figure 2.3). Certain participants (P7, P9, P10) show more consistent behavior than others (P1, P8, P11). Conflict geometry preference is implemented in Westin et al.'s consistency framework in the high-level decision stage. The heterogeneity in the population shows that geometry preference indeed varies per participant and is most likely part of a controller's strategy. In addition, P7 and P10 show a strict procedure that seems to drive the resolution decision rather than situation at hand.

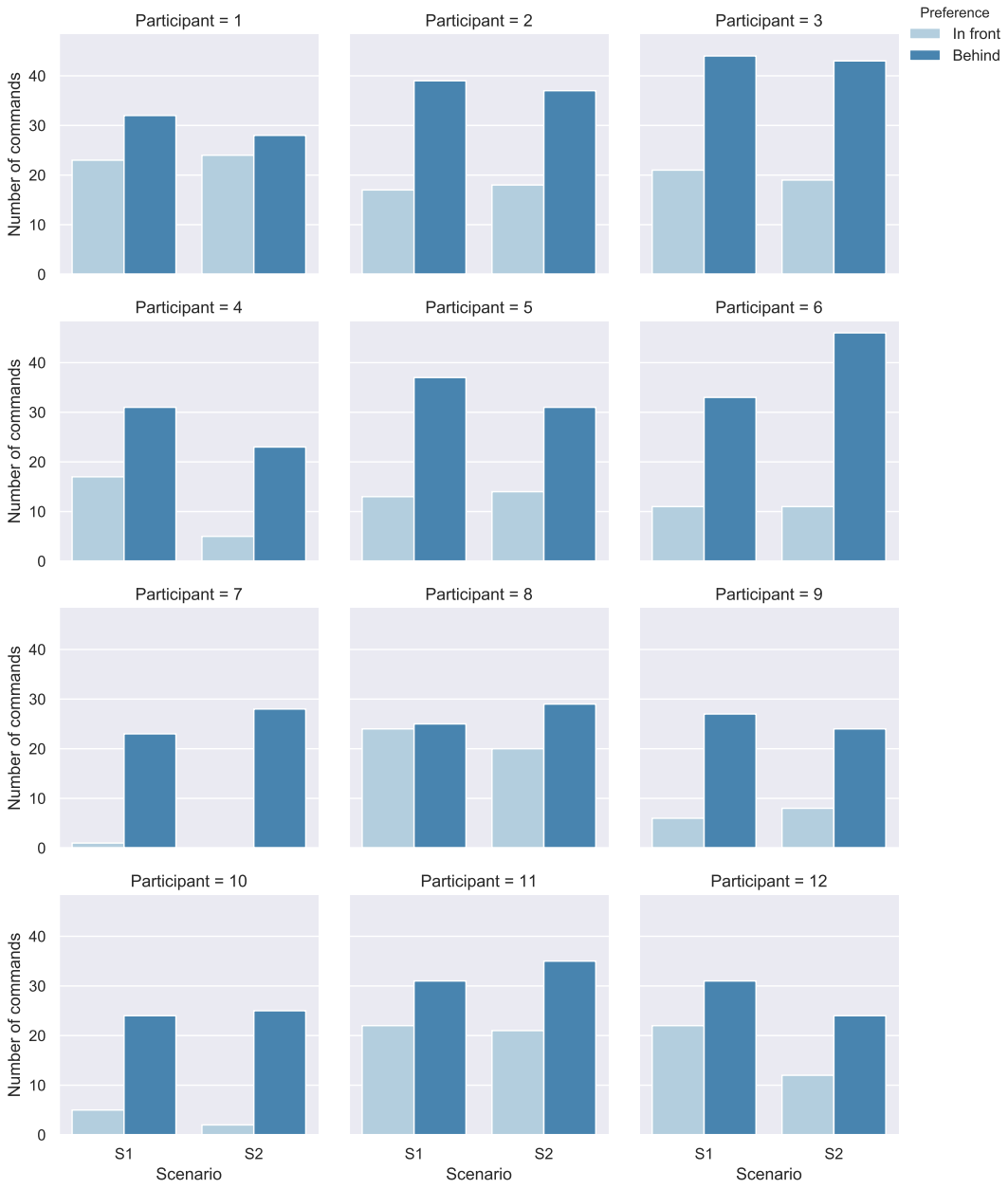


Figure A.6: Geometric preference per participant. Only the first command per aircraft is included.

A.7. Loss of Separation

Figure A.7 shows the number of losses of separation (LoS) per participant and SSD condition. The novices (P1-P6) and intermediate controllers (P7-P12) caused 11 and 10 losses of separation respectively, indicating no considerable difference in skill level in this regard. Furthermore, Figure A.7 shows considerably less LoSs with the SSD (6) than without (15). This implies that the SSD could indeed increase ATCo performance with regard to safety. In consequence, the SSD acts as its own type of strategic conformal automation through *information integration*, rather than action implementation. The SSD is conformal in the sense that the controller can always resolve the conflict in line with his or her personal strategy.

SSD availability during the experiment was determined by a Latin square thus training effects are not expected to have influenced these results.

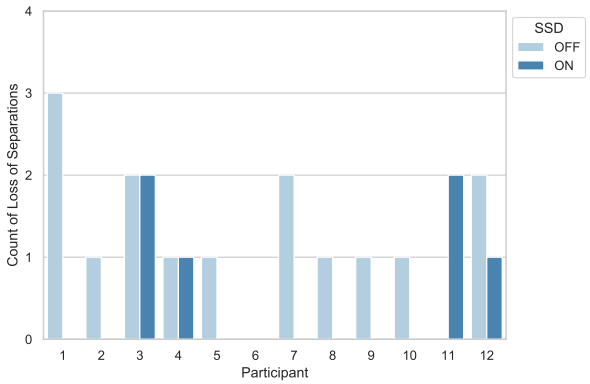


Figure A.7: Loss of separations during all runs. A loss of separation occurs when aircraft are closer together than the separation minima allow, i.e. when their protected zones ($D = 5$nm) make contact.

B

Training methodology

This chapter consists of three parts: B.1 illustrates the code architecture including two main methods in pseudo code. B.2 displays the results of data preprocessing methods and B.3 shows a visualization of the network weights of the first convolutional filter.

B.1. Code structure

The code structure of the thesis project is given in Figure B.1. It shows the data pipeline including data generation in *SectorX*, data preprocessing, model-training, model (cross-)validation and statistical analysis. Two modules in this pipeline have been highlighted using pseudo-code: model training (`train_model()`, Module 2) and cross-validation (`cross_validation()`, Module 3).

Module 2 Model Training Sequence *train_model()*

```
Initialize config
Set random seed {python, numpy, tensorflow}
Require: ssd_array, command_dataframe
for P_train in {P1 − P12, all} do                          ▷ 'all' selects all data for the general models
    for abstraction_level in {type, direction, value} do
        for ssd_condition in {off, on, both} do
            x_data, y_data ← prepare_training_set(P_train, abstraction_level, ssd_condition, all_dataframes)
            if P_train = all then
                Randomize and cap data                     ▷ Cap to average number of samples per participant
            Split data in k = 5 folds
            for fold in k-folds do
                x_train, x_val, y_train, y_val ← k-fold(x_data, y_data)
                model ← create_model()                     ▷ Define network architecture using Keras (Chollet, 2015)
                model ← compile_model(model)               ▷ Compile loss function and Adam optimizer
                for epoch in N_epochs do
                    for step in steps-per-epoch do
                        Compute network loss using Categorical Entropy Loss function
                        Update network parameters θ using first-order gradient descent parameter optimzer
                    Evaluate model validation metrics      ▷ Accuracy, MCC, Informedness, F1 Score
                    Calculate and save Confusion matrix
                    if validation MCC improved then
                        save model parameters to disk      ▷ In .h5py format
                    else
                        continue
Remove redundant underperforming models
```

