Are Relations Relevant in CNNs?

A Study Based on a Facial Dataset

Lisa Andrea Lengenfelder

A thesis presented for the degree
Bachelor of Science in Applied Computer Science

Faculty Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Reviewer: Prof. Dr. Ute Schmid
Matriculation number: 1948935

Abstract

As artificial intelligence plays an increasingly important role in people's lives beyond shopping or movie suggestions, it is important to understand how these systems work. One step closer to understanding how machine learning works is to take a closer look at one part of it, in this case Convolutional Neural Networks (CNN). This thesis focuses on whether relations are relevant for a traditional CNN and a Fully Convolutional Neural Network (FCN). The basis is a dataset constructed from the FASSEG dataset with artificial faces, where the features have been arranged either correctly or incorrectly. In summary the FCNs performed better than the CNNs and there is strong evidence that spatial relations are relevant for Convolutional Neural Networks in general.

Keywords: Machine Learning, Deep Learning, Convolutional Neural Network, Fully Convolutional Network, Explainable Artificial Intelligence, Spacial Relations

Contents

1 Introduction

Artificial intelligence (AI), and therefore machine learning (ML) and deep learning are gaining greater importance in our everyday lives. As these systems become more and more involved in procedures and decisions that affect human lives more than Netflix movie suggestions, it is important to understand how these mechanisms work and make their decisions. The following case is from a study from the 1990s, but it still shows the importance of Explainable Artificial Intelligence, for example in medical cases. At that time, artificial neural networks (ANNs) were used to predict the mortality risk of pneumonia patients. This prediction was then used to decide whether a patient should be treated in hospital (high risk) or as an outpatient (low risk). Although the ANNs gave more accurate results than the linear regression, which was also tested, they were considered too risky. The networks had learned that pneumonia patients with asthma had a lower risk of death than the rest of the population and that they should be treated as outpatients. However, this is medically counterintuitive, as these patients tend to have a higher risk. It turned out, that this occurred because in the training data, asthma patients with pneumonia did not have to go to hospital per se, but instead directly to the intensive care unit. Here, they were treated so well that their mortality risk was lower compared to the rest of the population. Hence the neural networks' wrong conclusion. [3]

Explainable Artificial Intelligence (XAI) attempts to demystify the black-box-models that are at the heart of current deep learning architectures [5], so that errors like the one described can be prevented. This thesis focuses on Convolutional Neural Networks (CNNs) as part of XAI and tries to illustrate whether relations are relevant for the learning process.

CNNs are a specialised kind of Artificial Neural Network (ANN) for processing data such as images [5]. The CNNs used in this experiment are Deep Neural Networks (DNNs), as they consist of several layers [5]. Besides the more common CNN which uses a dense layer to classify the images, also Fully Convolutional Neural Networks (FCN), which are a subtype of CNNs, are trained to tackle the research question. In contrast to the CNN, the FCN's classifying layers are also convolutional layers [9]. A more detailed description will be given in the following chapter.

Both kinds of CNNs are trained using images of automatically constructed faces, where the facial features are either in the correct position or switched up. They are then tested on similar and slightly altered data. This is done to understand if the spatial relations between the features are important for the CNNs to distinguish if the image contains a face.

2 Theoretical Background

This section describes conceptual basics and related work. Firstly, an explanation of what a Deep Neural Network is, followed by a differentiation between a common CNN and an FCN. Additionally, the concept of Explainable AI (XAI) is described. Finally, it will be shown how this thesis fits in with previous academic works.

2.1 Deep Neural Networks

An Artificial Neural Network (ANN) is a network of several neurons arranged in layers. There are three types of layers in an ANN: input, hidden and output layers. The input layer consists of a fixed number of neurons that correlate with the input data. For example, assuming one has an RGB image with a size of 50x50 pixels, then the input layer would consist of one neuron for each colour channel in every pixel. This results in 50x50x3 (7500) neurons. These neurons are per definition just input. This layer is followed by few (ANN) or many (Deep Neural Network) hidden layers. The result of each layer is the input for the next layer. A neuron in these following layers receives an input (x) that is being weighted (w) and added to a bias (b). The input is given, the weights and biases are initialised randomly. The result of this calculation is called netinput (n). [8]

$$w \cdot x + b = n \tag{1}$$

Using this netinput each neuron generates output by feeding it into the activation function of the neuron [8].

$$f_{activation}(n) = output \tag{2}$$

The number of neurons in hidden layers varies. The result of the last hidden layer is then used as the input for the last layer in an ANN: the output layer. This layer usually consists of as many neurons as there are categories to be classified. For example, if you want to divide images into “elephant”, “mouse” and “cat”, you have three output neurons. The result computed by this layer is called actual output. The aim is to approximate the actual output as closely as possible to the given desired output. [8]

In order to update the originally randomly set weights and biases, the so-called back-propagation is executed. The algorithm starts from the output layer and works its way to the input layer, calculating the error gradient after each layer. Such a forward and backward run of all training examples is call an “epoch”. If the gradient becomes so small that the training does not converge or only converges very slowly, it is called a vanishing gradient. This can be prevented by some activation functions. [16]

An activation function applies to all neurons in a layer. The following detailed descriptions are limited to the two activation functions that are used in this thesis: the ReLU and the softmax function.

The ReLU function is used in the hidden layers and is a non-linear activation function. ReLU is short for rectified linear unit [16]. With the ReLU function the vanishing gradient problem can be avoided and the learning speed of deep neural networks improved [6]. In [16] it is defined as

$$f_{ReLU}(n) = max(0, n) \tag{3}$$

The softmax function is used as a classifier. In [16] it is defined as

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N}}(i = 1, 2, ..., N) \tag{4}$$

The softmax function has the fortunate effect that the larger a value is, the smaller all other values are. This means that a single output is favoured with a very high probability [18].

These explanations are limited to supervised feedforward networks, as they are the only ones trained in this thesis. Feedforward means that the input is only ever forwarded to the next neuron, which means that there are no loops. [8] Supervised means that the input and desired output are known.

The two Deep Neural Networks, that are the subject of this paper, are the more common Convolutional Neural Network (CNN) and a slightly modified subtype of the CNN, the Fully Convolutional Neural Network (FCN). Both are subsequently described.

2.1.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a specialised kind of deep feedforward Artificial Neural Network (ANN) for processing data such as images [3].

The weight and bias values are the same for all hidden layers in a CNN, unlike in other DNNs [17].

As the name suggests, a major component of CNNs is convolution. The most important hidden layers in CNNs include convolutional layers with an activation function and pooling layers (see Figure 1). In convolutional layers, input images pass through a series of convolutional filters, each of which activates specific features from those images. Pooling simplifies the output by reducing the number of parameters to be learned by the network by means of non-linear down sampling. The most common activation function here is ReLU. These layers are repeated several times, each layer learning to identify different features. [17]

The feature learning block is followed by the classification block (see Figure 1). The last hidden layer, i.e. the penultimate layer, is a fully connected or dense layer that outputs a vector of K dimensions, where K represents the number of classes that can be predicted by the network. This vector contains the class probabilities for each image to be classified. [17]

The final layer of the CNN architecture uses a classification function such as softmax [17].

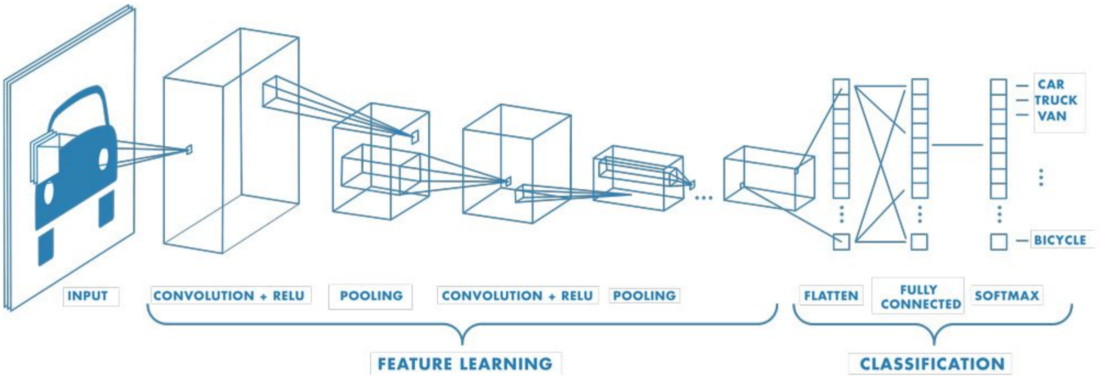


Figure 1: Example of a CNN architecture [17]

Since a convolutional layer consists of several feature maps, it is shown here as a cuboid [16].

