Dive into Deep Learning

xAI-Proj-B: Bachelor Project Explainable Machine Learning

Florian Gutbier*
Otto-Friedrich University of Bamberg
96047 Bamberg, Germany
XXX@stud.uni-bamberg.de

Marius Ludwig Bachmeier†
Otto-Friedrich University of Bamberg
96047 Bamberg, Germany
XXX@stud.uni-bamberg.de

Andreas Schwab‡
Otto-Friedrich University of Bamberg
96047 Bamberg, Germany
andreas-franz.schwab@stud.uni-bamberg.de

Abstract

//TODO: write abstract? Was muss da rein?

1 Introduction

The core of this project was to understand the key principles of deep learning and to apply them in a practical environment. Through the methodical training and evaluation of various techniques, we progressively deepened our understanding of this complex field. A crucial first step in developing an effective classification model was to thoroughly investigate and understand the dataset at hand. Therefore, our investigation began with an introduction to the well-known datasets MNIST (Deng, 2012) and MedMNIST (Yang et al., 2021), which served as the building blocks for our study.

The project was structured to take account of the different characteristics of the individual datasets. We started with MNIST, which was chosen due to its wide distribution and the numerous tutorials available, which facilitated our entry into the world of deep learning. With this dataset, we took on the challenge of developing a rudimentary Convolutional Neural Network (SimpleCNN) that was intentionally designed with a limited number of layers. The initial aim of this challenge was not to achieve peak performance, but rather to gain practical experience and understand the basics of the architecture of neural networks and their ability to distinguish between different digits.

As our knowledge increased, we shifted our focus to the more challenging MedMNIST dataset, focusing particularly on the PathMNIST subset. This phase formed the core of our project, in which we focused intensively on experimenting with different pre-trained models. Our investigations extended to testing a wide range of hyperparameters as well as implementing different strategies for data preprocessing and augmentation. The complexity and challenges of the PathMNIST dataset required the use of more advanced techniques and approaches, representing a significant advance over our initial experiments with MNIST.

1.1 MNIST

The "modified National Institute of Standards and Technology" dataset comprises a collection of 70,000 handwritten digits carefully divided into a training set of 60,000 images and a test set of

*Degree: B.Sc. AI, matriculation #: 12345678
†Degree: B.Sc. AI, matriculation #: 12345678
‡Degree: B.Sc. AI, matriculation #: 2017990

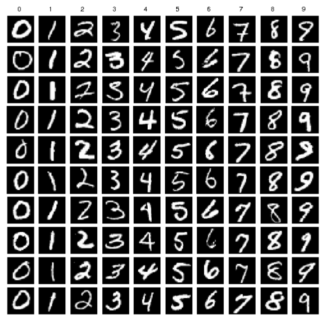

Figure 1: Example greyscale images of the MNIST dataset - CHANGE IMAGE - RENDER OWN!!!

10,000 images. Each digit is represented in a grayscale image of 28×28 pixels, and offers a wide range of styles and shapes. This feature makes MNIST not only an ideal starting point for novices in the field of machine learning, but also a benchmark for evaluating the performance of different image classification models. Some examples of the dataset can be seeen in Figure 1

In our project, the MNIST dataset provided a foundational experience that allowed us to understand the basics of neural network architecture and effectively implement them for the task of classifying numbers.

1.2 MedMNIST

//TODO

2 Methods

2.1 Model Architectures

2.1.1 SimpleCNN

//TODO

2.1.2 ResNet

//TODO

Resnet18 //TODO

Resnet50 //TODO

ResnetXX //TODO

2.1.3 Xception

As seen in Figure 2 the Architecture of Xception outlined by (Chollet, 2017).

3 Discussion

//TODO

4 Conclusion

//TODO

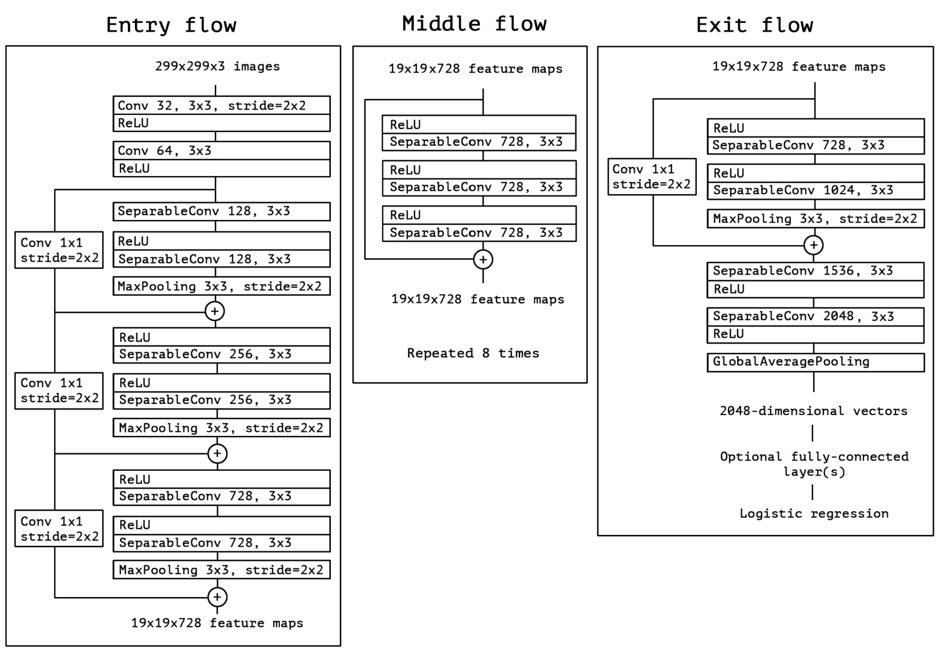

Figure 2: Xceptions architecture as outlined by Chollet.

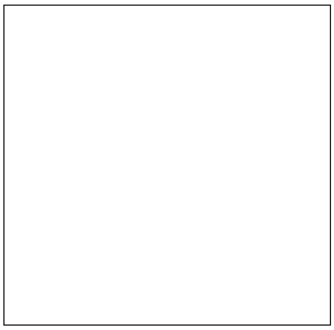

Figure 3: Sample figure caption.

Table 1: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

References

F. Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

J. Yang, R. Shi, and B. Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195, 2021.

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 23, 2024

(Place, Date) (Signature)

Bamberg, February 23, 2024

(Place, Date) (Signature)

Bamberg, February 23, 2024

(Place, Date) (Signature)

A Appendix

Initial version of our SimpleCNN, including two convulutional layers:

```
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels = 1,
                out_channels = 32,
                kernel_size = 5,
                stride=1,
                padding="same"
            ),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size=2),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(32,64,5,1,"same"),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size=2),
        )
        self.out = nn.Linear(64*7*7, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(-1, 64*7*7)
        output = self.out()
        return torch.log_softmax(output, dim=1)
```

Structure of the improved version of the SimpleCNN using three convolutional layers, Batch normalization and Dropout:

```
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=1, padding="same"),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
            nn.Dropout(0.25)
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding="same"),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Dropout(0.25)
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding="same"),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Dropout(0.25)
        )
        self.fc1 = nn.Linear(128 * 3 * 3, 256)
        self.fc_bn = nn.BatchNorm1d(256)
        self.dropout_fc = nn.Dropout(0.5)
        self.fc2 = nn.Linear(256, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = x.view(-1, 128 * 3 * 3)
        x = F.relu(self.fc_bn(self.fc1(x)))
        x = self.dropout_fc(x)
        x = self.fc2(x)
        return torch.log_softmax(x, dim=1)
```