Paper, Rock, and Scissors Recognition on Images

xAI-Proj-M: Master Project Explainable Machine Learning

Philipp Höfling *
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
philipp-andreas.hoefling@stud.uni-bamberg.de

Satyam Pant†
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
satyam.pant@stud.uni-bamberg.de

Tobias Koch‡
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
tobias.koch@stud.uni-bamberg.de

Abstract

(Tobias) This scientific report presents the results of a project aimed at classifying images of hand signs for rock, paper, and scissors. The study addressed three research questions: (1) the impact of transfer learning on the performance of convolutional neural network models, (2) the effect of various data augmentation techniques on the model's accuracy, and (3) the potential of model-agnostic explanations to enhance the interpretability of the convolutional neural network models. The experiments show that transfer learning using the pre-trained EfficientNet-V2 model leads to a 7.26% improvement in validation accuracy and a reduction of 855 minutes in training time compared to the baseline model. Moreover, applying data augmentation techniques, particularly random horizontal flip, further improves the performance of the Transfer Learning Model. Finally, model-agnostic explanations provided by LIME enable a better understanding of the model's decision-making process and identify areas for potential improvement in data engineering.

1 Introduction (Philipp)

This report is the outcome of the "Master Project Explainable Machine Learning" course at Otto-Friedrich University of Bamberg. The objective of the project was to create and evaluate a machine learning algorithm to classify rock-paper-scissors hand gestures on images. Rock-paper-scissors is a popular hand game played worldwide where each hand sign beats a particular other sign (Wik, 2023). The images to be classified include only one hand mark, and the occurrence of no hand signs is excluded. The project comprises the three main parts data engineering, model engineering, and model evaluation, following the CRISP-ML lifecycle (Studer et al.). Each part was managed by a different team member and focused on specific research questions. The data augmentation part aimed to determine "Which data augmentation methods can enhance the model's accuracy in classifying images of rock-paper-scissors hand signs?" The research question for model engineering was "Can transfer learning improve the performance of CNN models in classifying rock-paper-scissors images?" Finally, the model evaluation part investigated the effectiveness of Local Interpretable Model-agnostic Explanations (LIME) in enhancing the interpretability of CNN models for classifying rock-paper-scissors images.

*Degree: M.Sc. WI, matriculation #: 2086535
†Degree: M.Sc. AI, matriculation #: 2062179
‡Degree: M.Sc. WI, matriculation #: 2060386

2 Related Work

Data Augmentation (Satyam) Deep learning has made immense advancements in many fields. However, it has some limitations when dealing with scarcity of data, which results in a low degree of generalization on both validation and testing set Lei et al. (2019). Data Augmentation is a strategy that is used to prevent overfitting in a model by making use of regularization Maharana et al. (2022). As stated by Shorten and Khoshgoftaar (2019), image classification requires enormous amounts of data for training. However, many application domains don't have this much data so to overcome this issue, data augmentation techniques come in handy as they try to create "fake data" from the existing data by enhancing the size and quality of the training dataset without losing the semantics of the image. Nevertheless, while performing image augmentation labels must be transformed accordingly Ledig (2022-2023).

Transfer Learning (Tobias) Transfer learning has become more popular in recent years, especially in the research area. In (Ravishankar et al., 2017), they use for example the CaffeNet architecture pre-trained on the ImageNet dataset to solve the kidney detection problem in ultrasound images. Also, gaming is an interesting field for transfer learning where a model trained for a specific task can be used to learn another game. An AI for playing the strategy game MadRTS is developed using the CAse-Based Reinforcement Learner (CARL) architecture (Sharma et al., 2007). Here, the model is not pre-trained on another game but transfer learning is used to incorporate the knowledge from different tasks inside the game (Sharma et al., 2007). Even if there is much research in this field, the cases where transfer learning is implemented in real-world applications are limited.

Explainable AI - Local Interpretable Model-agnostic Explainations (Philipp) While deep learning is getting more popular over the last years deep learning methods are often referred to as black boxes. One of the reasons for this is the complexity of the models and the resulting problems when trying to understand how the model arrives at its results (Ribeiro et al.). LIME is an algorithm that can explain the predictions of any classifier or regressor in a faithful way, by approximating it locally with an interpretable model (Ribeiro et al.). LIME is used for example in medical research to make the classification of an image more interpretable e.g., the results of a Convolutional Neural Network which detects tumor tissue in patches extracted from histology whole slide images (Palatnik de Sousa et al., 2019). While in the original paper for LIME besides the approach to provide explainability for images the approach for text classification is presented (Ribeiro et al.), the Authors provided a LIME version for audio classification (Saumitra Mishra et al, 2017).

3 Datasets (Satyam)

During implementation a total of six datasets are considered which are taken from different open-source websites namely kaggle and robflow. However, a couple of local datasets have random images clicked by different people with numerous backgrounds and gestures. Also, some of the sources include kagglee and github which contains the most diverse images of rock, paper, and scissors with backgrounds.

3.1 Overview

While analyzing datasets, it is observed (see Fig.:2a) that dataset 1 and dataset 4 have the highest image distribution of around 3000 and 2200 images respectively whereas datasets 5 and 6 have the least images of approximately 150 and 160 respectively. On the other hand, dataset 2 contains close to 2200 images compared to dataset 3 which encloses 2350 images.

As all the datasets are diverse in terms of foreground and background, having an overview is critical. Some of the random images from the datasets are shown below along with some illustrations:(see Figure:1)

- **Dataset 1:** This set contains image data (see Fig:1a) posed against a white background with a pixel size of 300x300 in 24-bit color and is taken from robflow. Besides this, the dataset has a distribution of around 975 images per class.
- **Dataset 2:** This dataset contains images of hand gestures against a green screen (see Fig:1b) with a pixel size of 300x200 and is adapted from kaggle. The dataset is subdivided and has

around 750 images of 'Rock', 720 images of 'Paper', and approximately 755 images of 'Scissors'.

- **Dataset 3:** This dataset contains images of the human face along with hand gestures of rock, paper, and scissors which can be seen in Figure 1c) and it is taken from kagglee. Besides this, the dataset has around 830 images of 'Paper', approx 790 'Rock' images, and close to 755 images of 'Scissors'.

- **Dataset 4:**This dataset includes hand signs images of rock, paper, and scissors along with numerous backgrounds (see Fig.1d) adapted from github. Further, the class distribution is within the range of 800 to 830 images.

- **Dataset 5 and 6**:Both of these datasets were provided to us during later stages for validation and testing of the model. The images were captured via a mobile camera and have non-identical images of hand gestures with different complex backgrounds (refer to Fig.1e,1f). On top of that, the class distribution for both datasets is somewhat similar which is within the range of 50 to 70 images.