2.1.2 Fully Convolutional Neural Network

Fully Convolutional Neural Networks (FCNs) are a subtype of Convolutional Neural Networks (CNNs). FCNs differ from the more common CNNs by "replacing the fully connected layers with more convolutional layers" [9]. In contrast to CNNs, where "[t]he fully connected layers [. . .] have fixed dimensions and throw away spatial coordinates" [14], FCNs keep spatial information.

As already mentioned, in FCNs the dense or fully connected layers are replaced with 1x1 convolution layers. Another crucial component is the GlobalMaxPooling layer, or alternatively a GlobalAveragePooling layer. This ensures that the data is transformed from the actual height and width to a single pixel, "essentially taking [the] max or average of the values along height and width dimensions for every filter" [13]. This is then used as the input for the output layer, which is the softmax activation layer. The number of neurons in this layer usually corresponds to the number of classes. [13]

Although both DNNs are Convolutional Neural Networks, the fully connected CNNs will be referred to as "CNNs" and the Fully Convolutional Neural Networks as "FCNs" for better readability. When referring to both, "ConvNets" will be used.

2.2 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) “is a research field that aims to make AI systems results more understandable to humans” [1].

XAI has many application domains, such as transportation, healthcare, legal, finance and military [1].

DARPA defines the goal of XAI as “to create a suite of new or modified machine learning techniques that produce explainable models that, when combined with effective explanation techniques, enable end users to understand, appropriately trust, and effectively manage the emerging generation of Artificial Intelligence (AI) systems” [4].

One term often used to describe the incapability of Machine Learning (ML) and Artificial Intelligence systems to explain how a result was obtained, is called “the black-box-problem” [1]. XAI aims to overcome this problem.

Especially when talking about ML, the term “interpretable” is favoured over “explainable” to describe a “system where a user cannot only see but also study and understand how inputs are mathematically mapped to outputs” [1].

According to [1] the need for XAI stems from (at least) four reasons:

1. Explain to Justify (“to ensure that AI based decisions were not made erroneously”)
2. Explain to Control (“to prevent things from going wrong”)
3. Explain to Improve (“A model that can be explained and understood is one that can be more easily improved.”)
4. Explain To Discover (“Asking for explanations is a helpful tool to learn new facts, to gather information and thus to gain knowledge.”)

This thesis aims to explore the fourth point - explain to discover - to understand if and how relations are relevant in CNNs.

2.3 Related Work

There has been a lot of research in the field of XAI. This work sees itself as a logical extension of [10, 11, 12]. As “relations between constituents are crucial for a concept" [12] and use cases exist where "combinations and relations between parts" [12] need to be taken into account, further research as to whether relations are relevant in CNNs needs to be conducted.

This thesis should be considered a starting point for the analysis of this research question.

3 Approach

The images used to train and test the CNNs, were generated from images of the FASSEG dataset [7]. In the following this dataset will be referred to as "Picasso Dataset", as dubbed by Rabold et al. [11].

The facial features, such as eyes, mouth and nose, were programmatically extracted from the original images using the masks provided by the FASSEG dataset [7]. The images of the faces were altered by hand so that the features were removed and could serve as a blank "skin" canvas [11].

Then the features were reapplied to the images, either in the correct constellation or mixed up. The images with the correct constellation, irrespective of whether it was originally a left or right eye, constitute the positive ("face") class, the other images the negative ("no_face") class. Examples are shown in Figure 2. [11]

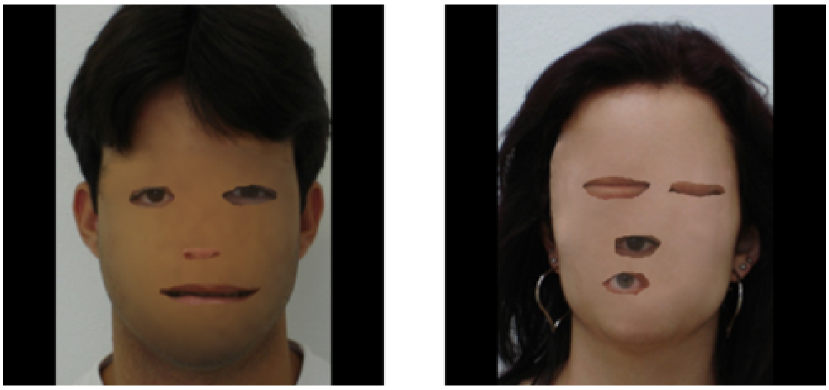


Figure 2: Picasso Dataset Basis Examples (left: "face", right: "no_face")

To estimate whether the performance varies if the features are positioned differently, three datasets were generated. Figure 2 shows examples of the most important dataset, the “baseline” dataset, where the cut-out facial features are placed in the position of the original features. It serves as a baseline for the overall performance of the ConvNets.

The second dataset contains faces, where the features were moved closer together. The original positions of the eyes were each shifted down 16 pixels and 8 pixels further to the middle. The mouth’s original position was moved up 12 pixels. The nose kept its position.

The third dataset consists of faces with features further apart than in the original. This time the eyes’ original positions were shifted up 20 pixels. The left eye was moved to the left by 10 pixels, the right eye to the right by only 5 pixels. This was because in many images, when the right eye was replaced by a mouth in the “no_face” class, parts of the mouth were already in the hair section of the image. This decision was made with the intention to make the images look as face-like as possible as it was believed that this would otherwise impact the validity of this dataset. Shifting the right eye slot even more to the right, would have aggravated the problem. The mouth was moved down 20 pixels and the nose again kept its place. Examples for the second and third datasets are shown in Figure 3.

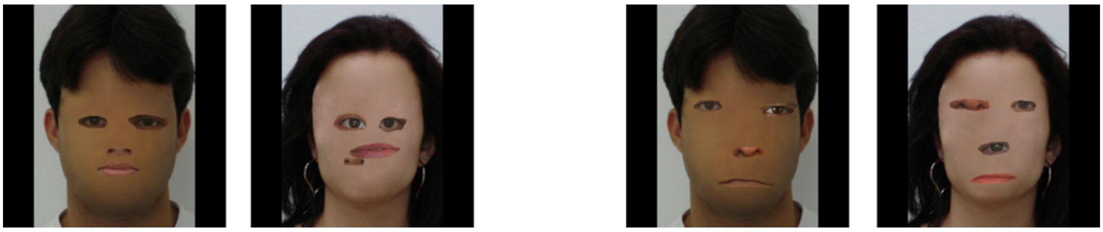
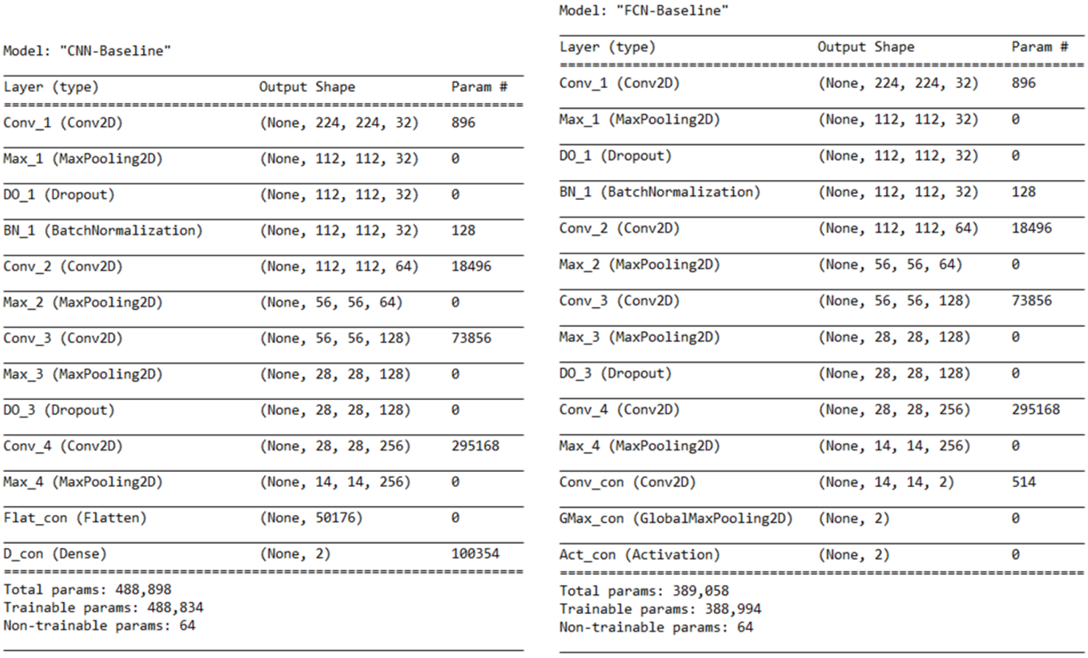


