***Rice Plant Leaf Disease Detection using InceptionResNetV2, InceptionV3, Xception***

***MUNGILI CHETAN SAI RAJU – 21BCE9409***

***VEERAPU VISHNU – 21BCE9254***

***Abstract:***

*In the crop protection system, early and precise identification of plant diseases is vital. In standard processes, identification is carried out either through visual observation or through laboratory testing. While the laboratory test is time-consuming and may not be able to produce the results in a timely manner, visual observation requires experience and can vary depending on the individual, which could cause an error. To overcome these issues, image based on deep learning approach is used to detect and identify plant diseases. We have focused specifically on rice plant diseases. The dataset that contains different variations of diseased symptoms is gathered. We have used a dataset that contains* ***4500*** *images belongs to nine classes (a) bacterial\_leaf\_blight (b) Brown Spot (c) Healthy (d) Hispa (d)Leaf Steak (e)Leaf\_blast (f)leaf\_scald (g)Narrow\_brown\_spot (h)Shath Blight (i)Tungro. We have used pretrained tensorflow and Keras models like InceptionResNetV2, XceptionNet, InceptionV3 as feature extractors. We have got very interesting results.*

***Introduction***

*Recent developments in machine learning and artificial intelligence have changed many industries, including agriculture. To increase crop yield and guarantee food security, farmers are now implementing sophisticated technologies. Plant disease identification in crops is a crucial part of agriculture since it can result in severe yield and quality losses. Among the many crops grown around the world, rice stands out as a staple diet for more than half of humanity. In this study, we investigate the use of deep learning models, particularly InceptionResNetV2, InceptionV3, and Xception, for the identification of leaf disease in rice plants.*

## ***Understanding the Importance of Disease Detection in Rice Plants:***

*Rice, being a major staple, plays a crucial role in the global food supply chain. However, the health of rice crops is susceptible to various diseases caused by pathogens such as fungi, bacteria, and viruses. Timely detection and management of these diseases are essential to minimize the impact on crop yield and quality. Traditional methods of disease detection are time-consuming and often require expert agronomists. The introduction of deep learning models brings a promising solution to automate this process and detect diseases accurately.*



***Overview of Models:***

***InceptionResNetV2:***

*A potent deep learning architecture called InceptionResNetV2 combines the advantages of Inception and ResNet. It belongs to the Inception architecture family and was created by Google.* InceptionResNetV2 is a well-liked option for spotting *illnesses in plant leaves due to its outstanding performance in the image classification tasks.*

***Inception V3:***

*Another well-known deep learning model, InceptionV3, has gained popularity for its precision and effectiveness in image identification tasks. It was created by Google as well and is predicated on the concept of factoring convolutions. Due to its capacity to recognize fine details in images, InceptionV3 has found extensive use in a variety of applications, including the diagnosis of plant diseases.*

### ***Xception:***

*Xception is an extension of the Inception architecture, known for its depth-wise separable convolutions. This unique approach reduces computational complexity while maintaining high accuracy, making it an attractive option for real-time applications like disease detection in rice plants.*

## ***Implementing Deep Learning Models for Rice Plant Disease Detection:***

### ***Data Collection and Preprocessing:***

*The first step in training deep learning models is to gather a diverse dataset of rice plant leaf images, including healthy leaves and leaves affected by various diseases. Proper preprocessing techniques, such as data augmentation and normalization, are applied to ensure the models can learn effectively from the data.*

### ***Transfer Learning:***

*Deep learning models like InceptionResNetV2, InceptionV3, and Xception require a significant amount of labeled data for training, which may not always be readily available for specific plant diseases. Transfer learning comes to the rescue in such scenarios. It involves using a pre-trained model and fine-tuning it on the target task with a smaller dataset. This approach helps to leverage the knowledge acquired from training on large-scale datasets and adapt it to the specific rice plant disease detection problem.*

### ***Training and Evaluation:***

*The labeled dataset is divided into training, validation, and testing sets. The deep learning models are trained on the training data and evaluated on the validation set to fine-tune their hyperparameters. The model's performance is assessed using metrics such as accuracy, precision. The goal is to achieve a high-performing model that can accurately identify the presence of diseases in rice plant leaves.*

***Literature review:***

***1. “Automatic Diagnosis of Rice Diseases Using Deep Learning”*** *by* ***Ruoling Deng, Ming Tao, Hang Xing, Xiuli Yang, Chuang Liu, Kaifeng Liao, and Long Qi.***