(a) Dataset 1

(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

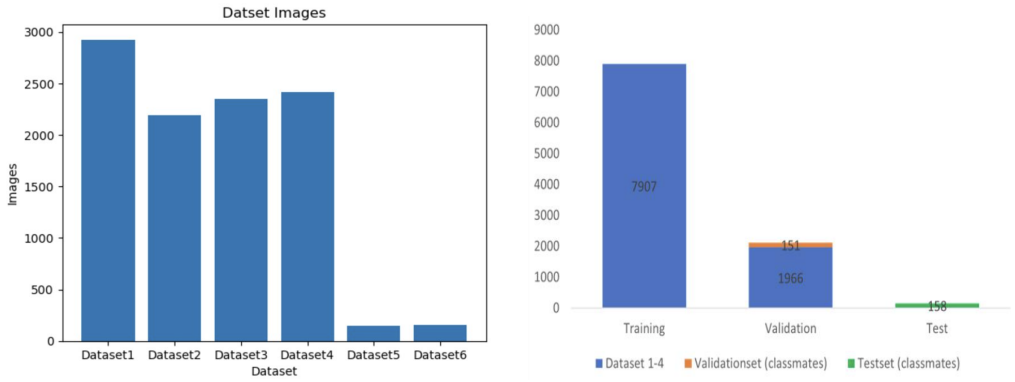
(e) Dataset 5

(f) Dataset 6

Figure 1: Dataset images

3.2 Preprocessing

Initially, all the images are converted from RGBA (Red Green Blue Alpha) to RGB (Red Green Blue) as sometimes multiple images are layered with some alpha value which represents the opacity ie. what part of colors from the lower region can be seen through the color at a given level (alpha composting or alpha blending) Maji and Nath (2014).
After that datasets are splitted (see Figure 2b), dataset 1 to dataset 4 are partitioned into train and validation set in the ratio of 80:20 respectively. Later, dataset 5 is introduced to extend the validation set, and eventually, dataset 6 is kept aside for the testing phase. So, in total training set has around 7900 images where as validation set comprises of approx. 2220 images and the test set contains 158 images.
Because a pre-trained model is used for further steps, it was necessary to preprocess the images in the same way the model was trained. As a result of this, the images are resized to a uniform size of (384x384) and then transformed into tensors. Moreover, the tensors are transformed to numbers between [0,1] and normalized with the help of mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].



(a) Image Datasets (b) Dataset Split

Figure 2: Dataset Distribution

4 Experiments

This chapter presents the development environment and the experiments to answer the three research questions mentioned in the introduction.

4.1 Development Environment (Tobias)

The neural networks are implemented in Python with PyTorch as the main library for Deep Learning. Besides Jupyter notebooks, which are used for quick tests and first trials, modular programming with Python files is realized as a software design technique to break down the code into smaller parts. The training of the neural networks is performed on the NVIDIA Quadro T1000 GPU with Intel® Core™ i7-10750H and 32 GB RAM. In addition, Microsoft Teams is used for task management and knowledge sharing and GIT as version control tool. The public GIT repository[4] with an MIT license includes a Readme with information on how to setup the environment and how to train an own model or test an existing one.

4.2 Model Engineering (Tobias)

In this section, the question if transfer learning can improve the performance of CNN models in classifying rock, paper, and scissors on images is explored. For this, the performance of two models (Baseline Model and Transfer Learning Model), which are based on the EfficientNetV2-S architecture, are trained and compared. Transfer learning involves leveraging the knowledge and representation learned by a pre-trained model on a large dataset, such as ImageNet-1K, to improve the performance of a model on a smaller dataset or a different task. In the case of classifying rock, paper, and scissors on images, a pre-trained model on ImageNet-1K could be useful because the model has probably learned to recognize and extract useful features such as edges, textures, shapes, and objects from images. Also, the fine-tuning process can be quicker and more efficient. In the following subsections, first, the model architecture is described and subsequently, both models are evaluated and compared.

4.2.1 EfficientNetV2-S Architecure

The EfficentNet architecture was first introduced in 2019 by Google AI and promises high precision with optimized efficiency (Tan and Le, 2019). To be able to scale it up easily, the developers focused on a simple construction of a basic model (see Figure 3), which consists of a series of mobile inverted bottleneck convolution (MBConv) blocks (Tan and Le, 2019).

MBConvs are often used in models to reduce the number of channels and thus increase efficiency. An MBConv usually consists of a 1x1 convolution layer with a ReLU activation function, followed by a depthwise convolution with a ReLU activation function and a linear 1x1 convolution (Sandler et al., 2018; Tan and Le, 2019). While the first 1x1 convolution layer extends the input to a higher

[4]Git repository: `https://github.com/koch-tobias/PaperScissorsRock_byHandmodels`

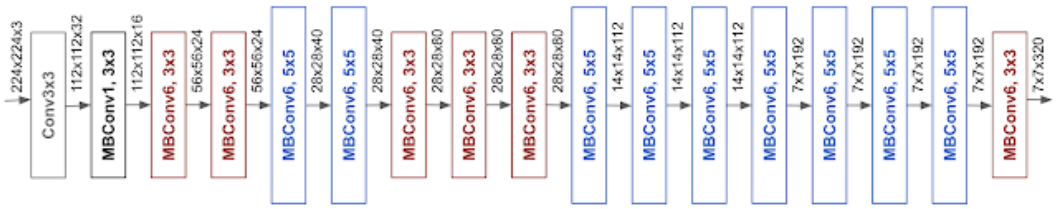


Figure 3: EfficientNetB0 basic network (Tan and Le, 2019)

dimension, the 1x1 convolution at the end is used for the projection back to a lower dimension (Gonzales, 2019). With the depthwise convolution in the center, spatial filtering of features from the higher dimensional input is achieved (Gonzales, 2019). Another property of MBConvs is the skip connection of the input to the output to address the problem of exploding and vanishing gradients. Figure 4 shows a comparison of the EfficientNet models with other architectures, which are often chosen for image classification. All models are trained on the ImageNet-1K dataset that contains over 1.4 million images with 1000 output classes (Russakovsky et al., 2015).