Figure 3: Examples for features closer together (left) and features further apart (right) datasets

Each dataset consists of 20000 images. The training data contains 16002 images, the validation data 998 images and the testing data 3000 images. Every subset has exactly 50% positive and 50% negative samples. Although the ConvNets have no VGG16 architecture they seemed to perform better when the images were pre-processed as if used in a VGG16 architecture.

With these images the training of all ConvNets was performed in two ways. Firstly, the CNNs were trained with 13 (Figure 4a) and the FCNs with 14 layers

(Figure 4b), both in 30 epochs. Additionally, a second set of ConvNets was trained in 20 epochs with 15 (CNN) or 16 layers (FCN). These ConvNets contain an additional convolutional and maxpooling layer.



```
Model: "CNN-Baseline"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv_1 (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
Max_1 (MaxPooling2D)         (None, 112, 112, 32)      0
_________________________________________________________________
DO_1 (Dropout)               (None, 112, 112, 32)      0
_________________________________________________________________
BN_1 (BatchNormalization)    (None, 112, 112, 32)      128
_________________________________________________________________
Conv_2 (Conv2D)              (None, 112, 112, 64)      18496
_________________________________________________________________
Max_2 (MaxPooling2D)         (None, 56, 56, 64)        0
_________________________________________________________________
Conv_3 (Conv2D)              (None, 56, 56, 128)       73856
_________________________________________________________________
Max_3 (MaxPooling2D)         (None, 28, 28, 128)       0
_________________________________________________________________
DO_3 (Dropout)               (None, 28, 28, 128)       0
_________________________________________________________________
Conv_4 (Conv2D)              (None, 28, 28, 256)       295168
_________________________________________________________________
Max_4 (MaxPooling2D)         (None, 14, 14, 256)       0
_________________________________________________________________
Flat_con (Flatten)           (None, 50176)             0
_________________________________________________________________
D_con (Dense)                (None, 2)                 100354
=================================================================
Total params: 488,898
Trainable params: 488,834
Non-trainable params: 64
_________________________________________________________________
```

(a) Example of a CNN model with 13 layers

```
Model: "FCN-Baseline"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv_1 (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
Max_1 (MaxPooling2D)         (None, 112, 112, 32)      0
_________________________________________________________________
DO_1 (Dropout)               (None, 112, 112, 32)      0
_________________________________________________________________
BN_1 (BatchNormalization)    (None, 112, 112, 32)      128
_________________________________________________________________
Conv_2 (Conv2D)              (None, 112, 112, 64)      18496
_________________________________________________________________
Max_2 (MaxPooling2D)         (None, 56, 56, 64)        0
_________________________________________________________________
Conv_3 (Conv2D)              (None, 56, 56, 128)       73856
_________________________________________________________________
Max_3 (MaxPooling2D)         (None, 28, 28, 128)       0
_________________________________________________________________
DO_3 (Dropout)               (None, 28, 28, 128)       0
_________________________________________________________________
Conv_4 (Conv2D)              (None, 28, 28, 256)       295168
_________________________________________________________________
Max_4 (MaxPooling2D)         (None, 14, 14, 256)       0
_________________________________________________________________
Conv_con (Conv2D)            (None, 14, 14, 2)         514
_________________________________________________________________
GMax_con (GlobalMaxPooling2D) (None, 2)                0
_________________________________________________________________
Act_con (Activation)         (None, 2)                 0
=================================================================
Total params: 389,058
Trainable params: 388,994
Non-trainable params: 64
_________________________________________________________________
```

(b) Example of an FCN model with 14 layers

Figure 4: Examples of used ConvNet models

As one can see in Figure 4 the underlying structure is the same except for the connecting layers. There are two dropout layers in every model to prevent overfitting [15], the dropout rate was set to 0.2. There is a batch normalisation layer included to normalize the data to a more fitting value range of 0 to 1 [6]. The convolutional and maxpooling layers as well as the respective connected layers are explained in chapter 2.1. The flatten layer prepares the input for the dense layer. To update weights and biases the optimiser Adam is used as an alternative to the classical stochastic gradient descent [2]. The learning rate was set to 0.0001. The ReLU function was used as the activation function for all convolutional layers. Softmax was used as the activation function for the classification layers. Each variant was trained and tested 3 times to rule out outlier results. The arithmetic mean was calculated for every variant's results.

The following sections will provide a detailed description of how each ConvNet was trained and tested and the results that were achieved.

3.1 Training and Testing Convolutional Neural Networks

3.1.1 Baseline Dataset

The CNNs in this section were trained on and then tested with the baseline dataset (see Figure 2). This means that the features in these images were in their original position or mixed up. This training process set the baseline for all following training runs in this paper.

As can be seen in Figure 5, the training and the validation accuracy of the CNNs reached close to 100% early in the process. Moreover, training and validation loss dropped with almost every step. Overall, the CNNs achieve a mean loss of $6.02 \cdot 10^{-6}$ and 100% mean accuracy. All 3000 tested images were classified correctly.

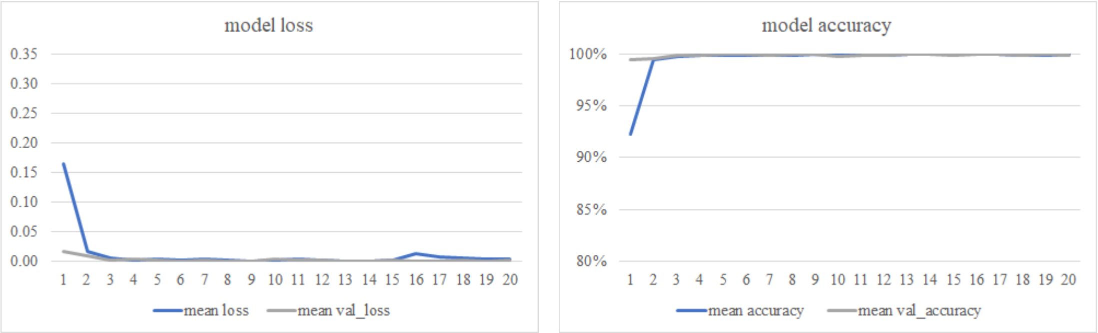


Figure 5: Average course of loss and accuracy for the baseline CNNs over 20 epochs

3.1.2 Features Closer Together Dataset

This time the CNNs were trained and tested with the images, where the features were closer together (FCT) than in the original positions (see Figure 3). There is hardly any difference in the learning process compared to 3.1.1 as can be seen in

Figure 6. After a slightly worse start the accuracy increased marginally quicker and the loss decreased faster than in the baseline CNNs.

On average all CNNs achieved 99.96% accuracy and $9.48 \cdot 10^{-4}$ loss. All "face" images were classified correctly, whereas only 1499 of the "no_face" images were identified correctly. The FCT CNNs therefore performed marginally worse.