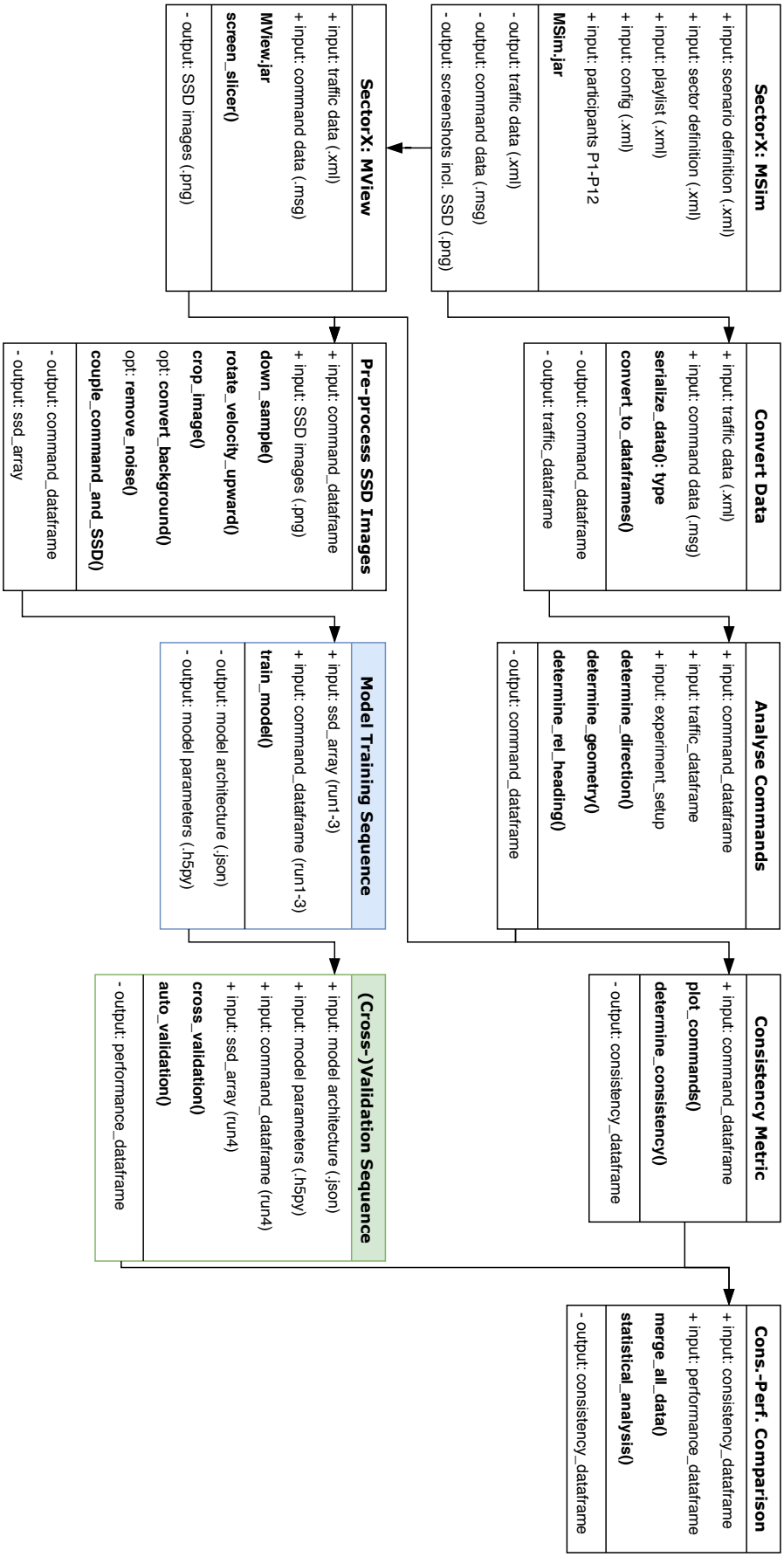


Figure B.1: Code environment from data generation to model analyses. SectorX is a Java-based application and the remaining codebase is written in Python. Keras (Chollet, 2015), based on TensorFlow (Abadi et al., 2016), is used for training the CNNs.

Module 3 Cross-Validation sequence *cross_validation()*

```
Initialize config
Require: model_parameters, model_architecture, test_data (ssd_array, command_dataframe)
for P_train in {P1 − P12, all} do                    ▷ 'all' selects all data for the general models
    for abstraction_level in {type, direction, value} do
        Load model_architecture                      ▷ Identical for all abstraction levels
        Load model_parameters P_model
        for P_val in {P1 − P12} do
            Load test_data P_val                     ▷ Experiment Run 4, specified for abstraction level
            Evaluate accuracy and MCC of model P_train
Save performance data to disk
```

B.2. Data preprocessing

The SSDs are preprocessed following the sequence in Figure B.1. As described in Part I, four data processing techniques are evaluated: cropping, background conversion, noise removal and size reduction. The results are shown in Figure B.2. preprocessing the SSDs results in minor performance improvements of a few MCC points. Although *direction* and *value* predictions improve slightly, *type* predictions obtain lower MCC values. Because the same input dataset is used for all abstraction levels, only cropping and down-sizing are applied to obtain considerably faster computing times.

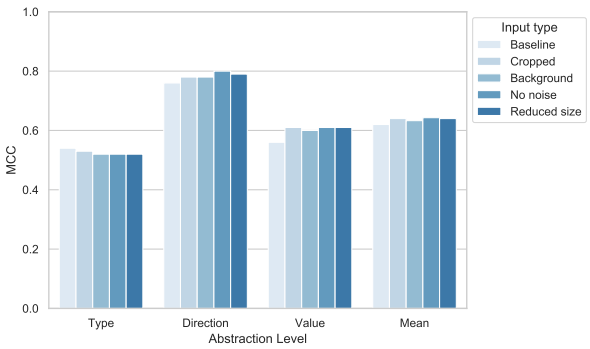


Figure B.2: Mean achieved model performance in MCC for all participants and conditions using different data preprocessing methods.

B.3. Convolutional Filters

The parameters of the first 32 convolutional filters are shown in Figure B.3. Every filter applies a different transformation matrix to the pixel values. Figure B.3 shows that certain filters capture only the velocity vector or exit waypoint bearing. On the other hand, the filters do not detect a binary (black–white) distinction between solution space or a Forbidden Beam Zone (FBZ). Certain filters (e.g. Filter 32) only detect edges, where a boundary between white and red exists. The visualization of these filters indicates that the network is able in detecting the most important elements in the SSD.

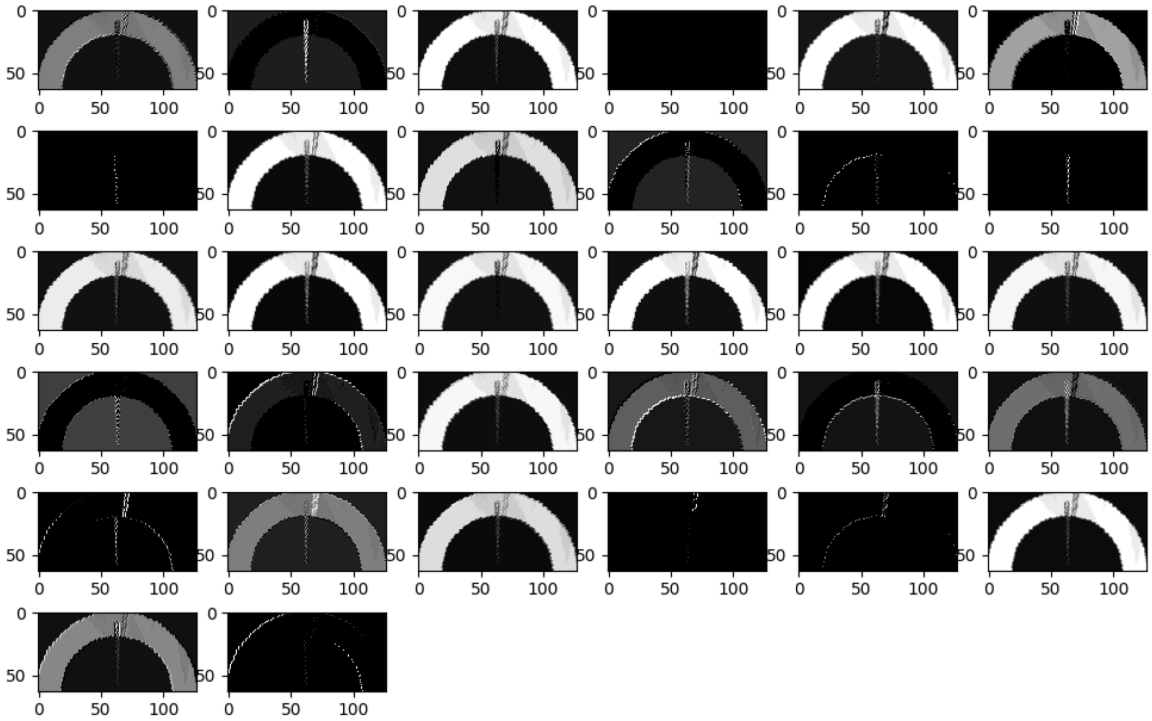


Figure B.3: Parameter (weight) visualization of the 32 filters of the first convolutional layer.

C

Validation per participant

This chapter provides the methods of validation for all participants. Section C.1 shows the MCC and accuracy values during training. Section C.2 shows the ultimately achieved MCC scores using k-fold validation, Section C.3 shows the confusion matrices of the test dataset and Section C.4 shows the cross-validation results.

C.1. Performance during training

Figure C.1 shows the MCC scores after each epoch during training. The MCC values are obtained by validating the current model with a validation set that was separated using stratified k-fold validation. Inter and intra participant performance is clearly visible. The large spread between folds indicates that data quantity is a limiting factor, because random sampling affects the achievable performance.

