ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Faculty of Engineering, Computer Science and Psychology
Institute of Neuroinformatics

Feedback and Forward networks for object recognition on the data with different noise types

Project work report at Ulm University.

Submitted for the degree of Master of Science (MSc) in Artificial Intelligence.

Submitted by:

Tatsiana Mazouka

tatsiana.mazouka@uni-ulm.de

Ulm, August 2023

Version February 20, 2023

Contents

1 Introduction

In this student project work, the research was conducted to answer the following question: "*Are feedback Deep Neural Network (DNN) more robust than forward DNNs when trained on one noise type and tested on another? If so, which feedback type performs best?*" The implementation for the study can be found in the Git repository [1].

VGG16 neural network was examined to find an answer to the given question. The model was trained on data without noise and on data with Gaussian noise. Also, the model was expanded with different kinds of feedback connections. To analyze performance, accuracy on the test split with different noise types was considered. The McNemar test [2] was applied to prove the statistical significance of the models' differences.

1.1 Problem statement

The problem statement was described in the paper about "Generalisation in humans and deep neural networks" [3]. According to the authors, the human visual system is more robust to almost all of the tested image manipulations in comparison to the current state-of-the-art object recognition models. In figure 1 the applied image augmentations are illustrated.

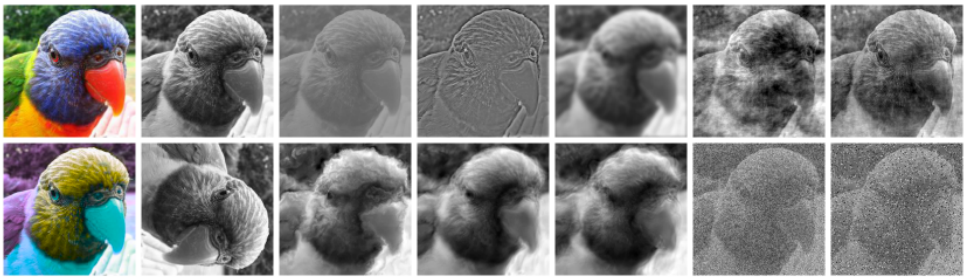


Figure 1: Image manipulations applied in the research about Generalisation in humans and deep neural networks. [3].

In the research, object classification models were trained on the data with one type of noise and tested on the data with another type of noise. For example, a model is trained on the images with uniform noise. It recognizes well the objects in the images with uniform noise but its test accuracy decreases in the images with Gaussian noise. However, for humans, it is not a problem to recognize the object on the same image with different noise types. Also, human eyes can barely recognize the difference in the images with some noise types as e.x. Salt-and-pepper noise and Gaussian noise. But the same image distortion plays a big difference in the training of machine learning models.

In figure 2 the accuracies of the models' evaluations on the data with different noise types are illustrated. In the research, the common state-of-the-art computer vision models are considered, which are trained on the data without noise. The figure shows, that the human visual system outperforms machine learning models on all architectures. Therefore, the conclusion can be derived, that the generalization performance of current computer vision models is worse than that of the human visual system.

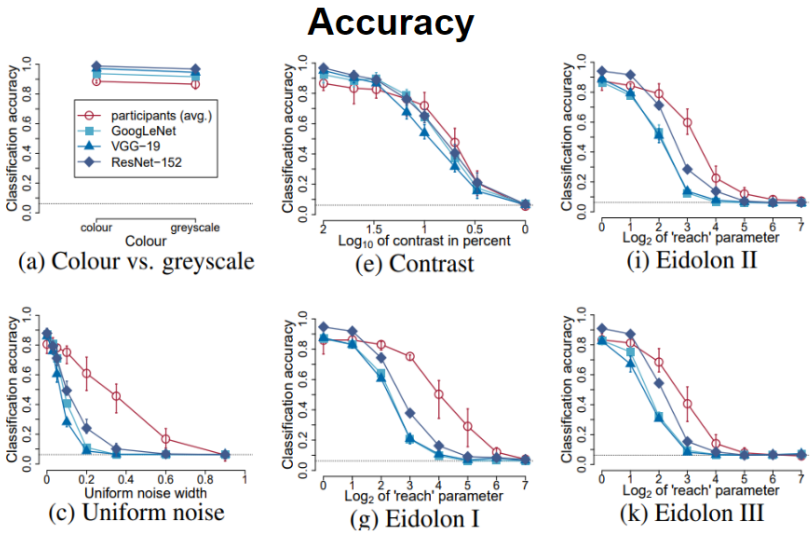


Figure 2: Evaluation performance of the computer vision models and participants of the research on the data with different noise types. [3].

According to the research, greater emphasis should be placed on the model's generalization performance. The challenge is to train or improve the models so, that they could generalize better and learn some abstract patterns from the image instead overfit on the data without noise. [3]

1.2 Solution with Feedback networks

In the paper about "Incorporating Feedback in Convolutional Neural Network (CNN) " [4] is proposed to build the Feedback connections to the CNN to improve models' stability and accuracy.

The purpose of the considered in the study models is to detect digits in images. In figure 3, the models' architecture is illustrated. The forward model consists of three convolutional layers. The Feedback model improves the architecture of the Forward model via three repeated Feedback steps, that create more convolutions between the first input and the output. The functional size of the neurons' receptive field is enhanced after each step. More information is sent to the first layer in this manner.

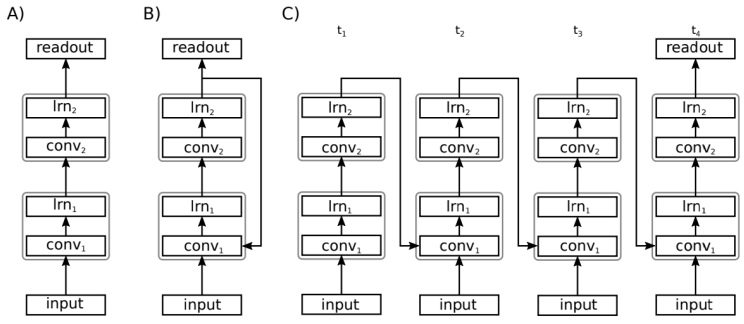
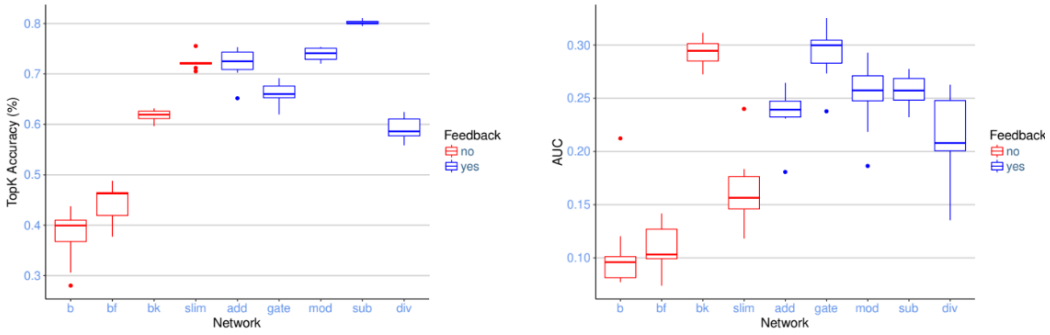


Figure 3: Forward and Feedback network architectures from the solution statement paper [4]. The image with label C represents the unrolled version of the Feedback network.