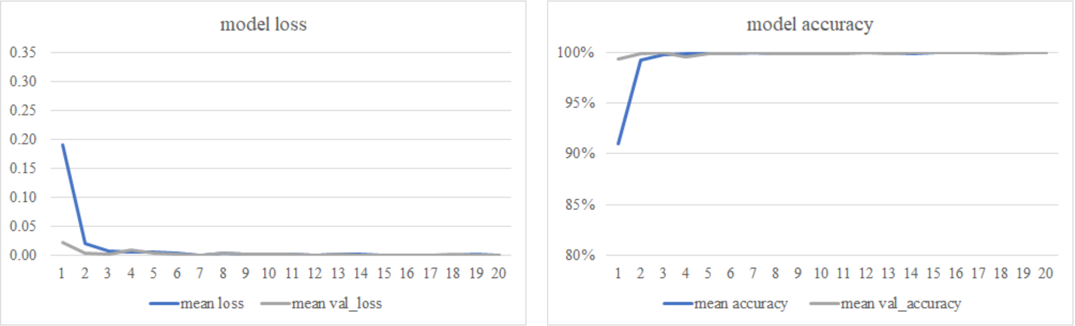


Figure 6: Average course of loss and accuracy for the FCT CNNs over 20 epochs

3.1.3 Features Further Apart Dataset

In this section the CNNs were trained and tested with images of faces with the facial features further apart (FFA) than in the baseline dataset (see Figure 3). The CNNs performed very similarly to 3.1.1 (see Figure 7). The start is similar to 3.1.1, the training and validation accuracies of the CNNs reach 100% within the first ten epochs. Again, training and validation losses decrease significantly with almost every epoch.

Altogether the CNNs achieved a mean accuracy of 99.99% and a loss of $6.37 \cdot 10^{-4}$. Here, all 3000 tested images were classified correctly, thus performing as well as the baseline CNNs.

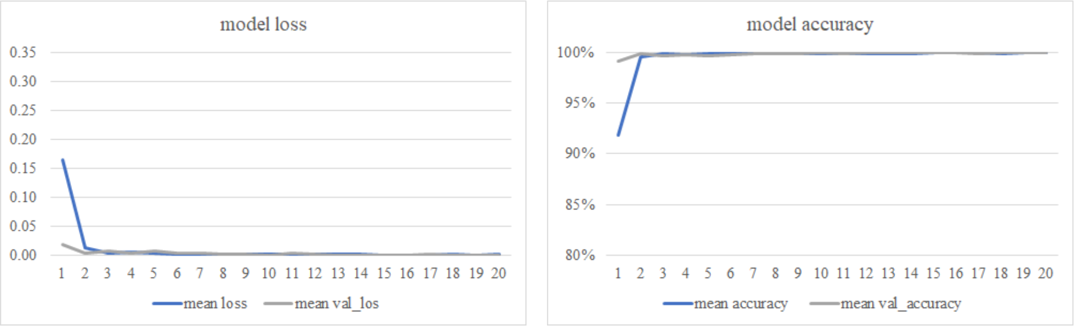


Figure 7: Average course of loss and accuracy for the FFA CNNs over 20 epochs

3.2 Training and Testing Fully Convolutional Neural Networks

3.2.1 Baseline Dataset

The FCNs, as the CNNs in 3.1.1, were trained and tested with the baseline dataset. While starting at approximately 10% less mean accuracy than the baseline CNNs, values after the second epoch were similar (see Figure 8). All in all, the average FCN learned quickly and achieved 100% accuracy within the first epochs. The overall mean accuracy also reached 100%. The loss value of $1.67 \cdot 10^{-5}$ was closer to the loss of the baseline CNNs than the values from the FCT (3.1.2) and FFA CNNs (3.1.3). All 3000 images were classified correctly. This was used as the baseline for all other FCNs.

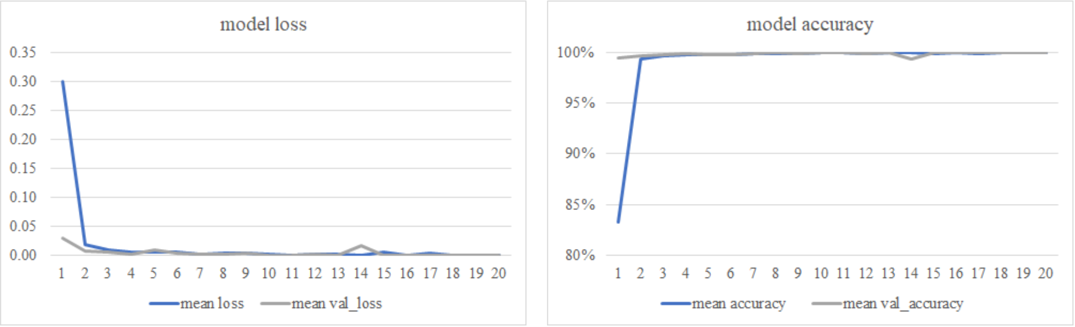


Figure 8: Average course of loss and accuracy for the baseline FCNs over 20 epochs

3.2.2 Features Closer Together Dataset

Here the FCNs were trained and tested with the images, in which the features are closer together than in the original. The mean accuracy after the first epoch is marginally lower than in the baseline FCNs, and overall, it achieves slightly worse results than the baseline FCN (see Figure 9). It also performs worse than the FCT CNNs (3.1.2). The average loss was $6.01 \cdot 10^{-3}$ and the accuracy 99.86%. All 1500 "no_face" images were identified correctly, 4 of the 1500 "face" images were misclassified. These are still acceptable values, but the worst ones so far.

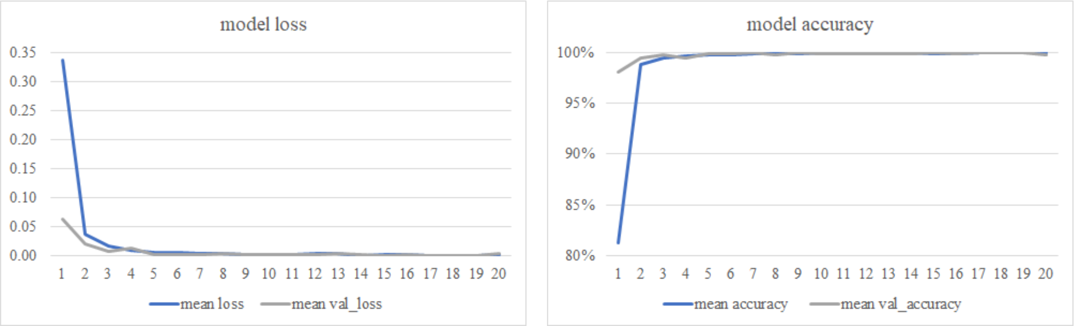


Figure 9: Average course of loss and accuracy for the FCT FCNs over 20 epochs

3.2.3 Features Further Apart Dataset

The last trained FCNs were trained and tested with the Features Further Apart Dataset. The mean accuracy after the first epoch was better than in the baseline and FCT FCNs. The average loss value was $1.36 \cdot 10^{-5}$ and therefore also better than the baseline FCNs from 3.2.1, the accuracy was the same. Again, all 3000 images were classified correctly.

3.3 Testing Different Data on the Pretrained Models

The previous sections already presented promising results. To further investigate whether relations between the features are truly relevant to classifying the images, the following tests were conducted. Since they are relevant here, also 30-epoch runs with 13/14-layers are included in the following sections.