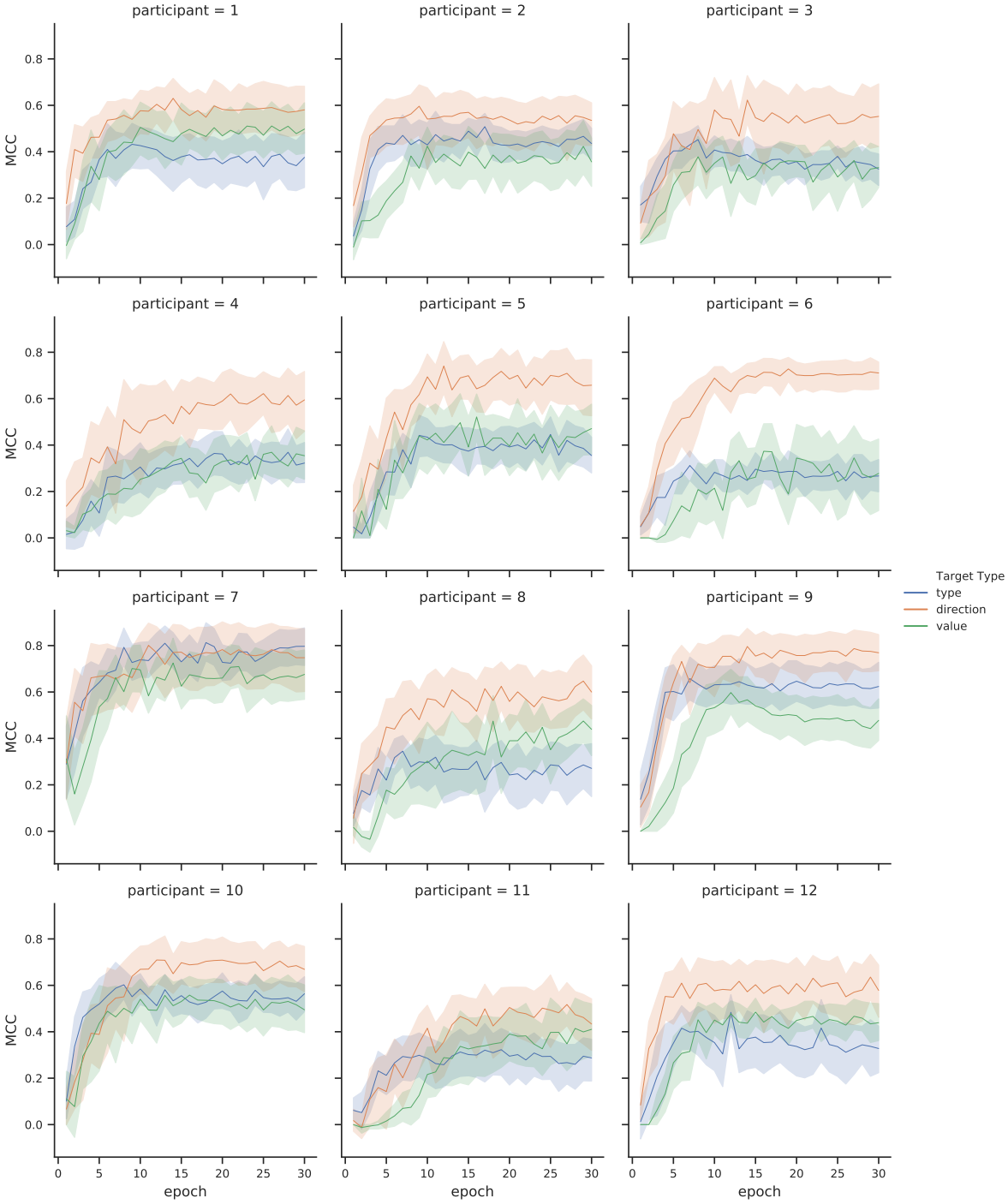


Figure C.1: MCC values during training for all participants and abstraction levels. The spread surrounding the trend lines is defined by the best and worst performing validation folds.

Figure C.2 shows the accuracy scores after each epoch during training. The trend lines are comparable to MCC scores, but the spread between folds appears to be smaller. In general, the accuracy scores are higher than the MCC scores and overfitting occurs more clearly. For example, The *type* accuracy of participant 3 deteriorates after 8 epochs. Contrarily, the *type* accuracy of participant 4 still shows an increasing trend at epoch 8. Therefore, training cannot be capped at a pre-specified epoch number. As a result, the k-fold validation performance scores are used to select the best performing model parameters, which are then saved and validated using the test dataset.

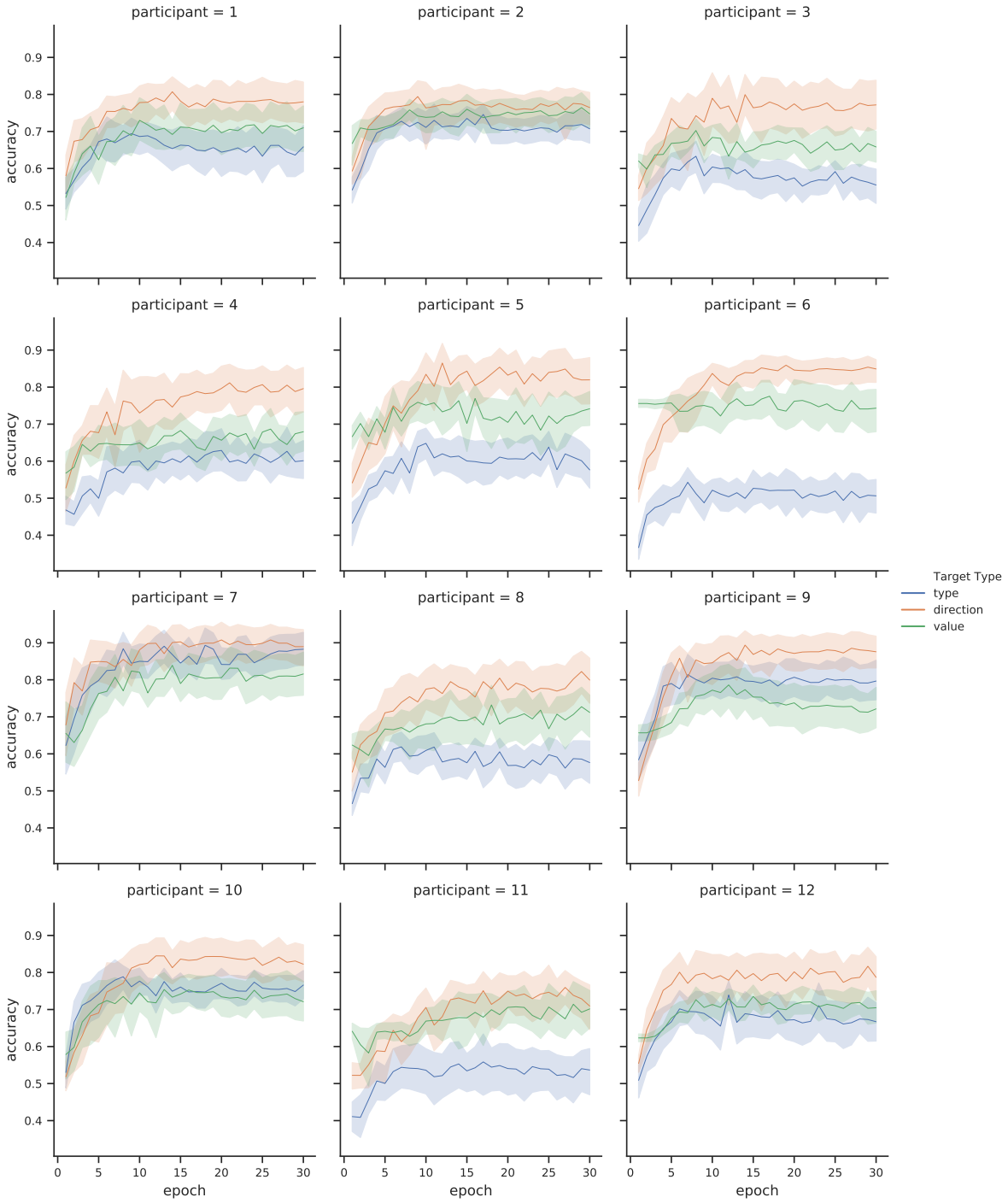


Figure C.2: Accuracy values during training for all participants and abstraction levels. The spread surrounding the trend lines is defined by the best and worst performing validation folds.

C.2. Performance per Participant

Figure C.3 shows the highest achieved MCC value per fold per abstraction level for all participants for both conditions. This chart gives insight in the effect of the availability of the SSD per participant per abstraction level. It shows that certain participants experience a positive effect due to SSD availability (e.g. P7 or P9), while others are relatively insensitive (P3, P12) or experience negative effects (e.g. P6). This indicates that the SSD can have a beneficial effect on controller consistency, depending on the controller. Intuitively, training with the SSD might influence its effect on controller consistency. The fact that P7 and P9 have extensive experience with the SSD substantiates this intuition.

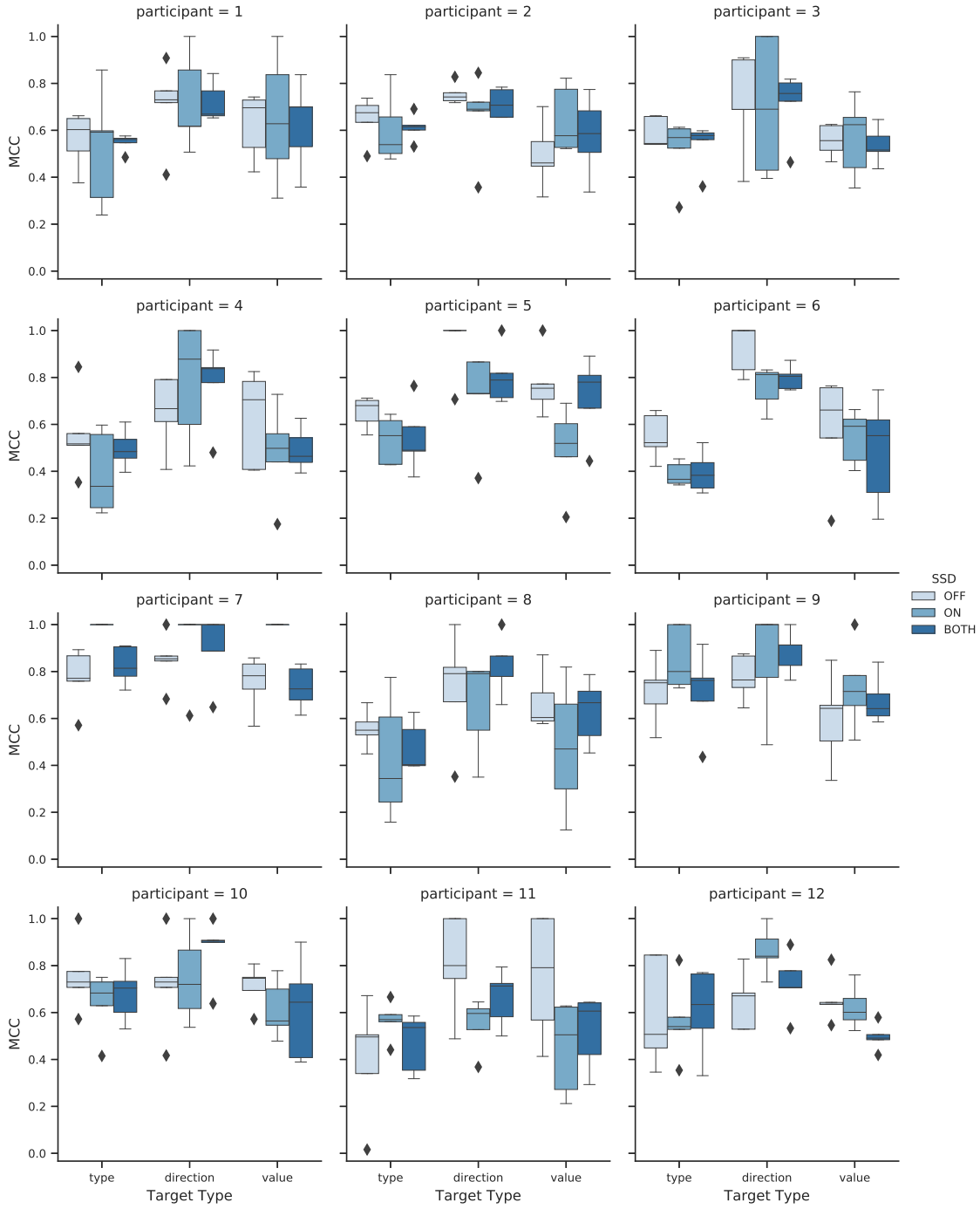


Figure C.3: Validation results per k-fold for all participants and abstraction levels.