Furthermore, various functions were investigated in the study to link the Feedback connections with previous layers: addition, division, subtraction, gating, and modulo operation.

In figure 4 the experimental results are illustrated, which show better performance of Feedback models than of Forward models. The graphic also demonstrates that the Feedback models are more resistant to data containing white noise.



Performance on the test set

Robustness against white noise

Figure 4: Evaluation results of the Feedback network and Forward network on the test dataset and dataset with white noise. [4].

In the current project work, different Feedback models are studied to improve the performance of the VGG16 network on the data with different noise types. Therefore, different experiments were conducted to solve the problem described in the section 1.1 using Feedback connections in CNN. The inspiration for various experiments was taken from the discussed solution paper. [4]

2 Prepare dataset

ImageNet dataset is used for the experimental procedure. It consists of over 14 million images, that are related to 1000 classes. Training on the entire dataset is a time-consuming and resource-intensive approach. For the research procedure, it is more practical to use just part of the dataset. It helps to confirm assumptions easier and reduces the training complexity.

The dataset preparation from the problem statement paper [3] was adapted for the experimental procedure. According to it, the ImageNet dataset was mapped to 16 categories [5]. The mapping information can be found in the project code from the paper “Generalisation in humans and deep neural networks” [3] on GitHub.

The models in the current project work were built using the TensorFlow [6] and Keras [7] software libraries. The desired dataset’s format is Tensorflow Dataset Prefetch, that parallelizes data preparation and training procedure.

2.1 Load ImageNet dataset

To work with the ImageNet dataset, firstly it should be uploaded locally. On the official ImageNet website two following files should be downloaded “ILSVRC2012_img_train.tar” and “ILSVRC2012_img_val.tar”.

The archives contain the train images, validation images, and their labels. The Tensorflow dataset can be prepared using them. For the project work the same version of the ImageNet dataset was used as in the paper with the problem statement. The implementation for data loading was taken from the article of Pascal Janetzky in Toward Data Science journal [8] and can be found in the listing 1.

The experimental procedure employs three datasets: train set, validation set, and test set. The validation set has initially 50000 samples. 50000 samples of the training set were taken to create the test dataset. The original test dataset from ImageNet does not have labels, because the dataset was used for competition. That is why the test dataset is created from the part of the training dataset. Finally, the training set contains 1231167 samples.

2.2 Preprocess dataset

The following steps are applied for dataset preparation:

- Normalize image colors by dividing them throw 255.
- Resize image to (224, 224).
- Change labels according to the mapping file.
- Remove images, that do not belong to the defined categories.
- Normalize the number of samples for every labeling class.

In this notebook, the implementation to preprocess the dataset without the normalization step can be found. The normalization is performed in the following notebooks. [1]

2.3 Normalize distribution of labels

The first training on the prepared dataset showed accuracy equal to 50%, which was learned during the first couple of epochs and does not improve further. The analysis of the dataset's distribution showed, that the biggest amount of samples belong to label 15. In figure 5, the distribution of labels after the first preprocessing procedure is illustrated.

Therefore, the assumption was derived that the model learns to predict just this one label. The assumption was proved because after the labels' amount was normalized, test and validation accuracies improved significantly. Normalization was performed throw setting the maximum number of samples for each label to the specified threshold. For validation and test datasets, the threshold is set to 500. For the training dataset, the threshold is set to 10000.

It is important to show the model a lot of different samples so that it could learn to predict different patterns. That is why the number of samples with frequent labels was reduced. In figure 6, the final distribution of labels in different data splits is illustrated.

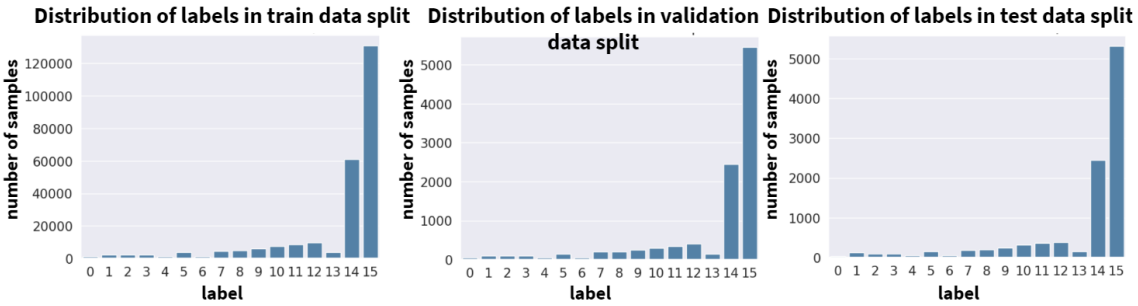


Figure 5: Distribution of labels in train, validation, and test data splits before normalization of labels' distribution.

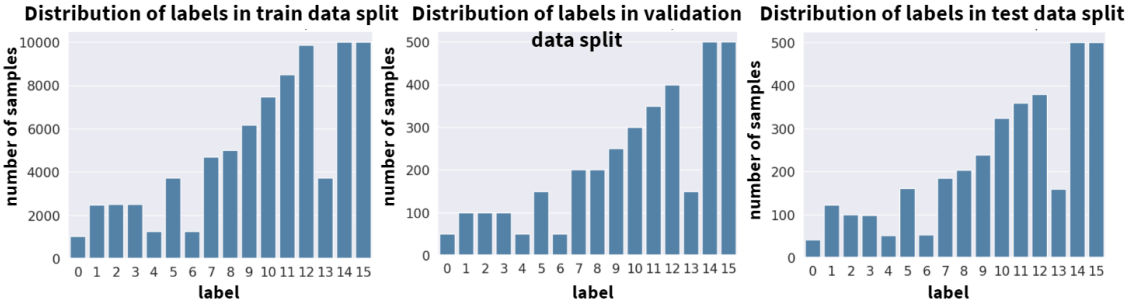


Figure 6: Distribution of labels in train, validation, and test data splits after all preprocessing steps. The presented dataset is used for all experiments.

2.4 Noise types

Every model was evaluated on the test dataset without noise, with Salt-and-pepper noise, and with Gaussian additive noise. The functions for the creation of both noise types are presented in the listings 3 and 2.

To apply the Gaussian additive noise to an image, first, the Gaussian normal distribution with mean 0 and variance 0,1 is created using a random function for every row, column, and color filter of the image. The calculated matrix is added to the original image.

The Salt-and-pepper noise is created via the setting of white and black points in the random position in the image. The frequency of the insertions can be controlled using the padding parameter.

Figure 7 illustrates the example of an image with different noise types.

3 Experimental methods

The described solution proposal in the paper "Incorporating Feedback in CNN" [4] gave great inspiration for different experiments to answer the research question. The tested approaches are given in this section.