*The goal of this research is to provide an automated detection system for rice illnesses using deep learning techniques. Existing disease diagnosis procedures for rice are frequently erroneous and need the use of specialised equipment. To solve this issue, the researchers developed a smartphone app using a big dataset including 33,026 photos of six different forms of rice illnesses. To increase illness recognition accuracy and reduce misunderstanding among different types of diseases, they used an Ensemble Model that combined three submodels, namely DenseNet-121, SE-ResNet-50, and ResNeSt-50. The Ensemble Model attained an overall accuracy of 91%, making it appropriate for field detection of rice illnesses through smartphone app, providing simplicity and efficiency.*

*2."****Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases Using an Improved CNN Model****" by* ***Ghazanfar Latif, Sherif E. Abdelhamid, Roxane Elias Mallouhy, Jaafar Alghazo, and Zafar Abbas Kazimi***

*A study published in the above general proposed a modified VGG19-based transfer learning method for detecting and classifying six types of rice diseases: healthy, narrow brown spot, leaf scald, leaf blast, brown spot, and bacterial leaf blight. Using a non-normalized augmented dataset, the updated approach attained an average accuracy of 96.08%.*

*3. The survey study "****Rice Plant Disease Detection and Classification Techniques: A Survey****" by* ***Tejas Tawde , Kunal Deshmukh , Lobhas Verekar , Ajay Reddy, Shailendra Aswale, Pratiksha Shetgaonkar*** *examined many tactics for rice leaf disease classification, including image capture, image preprocessing, feature extraction, feature selection, and classification. The report also explored the problems and future objectives for employing image processing techniques to identify rice plant illness.*

*4.* *A study titled "****Rice Leaf Disease Detection and Classification Using a Deep Neural Network****" proposed a method for detecting and classifying four types of rice diseases: healthy, bacterial leaf blight, brown spot, and leaf blast using a convolutional neural network (CNN) with four convolutional layers and two fully connected layers. On a dataset of 800 photos, the approach obtained an accuracy of 94.67%.*

**Methodology:**

*There are two primary phases to training a pre-trained model. To begin, a pre-trained model that has learnt from a huge dataset is used as a starting point. The model is then fine-tuned using a smaller dataset tailored to the job at hand. The model's weights are modified during fine-tuning to react to new data while keeping the useful information obtained during initial training. This technique allows the model to specialize and increase its performance on a single job without requiring a significant quantity of data or a long period of training. To produce better outcomes more efficiently, fine-tuning a pre-trained model is a commonly used approach in deep learning. The codes of the three models are given below:*

**InceptonResnetV2 code:**

import tensorflow as tf

from tensorflow.keras.applications import InceptionResNetV2

from tensorflow.keras.layers import GlobalAveragePooling2D, Dense

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Model

from tensorflow.keras.callbacks import ModelCheckpoint

# Set the number of classes for rice plant diseases

num\_classes = 9

# Load the pre-trained InceptionResNetV2 model without the top (fully connected) layers

base\_model = InceptionResNetV2(weights='imagenet', include\_top=False, input\_shape=(299, 299, 3))

# Add custom top layers

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu')(x)

predictions = Dense(num\_classes, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

# Freeze the base layers of the pre-trained model

for layer in base\_model.layers:

layer.trainable = False

# Compile the model

model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

train\_datagen = ImageDataGenerator(

rescale=1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

rotation\_range=20,

)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

'D://DEEP LEARNING DATASET/training',

target\_size=(299, 299),

batch\_size=32,

class\_mode='categorical'

)

validation\_generator = test\_datagen.flow\_from\_directory(

'D://DEEP LEARNING DATASET/testing',

target\_size=(299, 299),

batch\_size=32,

class\_mode='categorical'

)

# Fine-tuning: Unfreeze some layers for further training

for layer in model.layers[-50:]:

layer.trainable = True

# Re-compile the model after fine-tuning

model.compile(optimizer=Adam(learning\_rate=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])

# Define a ModelCheckpoint callback to save the best weights during training

checkpoint = ModelCheckpoint(

'best\_weights.h5',

monitor='val\_accuracy',

verbose=1,

save\_best\_only=True,

mode='max'

)

history = model.fit(

train\_generator,

steps\_per\_epoch=train\_generator.n // train\_generator.batch\_size,

epochs=25, # You can adjust the number of fine-tuning epochs as needed

validation\_data=validation\_generator,

validation\_steps=validation\_generator.n // validation\_generator.batch\_size,

callbacks=[checkpoint]

)