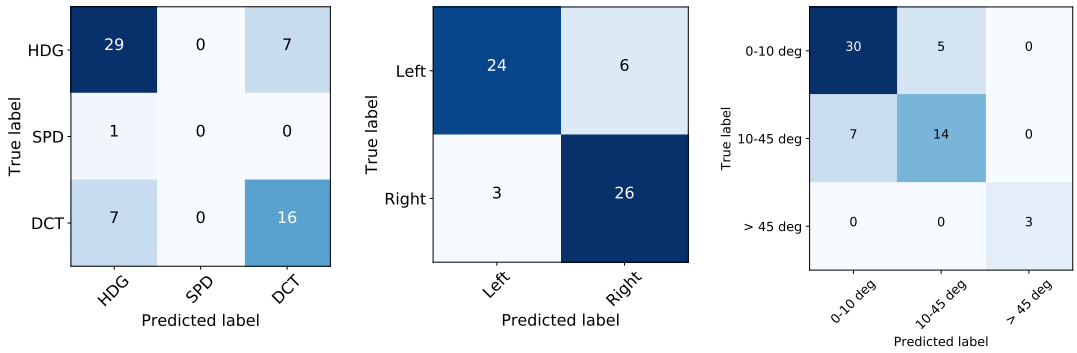
C.3. Confusion Matrices

This chapter shows the confusion matrices of the test results per participant. A confusion matrix gives insight in the distribution of True Positive (TP)s, False Positive (FP)s, True Negative (TN)s and False Negative (FN)s by comparing the *true* and *predicted* commands properties. A darker diagonal in the matrix indicates relatively high MCC scores.

Certain confusion matrices, e.g. P2 *value* predictions, are incomplete because the participant did not use all classes. The confusion matrices provide a means gain insight in incorrect predictions. For example, the *type* prediction model of Participant 8 oftentimes predicts SPD or DCT commands while a HDG was given. These errors might be explained by the change of strategy during Run 4 as shown in Figure A.3. As the confusion matrix shows, a test dataset containing two strategies is does not result in accurate validation results.

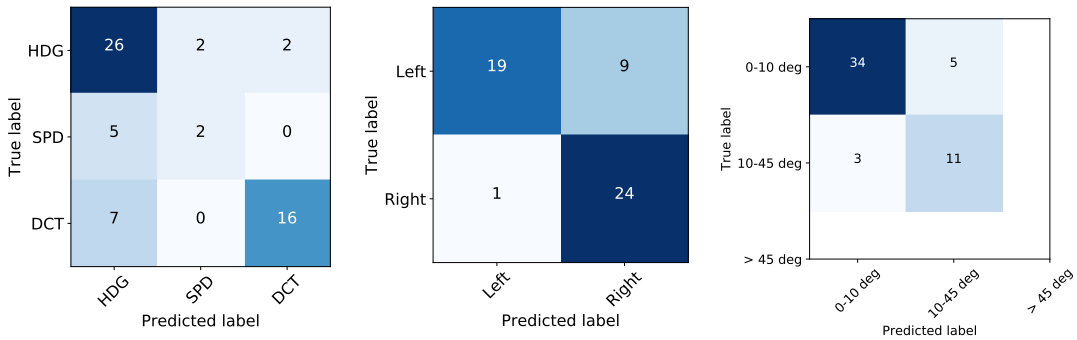
Another peculiar example is *type* prediction of P1, P4 and P9, where SPD commands were never predicted correctly. This can be explained using Figure A.3. P1, P4 and P9 sparsely or very irregularly used SPD commands, making a correct prediction difficult. P10 and P12 test datasets did not include speed commands caused by the fact that P10 and P12 used 5 and 0 speed commands during Runs 1-3 respectively.

The value predictions of P12 are not accurate relative to other participants, as confirmed by its confusion matrix. An explanation for this might be the irregular distribution of True labels, as only very small or very large heading deviations are used. This 'non-linear strategy' might be difficult to capture by the input–output mapping of the neural network.



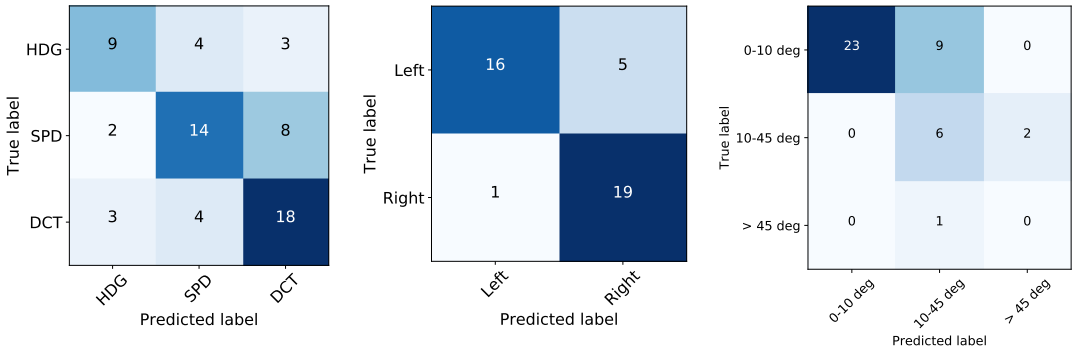
(a) Type (b) Direction (c) Value

Figure C.4: Participant 1: Confusion matrices of test results.



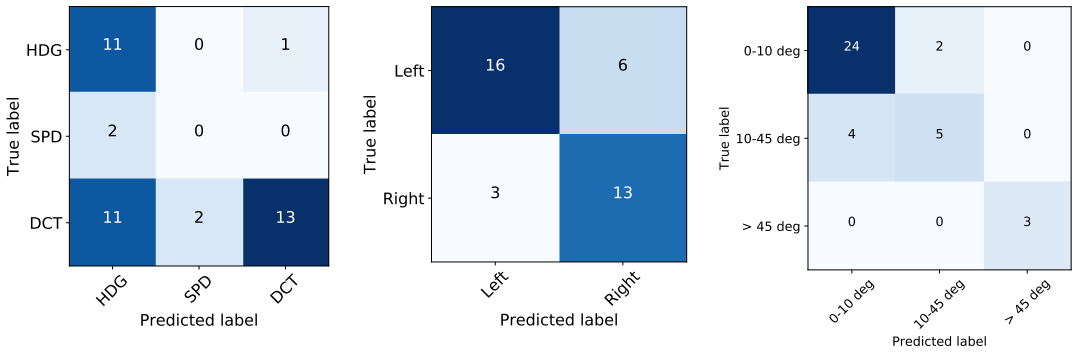
(a) Type (b) Direction (c) Value

Figure C.5: Participant 2: Confusion matrices of test results.



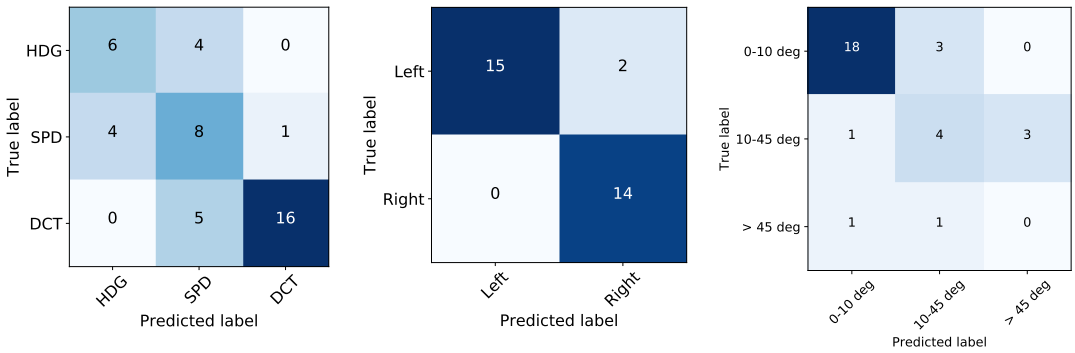
(a) Type (b) Direction (c) Value

Figure C.6: Participant 3: Confusion matrices of test results.



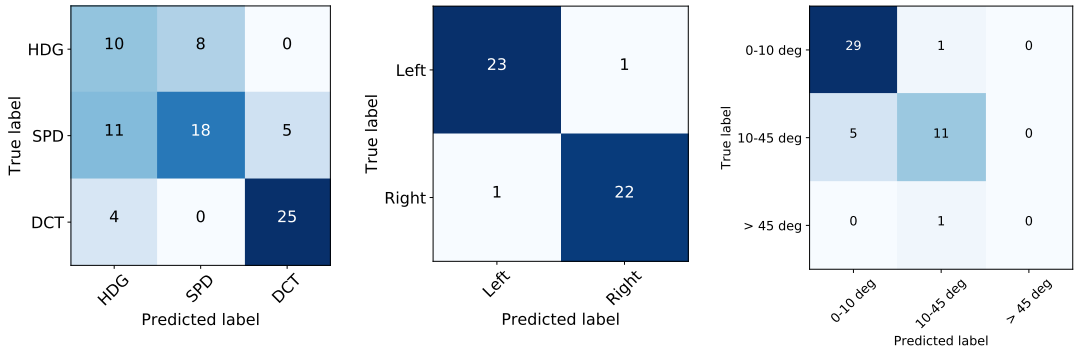
(a) Type (b) Direction (c) Value

Figure C.7: Participant 4: Confusion matrices of test results.