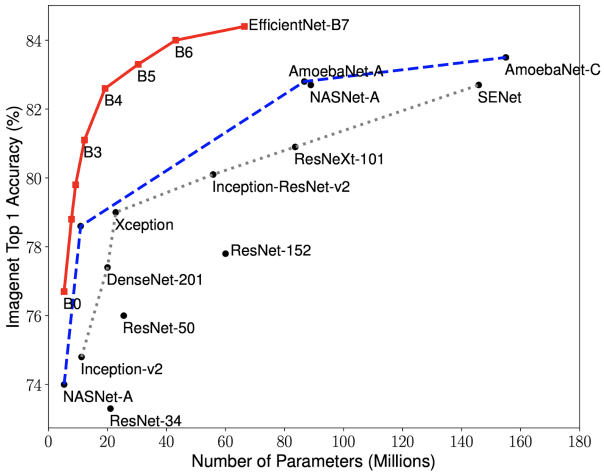


Figure 4: Model comparison trained on ImageNet dataset (Tan and Le, 2019)

It can be observed that even the upscaled version EfficientNet-B4 achieves a high top-1 accuracy while using comparatively few parameters. However, apart from the benefits in performance, the EfficientNet models also have a drawback related to the high training time due to the MBConvs. Therefore, the publishers of the EfficientNet architecture have developed an adapted version in 2021, which should accelerate the training without performance loss. To achieve this, they replaced some of the first MBConv blocks with FusedMBConv blocks. FusedMBConv blocks have the advantage over MBConv blocks that the depthwise convolutions, which are slow at high dimensionality, are replaced by simple convolution layers (Tan and Le, 2021). This structural change achieves an acceleration of the training time as well as a reduction of the model size (up to 6.8 times smaller) compared to other state-of-the-art models (Tan and Le, 2021). The EfficientNetV2 model exists in three versions: small (S), medium (M), and large (L). For this experiment, the EfficientNetV2-S version is used to reduce training time while maintaining high accuracy. The layers of the EfficientNetV2-S model are listed in Table 1. Comparing the EfficientNetV2-S model with the EfficientNet-B7 model, the EfficientNetV2-S achieves on the ImageNet dataset a top-1 accuracy of 83.9%, uses 22 million parameters, and trains 7.1 hours whereas the EfficientNet-B7 achieves a top-1 accuracy of 84.7%, uses 66 million parameters, and trains 139 hours (Tan and Le, 2021).

4.2.2 Algorithms, Activation Function, and Parameters

In the context of training a deep neural network, it is essential to specify appropriate components, including a loss function, an optimizer, and an activation function. Given the problem of classifying

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4 SE0.25, k3x3	2	128	6
5	MBConv6 SE0.25, k3x3	1	160	9
6	MBConv6 SE0.25, k3x3	2	256	15
7	Conv1x1 + Pooling + FC		1280	1

Table 1: EfficientV2-S architecture

rock, paper, and scissors images, which constitutes a multi-class classification task, it is necessary to use a suitable loss function. Therefore, the cross-entropy function is often chosen for computing the loss associated with multi-class classification problems. The formula for the cross-entropy loss is shown in equation 1, where **t** are the ground truth labels and **p** the output probabilities. In this case, the probabilities **p** are calculated with the softmax activation function.

$$\mathcal{L}(\mathbf{t}, \mathbf{p}) = -\sum_{j=1}^{K} t_j log(p_j) \quad with \quad p_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

For adjusting the weights and searching for the global optimum, the Adaptive Moment Estimation (Adam) optimizer is used. This optimizer is widely used for image classification because it achieves high performance in a comparatively short time. It combines the advantages of the Adaptive Gradient (AdaGrad) algorithm and the Root Mean Square Propagation (RMSProp), which are both extensions of the stochastic gradient descent (SGD) algorithm (Ruder, 2016).

In order to optimize the performance and prevent overfitting of the deep neural network model, the hyperparameters are tuned using a grid search methodology. Nevertheless, due to the extensive training time associated with the neural network, a limited number of hyperparameter combinations are evaluated. The selected hyperparameter configurations are presented in Table 2. Because of the large number of trainable parameters in the Baseline Model and our limited GPU capacity, a maximum batch size of two could be chosen for comparison. However, to test the effect of a higher batch size on the model performance, the best Transfer Learning Model is additionally trained with a batch size of 64.

Epochs	Batch Size	Patience	Dropout	Learning Rate	Seed	Workers
100	2	4	[0.1,0.2]	[0.001, 0.01]	42	4

Table 2: Grid search hyperparameters

Furthermore, an early stopping criterion is implemented to prevent a termination of the model before it reaches a desired level of convergence. Specifically, the criterion is designed to avoid stopping the training process before epoch 9, since the Baseline Model requires several iterations to demonstrate noticeable improvement in performance on the validation dataset.

4.2.3 Evaluation Baseline Model

The Baseline Model serves as a comparative model to interpret the performance of the Transfer Learning Model. Here, the EfficientNetV2-S model with the architecture shown in Table 1 is used without pre-trained weights, which means that all 22 million parameters are trained with the rock, paper, and scissors dataset. The last layer is adjusted to the classification problem so that the network returns three output probabilities instead of 1000. After training the model using the grid search hyperparameter tuning, the results shown in Table 3 are achieved.

Considering the top-1 validation accuracy, the best Baseline Model (1) with 90.16% is achieved using a dropout of 0.1 and a learning rate of 0.001. This model is trained for 51 epochs in a total training time of 946 minutes.

	Trained Epochs	Dropout	Learning Rate	Total Train Time	Train Loss	Train Acc	Validation Loss	Validation Acc
1	51	0.1	0.001	946	0.1156	0.9579	0.3072	0.9016
2	45	0.1	0.01	835	0.234	0.8435	0.4010	0.7830
3	25	0.2	0.001	460	0.8853	0.6965	1.0055	0.6243
4	55	0.2	0.01	1020	0.1598	0.9332	0.3426	0.8872

Table 3: Hyperparameter optimization of the Baseline Model

4.2.4 Evaluation Transfer Learning Model

For training the Transfer Learning Model, most layers of the EfficientNet2-S model are frozen. Only the last 3 layers are set as trainable, which means that 2.3 million parameters are fine-tuned by training the model with the paper, scissors, and rock dataset. Also here, the last layer is adjusted to fit the new classification problem. Table 4 presents the results of training the model with the hyperparameters displayed in Table 2. Here, the best validation accuracy of 97.42% is achieved by Transfer Learning Model 1 after training for 19 epochs in a total training time of 91 minutes.

	Trained Epochs	Dropout	Learning Rate	Total Train Time	Train Loss	Train Acc	Validation Loss	Validation Acc
1	19	0.1	0.001	91	0.0510	0.9824	0.0974	0.9742
2	20	0.2	0.001	96	0.0518	0.9827	0.0949	0.9709
3	09	0.1	0.01	40	0.1389	0.9575	0.1529	0.9563
4	16	0.2	0.01	77	0.0900	0.9700	0.9581	0.9582

Table 4: Hyperparameter optimization of the Transfer Learning Model

As already mentioned in Section 4.2.2 the models are trained with a batch size of two because of the limited GPU capacity. Therefore, solely the best Transfer Learning Model has been trained additionally with a batch size of 64 to test the effect on the model performance. All other parameters are not changed. The change of the batch size results in an accuracy of 98.18% on the validation set, as shown in Table 5.