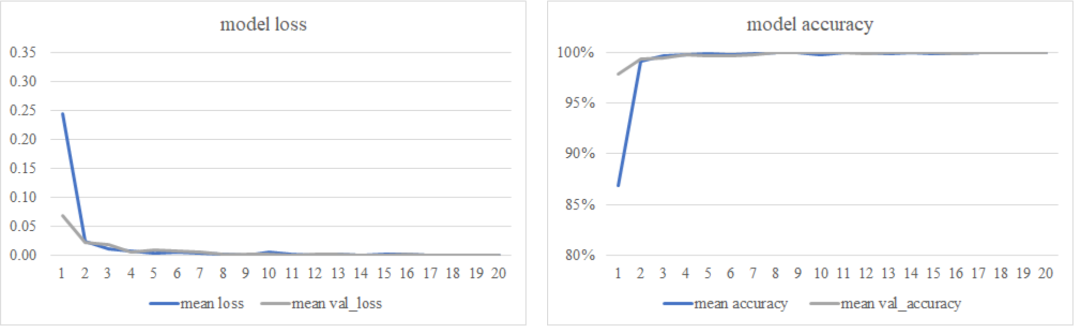
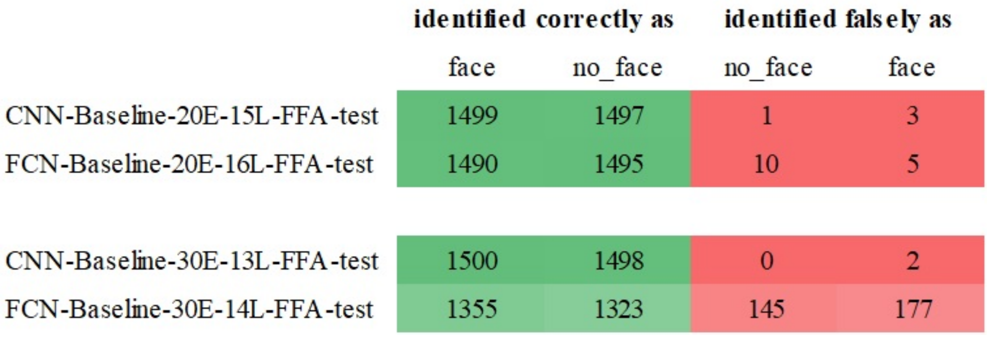


Figure 10: Average course of loss and accuracy for the FFA FCNs over 20 epochs

3.3.1 Testing Features Further Apart Data on Baseline Models

The networks in this part are the ones trained in 3.1.1 and 3.2.1 with baseline data, which achieved 100% accuracy when tested with the baseline data.

Tested with FFA data, the CNNs achieved 99.9% accuracy and less than 0.006 loss for both the 15- and the 13-layer models. The FCNs on the other hand still performed well but not as well as the CNNs. The 16-layer model achieved 99.5% accuracy and less than 0.02 loss, whereas the 14-layer model only reached 89.3% accuracy and displayed a loss of 0.35.



	identified correctly as		identified falsely as	
	face	no_face	no_face	face
CNN-Baseline-20E-15L-FFA-test	1499	1497	1	3
FCN-Baseline-20E-16L-FFA-test	1490	1495	10	5
CNN-Baseline-30E-13L-FFA-test	1500	1498	0	2
FCN-Baseline-30E-14L-FFA-test	1355	1323	145	177

Figure 11: Results from baseline CNNs and FCNs tested with FFA data

As can be seen in Figure 11, the first three ConvNets perform very closely to the training results from 3.1.1 and 3.2.1. Interestingly one image was always misclassified as a face in all tested CNNs (see Figure 12) even though it is clearly identifiable as "no_face" to the human eye.

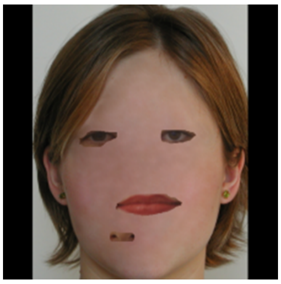
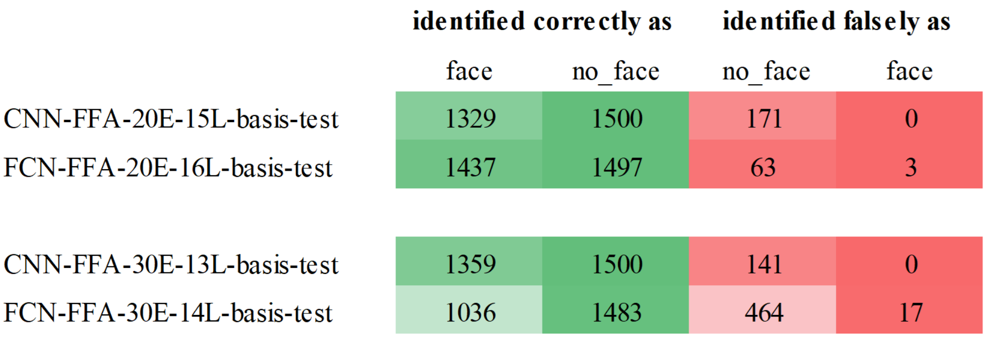


Figure 12: Image 09274.png

Overall, most of the images were identified correctly in all ConvNets. Even when the features were further away from the positions the model was trained with originally, both ConvNets performed reasonably well.

3.3.2 Testing Baseline Data on Features Further Apart Models

This time the networks trained in 3.1.3 and 3.2.3 with FFA data were used. In this case they were tested with baseline data. In this test both ConvNets did not perform as well as in 3.3.1. The CNNs achieved on average 94.3% accuracy in the 20-epoch runs, where the FCNs accomplished 97.8%. During the 30-epoch runs the CNNs performed better with an accuracy of 95.3%, whereas the FCNs only achieved 84.0%. The overall classification results can be seen in Figure 13.



	identified correctly as		identified falsely as	
	face	no_face	no_face	face
CNN-FFA-20E-15L-basis-test	1329	1500	171	0
FCN-FFA-20E-16L-basis-test	1437	1497	63	3
CNN-FFA-30E-13L-basis-test	1359	1500	141	0
FCN-FFA-30E-14L-basis-test	1036	1483	464	17

Figure 13: Results from FFA CNNs and FCNs tested with baseline data

The “no_face” class appears to be easier to identify than the “face” class in all cases. Altogether, both ConvNets again coped similarly well with the features being back in the original position. Even though it must be noted that the other way round (see 3.3.1) clearly works better.

3.3.3 Testing on Shifted Data

As part of the last test, it was attempted to discover whether the absolute positions of or the relations between the facial features in the images are relevant for the classification. For this the entire "face-part" of the image was moved to the left or right. The generated test data consisted of 1000 images with the faces on the left side, 1000 images positioned in the middle as before and 1000 images with the faces on the right side (see Figure 14). All three categories were divided into 50% "face" and 50% "no_face" data. This data was tested on all CNNs and FCNs that were described in 3.1 and 3.2.

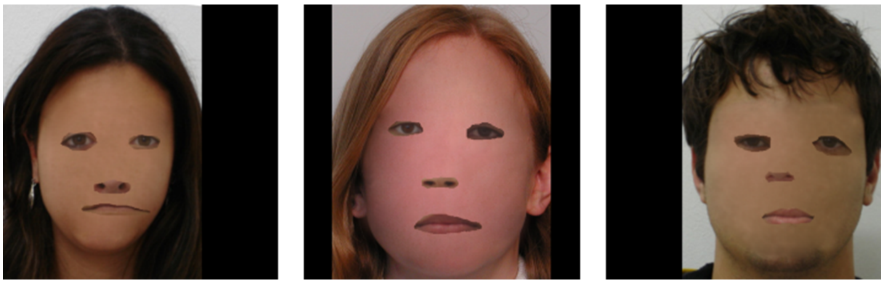
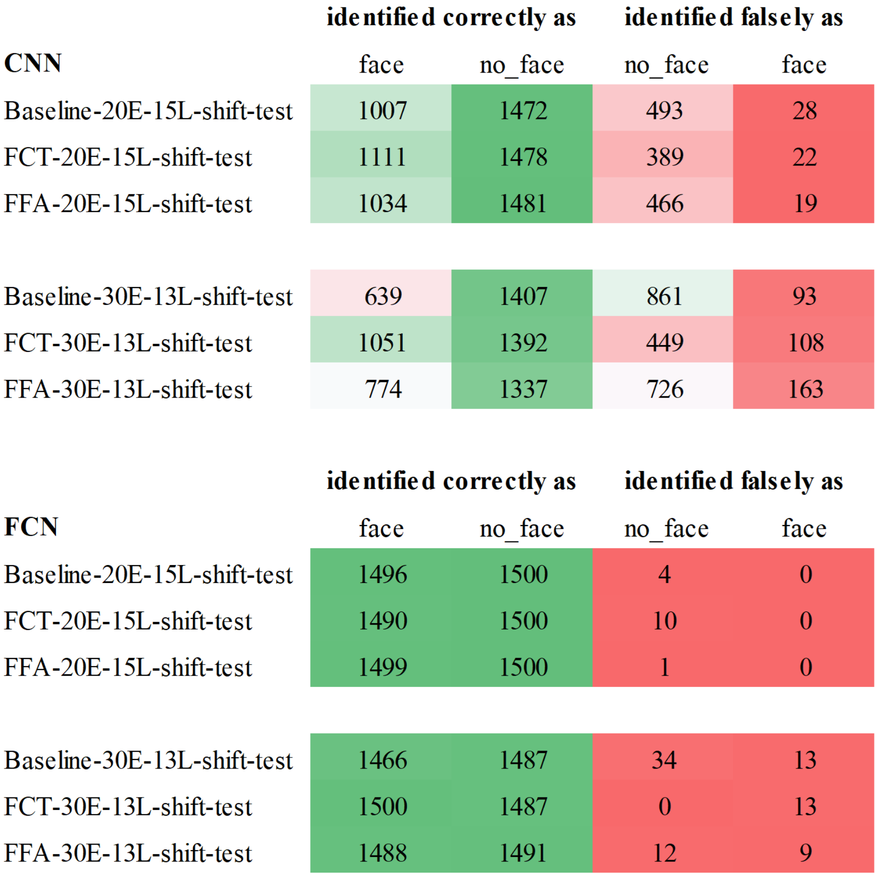