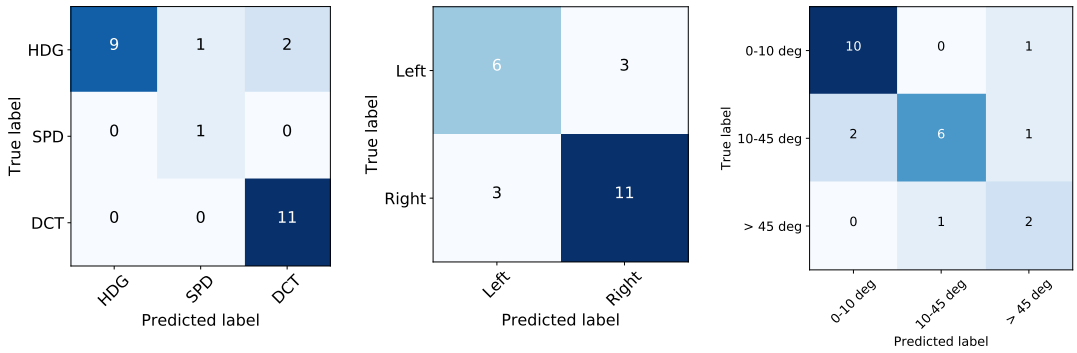
(a) Type (b) Direction (c) Value

Figure C.8: Participant 5: Confusion matrices of test results.



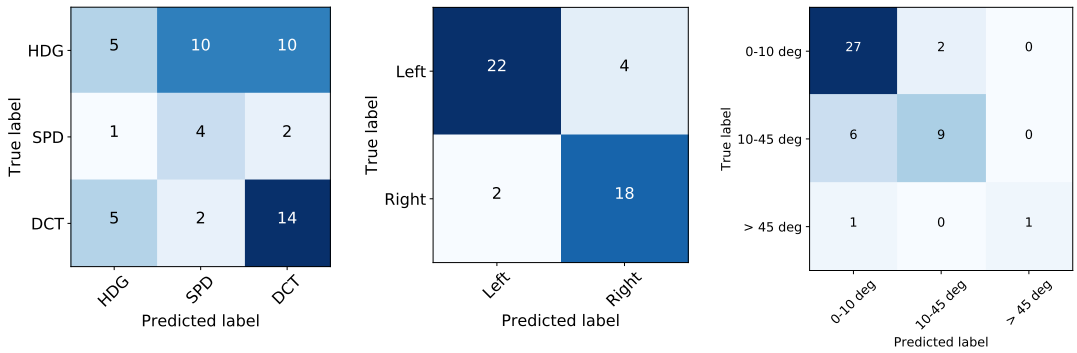
(a) Type (b) Direction (c) Value

Figure C.9: Participant 6: Confusion matrices of test results.



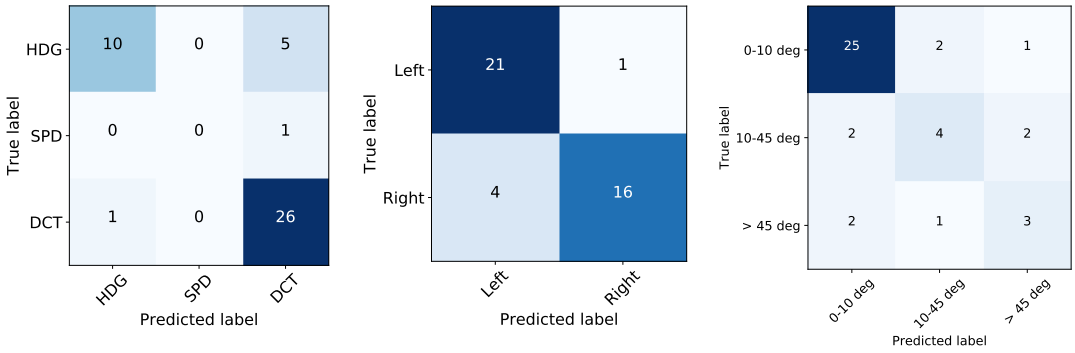
(a) Type (b) Direction (c) Value

Figure C.10: Participant 7: Confusion matrices of test results.



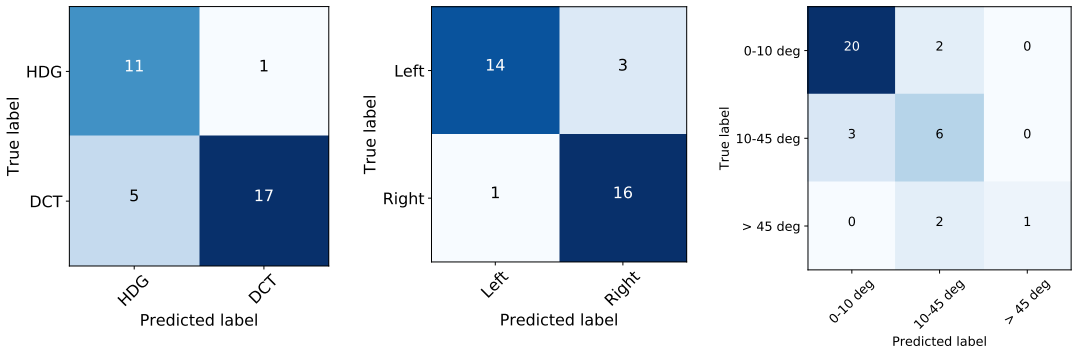
(a) Type (b) Direction (c) Value

Figure C.11: Participant 8: Confusion matrices of test results.



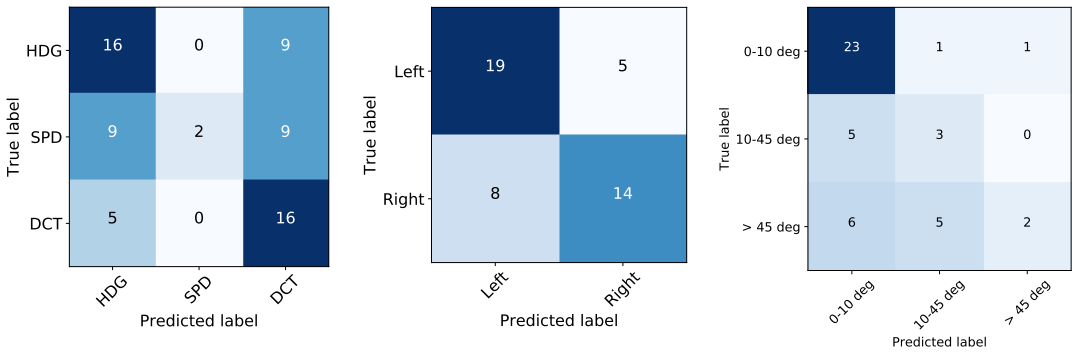
(a) Type (b) Direction (c) Value

Figure C.12: Participant 9: Confusion matrices of test results.



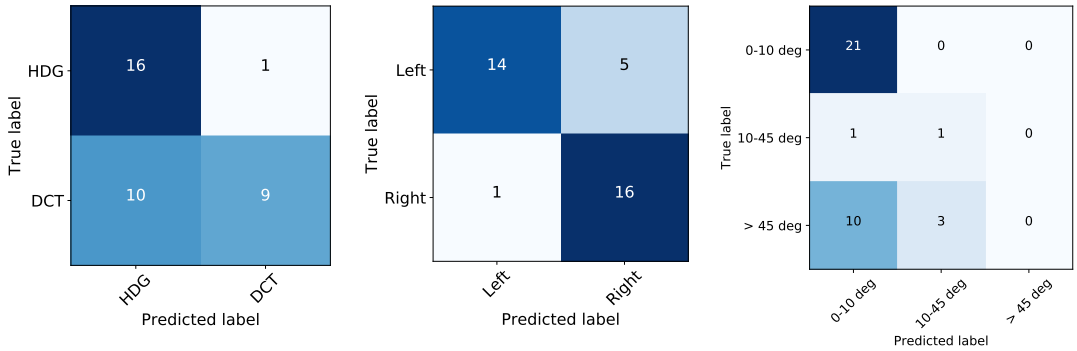
(a) Type (b) Direction (c) Value

Figure C.13: Participant 10: Confusion matrices of test results.



(a) Type (b) Direction (c) Value

Figure C.14: Participant 11: Confusion matrices of test results.



(a) Type

(b) Direction

(c) Value

Figure C.15: Participant 12: Confusion matrices of test results.

C.4. Cross-Validation Results

This section illustrates the cross-validation results per participant. Figure C.16 shows the MCC values obtained by validating a personalized model with all other participants' test datasets. The chart gives insight in how 'personal' the individual models are and which participants have comparable strategies. For instance, P5's directional strategy appears to be conformal to P3-P7's actions, indicated by the high MCC values for direction predictions with P5's model. Insight in similarity between ATCo strategies can be used to group participants' datasets to obtain larger training sets.