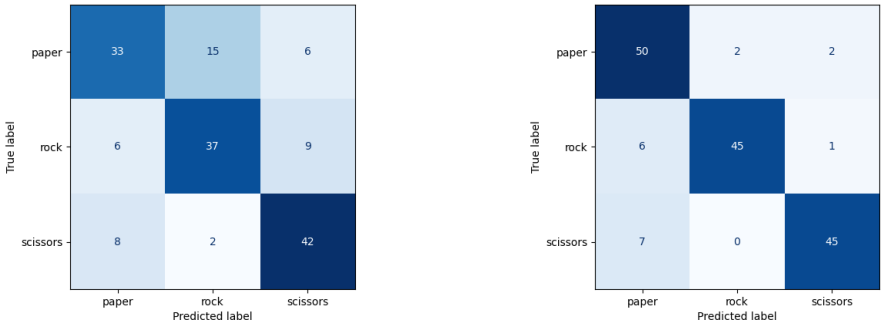
	Trained Epochs	Dropout	Learning Rate	Total Train Time	Train Loss	Train Acc	Validation Loss	Validation Acc
1	23	0.1	0.001	77	0.0102	0.9968	0.0706	0.9818

Table 5: Best Transfer Learning Model trained with a batch size of 64

4.2.5 Comparison

In this section, the best model of the Baseline Model and the best model of the Transfer Learning Model with a batch size of two are compared on the validation and the test set. In addition, the Transfer Learning Model trained with the higher batch size is considered. Beginning with the comparison of the best Baseline Model and the best Transfer Learning Model, Figure 13 shows the training and validation loss and also the accuracy of both models. It is interesting that the Baseline Model needs around 9 epochs until improvements are visible. This could be due to the high number of parameters that have to be trained. Also noticeable are the jumps in loss and accuracy on the validation set when training the Transfer Learning Model. This could probably be improved by using more parameters in the hyperparameter tuning process or by adding more data with a complex background. Comparing the number of trained epochs and the total training time in Figure 3 and 4, the Transfer Learning Model trained 855 minutes less than the Baseline Model and achieves a substantially higher accuracy (+7.26%) on the validation set.

The confusion matrices in Figure 5 provide an overview of the performance of the models on the test set for each output class. Calculated from this, the accuracy is 70.89% for the Baseline Model and 87.97% for the Transfer Learning Model, which is a difference of almost 18%.



(a) Baseline Model (b) Transfer Learning Model

Figure 5: Confusion matrix comparison on the test set

Figure 14 shows the loss and accuracy plots and Figure 6 the confusion matrix for the Transfer Learning Model, which is trained with a batch size of 64. Comparing these results with those of the best Transfer Learning Model with a batch size of 2, the model performance improved slightly on the validation set whereas on the test set no substantial differences occurred.

Taking all confusion matrices into account, it is noticeable that images with the ground truth "scissors" are often wrongly classified as "paper" and images with the ground truth "paper" as "rock". The reasons will be discussed in more detail in the model evaluation experiment using explainable artificial intelligence methods.

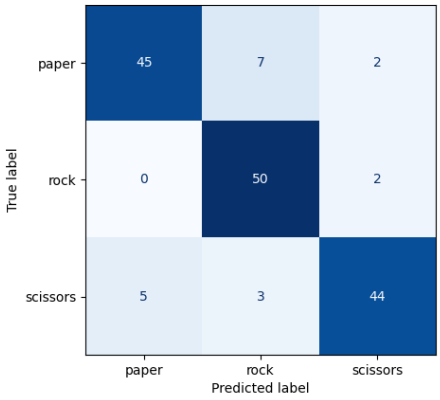


Figure 6: Confusion matrix on the test set of the Transfer Learning Model trained with a batch size of 64

4.3 Data Engineering (Satyam)

In this section, the research question is formulated as "Which data augmentation methods can increase the model's performance in classifying images of „Rock, Paper, Scissors“ hand signs?" As there are numerous augmentation techniques but to justify the research question here a combination of image augmentation methods are taken into consideration and then model's performance is eventually evaluated using accuracy.
When doing data augmentation, the main challenge is obtaining the correct data and determining the appropriate relevance to our research question. Thus, applying a variety of augmentation techniques allows us to obtain images of varying sizes, poses, colors, and lighting. In the following section some of the augmentation techniques are covered which show the variability in the image visuals.Khosla and Saini (2020)

(a) Original Image (b) Rotation (c) Horizontal Flip

(d) Noise (e) Shift (f) Colour Jitter

Figure 7: Image Augmentations

4.3.1 Augmentation Techniques

During this project libraries like torchvision and albumentations are used for augmenting the images. However, more emphasis was given to torchvision as we adapted to it during the early stages and also obtained some amazing results from it. Some of the augmentation methods are listed in (see Fig. 7), which compares the original image with different augmentations like rotation, flipping, color jitter, etc.

1. **Rotation:** Depending on our requirements, we can select the rotating parameter value, for example, 45 degrees or orient at minute angles. While rotating the image one must also consider the background of the image. If it is either black or white, then the newly added color will introduce some kind of noise into the image. In contrast, the images with different color backgrounds will not allow the noise to blend and therefore, allowing the network to learn insights from the image. Moreover, the degree of rotation should be selected wisely, otherwise the labels won't be preserved Khosla and Saini (2020). Figure 7b shows how rotation is applied to an image.
2. **Horizontal Flip:** We have only considered horizontal flipping here, since vertical flipping is restricted in some frameworks. Further, flipping produces images by rotating images by multiples of 90 degrees. For our use case, flipping plays an essential role in preserving the labels Khosla and Saini (2020). Figure 7c shows the result after horizontally flipping the original image.
3. **Noise:** Noise refers to the addition of grains in terms of white and black dots. In general, this technique prevents overfitting, which improves the performance of the model. Figure 7d shows grains on the image.
4. **Shift:** Image shifting tends to move the image along the X or Y direction or both. This method can cause some additional noise in the image as the network tries to look everywhere in the image. By shifting the image left, right, up, and down, positional bias can be avoided. Figure 7e shows how the image is horizontally shifted.

5. **Colour Jitter:** This method changes the pixel value rather than pixel position.This augmentation allow us to vary the brightness, contrast, hue, and saturation of the images. Depending on the brightness parameter, the image will either be dark or bright. While the contrast parameter determines the contrast between light and dark colors in an image.Moreover, hue alters the shade of the image, while saturation determines the amount of color in an image. Figure 7f shows the color jitter approach.