Figure 14: Examples of the shifted test data

In all other tests, the 30-epoch FCNs with 14 layers generally achieved similar if not worse results than the ones trained in 20 epochs and with 16 layers. Additionally, they performed similarly or worse than the CNNs trained in 20 and 30 epochs. They were omitted in sections 3.1 and 3.2 for this reason. While this was true for all other conducted tests, when tested with the shifted data they performed completely differently.

In these tests, the FCNs achieved an accuracy of 99.5% on average, while the CNNs achieved a mean accuracy of 78.8%. Especially when comparing the performance of the 14-layer-30-epochs-FCNs with the one of the 15-layer-20-epochs-CNNs, it is apparent that there is a large discrepancy between the FCNs' mean accuracy of 99.1% and the CNNs' mean accuracy of 84.3%. All mean accuracy



CNN	identified correctly as		identified falsely as	
	face	no_face	no_face	face
Baseline-20E-15L-shift-test	1007	1472	493	28
FCT-20E-15L-shift-test	1111	1478	389	22
FFA-20E-15L-shift-test	1034	1481	466	19
Baseline-30E-13L-shift-test	639	1407	861	93
FCT-30E-13L-shift-test	1051	1392	449	108
FFA-30E-13L-shift-test	774	1337	726	163

FCN	identified correctly as		identified falsely as	
	face	no_face	no_face	face
Baseline-20E-15L-shift-test	1496	1500	4	0
FCT-20E-15L-shift-test	1490	1500	10	0
FFA-20E-15L-shift-test	1499	1500	1	0
Baseline-30E-13L-shift-test	1466	1487	34	13
FCT-30E-13L-shift-test	1500	1487	0	13
FFA-30E-13L-shift-test	1488	1491	12	9

Figure 15: Results from CNNs and FCNs tested with the shifted data

and loss values can be found in Figure 16. It should be noted again, that the CNNs seem to have hardly any problems identifying the "no_face" class but appear to only guess when it comes to the "face" class.

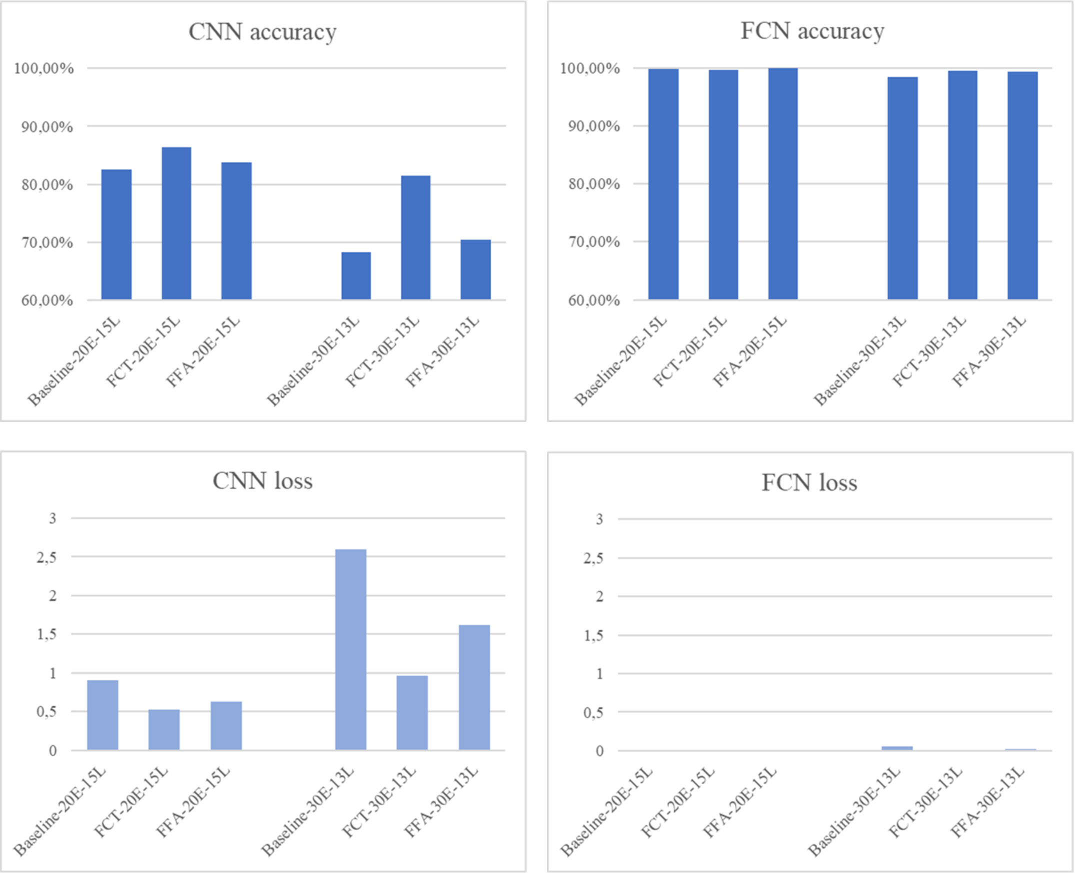
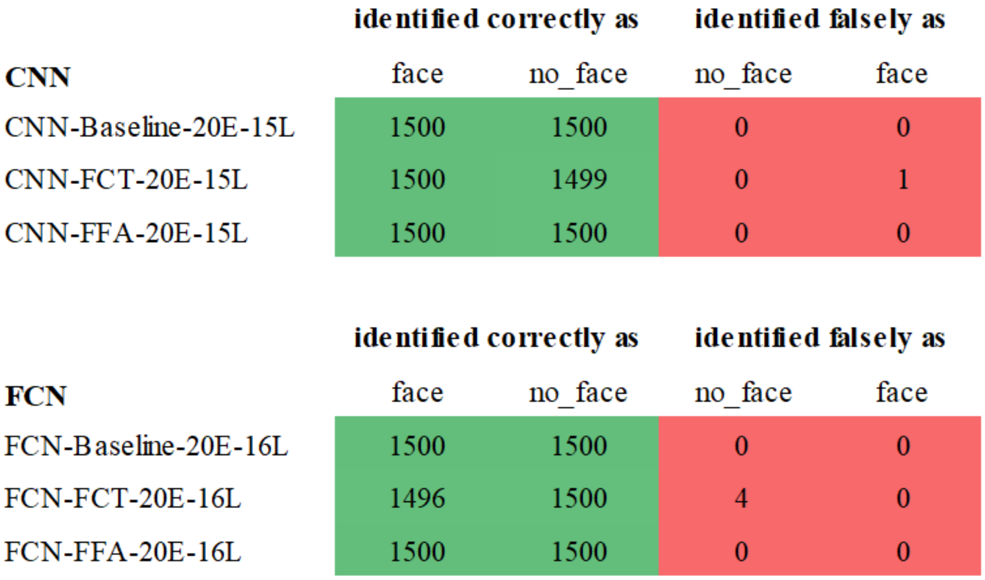