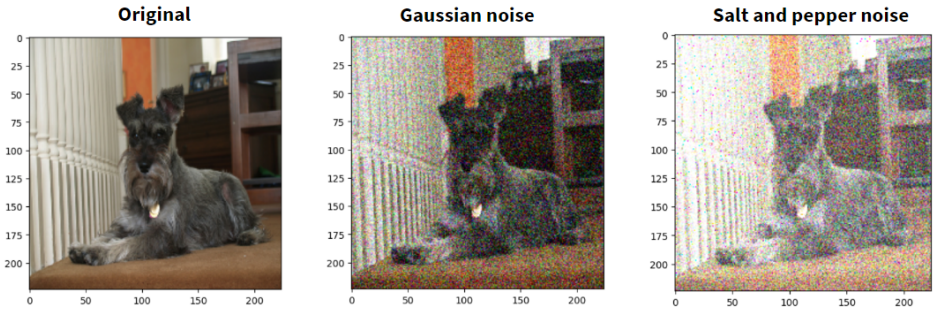


Figure 7: Two noise types are applied to the image: additive Gaussian noise and Salt-and-pepper noise. The models in the project work are evaluated on the test dataset without noise and with illustrated distortion types.

3.1 Model types

VGG16 [9] model was chosen as a baseline to study Feedback networks. The model's variations were trained on data without noise and on data with Gaussian noise. The main experiments were conducted on the model with frozen layers using Transfer Learning.

Experiments were also carried out on the model without frozen layers. In the study, two main types of models have been compared: the Forward model and the Feedback model. The Feedback Model takes over the architecture of the corresponding Forward model and adds Feedback connections to it. So, the compared Feedback and Forward models have the same frozen layers.

The smaller dataset is considered during this project work, than the original ImageNet dataset, on which the VGG16 is trained initially. The subset of Imagenet can be not sufficient to train the whole model. That is why the penultimate layer of the VGG16 network is replaced with the fully-connected dense layer with 256 neurons. The last layer is replaced with the fully-connected layer with 16 neurons to fit the number of classes in dataset.

3.2 Transfer Learning

VGG16 is a huge neural network that requires a large amount of data to train. The optimization process may freeze and fail to discover a reasonable solution if there is not enough data provided. That is why mostly fine-tuned variations of VGG16 [9] model were studied.

For frozen models, weights were pre-trained on the whole ImageNet dataset. It was decided to run the experiments with Transfer learning because it has the following advantages, that should be investigated:

- Efficient training procedure.
- Better performance in some cases.
- Requiring fewer data.

The experiments were conducted with the models with different amounts of frozen layers. The variation of frozen layers helps to test the following assumption: If just a couple of the first layers

are frozen, the model has a possibility to learn a more sophisticated function since more layers are left for training. Or the model may show worse performance because VGG16 [9] with a couple of frozen layers can be too complex for training with the prepared dataset for this research.

3.3 Feedback VGG16 model

The following parameters in the Feedback models can be changed to study different versions of it:

- Blocks for Feedback connection.
- Number of projection convolutions.
- Number of Feedback steps.
- Number of frozen Layers.
- Feedback type.

The number of feedback steps was set to 3 and the additive Feedback was chosen for all experiments. Other parameters were variated as part of the project work.

In figure 8 is presented the Frozen Feedback VGG16 [9] model with the Feedback connection from block 4 to block 1. To perform such a projection, a large upsampling was applied. Therefore, a lot of information could be lost and the following convolution size is not enough to learn a complicated function. To fix this artifact, three convolution layers are applied after each other. The implementation of this Feedback Model can be found in listing 4. Feedback models are compared with Forward models, which have the same frozen layers. For example, this Feedback model is compared with the Forward model, which has frozen weights in 1,2,3, and 4 blocks. It is illustrated in figure 9.

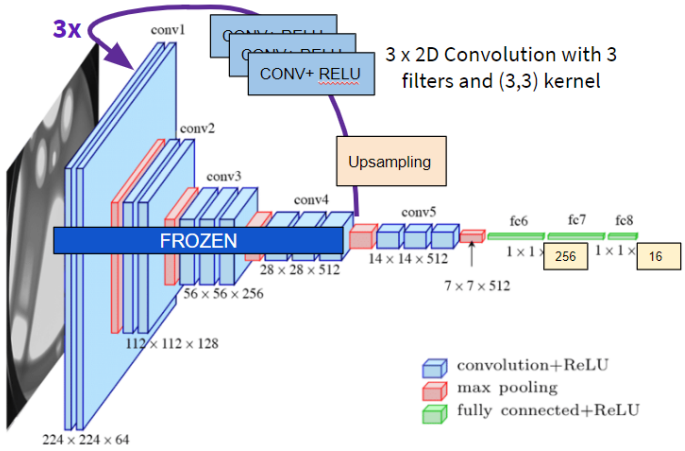


Figure 8: Frozen Feedback VGG16 [9] model with the additive Feedback connection from block 4 to block 1.

Other tested types of Feedback networks have just one convolution layer because they have smaller upsampling and more filters inside the projection, which are able to build enough sophisticated functions. Figure 10 presents the example of the Feedback VGG16 [9] model with the Feedback

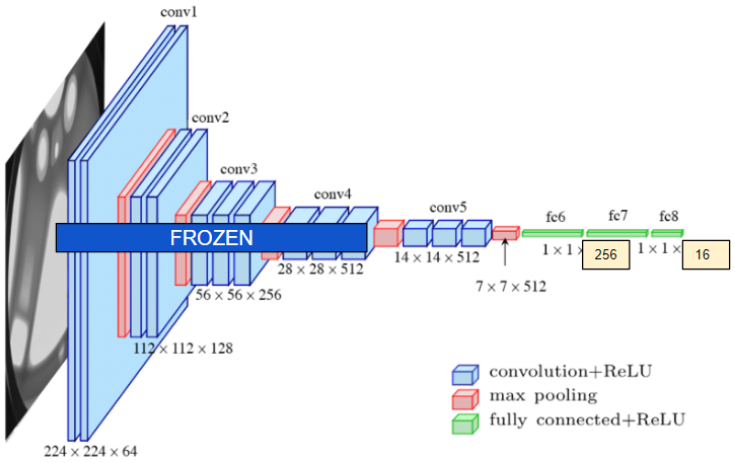


Figure 9: Frozen Forward VGG16 [9] model with not frozen 5th block.

connection from block 5 to block 3. Its projection convolution has 128 filters and therefore just one convolution layer was applied.

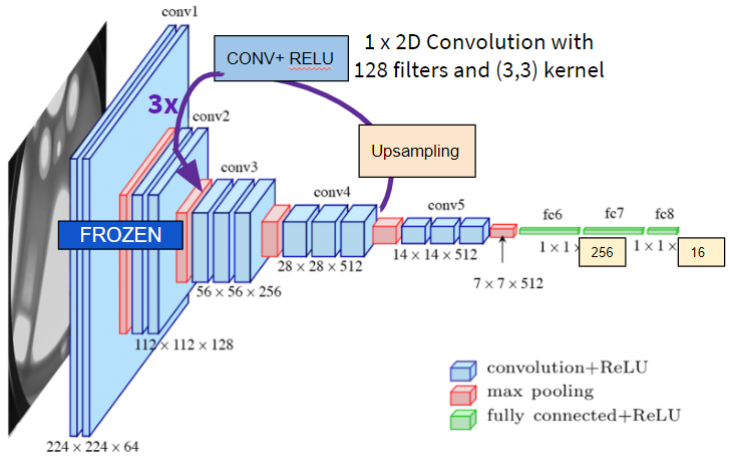


Figure 10: Frozen Feedback VGG16 [9] model with the additive Feedback connection from block 5 to block 3.