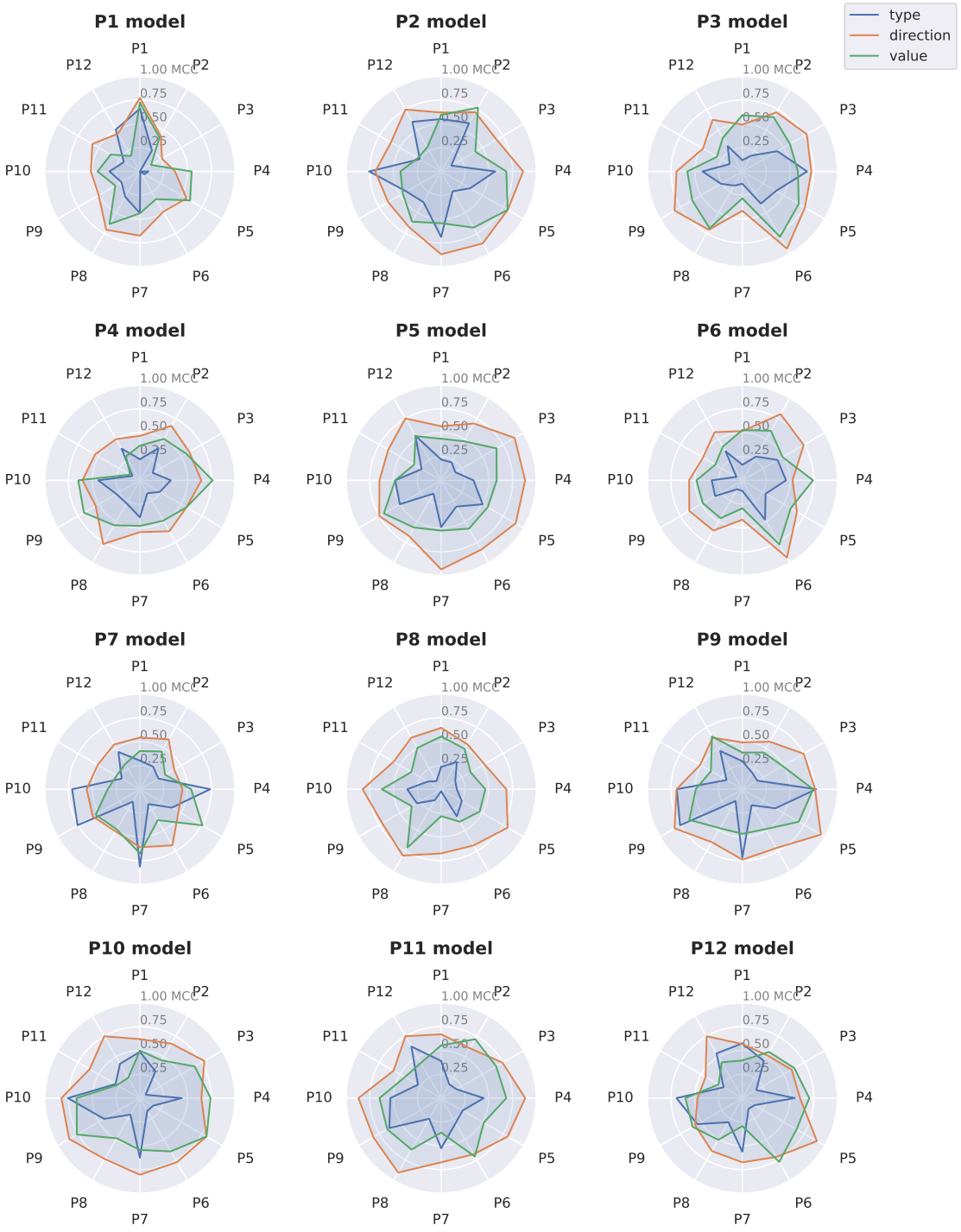


Figure C.16: Spider plot per participant showing model performance for all abstraction levels of cross-validation for each participant.

Figure C.17 combines all abstraction levels in one averaged indicator for an inter-participant comparison. Especially P1, P6 and P7 show clearly distinguishable peaks, indicating that these participants have the most differentiating strategies. Others participants, e.g. P11 and P12, appear to have a more generic strategy. However, P11 and P12's individual model performance is also below average, implying that they did not follow a consistent control strategy, which is confirmed by the consistency metrics. Similar spider plots of professional ATCos could point out their strategy homo- or heterogeneity as a group to determine whether strategic conformal automation is beneficial for trained professionals.

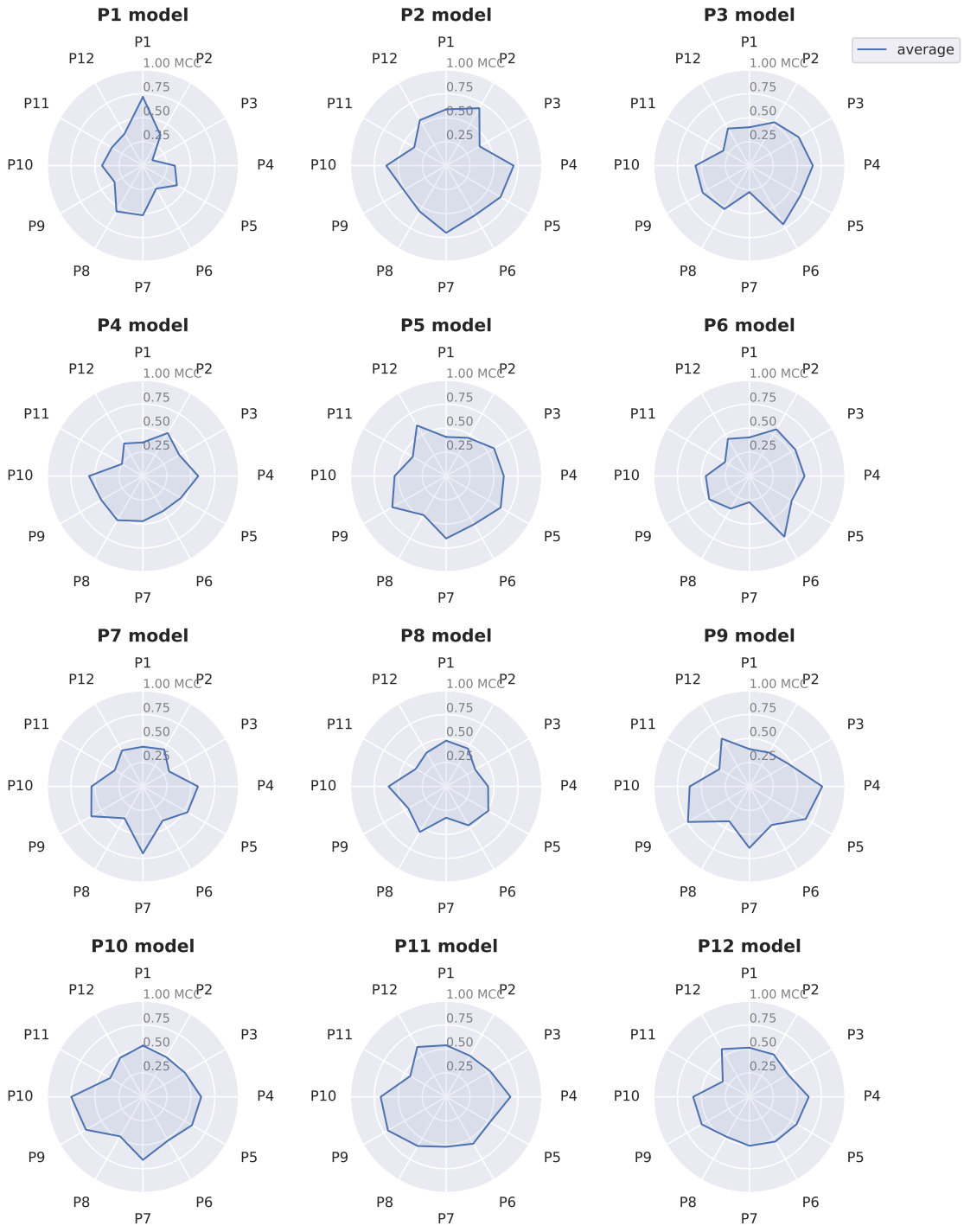


Figure C.17: Spider plot per participant showing the mean model performance of cross-validation for each participant.

IV

Conclusions and recommendations

Conclusions and recommendations

This research evaluates how strategic conformal automation for air traffic control can be achieved using convolutional neural networks. To accomplish this, a literature review, a preliminary analysis and a human-in-the-loop experiment are performed.

A **literature study** found that *Conflict Detection & Resolution* (CD&R) is a suited task given its importance and its ability to be simplified. Considering only horizontal plane conflicts limits the solution space while not necessarily lowering task difficulty. It was hypothesized that the parameters that influence CD&R decision-making are for the majority contained in an image of a solution space diagram (SSD). As a result, convolutional neural networks are applied because they are found to perform well using visual inputs. In addition, using visual inputs eliminates the need for hand-crafting features based on assumptions or prior knowledge.

The **preliminary analysis** confirmed this hypothesis by obtaining considerable prediction accuracy scores using artificially generated data. A conflict resolution algorithm (Modified Voltage Potential) was used to resolve approximately 6000 simulated conflicts in ATM Simulator BlueSky. Heading command *direction* and *value* were predicted for horizontal plane two-aircraft conflicts. It was found that heading direction prediction accuracies up to 90% are achievable using only SSD images as input. In addition, randomizing part of the dataset to simulate random human behavior proved the robustness of the convolutional neural networks. Besides, insight in network architectures, hyperparameters and preprocessing techniques was obtained which is subsequently used in the final phase.

In the **final phase**, the obtained knowledge is combined with a human-in-the-loop experiment to evaluate personalized prediction feasibility using human-generated data. Results confirm that the participants are sufficiently consistent to predict command *type*, *direction* and *value* with up to 88%, 96% and 85% accuracy respectively. Additionally, controller consistency and achieved model performance are positively correlated, confirming the hypothesis that consistent controllers are more suited for strategic conformal automation. Moreover, personalized models obtain significantly higher prediction accuracies than general models, indicating that controllers in this experiment exhibit differentiating strategies, i.e. are not homogeneous as a group. This is a critical assumption for strategic conformal automation.

Recommendations focus on three elements: learning methodology, experiment design and implementation. The appendices show population heterogeneity in the higher abstraction levels, such as resolution geometry and aircraft selection. More (visual) sector information could be added to the model input, incorporating these decisions in the model predictions. Additionally, as data quantity is limited in human-in-the-loop experiments, inverse reinforcement learning could provide a means to use the available data more effectively by learning a personalized reward function. Besides, the cross-validation results indicate that participants' data can be combined to obtain larger datasets.

Secondly, a future experiment can improve on population, scenarios and training. As achievable prediction accuracies are higher for more consistent controllers, using professional air traffic controllers could confirm this finding and provide insight in real-world applicability. Furthermore, as the simplified scenarios proved to be predictable, the next step is to create higher fidelity scenarios, without constraints on conflicts angles or altitude. Lastly, the appendices show that changes of strategy occurred during the experiment, which could be mitigated with more extensive training.