4.3.2 Augmentation methods used in combination

As stated in Yang et al. (2022), the selection and combining of data augmentation techniques to generate new data can be seen as a critical criterion but as per Pawara et al. (2017), techniques when used in combination produce better results than a single method. Moreover, as proposed by Elgendi et al. (2021), we used some combinations along with their parameter values and adjusted them according to our model :

- **Combination 1:** This augmentation include only rotation operation within range [-15,15]. So, the pseudo-code for this augmentation looks like:

  "Rand Rotation,"[-15,15]

- **Combination 2:** The parameters and the combinations for this augmentation are adapted from Nishio et al. (2020) which includes rotation within the range [-15,15], translating image within range 15-30%, scaling within the range [0.85,1.15] and shear within 85-115%. The pseudo-code for this augmentation is as follows:

  ”RandRotation,"[-15,15],...
  "RandTranslation,"[0.15,0.3],...
  "RandScale,"[0.85,0.3],...
  "RandShear,"[0.85,1.15]

- **Combination 3:**This augmentation method contains combinations of random rotation within range[-90,90], translating the image within 15-30%, shearing it around 85-115%. All together, these methods can be summarised under pytorch's function Random Affine. The pseudo-code for this augmentation is as follows:

  ”RandRotation,"[-90,90],...
  "RandTranslation,"[0.15,0.3],...
  "RandShear,"[0.85,1.15]

- **Combination 4:** This augmentation step includes only flipping of image horizontally with a probability of 40%. The pseudo code for this augmentation looks like:

  "RandHFlip,"[p=0.4]

- **Combination 5:**This augmentation step contains a combination of rotating the image within the range of[-30,30] and flipping with a probability of 50% and it is adapted from Xin Tie et al.. The pseudo-code for this augmentation is as follows:

  "RandRotation,"[-30,30],...
  "RandYReflection,"[p=0.5]

- **Combination 6:**This augmentation step shows variations of color values in terms of HSV and brightness values ie. the combination includes brightness, contrast, saturation, and hue with values of 1.0,0.5,1.0 and 0.1 respectively which all together can be comprised under pytorch's Color Jitter function. The pseudo-code for this augmentation is as follows:

  "RandBrightness,",[1.0],...

"RandContrast,",[0.5],...
"RandSaturation,",[1.0],...
"RandHue,",[0.1]

4.3.3 Results

The Transfer Learning Model as described in section 4.2.1 is now fed with different image combinations. This model is trained with a maximum of 100 epochs and with a dropout and learning rate of 0.1 and 0.01 respectively. Moreover, we have also introduced a patience rate while model training which is taken as 4. In spite of the fact that the maximum number of training epochs is 100, the patience level acts as a setting for early stopping. This ensures the model training to stop if there is no improvement in the models performance over 4 epochs on the validation set. Therefore, the trained epochs here justify how long our model has actually been trained over epochs.
Table 6 shows our mode‘s performance in several combinations. From this, we can deduce that our model performed well on combination 4 having a validation accuracy of 98.88% which is slightly better than other combination results. In contrast, when compared with test accuracy the model with combination 5 has the highest accuracy of 91.7%. However, we refer to validation accuracy for model evaluation.

	Trained Epochs	Dropout	Learning Rate	Total Train Time (mins)	Train Loss	Train Acc	Validation Loss	Validation Acc	Test Acc
1	6	0.1	0.001	21.14	0.0831	0.9719	0.05840	0.98621	0.902
2	7	0.1	0.01	24.89	0.0887	0.9716	0.0665981	0.9797	0.8797
3	17	0.1	0.001	71.71	0.07037	0.9733	0.067952	0.977941	0.875
4	17	0.1	0.01	60.01	0.01826	0.9931	0.060422	0.988051	0.905
5	10	0.1	0.01	36.44	0.04833	0.9823	0.0734206	0.982536	0.917
6	16	0.1	0.01	54.54	0.06760	0.9761	0.0701107	0.981617	0.886

Table 6: Results after Model Training

4.3.4 Comparison

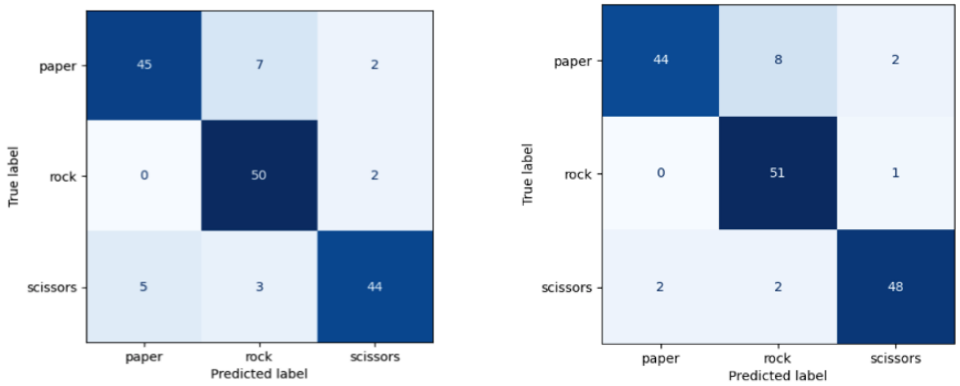
In this section, a comparison of the Transfer Learning Model with and without exposure to combination techniques is observed. In figure 15a, we can observe that the model has significant train and validation accuracy of around 99% and 98% respectively (also refer Table 5). On the other hand, in figure 15b, we can see that the best combination method with the Transfer Learning Model has similar train accuracy to the method when we don‘t apply any combinations. However, the validation accuracy differs a bit as combination 4 resulted in an accuracy value of approximately 99% which is around 1% more than when we don‘t use combinations with transfer learning (also refer Table 5 and Table 6).
The confusion matrices in Figure 8 provide an overview of the performance of the models with combination and without combination on the test set. When calculated, the accuracy of the best Transfer Learning Model without combination resulted in around 88% while after introducing the combination methods, the accuracy jumped up by approx. 3% proving that the introduction of data augmentation methods with some combinations can enhance the models' performance.

4.4 Model Evaluation (Philipp)

In this section, methods of model evaluation are explored. First, the performance measurements chosen for this project are discussed. Afterward, the question if "local interpretable model-agnostic explanations (LIME) can make CNN models for classifying rock, paper, scissors images more explainable" is explored. After a short introduction to LIME, the results of the LIME implementation are visualized and discussed.