3.4 Evaluation methods

Models were compared with each other based on the test accuracy of the dataset without noise, the dataset with Gaussian noise, and the dataset with Salt-and-pepper noise. The goal was to analyze the difference in the accuracies, that were calculated on data with different noise types.

If the accuracy metrics do not differ a lot, McNemar's test is applied to examine, whether the models' difference is statistically significant or not.

McNemar's test consists of the following steps:

1. Contingency table is created, that displays how many predictions of the models are correct or wrong. In figure 11 the example of a contingency matrix is displayed.

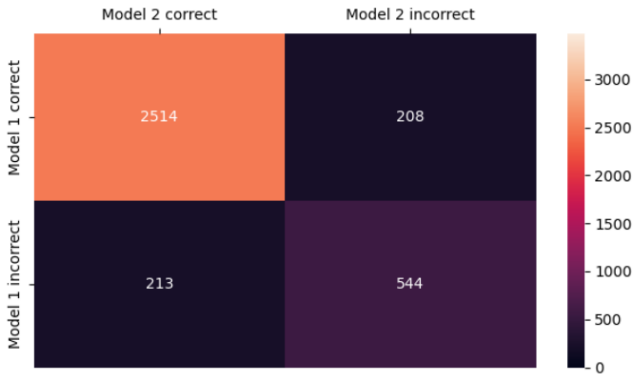


Figure 11: Example of the contingency table that is used by McNemar's test to prove or reject significant difference between two models.

2. The test statistic is performed to prove or reject the hypothesis if the models have the same proportion of errors. This characteristic defines the significant difference in the predictions of the models. The formula for test statistics is following:

$$statistic = \frac{(\#model1\_correct\_model2\_incorrect - \#model1\_incorrect\_model2\_correct)^2}{\#model1\_correct\_model2\_incorrect + \#model1\_incorrect\_model2\_correct} \quad (1)$$

The statistic considers the number of samples, where one of the models performs correctly and another one wrong. For calculation, each cell in the contingency table should have at least 25 values.

3. To reject or prove the hypothesis the value α is set to 0,05. If the p value is higher than α, the hypothesis is rejected. It means, that the models have a different proportion of errors and therefore have a statistically significant difference in predictions. [10]

3.5 Training procedure

The models are optimized using the Stochastic gradient descent with a learning rate of 0,001. The first experiments were conducted with the Adam Optimisation algorithm, but this optimization method works just for the Frozen Forward VGG16 model. For all other models, the optimization technique failed to find the optimum minimum. This conclusion was derived after analyzing the changes in training and validation accuracies after each epoch, which is illustrated in figure 12. The accuracy fluctuates around a very small number and does not improve. This problem was resolved by switching to a Stochastic gradient descent optimizer. Also, For training was taken the Sparse categorical cross-entropy loss function.

The training of models with frozen layers was executed for 40 epochs. The whole implementation of the training procedure can be found in listing 5.

The training procedure for not frozen models consists of two steps. Firstly, the frozen model is trained for 10 epochs. Secondly, all layers are unfrozen and the model is trained for the next 20 epochs. Because the VGG16 network is so vast and complex, it was not able to learn the optimum minimum on the used dataset without additional pretraining.

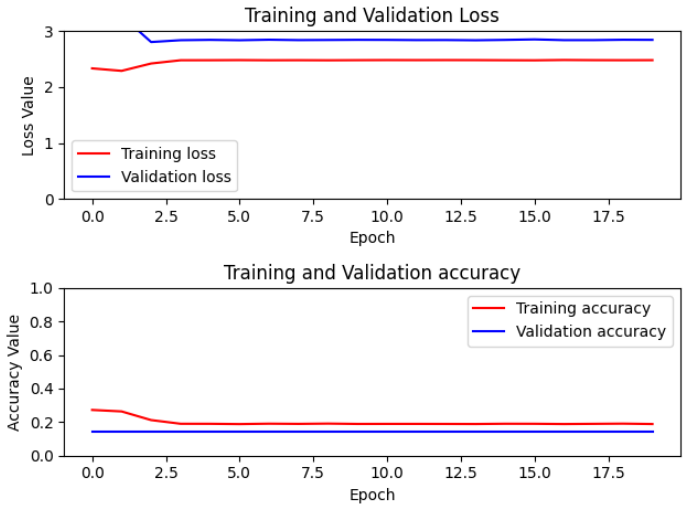


Figure 12: Loss and accuracy of the training procedure of the not frozen VGG16[9] model with the Feedback connection from block 4 to block 1.

The training of models was performed on the bwUniCluster using the Slurm batch jobs on the 4 graphics processing units.

4 Results

4.1 Answer on the research question

The models in the figures 8 and 9 were considered to answer the research question. The displayed architectures are called Feedback and Forward models in this subsection.

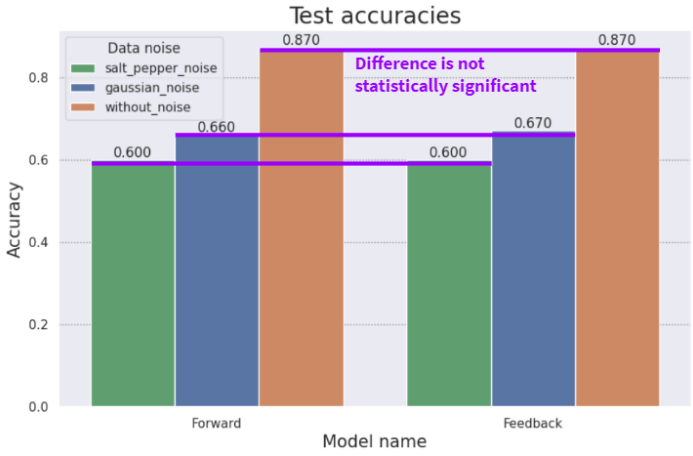


Figure 13: Compare Feedback and Forward models trained on data without noise.

In the first experiment, the Feedback and Forward models are trained on the data without noise. The results are presented in figure 13.

The Violet line defines in this and next figures the accuracies with a not statistically significant difference based on McNemar's test. The experiment showed, that there is no significant difference.

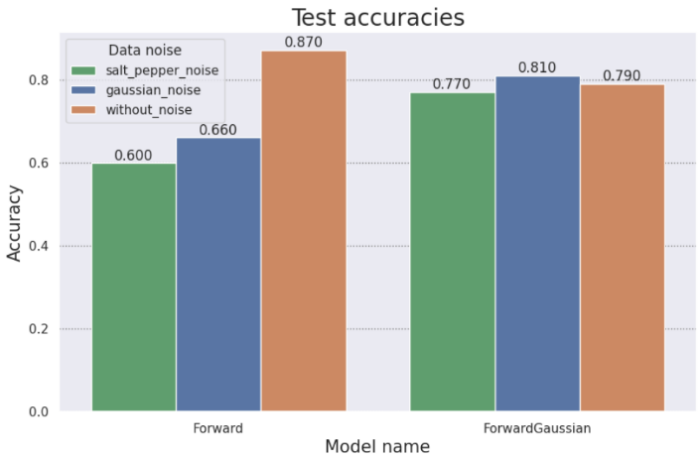


Figure 14: Compare Forward models trained on data without noise and on data with Gaussian noise.