Figure 16: Mean accuracy and loss of CNNs and FCNs

4 Results

All in all, the results are promising. During training and testing in 3.1 and 3.2, there was only very little misclassification (Figure 17). In addition, the mean loss and accuracy values showed good performance of the ConvNets (see Figure 18). Also, there was little variance in the values. It should be noted, that the ConvNets trained with the Features Closer Together dataset all appear to perform marginally worse than the other two networks. This could be explained by the distances between the features being too short in some cases, but the collected data is not sufficient to confirm this.



	identified correctly as		identified falsely as	
CNN	face	no_face	no_face	face
CNN-Baseline-20E-15L	1500	1500	0	0
CNN-FCT-20E-15L	1500	1499	0	1
CNN-FFA-20E-15L	1500	1500	0	0

	identified correctly as		identified falsely as	
FCN	face	no_face	no_face	face
FCN-Baseline-20E-16L	1500	1500	0	0
FCN-FCT-20E-16L	1496	1500	4	0
FCN-FFA-20E-16L	1500	1500	0	0

Figure 17: Overview of classification results of CNNs and FCNs

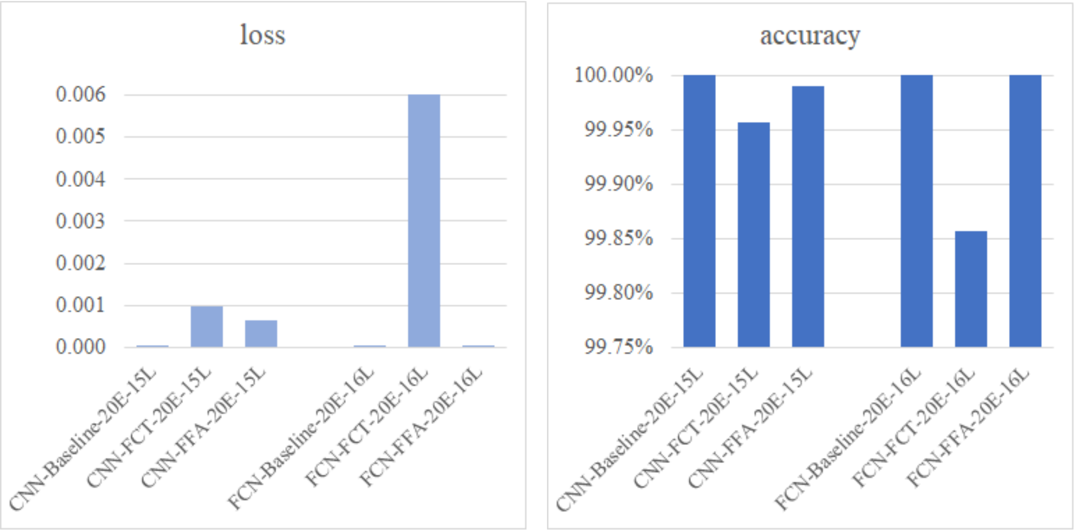


Figure 18: Overview of loss and accuracy of CNNs and FCNs

With these results it appears, that the ConvNets take the relations between the facial features into consideration (theory 1). Another explanation could be, that the ConvNets learn the general, absolute position of the facial features in the image instead of the spatial relations between them (theory 2). To explore, which option is more plausible, more tests were conducted in 3.3.

ConvNets in 3.3.1 were trained on baseline data and tested with FFA data. As it can be seen in Figure 11, the performance of both 20 epoch ConvNets is, on average, close to their performance when tested with baseline data (Figure 17). On average only 4 (CNN) and 15 (FCN) images were misclassified, which equals an error rate of at most 0.5%.

The results from 3.3.2 paint another picture. Contrary to 3.3.1, results were switched. ConvNets, that were trained on FFA data, which were then tested with baseline data conversely performed significantly worse. The mean number of misclassified images was 171 (CNN) and 66 (FCN) (Figure 13), amounting to a worst-case misclassification of 5.7% of all tested images.

These results support both theories. The results seen in Figure 19 argue in favour of theory 2. In theory the ConvNets could be dividing the images into nine sections. In this case, the three sections of the middle column would contain the relevant features. As one can see, these sections could vary in size and position from one dataset to another. An image from the FFA dataset could be divided into sections as illustrated by the yellow lines in Figure 19. All relevant information is included in these three middle sections: The upper part, containing the two eyes, the middle part, containing the nose, and the bottom part, containing the mouth. The same applies to the baseline images, where the red lines show a possible division.

When applying the red grid to the FFA image, the facial features still mostly fit the described segmentation. On the other hand, when applying the yellow grid to the baseline image, one can see that the grid no longer separates each facial feature into its own grid piece. This would explain, why the model trained on baseline images performed similarly well when tested with FFA images, but the model trained on FFA images had a larger error rate when tested with baseline images. One possible explanation for why the error rate of the latter is not larger, could be that the relations between the different facial features were taken into consideration (theory 1).

To investigate both theories further, all models that were trained in 3.1 and 3.2, were tested with images, which were shifted to the left or right (see Figure

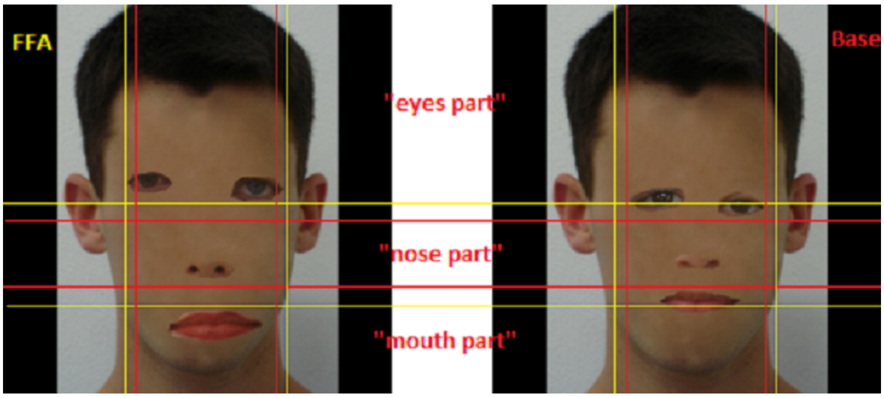


Figure 19: Example for theory 2 (FFA (yellow lines) left, baseline (red lines) right)

14). Figure 20 shows the red baseline grid applied to a baseline image which was shifted to the left.

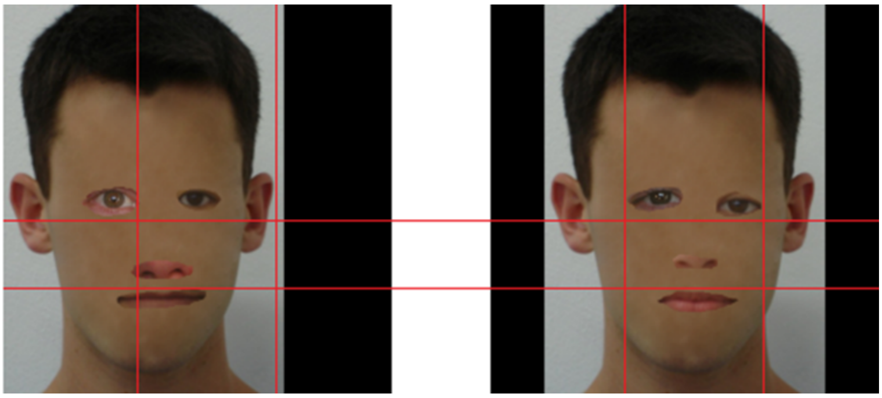


Figure 20: Example for theory 2 with shifted data

As one can see, when applying the red grid to a shifted image not all facial features fit the middle grid column anymore. This could explain, why the performance of the CNNs worsens significantly with shifted images (Figure 15). In the 20 epoch runs tested with shifted baseline, FCT and FFA images the average number of misclassified images was 472. That amounts to 15.7% of all tested images. It is noteworthy that the CNNs trained and tested with FCT data now delivered the best results, even though they previously performed worse than the other two types (chapter 3.1.2). It is to be added, that many or most "no_face" images were identified correctly, whereas in some cases even more than half of the “face” images were wrongly classified. This could be explained by the network misclassifying all “face” images, where the features were outside of the mentioned middle column of the red grid (Figure 20).