4.4.1 Performance Measurements

After completing the model and data engineering part, multiple models are available for the classification of hand images displaying rock, paper, and scissors gestures. The next step involves comparing



(a) Transfer Learning without Data Augmentation

(b) Transfer Learning with Data Augmentation

Figure 8: Confusion Matrix comparison on the basis of Test Accuracy

these models to select the most suitable one. In this chapter, the performance measurements used in this project are explained, and the comparison of each model (after data enrichment/model engineering) is performed in the respective chapters. The analysis conducted during the data engineering phase has revealed that the data distribution is almost balanced. Therefore, there is no requirement for performance measures meant for unbalanced datasets, such as weighted or micro-averaging measurements.
Additionally, the task of classifying rock, paper, and scissors has no need to avoidance of specific model classifications. In contrast in medical environments, this avoidance is frequently required as an incorrect diagnosis, such as a false-negative cancer diagnosis, can have fatal consequences for a patient, requiring a high sensitivity. (Campanella et al., 2019)
We tested the following Measurements for different models (the equation of each measurement is adopted from (Sokolova and Lapalme, 2009)).
Average Accuracy

$$\frac{\sum_{i=1}^{l} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{l} \tag{2}$$

Macro averaging recall

$$Recall_M \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l} \tag{3}$$

Macro averaging precision

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fn_i}}{l} \tag{4}$$

Macro-averaging Fscore

$$Fscore_M = \frac{2 * Precision_M * Recall_M}{Precision_M + Recall_M} \tag{5}$$

In this project the average accuracy in combination with the cross-entropy loss, as already stated in the model engineering part (see Figure 1), will be used as measurements. The consideration of the loss gives an indication about the level of overfitting during training and when choosing between models with similar accuracy a higher loss can have a benefit on the generalizability of the model. Besides the above-stated arguments for average accuracy, the main reason for the selection is the more intuitive interpretation of the measurement as opposed to other more advanced performance measurements like the F1-Score. An additional useful approach for evaluating the models' performance was the examination of the confusion matrices. This allowed for gaining insight into the models' generalizability across all classes, as well as identifying common errors made by the models. For instance, in the final model (see 8), 8 paper images from the test dataset were misclassified as rock images. In this project, the selection of models is based both on the cross-entropy loss and the average accuracy.

4.4.2 Local Interpretable Model-Agnostic Explanations

Following the question, if local interpretable model-agnostic explanations (LIME) can make CNN models for rock, paper, and scissors more explainable, is discussed. Users often don't trust the predictions of a machine learning model or the machine learning model itself, mainly because the model is seen as a black box. (Ribeiro et al.) LIME is an algorithm that can explain the predictions of any classifier or regressor in a faithful way, by approximating it locally with an interpretable model. (Ribeiro et al.) By "explaining a prediction", we mean presenting textual or visual artifacts that provide a qualitative understanding of the relationship between the instance's components (e.g. words in texts, patches in an images) and the model's prediction. (Ribeiro et al.) LIME first divides a given image into superpixels. A distribution of perturbed images is then generated by randomly hiding some superpixels. (Palatnik de Sousa et al., 2019) After running the pertubated images through the prediction method of our model the perturbed images and probabilities were then presented to a linear regression model that estimated how each superpixel contributed to the overall prediction. In other words, the perturbed images formed a vicinity around the original image where the linear model (the regression) learned to approximate the non-linear model (the CNN). (Palatnik de Sousa et al., 2019) In this project the original LIME Implementation (https://lime-ml.readthedocs.io/en/latest/lime.html) is used.

First To employ the LIME technique for an explanation of model predictions, it was necessary to preprocess the image in the same manner as the model was trained. Next, a LIME Explainer Instance was generated, where the number of pertubated images to be generated (1000) and the model prediction method were specified. The standard SK-Image Quickshift Algorithm was used as a segmentation method, and the standard cosine distance was used as the distance metric. The get_image_and_mask method was utilized to generate the mask for visualizing the LIME explainer. Due to the high probabilities for nearly all cases only positive contributions to the classification were considered and three superpixels are included in our explanation.

4.4.3 Results

In figure 9 is an example of the output of LIME illustrated. See that for rock the model decides on 2 Fingers close together and for paper single parts of different fingers are used for the classification As

(a) Model Prediction: Paper with 0.999 Probability

(b) Model Prediction: Rock with 0.972 Probability

Figure 9: LIME on a Rock and Paper hand gesture image

already noted the model tends to misclassify paper images as rock. In Figure 11 two cases of this misclassification are illustrated. In both cases, one of the decision boundaries contains more than one finger. This may indicate that our model over generalizes for Rock Predictions. Furthermore, a case is presented where the prediction of the model matches the label of the image, however, the decision boundary provided by LIME showed us that the model was unable to detect the hand in the image and classified based on a face in the background(see figure 10). These cases of "wrong classification" are not detectable with classic methods mentioned above, due to the matching prediction and ground truth label.

Figure 10: Model Prediction: Rock with 0.998 Probability

(a) Model Prediction: Rock with 0.992 Probability

(b) Model Prediction: Rock with 0.999 Probability

Figure 11: LIME on two misclassified paper images

Moreover, the present study compared LIME with Saliency Maps, which are heat maps that represent the gradients of each pixel for a given image prediction (Simonyan et al.). As depicted in Figure 12, a Saliency Map was applied to an accurately classified image of a scissor. The resulting heat map highlighted the scissor's shape and showed a higher density of high gradients in the fingertips. However, in most cases, the Saliency Map visualizations were not particularly informative, and noticeable patterns or edges could not be identified in the heat maps. Consequently, the LIME method was deemed more appropriate for the purposes of this study.

While LIME is a powerful tool to visualize which parts of an image the Deep Learning Model uses for the classification of an image, it is restricted to single images. We can make single predictions of our model more explainable, but to make the model more explainable we have to randomly select test images and let them run through LIME. Overall LIME was mainly used to test random samples or random missclassified samples to understand what part of the image led to the classification of the model. In the most cases except figure 11 the model decided on parts of the hands or fingers, this realization could raise the trust of a user in the model. In summary, LIME has made the model more explainable with individual examples and the implementation of LIME in the final product (e.g. an app) could improve the trust of the user in the model decisions.

5 Discussion (Philipp)

This work can give a good overview of some best practices and also methods that should not be pursued further in the field of image classification with convolutional neural networks. As suspected the outcomes of this project show a clear advantage of transfer learning and using established

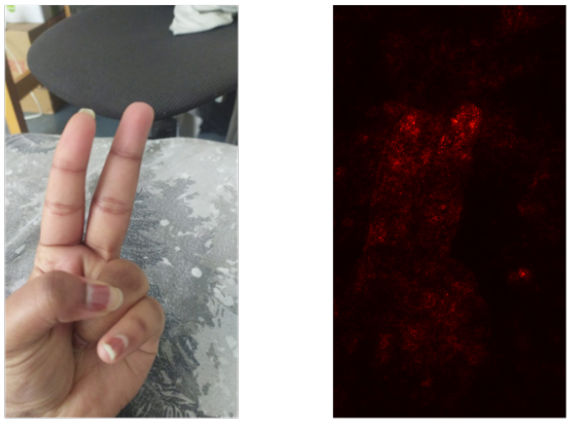


Figure 12: Saliency Map on a scissors image

architectures for a short-span project. Furthermore, we could see an improvement in performance by using data augmentation. Finally, we used LIME to make our black-box model more explainable.