In the next experiment, the Forward model was also trained on the dataset with Gaussian noise and compared with the Forward model, trained on data without noise. The results are presented in figure 14. The experiment showed that the model is more robust to other noise types if it is trained on the data with noise. Although the accuracy became lower, the difference between the accuracies on different data types decreases significantly.

After, the Feedback and Forward models are trained on the data with Gaussian. Figure 15 depicts the results. Although the Feedback connection improved slightly the accuracy on the data with

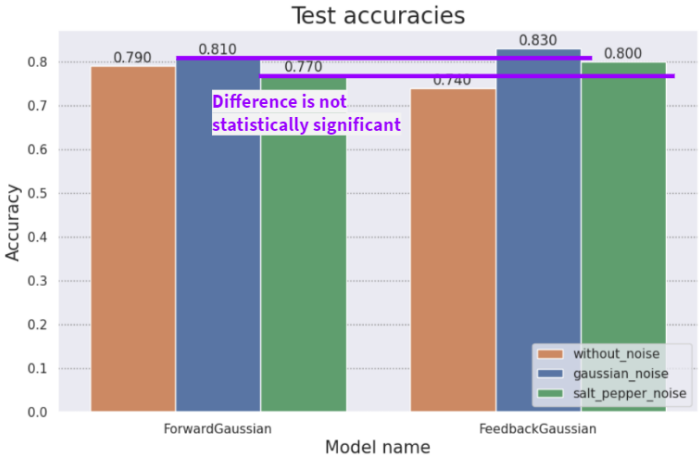


Figure 15: Compare Feedback and Forward models trained on data without noise.

noise, McNemar's test showed that the improvement is not statistically significant. The accuracy on data without noise decreased.

Based on the analyzed graphs, the conclusion can be derived, that the Feedback DNNs are not more robust than Forward DNNs when trained on one noise type and tested on another. The study showed, that the Forward DNNs trained on the data with noise are more robust than the Forward

DNNs trained on the data without noise

The second part of the research question was to find out which type of Feedback connection works best. Therefore, other experiments were conducted to find the answer.

4.2 Compare the number of projection layers

The upsampling is performed inside the Feedback connection, which is why part of the information is lost. The projection function can process the remaining values using convolution layers. The number of layers can be varied and influence the models' training. This assumption is proved by the experiments, described in this subsection. All models in these experiments are trained on data without noise.

First, the number of projection convolution layers is varied for the model presented in figure 9, which has a Feedback connection from block 4 to block 1. The version of the model used in previous experiments has 3 layers. The models with 1,2 and 3 projection layers are compared and the results are illustrated in figure 16. The model with one projection layer was not able to learn and

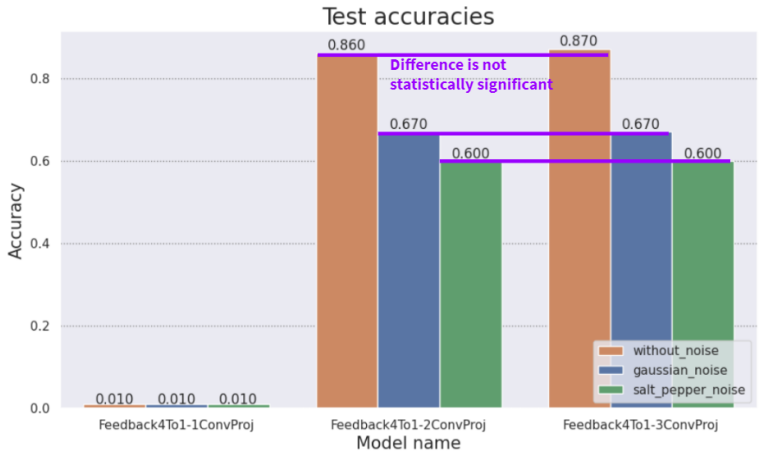


Figure 16: Comparison of Feedback models with 1,2 and 3 convolution projection layers. The baseline Feedback model has a Feedback connection from block 4 to block 1. The complete model's architecture is presented in figure 8.

has very low accuracy. Also, the considered Feedback connection from block 4 to block 1 has just 3 filters and a kernel with the size (3,3), which makes it impossible to learn a sufficiently complex projection function. Nevertheless, the models with 2 and 3 projection layers do not show a significant difference. The statement can be derived, that the Feedback model can not learn if the projection layer is small. However, the model's results do not improve after a sufficient number of projection layers are applied. The difficulty is to find a sufficient number of layers for different Feedback models.

Also, for the model shown in figure 10, which contains a Feedback link from block 3 to block 5, the number of projection layers was altered. The architecture utilized in prior experiments has one layer. Figure 17 shows the results of comparing models with 1, 2, and 3 projection layers. The graph shows, that the Feedback model with 3 projection layers has a small improvement in recognizing the object in images with Gaussian noise. Therefore, more models with more projection layers should be tested to find the optimum number of layers for this type of Feedback model.

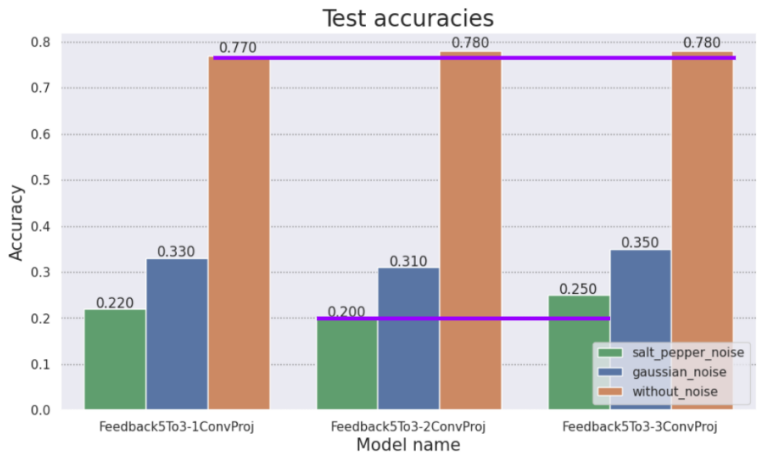


Figure 17: Feedback models with 1, 2, and 3 convolution projection layers are compared. The baseline Feedback model has a Feedback connection from block 5 to block 3. The complete model's architecture is presented in figure 10.

4.3 Training of not frozen models

Transfer learning reduces the complexity of the training procedure. But the VGG16 model may show better results if all layers are trained.

Accordingly, the not frozen Forward VGG16 model is compared with its alternative, which has a Feedback connection from block 4 to block 1. Namely, the architecture is taken over from the models displayed in figures 8 and 9, but there are no frozen layers. The results of the experiments are shown in figure 18. The considered forward model showed statistically significantly better accuracy on data with and without noise.