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

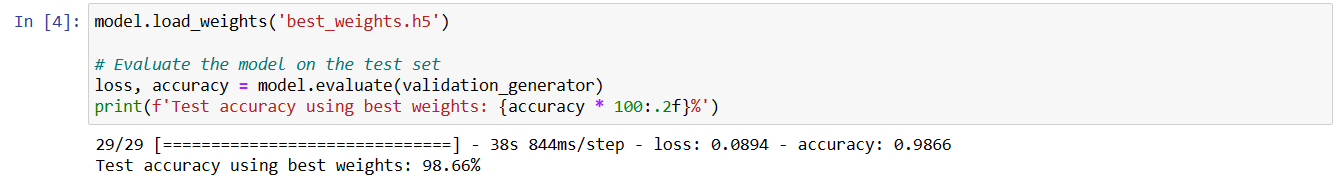
plt.ylabel('Accuracy')

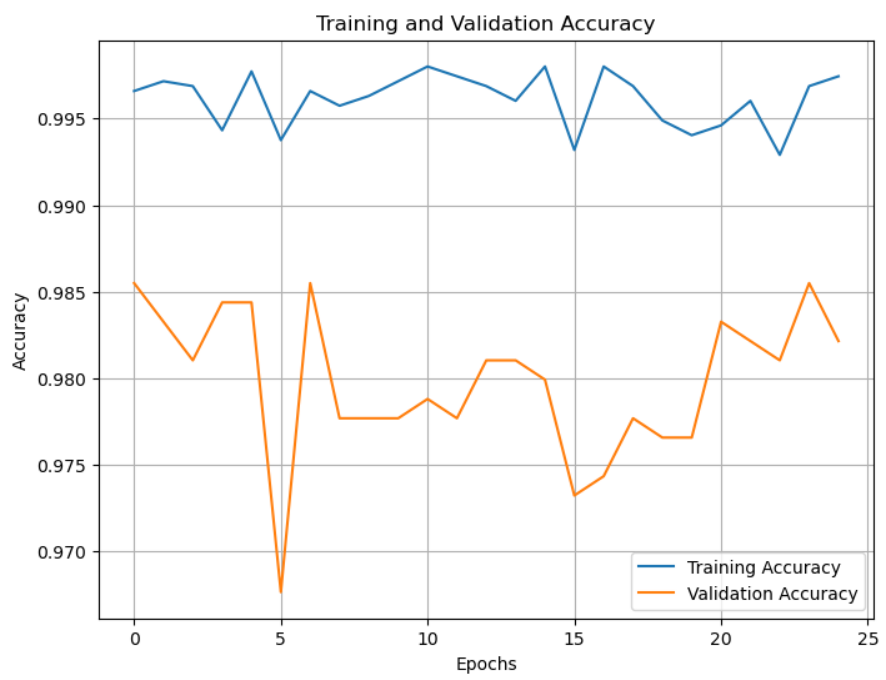
plt.title('Training and Validation Accuracy')

plt.legend()

plt.grid()

plt.show()





**InceptionV3 code:**

import tensorflow as tf

from tensorflow.keras import layers, Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

from tensorflow.keras.callbacks import ModelCheckpoint

# Set the number of classes for rice plant diseases

num\_classes = 9

# Load the pre-trained InceptionV3 model without the top (fully connected) layers

base\_model = tf.keras.applications.InceptionV3(weights='imagenet', include\_top=False, input\_shape=(299, 299, 3))

# Add custom top layers

x = base\_model.output

x = layers.GlobalAveragePooling2D()(x)

x = layers.Dense(1024, activation='relu')(x)

predictions = layers.Dense(num\_classes, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

# Freeze the base layers of the pre-trained model

for layer in base\_model.layers:

layer.trainable = False

# Compile the model

model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

# Data augmentation and loading the dataset

train\_datagen = ImageDataGenerator(

rescale=1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

rotation\_range=20,

)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

'/content/drive/MyDrive/DEEP LEARNING DATASET/training',

target\_size=(299, 299),

batch\_size=32,

class\_mode='categorical'

)

validation\_generator = test\_datagen.flow\_from\_directory(

'/content/drive/MyDrive/DEEP LEARNING DATASET/testing',

target\_size=(299, 299),

batch\_size=32,

class\_mode='categorical'

)

checkpoint = ModelCheckpoint(

'weights\_checkpoint.h5',

monitor='val\_accuracy',

verbose=1,

save\_weights\_only=True,

save\_best\_only=True,

mode='max'

)

history = model.fit(

train\_generator,

steps\_per\_epoch=train\_generator.samples // train\_generator.batch\_size,

epochs=10,

validation\_data=validation\_generator,

validation\_steps=validation\_generator.samples // validation\_generator.batch\_size,

callbacks=[checkpoint]

)