On the other hand, it can be observed that the FCN testing results surpassed the CNN testing results (Figure 15). With an average misclassification of 5 (0.2%) out of 3000 images averaged over the three types in all 20-epoch runs, they performed notably better. Additionally, it should be noted that even the 30-epoch FCN testing runs still outperformed the 20-epoch CNN runs. The mean number of misclassified images in 30-epoch FCNs was only 27 (0.9%) out of all 3000 tested images and was therefore still better. This is remarkable as this never occured during any other tests that were conducted for this thesis.

With these insights it seems that the CNNs learned something different than the FCNs. While the CNNs seem to be more focused on absolute positions, the FCNs appear to take relations between the facial features into consideration. The reason for this could be that CNNs “throw away spatial coordinates” [14], whereas FCNs retain spatial information.

5 Lessons Learned

In addition to the gathered results concerning the research question, the author aquired further knowledge in the process of writing this thesis. Where the CNNs were already doing very well with two convolution and one maxpooling layers, the FCNs had enormous difficulty learning anything. As the FCNs always showed around 50% accuracy after training, it was assumed that the FCNs were simply guessing. This led to the expectation that all four categories, correctly and incorrectly classified “face” and “no_face” images, would contain roughly the same number of test images. It turned out that, depending on the initial classification, the FCNs assigned almost all test images to either “face” or “no_face” categories. Even after a run with 400 epochs, the FCNs did not reach more than 75% accuracy. As a consequence of this problem, the author realised that one should work with more convolutional and maxpooling layers. This led to the results shown in chapters 3 and 4 of this paper.

Also, the computational effort was underestimated. Especially the training of the FCN with over 400 epochs took about 8 hours. The author's graphics card barely managed to train the CNNs specified in 3.1, there was too little GPU RAM available for the training of the FCNs (3.2). This problem was solved by using external hardware which had enough GPU RAM to run all trainings.

Problems also occurred when creating the test images. Some features on the generated images were placed on top of each other by the script. Debugging the script and moving individual pixels in the code was unsuccessful. After a significant amount of time, alias pixels that were hidden in the labels of the FASSEG dataset [7] became apparent. Once the alias pixels were removed manually, all the features were placed correctly.

6 Evaluation

In addition to the shortcomings and caveats mentioned in the previous chapters, several more points need to be mentioned.

The time frame and troubleshooting limited the scope and detail of this bachelor thesis. For these reasons and to make the results reproducible, only one seed was used. In addition, only 3 runs were performed per type of dataset and ConvNet, as the time frame was further limited by the long runtime of each ConvNet. The necessity of high-performance graphics cards was the biggest limiting factor. To increase the statistical validity, more runs with different seeds should be conducted.

Another factor that needs improvement is that the theories from chapter 4 are only based on this data set. This leads to the fact that the theories put forward are not complete and could not be fully tested due to the reasons already described.

7 Conclusion and Future Work

Both types of Convolutional Neural Networks performed exceptionally well after training with images with facial features positioned in their original slots as well as with facial features arranged closer or further apart. Even when networks were trained with facial features positioned in their original slots and then tested with images with facial features arranged further apart, or vice versa, the results were not as good as in the previous tests but still reasonably good. Only the tests with shifted images showed clear differences between the ConvNets. While the Fully Convolutional Neural Networks continued to perform as before, the fully connected Convolutional Neural Networks performed worse.

Overall, the results of the tests indicate that the relations between facial features are relevant for ConvNets. However, there is a difference between Fully Convolutional Neural Networks and the fully connected Convolutional Neural Networks. For the FCN, the relations seem to be more relevant than for the more common fully connected CNN.

Since this work could not conclusively clarify whether spatial relations are relevant for CNNs, the following additional tests should be carried out in the future:

Firstly, instead of only moving the images to the left and right, one could also move them up and down and test the trained CNNs with this dataset.

Furthermore, networks trained with images already shifted to the left or right should be tested with images shifted in the other direction. This method would most likely ensure that the facial features are no longer too close to their original position, which should make it more difficult for the ConvNets to identify faces by their absolute position within the images.

Finally, the number of convolutional and maxpooling layers could be increased. It is possible that the fully connected CNN will only learn to take relations more into account in a later filter.

References

[1] Adadi, A., Berrada, M.: Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access **6**, 52138–52160 (2018). https://doi.org/10.1109/ACCESS.2018.2870052

[2] Brownlee, J.: Gentle Introduction to the Adam Optimization Algorithm for Deep Learning (2017), `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/`

[3] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: Intelligible Models for HealthCare. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1721–1730. Association for Computing Machinery, Sydney, NSW, Australia (2015). https://doi.org/10.1145/2783258.2788613, `https://doi.org/10.1145/2783258.2788613`

[4] DARPA: Broad Agency Announcement Explainable Artificial Intelligence (XAI) DARPA-BAA-16-53 (2016), `http://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf%0Apapers3://publication/uuid/AF93BD83-DA5C-48BE-A0B4-212DC3C78A31`

[5] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`

[6] Ide, H., Kurita, T.: Improvement of learning for CNN with ReLU activation by sparse regularization. Proceedings of the International Joint Conference on Neural Networks **2017-May**, 2684–2691 (2017). https://doi.org/10.1109/IJCNN.2017.7966185

[7] Khan, K., Ahmad, N., Ullah, K., DIn, I.: Multiclass semantic segmentation of faces using CRFs. Turkish Journal of Electrical Engineering and Computer Sciences **25**(4), 3164–3174 (2017). https://doi.org/10.3906/elk-1607-332

[8] Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015), `http://neuralnetworksanddeeplearning.com`

[9] Ozturk, O., Saritürk, B., Seker, D.Z.: Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries. International Journal of Environment and Geoinformatics **7**(3), 272–279 (2020). https://doi.org/10.30897/ijegeo.737993

[10] Rabold, J., Deininger, H., Siebers, M., Schmid, U.: Enriching visual with verbal explanations for relational concepts – combining LIME with aleph. In: Communications in Computer and Information Science. vol. 1167 CCIS, pp. 180–192. Spriger International Publishing (10 2020). https://doi.org/10.1007/978-3-030-43823-4_16, `http://arxiv.org/abs/1910.01837`

[11] Rabold, J., Schwalbe, G., Schmid, U.: Expressive explanations of dnns by combining concept analysis with ILP. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **12325 LNAI**, 148–162 (2020). https://doi.org/10.1007/978-3-030-58285-2_11

[12] Rabold, J., Siebers, M., Schmid, U.: Explaining Black-Box Classifiers with ILP – Empowering LIME with Aleph to Approximate Non-linear Decisions with Relational Rules. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 11105 LNAI, pp. 105–117. Springer Verlag (2018). https://doi.org/10.1007/978-3-319-99960-9_7

[13] Rawlani, H.: Understanding and implementing a fully convolutional network (FCN) (2020), `https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b`

[14] Shelhamer, E., Long, J., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(4), 640–651 (2017). https://doi.org/10.1109/TPAMI.2016.2572683

[15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014)

[16] Steinwender, J., Schwaiger, R.: Neuronale Netze programmieren mit Python. Rheinwerk Verlag, 2 edn. (2020)

[17] The MathWorks: Convolutional Neural Network Drei Dinge, die Sie über Convolutional Neural Networks wissen sollten, `https://de.mathworks.com/discovery/convolutional-neural-network-matlab.html`

[18] Trask, A.W.: Neuronale Netze und Deep Learning kapieren – Der einfache Praxiseinstig mit Beispielen in Python. mitp Verlag (2019), `https://learning.oreilly.com/library/view/neuronale-netze-und/9783747500170/`

List of Figures

Appendix

The code and datasets used in this thesis can be found at: `https://github.com/LisaCodes5/thesis-are-relations-relevant-in-cnns`

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

(Datum) (Unterschrift)