This work was limited by the hardware of our project members, as we were not able to compare higher Batch sizes for our Baseline model, a comparison of a baseline model trained with batch size 64 with a transfer learning model with batch size 64 would allow a better comparison between the methods. Also time was a limited resource as the project took place within about 4 months, so no back iteration in the CRISP-ML circle was conducted. Possible next steps to further increase the performance of the model are for example to raise the amount of noisy training data for Paper/Rock images to tackle the issue presented in the Model Evaluation part. This are interesting topics for further research and projects in this field. Also other methods which are not compared or implemented in this project could have a benefit on the performance of the model.

For future projects in the field of image classification, we would recommend a more iterative process with less focus on the model engineering part, an established architecture works well, and a focus on model evaluation and data engineering part, to ensure that the model can generalize on unseen noisy images.

6 Conclusion (Satyam)

We conclude that the demand for training datasets is continuously increasing as deep learning develops. Therefore, we need to rely on image augmentation techniques such as rotation, flipping, shifting, etc. to generate new data in order to overcome the shortage of data. Moreover, using augmentation techniques in combination can result in better model performance. However, selecting the right parameter value with the right augmentation methods is always a key factor for performing data augmentation Yang et al. (2022). Furthermore, model selection is also crucial, and we used EfficientNetV2-S as a transfer learning model since it reduced training time while still maintaining high accuracy when compared with the baseline model. However, there were some misclassifications, so LIME was introduced to address them. LIME provided the model-agnostic explanations by demonstrating the superpixles that contributed to the prediction. Additionally, Saliency maps were also used and applied to some accurately classified images. Although the heat maps showed a higher density around the area of interest, sometimes it had trouble noticing patterns and edges. Eventually, multiple methods for measuring model performance were studied but then average accuracy in combination with cross-entropy loss was considered for all of the measurements.

References

Rock paper scissors, 2023. URL https://en.wikipedia.org/w/index.php?title=Rock_paper_scissors&oldid=1139717311.

G. Campanella, M. G. Hanna, L. Geneslaw, A. Miraflor, V. Werneck Krauss Silva, K. J. Busam, E. Brogi, V. E. Reuter, D. S. Klimstra, and T. J. Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature Medicine*, 25(8):1301–1309, 2019. ISSN 1078-8956. doi: 10.1038/s41591-019-0508-1.

M. Elgendi, M. U. Nasir, Q. Tang, D. Smith, J.-P. Grenier, C. Batte, B. Spieler, W. D. Leslie, C. Menon, R. R. Fletcher, et al. The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective. *Frontiers in Medicine*, 8:629134, 2021.

L. Gonzales. A Look at MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://medium.com/@luis_gonzales/a-look-at-mobilenetv2-inverted-residuals-and-linear-bottlenecks-d49f85c12423s, 2019. Accessed: 2022-12-27.

C. Khosla and B. S. Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 79–85. IEEE, 2020.

P. D. C. Ledig. Deep learning. Lecture 8,Slide 5-6, 2022-2023.

C. Lei, B. Hu, D. Wang, S. Zhang, and Z. Chen. A preliminary study on data augmentation of deep learning for image classification. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, pages 1–6, 2019.

K. Maharana, S. Mondal, and B. Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 2022.

S. Maji and A. Nath. Scope and issues in alpha compositing technology. 2014.

M. Nishio, S. Noguchi, H. Matsuo, and T. Murakami. Automatic classification between covid-19 pneumonia, non-covid-19 pneumonia, and the healthy on chest x-ray image: combination of data augmentation methods. *Scientific reports*, 10(1):17532, 2020.

I. Palatnik de Sousa, M. Maria Bernardes Rebuzzi Vellasco, and E. Da Costa Silva. Local interpretable model-agnostic explanations for classification of lymph node metastases. *Sensors*, 19(13):2969, 2019. doi: 10.3390/s19132969.

P. Pawara, E. Okafor, L. Schomaker, and M. Wiering. Data augmentation for plant classification. In *Advanced Concepts for Intelligent Vision Systems: 18th International Conference, ACIVS 2017, Antwerp, Belgium, September 18-21, 2017, Proceedings 18*, pages 615–626. Springer, 2017.

H. Ravishankar, P. Sudhakar, R. Venkataramani, S. Thiruvenkadam, P. Annangi, N. Babu, and V. Vaidya. Understanding the mechanisms of deep transfer learning for medical images. *CoRR*, abs/1704.06040, 2017. URL http://arxiv.org/abs/1704.06040.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. URL http://arxiv.org/pdf/1602.04938v3.

S. Ruder. An overview of gradient descent optimization algorithms, 2016. URL https://arxiv.org/abs/1609.04747.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. doi: 10.48550/ARXIV.1801.04381. URL https://arxiv.org/abs/1801.04381.

Saumitra Mishra et al. *Local interpretable model-agnostic explanations for music content analysis*. 2017. URL https://archives.ismir.net/ismir2017/paper/000216.pdf.

M. Sharma, M. Holmes, J. Santamaría, A. Irani, C. Jr, and A. Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. pages 1041–1046, 01 2007.

C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. URL http://arxiv.org/pdf/1312.6034v2.

M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. ISSN 03064573. doi: 10.1016/j.ipm.2009.03.002.

S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Mueller. Towards crisp-ml(q): A machine learning process model with quality assurance methodology. URL http://arxiv.org/pdf/2003.05155v2.

M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL http://arxiv.org/abs/1905.11946.

M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021. URL https://arxiv.org/abs/2104.00298.

B. Xin Tie, C. Zhang, T. K. Song, J. D. Nadig, and M. L. Schiebler. Diagnosis of covid-19 pneumonia using chest radiography: Value of artificial intelligence.