# Plotting training and validation accuracy

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy (Pre-training)')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy (Pre-training)')

plt.plot(history\_fine\_tuned.history['accuracy'], label='Training Accuracy (Fine-tuning)')

plt.plot(history\_fine\_tuned.history['val\_accuracy'], label='Validation Accuracy (Fine-tuning)')

plt.xlabel('Epochs')

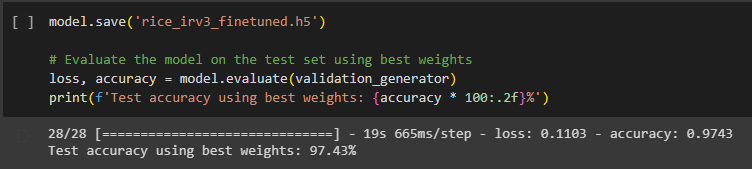
plt.ylabel('Accuracy')

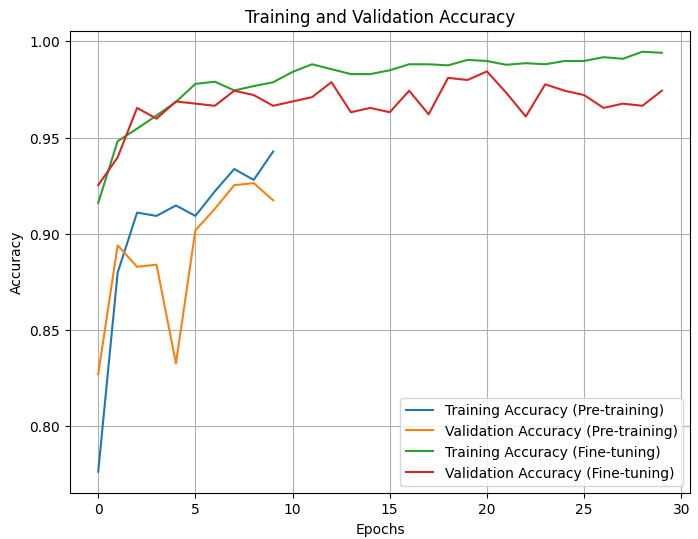
plt.title('Training and Validation Accuracy')

plt.legend()

plt.grid()

plt.show()





**Xception code:**

import numpy as np

from keras.applications import Xception

from keras.models import Model

from keras.layers import Dense, GlobalAveragePooling2D

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import Adam

from keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt

# Set the number of classes for rice plant diseases

num\_classes = 9

# Load the pre-trained Xception model without the top (fully connected) layers

base\_model = Xception(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Add custom top layers

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu')(x)

predictions = Dense(num\_classes, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

# Freeze the base layers of the pre-trained model except for the last few layers

for layer in base\_model.layers[:-20]:

layer.trainable = False

# Compile the model with a lower learning rate

model.compile(optimizer=Adam(lr=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])

# Data preprocessing and augmentation

train\_datagen = ImageDataGenerator(rescale=1./255, shear\_range=0.2, zoom\_range=0.2, horizontal\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

'C:\\Users\\vishn\\Desktop\\Folders\\deep\_learning\\training',

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical')

validation\_generator = test\_datagen.flow\_from\_directory(

'C:\\Users\\vishn\\Desktop\\Folders\\deep\_learning\\testing',

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical')

# Define a ModelCheckpoint callback to save the best weights during training

checkpoint = ModelCheckpoint(

'best\_weights.h5',

monitor='val\_accuracy',

verbose=1,

save\_best\_only=True,

mode='max'

)

# Train the model with fine-tuning and save best weights

history = model.fit(

train\_generator,

steps\_per\_epoch=train\_generator.n // train\_generator.batch\_size,

epochs=100,

validation\_data=validation\_generator,

validation\_steps=validation\_generator.n // validation\_generator.batch\_size,

callbacks=[checkpoint]

)

# Save the entire model (architecture and weights)

model.save('rice\_plant\_disease\_model.h5')