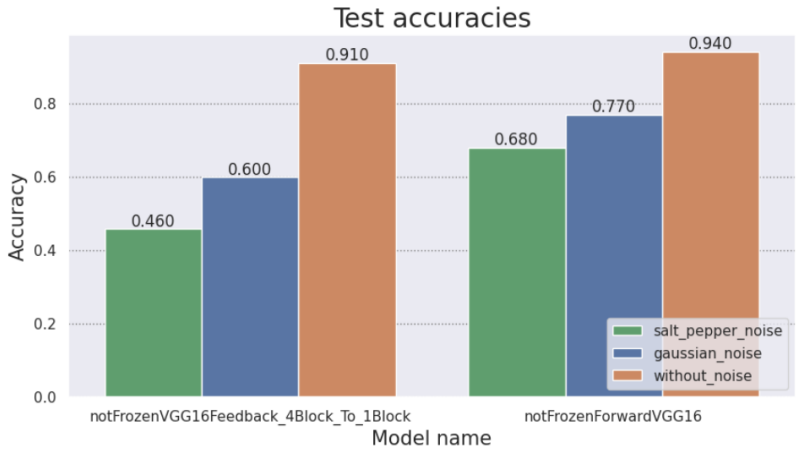


Figure 18: Comparison of Feedback and Forward models without frozen layers.

4.4 Comparison of different Feedback models

In the context of project work, the variations of Feedback models were compared, which have Feedback connections from different blocks to different blocks. During this experiment, the assumption was tested, if the location of the Feedback connection plays a role.

The models with frozen layers were considered in this experiment. Models with the following Feedback connections were compared:

1. From block 4 to block 1 (architecture is illustrated in figure 8).
2. From block 4 to block 4 (architecture is illustrated in figure 20).
3. From block 5 to block 3 (architecture is illustrated in figure 10).
4. From block 5 to block 4 (architecture is illustrated in figure 21).
5. From block 5 to block 5 (architecture is illustrated in figure 19).

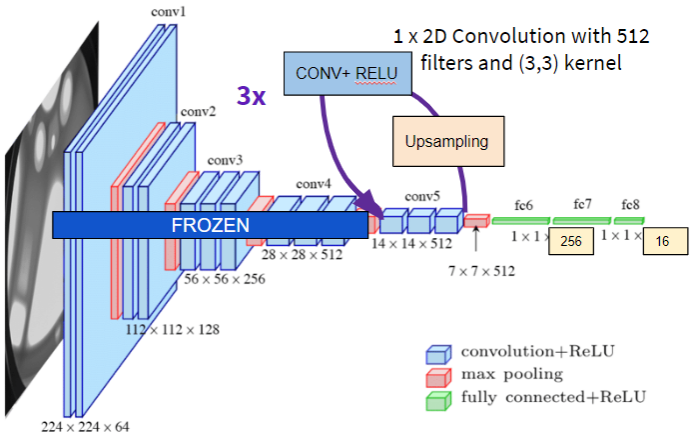


Figure 19: Frozen Feedback VGG16 [9] model with the additive Feedback connection from block 5 to block 5.

The comparison of the models is presented in figure 22. The following models showed the best evaluation result: the model with the Feedback connection from block 4 to block 1 and the model with the Feedback connection from block 5 to block 5. However, they also have the same frozen layers. That is why each Feedback model was compared with the corresponding Forward model, which has the same frozen layers. The result of this trial is illustrated in figure 23. It is shown, that the difference between each Feedback network and its corresponding Forward alternative is not significantly different.

Therefore the conclusion can be derived, that the number of frozen layers influences the model's performance and not the specific Feedback connection. This result demonstrates the necessity to compare the Feedback model with the Forward model, which contains the same frozen layers, in order to make the comparison valid.

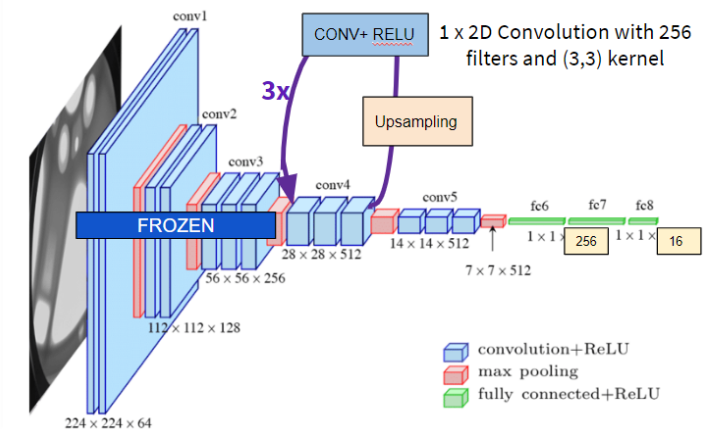


Figure 20: Frozen Feedback VGG16 [9] model with the additive Feedback connection from block 4 to block 4.

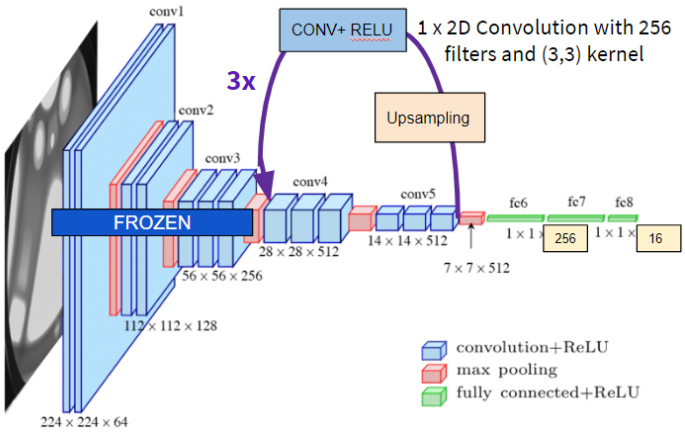


Figure 21: Frozen Feedback VGG16 [9] model with the additive Feedback connection from block 5 to block 4.

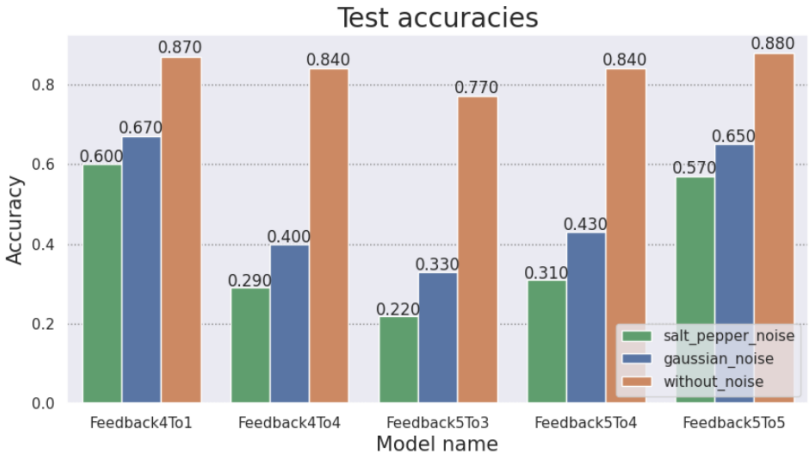


Figure 22: Comparison of Feedback models with different Feedback connections.