Finally, this research provides insight in the feasibility of strategic conformal automation for air traffic control using machine learning. A human population proved to be sufficiently homogeneous per controller and heterogeneous as a group for conformal automation to be beneficial. In addition, machine learning appears to be an effective method to predict human actions based on visual inputs. However, the acceptance feedback loop should be closed to assess if these individual-sensitive predictions indeed increase trust and acceptance and thus system-use. After all, machines are becoming more intelligent every day, but as the incredible cognitive models of humans prove difficult to match, the interaction between human and automation is more relevant than ever.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Isard, M. (2016). Tensorflow: a system for large-scale machine learning. In *Osdi* (Vol. 16, pp. 265–283).

Agogino, A. K., & Tumer, K. (2012). A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, *24*(1), 1–25. doi: 10.1007/s10458-010-9142-5

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A Brief Survey of Deep Reinforcement Learning. *IEEE Signal Processing Magazine*(Special Issue on Deep Learning for Image Understanding), 1–16. Retrieved from http://arxiv.org/abs/1708.05866{%}0Ahttp://dx.doi.org/10.1109/MSP.2017.2743240 doi: 10.1109/MSP.2017.2743240

Bainbridge, L. (1983). Ironies of automation. *Automatica*, *19*(6), 775–779. doi: 10.1016/0005-1098(83)90046-8

Bekier, M., Molesworth, B. R., & Williamson, A. (2012). Tipping point: The narrow path between automation acceptance and rejection in air traffic management. *Safety Science*, *50*(2), 259–265. doi: 10.1016/j.ssci.2011.08.059

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. In *34th international conference on machine learning*. Sydney, Australia: PMLR 70. Retrieved from http://arxiv.org/abs/1707.06887

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 679–684.

Bloem, M., & Bambos, N. (2015). Ground Delay Program Analytics with Behavioral Cloning and Inverse Reinforcement Learning. *Journal of Aerospace Information Systems*, *12*(3), 299–313. Retrieved from http://arc.aiaa.org/doi/10.2514/1.I010304 doi: 10.2514/1.I010304

Borst, C., Flach, J. M., & Ellerbroek, J. (2015). Beyond ecological interface design: Lessons from concerns and misconceptions. *IEEE Transactions on Human-Machine Systems*, *45*(2), 164–175. doi: 10.1109/THMS.2014.2364984

Bottou, L. (1998). *Online Learning and Stochastic Approximations* (revised, O ed.; D. Saad, Ed.). Cambridge, UK: Cambridge University Press.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168). ACM.

Chen, Z., & Huang, X. (2017). End-To-end learning for lane keeping of self-driving cars. *IEEE Intelligent Vehicles Symposium, Proceedings*, 1856–1860. doi: 10.1109/IVS.2017.7995975

Chollet, F. (2015). *Keras*. Retrieved from https://keras.io

CireşAn, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural networks*, *32*, 333–338.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games* (pp. 72–83). Springer.

Cruciol, L. L., Weigang, L., De Barros, A. G., & Koendjbiharie, M. W. (2014). Air holding problem solving with reinforcement learning to reduce airspace congestion. *Journal of Advanced Transportation*, *49*, 616–633. doi: 10.1002/atr.1293

De Boer, P. T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, *134*(1), 19–67. doi: 10.1007/s10479-005-5724-z

Dean, G., Fron, X., Miller, W., & Nicolaon, J. P. (1995). Arc 2000: An investigation into the feasibility of automatic conflict detection and resolution. *Air Traffic Control Quarterly*, *3*(4), 229–259.

D'Engelbronner, J. G. (2010). The Use of the Dynamic Solution Space to Assess Air Traffic Controller Workload. *AIAA Guidance, Naviagtion and Control Conference*(August), 1–21. Retrieved from http://arc.aiaa.org/doi/pdf/10.2514/6.2010-7542 doi: 10.2514/6.2010-7542

Duggan, G. B., Banbury, S., Howes, A., Patrick, J., & Waldron, S. M. (2004). Too much, too little or just right: Designing data fusion for situation awareness. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 48, pp. 528–532). SAGE Publications Sage CA: Los Angeles, CA.

Eby, M. S. (1994). A Self-Organizational Approach for Resolving Air Traffic Conflicts. *Lincoln Laboratory Journal*.

Endsley, M. R., & Kiris, E. O. (1995). The out-of-the-loop performance problem and level of control in automation. *Human factors*, *37*(2), 381–394.

Endsley, M. R., & Rodgers, M. D. (1994). Situation awareness information requirements for en route air traffic control (Tech. Rep. DOT/FAA/AM-94/27). *US Department of Transportation, Office of Aviation Medicine, Washington, DC*.

Endsley, M. R., & Rodgers, M. D. (1998). Distribution of attention, situation awareness and workload in a passive air traffic control task: Implications for operational errors and automation. *Air Traffic Control Quarterly*, *6*(1), 21–44.

Erzberger, H. (2004). *Transforming the NAS : The Next Generation Air Traffic Control System* (Tech. Rep. No. October). Ames Research Center: National Aeronautics and Space Administration (NASA). doi: NASA/TPâĂŞ2004-212828

EUROCONTROL. (1996). Model for Task and Job Descriptions of Air Traffic Controllers. *European Air Traffic Control Harmonisation and Integration Programme*.

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, *17*(7), 760–772.

Fothergill, S., & Neal, A. (2013). Conflict-resolution heuristics for en route air traffic management. *Proceedings of the Human Factors and Ergonomics Society*, 71–75. doi: 10.1177/1541931213571018

Galster, S. M., Duley, J. A., Masalonis, A. J., & Parasuraman, R. (2001). Air traffic controller performance and workload under mature free flight: Conflict detection and resolution of aircraft self-separation. *International Journal of Aviation Psychology*, *11*(1), 71–93. doi: 10.1207/S15327108IJAP1101_5

Ghazizadeh, M., Lee, J. D., & Boyle, L. N. (2012). Extending the Technology Acceptance Model to assess automation. *Cognition, Technology & Work*, *14*(1), 39–49.

Göritzlehner, R., Borst, C., Ellerbroek, J., Westin, C., Van Paassen, M., & Mulder, M. (2014). Effects of transparency on the acceptance of automated resolution advisories. In *Conference proceedings - ieee international conference on systems, man and cybernetics* (Vol. 2014-Janua, pp. 2965–2970). doi: 10.1109/smc.2014.6974381

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., … Chen, T. (2017). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377. Retrieved from `https://doi.org/10.1016/j.patcog.2017.10.013` doi: 10.1016/j.patcog.2017.10.013

Gu, S., Lillicrap, T., Sutskever, I., & Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *33rd international conference on machine learning* (pp. 2829–2838). New York, USA.

Guo, X., Singh, S. P., Lee, H., Lewis, R. L., & Wang, X. (2014). Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Advances in neural information processing systems* (pp. 3338–3346).

Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., … LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, *26*(2), 120–144.

Hermes, P., Mulder, M., Van Paassen, M., & Huisman, H. (2009). Solution-Space-Based Analysis of the Difficulty of Aircraft Merging Tasks. *Journal of Aircraft*, *46*(6), 1995–2015. doi: 10.2514/1.42886

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., … Gruslys, A. (2017). Deep Q-learning from Demonstrations. *Association for the Advancement of Artificial Intelligence (AAAI)*. Retrieved from `https://arxiv.org/pdf/1704.03732.pdf`

Hilburn, B. (2002). Evaluating Human Interaction with Advanced Air Traffic Management Automation. *RTO Meeting Proceedings*(MP-088), 1–12.

Hilburn, B., Jorna, P. G., Byrne, E. A., & Parasuraman, R. (1997). The effect of adaptive air traffic control (ATC) decision aiding on controller mental workload. *Human-automation interaction: Research and practice*, 84–91.

Hilburn, B., Westin, C., & Borst, C. (2014). Will Controllers Accept a Machine That Thinks Like They Think? The Role of Strategic Conformance in Decision Aiding Automation. *Air Traffic Control Quarterly*, *22*(2), 115–136.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks. IEEE Press*. doi: 10.1109/9780470544037.ch14

Hoekstra, J. M., & Ellerbroek, J. (2016). BlueSky ATC Simulator Project: an Open Data and Open Source Approach. *7th International Conference on Research in Air Transportation*, 1–8.

Hoekstra, J. M., Van Gent, R. N. H. W., & Ruigrok, R. C. J. (2002). Designing for safety: the free flight air traffic management concept. *Reliability Engineering & System Safety*, *75*(2), 215–232. doi: http://dx.doi.org/10.1016/S0951-8320(01)00096-5

ICAO. (2016). *Air Traffic Management: Procedures for Air Navigation Services* (16th editi ed., Vol. Doc 4444).

Jacobs, J. F. (2015). *Selection and development of innovative design alternatives: Ethical, social and uncertainty issues.* TU Delft, Delft University of Technology.

Jenie, Y. I., van Kampen, E.-J., de Visser, C. C., Ellerbroek, J., & Hoekstra, J. M. (2015). Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, *38*(6), 1140–1146.

Joint Planning and Development Office (JPDO), & Next Generation Air Transportation System (NextGen). (2011). *Concept of operations for the next generation air transportation system* (Tech. Rep.).

Kidwell, B., Calhoun, G. L., Ruff, H. A., & Parasuraman, R. (2012). Adaptable and adaptive automation for supervisory control of multiple autonomous vehicles. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 56, pp. 428–432). SAGE Publications Sage CA: Los Angeles, CA.

Kimball, K. A. (1970, feb). Estimation of Intersection of Two Converging Targets as a Function of Speed and Angle of Target Movement. *Perceptual and Motor Skills*, *30*(1), 303–310. Retrieved from `https://doi.org/10.2466/pms.1970.30.1.303` doi: 10.2466/pms.1970.30.1.303

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Kirwan, B., & Flynn, M. (2002). *Investigating air traffic controller conflict resolution strategies* (Vol. 1; Tech. Rep.). Brussels, Belgium: EUROCONTROL.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, *31*, 249–268. doi: 10.1115/1.1559160

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25* (pp. 1097–1105).

Law, D. J., Pellegrino, J. W., Mitchell, S. R., Fischer, S. C., McDonald, T. P., & Hunt, E. B. (1993). Perceptual and cognitive factors governing performance in comparative arrival-time judgments. *Journal of Experimental Psychology: Human Perception and Performance*, *19*(6), 1183.