# Plotting training and validation accuracy

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

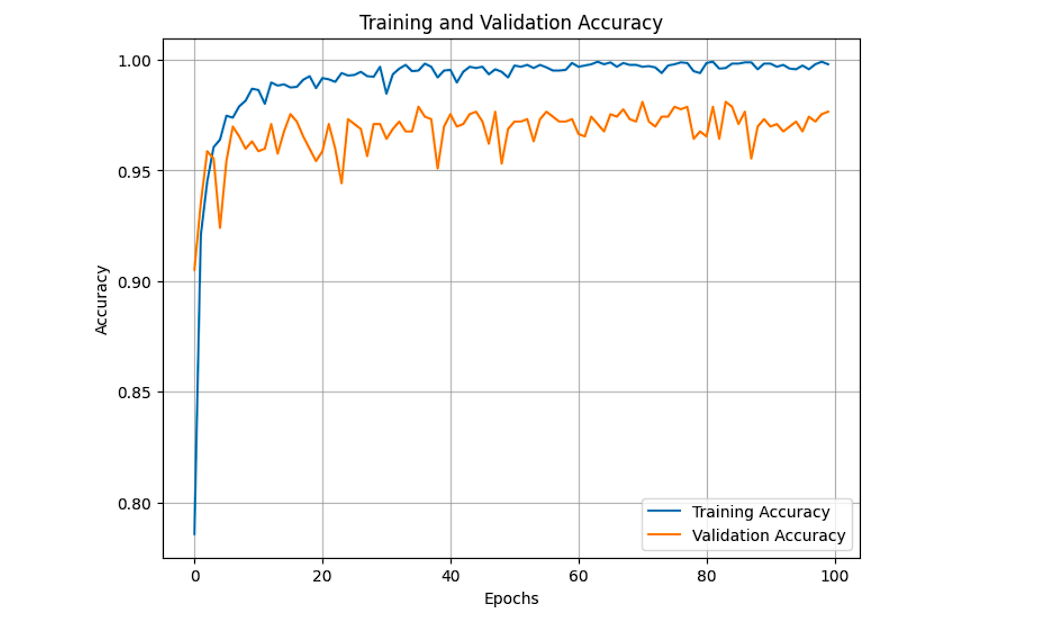
plt.ylabel('Accuracy')

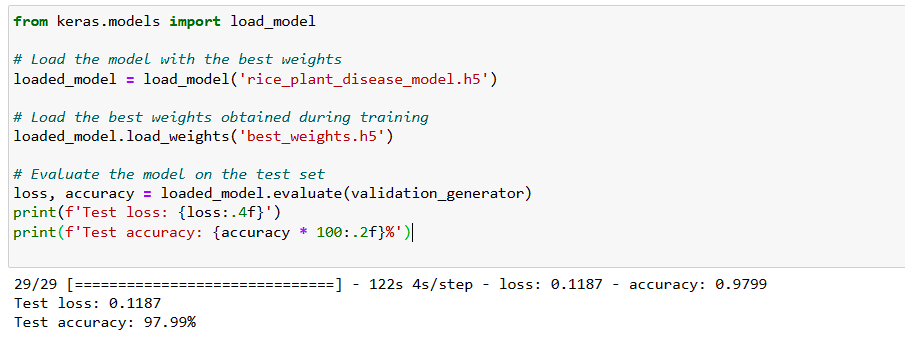
plt.title('Training and Validation Accuracy')

plt.legend()

plt.grid()

plt.show()





## ***Benefits and Challenges of Using Deep Learning for Rice Plant Disease Detection***

### ***Benefits***

* ***Accuracy****: Deep learning models have demonstrated impressive accuracy in image recognition tasks, making them reliable for detecting diseases in rice plants.*
* ***Speed****: Once trained, these models can process images rapidly, enabling quick and timely disease detection in large crop fields.*
* ***Cost-Effective****: Deep learning models can lead to cost savings by reducing the need for manual labor and expert consultation.*

### ***Challenges***

* ***Data Scarcity****: Obtaining a diverse and extensive dataset of labeled rice plant images can be challenging, especially for rare diseases.*
* ***Model Complexity****: Deep learning models often require significant computational resources and expertise to train and deploy.*

## ***Conclusion***

*In conclusion, deep learning models such as InceptionResNetV2, InceptionV3, and Xception offer a promising solution for rice plant leaf disease detection. By leveraging the power of artificial intelligence and transfer learning, farmers and agronomists can efficiently monitor the health of rice crops and take timely actions to mitigate potential losses. Embracing these advanced technologies will undoubtedly contribute to the improvement of agricultural practices and global food security.*

***References:***

1. [***https://www.frontiersin.org/articles/10.3389/fpls.2021.701038/full***](https://www.frontiersin.org/articles/10.3389/fpls.2021.701038/full)
2. [***https://www.mdpi.com/2223-7747/11/17/2230***](https://www.mdpi.com/2223-7747/11/17/2230)
3. [***https://www.ijert.org/rice-plant-disease-detection-and-classification-techniques-a-survey***](https://www.ijert.org/rice-plant-disease-detection-and-classification-techniques-a-survey)
4. [***https://link.springer.com/chapter/10.1007/978-3-031-21750-0\_20***](https://link.springer.com/chapter/10.1007/978-3-031-21750-0_20)