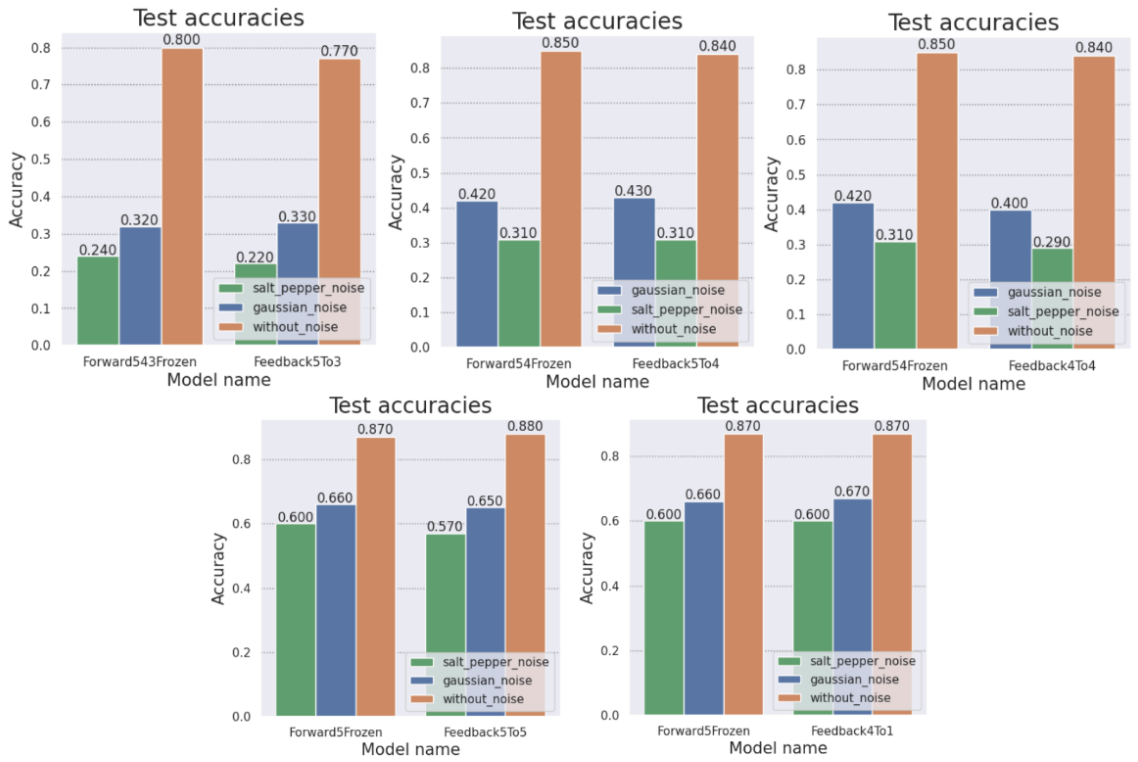


Figure 23: Comparison of each considered Feedback model with the Forward model, which has the same frozen layers.

5 Discussions

The research question brings up a wide range of possibilities for various experiments. All investigations cannot be carried out in the context of the project work. Nevertheless, This section discusses potential project improvements.

Distribution of labels In figure 6 the final distribution of labels is illustrated. The number of samples for each label is different. This aspect can impact training and make the model predict better the labels that occur more often.

To test this drawback, confusion matrices were created for the models displayed in the figures 8 and 9. The version of models trained on Gaussian data and data without noise were compared. The results were compared on the data without noise, on the data with Gaussian noise, and on the Salt-and-Pepper noise.

All confusion matrices showed a similar result: most values are located on a diagonal and the model predicts more correct samples for labels, that have more samples. In figure 24 the confusion matrix of the Forward model with frozen 1-4 blocks, trained on data without noise and tested on data with Gaussian noise is presented. The results for other models can be found in this PDF-File in the project implementation on GitHub.

The confusion matrices show, that the right predictions for different labels are distributed equally depending on the number of labels' occurrences. Nevertheless, to make a stronger assumption,

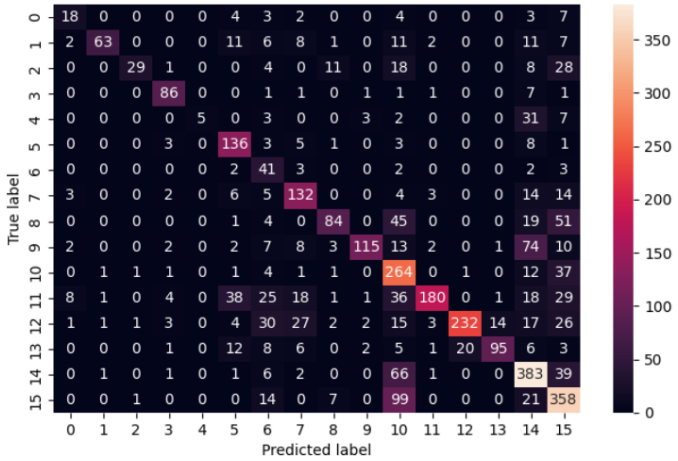


Figure 24: Confusion matrix of the Forward model with frozen 1-4 blocks (displayed in figure 9), trained on data without noise and tested on data with Gaussian noise.

the number of labels in the test dataset should be also distributed equally.

Training not frozen models Most resulting statements were derived based on training using transfer learning. However, the Feedback models may learn better patterns, if they would be trained without frozen layers and throw more epochs. The training of not frozen models is computationally very complex, which is why just a few experiments with it were carried out.

Not enough experiments to answer the research question Feedback models represent a complex architecture that can be varied in different ways. During this project work, just a couple of assumptions about Feedback models could be tested due to time and resource constraints.

Also, just one type of architecture was considered as the baseline - VGG16. Alternative models, such as ResNet or VGG19, might be explored as well.

In the study, only the additive Feedback connection was examined. Experiments with different types of connections (e.x. modulating or subtractive) may result in improved model performance.

6 Conclusion

Feedback connections present a novel architecture type. They have a broad spectrum of parameter settings, that can be adapted for a specific model.

In the context of the project work, the additive Feedback connection with three steps was applied to the VGG16 model to classify the object on the image. Other settings(location, number of frozen layers, number of projection layers) of the Feedback connection were varied to analyze the functionality of the network. The goal of the project is to prove or reject the assumption, that the Feedback connections can help the computer vision model to be more robust to different noise types.

The carried study shows that the addition of a Feedback connection does not improve the performance and robustness of the initial Forward model. But it requires more time and resources for

training. However, to develop a precise assumption regarding Feedback models, further experiments must be done as the current study was limited due to the time constraints of the student project work.