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi: 10.1038/nature14539

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Muller, U., Ben, J., Cosatto, E., & Flepp, B. (2005). Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Scholkopf, & J. Platt (Eds.), *Advances in neural information processing systems (nips 2015)* (Vol. 18).

LeCun, Y., Pfeifer, R., Schreter, Z., Fogelman, F., & Steels, L. (1989). *Generalization and network design strategies* (Tech. Rep.). Zurich, Switserland: Elsevier.

Lin, L.-J. (1993). *Reinforcement learning for robots using neural networks* (Tech. Rep.).

Majumdar, A., Ochieng, W. Y., McAuley, G., Michel Lenzi, J., & Lepadatu, C. (2004). The Factors Affecting Airspace Capacity in Europe: A Cross-Sectional Time-Series Analysis Using Simulated Controller Workload Data. *Journal of Navigation*, *57*(3), 385–405. Retrieved from `https://www.cambridge.org/core/article/factors-affecting-airspace-capacity-in-europe-a-crosssectional-timeseries-analysis-using-simulated-controller-workload-data/C7AB3EEB35A22F8B1FFB2B27F790FC26` doi: DOI:10.1017/S0373463304002863

Mark, G., & Kobsa, A. (2005). The effects of collaboration and system transparency on CIVE usage: an empirical study and model. *Presence: Teleoperators & Virtual Environments*, *14*(1), 60–80.

Mercado-Velasco, G., Mulder, M., & Van Paassen, M. (2010). Analysis of Air Traffic Controller Workload Reduction Based on the Solution Space for the Merging Task. *AIAA Guidance, Navigation, and Control Conference*, *AIAA 2010*-(August), 1–18. doi: 10.2514/6.2010-7541

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *33rd international conference on machine learning* (Vol. 48). New York, USA. doi: 10.1177/0956797613514093

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing

atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529. Retrieved from http://dx.doi.org/10.1038/nature14236 doi: 10.1038/nature14236

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *International conference on machine learning* (pp. 663–670).

Nguyen-Duc, M., Briot, J.-P., & Drogoul, A. (2003). An application of multi-agent coordination techniques in air traffic management. In *Intelligent agent technology, 2003. iat 2003. ieee/wic international conference on* (pp. 622–625). IEEE.

Pineau, J. (2018). Automation of Air Traffic Management Using Fuzzy Logic Algorithm to Integrate Unmanned Aerial Systems into the National Airspace. *International Journal of Electrical and Computer Engineering (IJECE)*, *8*(5).

Prevot, T., Homola, J. R., Martin, L. H., Mercer, J. S., & Cabrall, C. D. (2012). Toward automated air traffic control: investigating a fundamental paradigm shift in human/systems interaction. *International Journal of Human-Computer Interaction*, *28*(2), 77–98.

Rahman, S. M. A., Borst, C., Mulder, M., & Van Paassen, M. (2010). Measuring Sector Complexity : Solution Space-Based Method. *Advances in Air Navigation Services*, *8*, 11–34. doi: 10.5772/2574

Rantanen, E. M., McCarley, J. S., & Xu, X. (2004). Time Delays in Air Traffic Control Communication Loop: Effect on Controller Performance and Workload. *The International Journal of Aviation Psychology*, *14*(4), 369–394.

Rantanen, E. M., & Nunes, A. (2005). Hierarchical Conflict Detection in Air Traffic Control. *The International Journal of Aviation Psychology*, *15*(4), 339–362. doi: 10.1207/s15327108ijap1504

Regtuit, R. M., Borst, C., Van Kampen, E.-J., & van Paassen, M. R. M. (2018). Building Strategic Conformal Automation for Air Traffic Control Using Machine Learning. In *Aiaa scitech forum*. Kissimmee, Florida: AIAA Information Systems.

Remington, R. W., Johnston, J. C., Ruthruff, E., Gold, M., & Romera, M. (2000). Visual search in complex displays: factors affecting conflict detection by air traffic controllers. *Human factors*, *42*(3), 349–366. doi: 10.1518/001872000779698105

Rogers, E. M. (1983). *Diffusion of innovations* (3rd ed. ed.). New York: Free Press.

Rosenblatt, F. (1958). *Two theorems of statistical separability in the perceptron*. United States Department of Commerce.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0

Sáez Nieto, F. J. (2016). The long journey toward a higher level of automation in ATM as safety critical, sociotechnical and multi-Agent system. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *230*(9), 1533–1547. doi: 10.1177/0954410015596763

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized Experience Replay. In *International conference on learning representations* (pp. 1–21). San Juan, Puerto Rico. Retrieved from http://arxiv.org/abs/1511.05952 doi: 10.1038/nature14236

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897).

Seamster, T. L., Redding, R. E., Cannon, J. R., Ryder, J. M., & Purcell, J. A. (1993). Cognitive task analysis of expertise in air traffic control. *The international journal of aviation psychology*, *3*(4), 257–283.

SESAR Consortium. (2007). The Concept of Operations at a glance. *Single European Sky*.

Sheridan, T. B., & Verplank, W. L. (1978). *Human and computer control of undersea teleoperators* (Tech. Rep.).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. Retrieved from http://dx.doi.org/10.1038/nature16961 doi: 10.1038/nature16961

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *31st international conference on machine learning*. Beijing, China.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. Retrieved from http://dx.doi.org/10.1038/nature24270 doi: 10.1038/nature24270

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for Simplicity: The All

Convolutional Net. In *International conference on learning representations*. Freiburg, Germany. Retrieved from http://arxiv.org/abs/1412.6806 doi: 10.1163/_q3_SIM_00374

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. doi: 10.1214/12-AOS1000

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1) (No. 1). MIT press Cambridge.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA: MIT Press.

Sutton, R. S., Barto, A. G., & Williams, R. J. (1992). Reinforcement Learning is Direct Adaptive Optimal Control. *IEEE Control Systems*, *12*(2), 19–22.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (pp. 1057–1063).

Tesauro, G. (1995). Td-gammon: A self-teaching backgammon program. In *Applications of neural networks* (pp. 267–285). Springer.

Tieleman, T., & Hinton, G. E. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.

Tomlin, C., Pappas, G. J., & Sastry, S. (1998). Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Trans. Autom. Control*, *vol*(4), 42no4pp509—-521. doi: 10.1109/9.664154

Van Dam, S. B. J., Abeloos, A. L., Mulder, M., & Van Paassen, M. (2004). Functional presentation of travel opportunities in flexible use airspace: An EID of an airborne conflict support tool. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, *1*(January), 802–808. doi: 10.1109/ICSMC.2004.1398401

Van Dam, S. B. J., Mulder, M., & van Paassen, M. (2008). Ecological Interface Design of a Tactical Airborne Separation Assistance Tool. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *38*(6), 1221–1233. doi: 10.1109/TSMCA.2008.2001069

Van Gent, R. N. H. W., Hoekstra, J. M., & Ruigrok, R. C. J. (1997). Free flight with airborne separation assurance. In *Confederation of european aerospace societies (ceas)*. Amsterdam, The Netherlands: 10th European Aerospace Conference.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. In *30th aaai conference on artificial intelligence* (pp. 2094–2100). Phoenix, AZ.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly*, 425–478.

Vicente, K. J. (2002). Ecological interface design: Progress and challenges. *Human factors*, *44*(1), 62–78.

Vicente, K. J., & Rasmussen, J. (1992). Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man and Cybernetics*, *22*(4), 589–606. doi: 10.1109/21.156574

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. In *International conference on learning representations*. San Juan, Puerto Rico. Retrieved from http://arxiv.org/abs/1511.06581 doi: 10.1109/MCOM.2016.7378425

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Cambridge, UK: King's College.

Weigang, L., de Souza, B. B., Crespo, A. M. F., & Alves, D. P. (2008). Decision support system in tactical air traffic flow management for air traffic flow controllers. *Journal of Air Transport Management*, *14*(6), 329–336. doi: 10.1016/j.jairtraman.2008.08.007

Werbos, P. J. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. thesis, Harvard University, Cambridge, MA, 1974.

Westin, C. (2017). *Strategic Conformance: Exploring Acceptance of Individual-Sensitive Automation for Air Traffic Control*. PhD Thesis. Delft University of Technology, Netherlands. doi: 10.4233/uuid

Westin, C., Borst, C., & Hilburn, B. (2015). An empirical investigation into three underlying factors affecting automation acceptance. *Proceedings of the Fifth SESAR Innovation Days*, 1–9.

Westin, C., Borst, C., & Hilburn, B. (2016). Strategic Conformance: Overcoming Acceptance Issues of Decision Aiding Automation? *IEEE Transactions on Human-Machine Systems*, *46*(1), 41–52. doi: 10.1109/THMS.2015.2482480

Westin, C., Hilburn, B., & Borst, C. (2011). Mismatches between Automation and Human Strategies: An Investigation into Future Air Traffic Management (ATM) Decision Aiding. *Proceedings of the first SESAR*

Innovation days, *12*, 1–6.

Westin, C., Hilburn, B., & Borst, C. (2015). Air Traffic Controller Decision-Making Consistency and Consensus in Conflict Solution Performance. *5th CEAS Air and Space Conference*(110), 1–14.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229–256.

Wulfmeier, M., Ondruska, P., & Posner, I. (2015). Maximum Entropy Deep Inverse Reinforcement Learning. Retrieved from http://arxiv.org/abs/1507.04888

Wulfmeier, M., Rao, D., Wang, D. Z., Ondruska, P., & Posner, I. (2017). Large-scale cost function learning for path planning using deep inverse reinforcement learning. *International Journal of Robotics Research*, *36*(10), 1073–1087. doi: 10.1177/0278364917722396

Xu, H., Gao, Y., Yu, F., & Darrell, T. (2016). End-to-end Learning of Driving Models from Large-scale Video Datasets. , 2174–2182. Retrieved from http://arxiv.org/abs/1612.01079 doi: 10.1109/CVPR.2017.376

Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. *AAAI Conference on Artificial Intelligence*, *8*, 1433–1438.