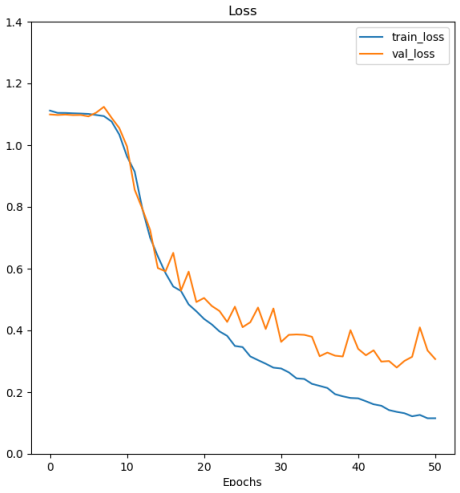
S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen. Image data augmentation for deep learning: A survey. *arXiv preprint arXiv:2204.08610*, 2022.

Declaration of Authorship

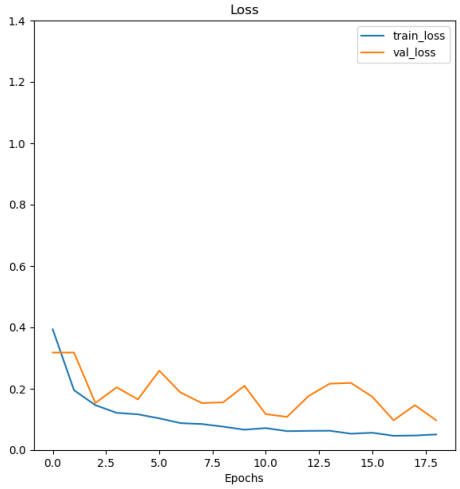
Ich erkläre hiermit gemäß § 9 Abs. 12 APO, dass ich die vorstehende Projektarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Projektarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Bamberg, February 28, 2023	
(Place, Date)	(Signature)
Bamberg, February 28, 2023	
(Place, Date)	(Signature)
Bamberg, February 28, 2023	
(Place, Date)	(Signature)

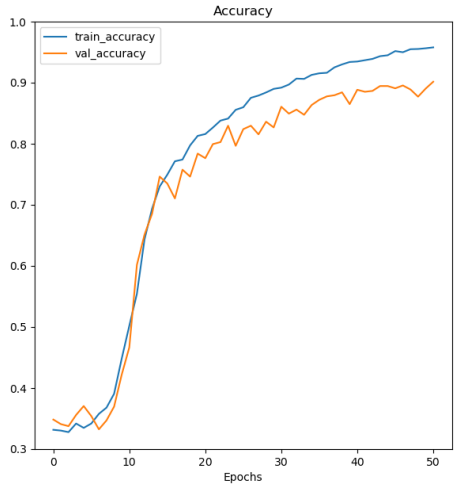
A Appendix



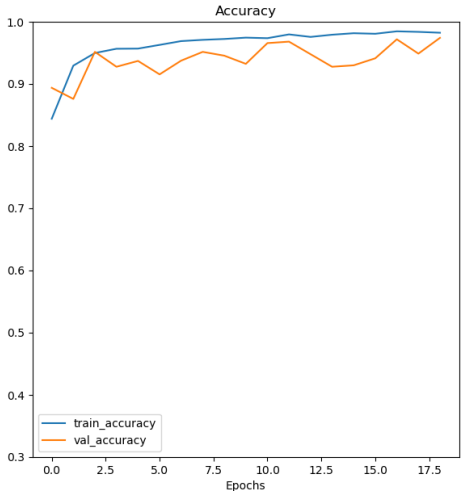
(a) Baseline Model (loss)



(b) Transfer Learning Model (loss)



(c) Baseline Model (accuracy)



(d) Transfer Learning Model (accuracy)

Figure 13: Loss and accuracy comparison on the training (blue) and validation (orange) set

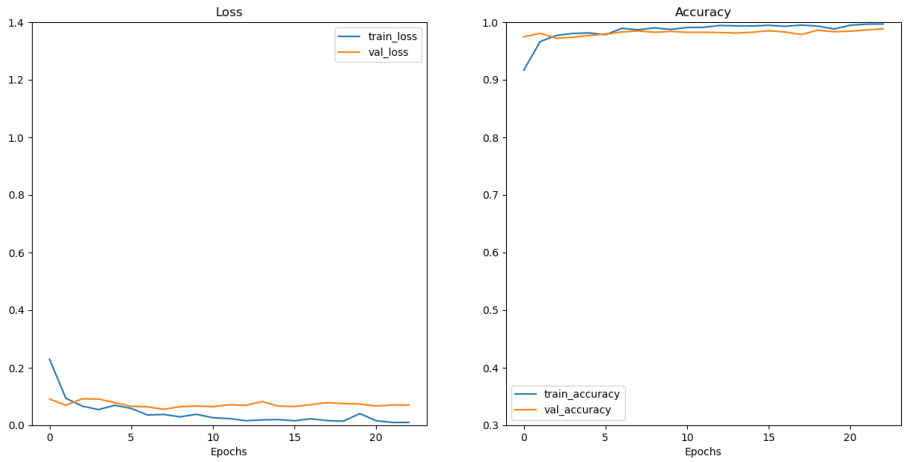
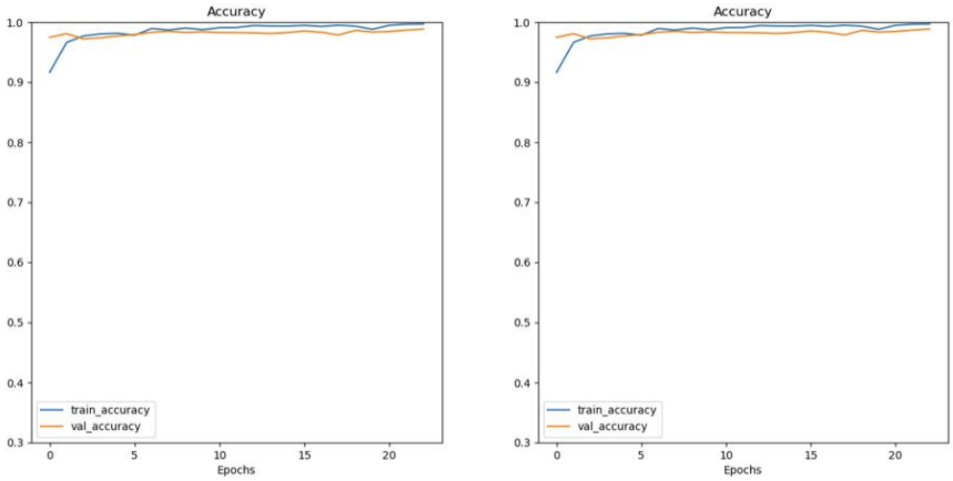


Figure 14: Train and validation loss and accuracy of the Transfer Learning Model trained with a batch size of 64



(a) Transfer Learning without Data Augmentation

(b) Transfer Learning with Data Augmentation

Figure 15: Comparison on the basis of Validation Accuracy