7 Appendix

Listing 1: Loading procedure of ImageNet dataset

```
# directory where you downloaded the tar files to
dataset_dir = '../../dataset_files'
temp_dir = '../temp_dataset_data'

download_config = tfds.download.DownloadConfig(
    extract_dir=os.path.join(temp_dir, 'extracted'),
    manual_dir=dataset_dir
)

tfds.builder("imagenet2012").download_and_prepare(
                                download_config=download_config)


builder = tfds.builder("imagenet2012")
raw_valid_ds, raw_test_ds, raw_train_ds = builder.as_dataset(
                                            ['validation',
                                            'train[0:50000]',
                                            'train[50000:]'],
                                            as_supervised=True)
```

Listing 2: Application of Gaussian noise on the image

```
def gaussian_noise(image):
    row,col,ch= image.shape
    mean = 0
    var = 0.01
    sigma = var**0.4
    gauss = np.random.normal(mean,sigma,(row,col,ch))
    gauss = gauss.reshape(row,col,ch)
    noisy = image + gauss
    return noisy
```

Listing 3: Application of Salt-and-pepper noise on the image

```
def salt_pepper_noise(img, pad = 150):
    show = 1
    noise = np.random.randint(pad, size = (img.shape[0],
```

```
                                            img.shape[1],
                                            3))
    img = np.where(noise == 0, 0, img)
    img = np.where(noise == (pad-1), 1, img)
    noise = noise / 255
    return noise + img
```

Listing 4: Frozen Feedback VGG16 model with the Feedback connection from block 4 to block 1

```
class VGG16FeedbackFrozen4BlockTo1Block(keras.Model):
    def __init__(self, projection_3_layers = True, **kwargs):
        super(VGG16FeedbackFrozen4BlockTo1Block, self)\
                        .__init__(name="FFmodel", **kwargs)
        VGG = VGG16(weights='imagenet', include_top=False)
        self.VGG_before_feedback = keras.Model(VGG.input,
                                    VGG.layers[-6].output)
        self.VGG_before_feedback.trainable = False

        self.maxpooling1 = MaxPooling2D(pool_size=(2,2),
                                        strides= (2,2))
        self.conv1 = Conv2D(512,
                        kernel_size=(3,3),
                        padding= 'same',
                        activation= 'relu')
        self.conv2 = Conv2D(512,
                        kernel_size=(3,3),
                        padding= 'same',
                        activation= 'relu')
        self.conv3 = Conv2D(512,
                        kernel_size=(3,3),
                        padding= 'same',
                        activation= 'relu')

        self.flatten = Flatten()
        self.dense = Dense(256, activation= 'relu')
        self.dropout = Dropout(0.5)
        self.maxpooling2 = MaxPooling2D(pool_size=(2,2),
                                        strides= (2,2))
        self.output_layer = Dense(num_classes,
                                  activation='softmax')

        self.upsampling = UpSampling3D(size=8)
        self.project_conv1 = Conv2D(3,
                                    kernel_size=(3,3),
                                    padding= 'same',
```

```
                                    activation= 'relu')
        self.project_conv2 = Conv2D(3,
                                    kernel_size=(3,3),
                                    padding= 'same',
                                    activation= 'relu')
        self.project_conv3 = Conv2D(3,
                                    kernel_size=(3,3),
                                    padding= 'same',
                                    activation= 'relu')

        self.projection_3_layers = projection_3_layers

    def call(self, inputs):
        num_timesteps = 3
        x = inputs
        for i in range(num_timesteps):
            x = self.VGG_before_feedback(x)
            if i != num_timesteps - 1:
                x = self.upsampling(x)
                x = self.project_conv1(x)
                if self.projection_3_layers:
                    x = self.project_conv2(x)
                    x = self.project_conv3(x)
                x = Add()([inputs, x])

        x = self.maxpooling1(x)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.maxpooling2(x)
        x = self.flatten(x)
        x = self.dense(x)
        x = self.dropout(x)
        x = self.output_layer(x)
        return x
```

Listing 5: Example of the training procedure

```
epochs = 40
train_ds = train_ds_prepared_without_batch\
                                .shuffle(10000)\
                                .batch(batch_size)\
                                .prefetch(4)
valid_ds = valid_ds_prepared_without_batch\
                                .batch(batch_size)\
```

```
                                .prefetch(4)
test_ds = test_ds_prepared_without_batch\
                                .batch(batch_size)\
                                .prefetch(4)
test_ds_gaussian_noise = test_ds_prepared_without_batch\
                                .map(tf_gaussian_noise)\
                                .batch(batch_size)\
                                .prefetch(4)
test_ds_salt_pepper_noise = test_ds_prepared_without_batch\
                                .map(tf_salt_pepper_noise)\
                                .batch(batch_size)\
                                .prefetch(4)

sample = next(iter(test_ds))[0]
model = model_class()
model.build(input_shape)
model(sample)
model.summary(show_trainable=True)

model.compile(
    optimizer=tf.keras.optimizers.SGD(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',
    metrics=["accuracy"])
history = model.fit(train_ds,
                    epochs=epochs,
                    validation_data=valid_ds,
                    callbacks=[model_checkpoint_callback],
                    verbose=1)
results = model.evaluate(test_ds, batch_size=batch_size)
results_gaussian_noise = model.evaluate(test_ds_gaussian_noise,
                                        batch_size=batch_size)
results_salt_pepper_noise = model.evaluate(
                                    test_ds_salt_pepper_noise,
                                    batch_size=batch_size)

test_accuracies = {
"test_acc_original_data": results[1],
"test_acc_gaussian_noise": results_gaussian_noise[1],
"test_acc_salt_pepper_noise": results_salt_pepper_noise[1]
}

persist_evaluation_results(accuracy=test_accuracies,
                           history=history.history,
                           path_to_persist=path_to_persist_results)
```

References

[1] T. Mazouka, Implementation of the project “Feedback and forward networks for object recognition on the data with different noise types”, https://github.com/Tmozovka/ProjectFeedbackNetworks, [Online; Stand 9. Februar 2023], **2022**.

[2] Wikipedia, McNemar-Test — Wikipedia, die freie Enzyklopädie, [Online; Stand 9. Februar 2023], **2019**.

[3] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, F. A. Wichmann, *CoRR* **2018**, *abs/1808.08750*.

[4] C. Jarvers, H. Neumann, Incorporating Feedback in Convolutional Neural Networks, https://www.informatik.uni-ulm.de/ni/staff/HNeumann/publicationsYear/PDFs/CONFERENCES/CCN19-JarversNeumann-feedbackCNN.pdf, [Online; Stand 9. Februar 2023], **2019**.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, **2009**, 248–255.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, Software available from tensorflow.org, **2015**.

[7] F. Chollet et al., Keras, https://keras.io, **2015**.

[8] P. Janetzky, Preparing the ImageNet dataset with TensorFlow, https://towardsdatascience.com/preparing-the-imagenet-dataset-with-tensorflow-c681916014ee, [Online; Stand 9. Februar 2023], **2021**.

[9] M. Ferguson, R. ak, Y.-T. Lee, K. Law in **2017**, pp. 1726–1735, DOI 10.1109/BigData.2017.8258115.

[10] J. Brownlee, How to Calculate McNemar’s Test to Compare Two Machine Learning Classifiers, https://machinelearningmastery.com/mcnemars-test-for-machine-learning/, [Online; Stand 9. Februar 2023], **2018**.

8 Abbreviations

DNN Deep Neural Network

CNN Convolutional Neural Network