

---

# KuchBhiRakhDo

**xAI-Proj-B: Bachelor Project Explainable Machine Learning**

**A Semester in the Life of a Deep Learning Engineer**

---

**Shehroz Shafiq Khan\***

Otto-Friedrich University of Bamberg

96049 Bamberg, Germany

`shehroz.shafiq-khan@stud.uni-bamberg.de`

**Hafiz Ahmer Saeed Khan<sup>†</sup>**

Otto-Friedrich University of Bamberg

96049 Bamberg, Germany

`hafiz.ahmer.saeed-khan@stud.uni-bamberg.de`

**Muhammad Junaid<sup>‡</sup>**

Otto-Friedrich University of Bamberg

96049 Bamberg, Germany

`muhammad.junaid@stud.uni-bamberg.de`

---

\*Degree: M.Sc. International Software System Science, matriculation #: 2093254

<sup>†</sup>Degree: M.Sc. International Software System Science, matriculation #: 2092973

<sup>‡</sup>Degree: M.Sc. International Software System Science, matriculation #: 2097636

## Abstract

In this project, we combined the power of deep learning techniques to excel in recognizing single-digit numbers. Our project consists of two main parts, each focusing on different datasets and architectures. In the first part, we explored the MNIST dataset, which contains a large collection of handwritten digits. We trained a customized Convolutional Neural Network (CNN) to accurately understand the intricacies of these handwritten digits. In the second part, we explored street house numbers captured in real-world images. We used the VGG-16 architecture and around 700 images to successfully recognize these street digits with high accuracy. This shows that our techniques work well in different areas. Our work combines technology and perception to achieve successful recognition of single-digit numbers. We've put all our work in our Git repository so others can see and learn from it: <https://github.com/shehrozshafiqh/xAI-deep-learning>

## 1 Introduction

### "A semester in life of a Deep learning Engineer".

We're showing a cool mix of machine learning and computer vision. Our aim is to figure out the secret of spotting single numbers using deep learning techniques.

We started with a well-known dataset of handwritten numbers called MNIST. People who like computers and pictures have loved this for a long time. We've used deep learning techniques that use Convolutional Neural Networks (CNN) to get the numbers right and also to understand why they look the way they do.

But we did not just stick to computers but also went to real street house numbers where there are single numbers on signs. We got ideas from colorful pictures and used a popular CNN architecture called VGG-16 to predict these street numbers.

### 1.1 Background

In this project, we tackled 2 problems regarding single-digit recognition using MNIST and Street Numbers datasets.

#### 1.1.1 MNIST

The MNIST dataset is a widely used collection of handwritten digit images commonly employed for machine learning and computer vision tasks. It contains a set of 28x28 grayscale images, each depicting a single digit from 0 to 9. The dataset is divided into two main parts: a training set and a test set.

The training set consists of thousands of images, where each image is labeled with its corresponding digit (0 to 9). Machine learning models can use this labeled data to learn the patterns and characteris-

tics of each digit. The model can then use this learned knowledge to predict the digit in new, unseen images.

The test set serves as a benchmark to evaluate the model's performance. It contains images that the model has never encountered during training. The model's predictions are compared to the actual labels of the test images to measure its accuracy.

The MNIST dataset is commonly used to develop and test various machine learning algorithms, especially for image classification tasks. It's a fundamental resource for researchers and practitioners to assess the effectiveness of different techniques in the realm of digit recognition and image analysis.

### **1.1.2 Street Numbers**

We gathered the StreetNumbers dataset firsthand by venturing into various neighborhoods across Bamberg. Our group with other teams captured a collection of over 700 high-resolution images. These images were initially in high resolution and were subsequently resized to dimensions of 256 by 256 pixels. All pictures were in RGB format. These images represent real-world scenarios and feature numbers displayed on signs and buildings. The primary objective of utilizing this dataset was to evaluate the performance of our pre-existing model on authentic, real-world data.

Utilizing cutting-edge neural networks, we embarked on the mission of training our system to visually comprehend and identify street house numbers.

## **1.2 Related Work**

We have extensively reviewed and analyzed two prominent research papers. These comprehensive studies have significantly contributed to our understanding of the domain of deep learning and provided valuable insights into the topic.

- Very Deep Convolutional Networks for Large-Scale Image Recognition  
<https://arxiv.org/abs/1409.1556>  
Karen Simonyan, Andrew Zisserman
- Deep Residual Learning for Image Recognition  
<https://arxiv.org/abs/1512.03385>  
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

## **1.3 Contribution**

Throughout this project, our main objective was to develop a robust solution for accurately recognizing single-digit numbers using deep learning techniques. We have made significant contributions in terms of innovative methodologies and practical implementations that push the boundaries of the current state of the art in this field.

Here is an overview of our achievements:

We extensively explored cloud-based training platforms, such as GCP and Google Colab, seamlessly integrating them into our model training process. This integration allowed us to efficiently utilize substantial computational resources for experimenting with intricate neural network architectures.

To counter overfitting challenges, we introduced inventive data augmentation techniques, including rotation, flipping, and cropping. By strategically enhancing our training dataset, we significantly improved our models' ability to generalize effectively across diverse real-world data.

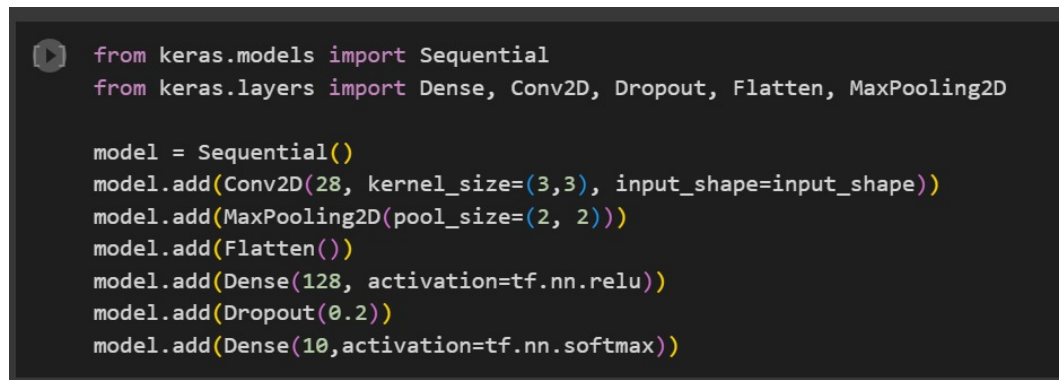
By leveraging Keras's transfer learning capabilities, we seamlessly integrated pre-trained models like VGG-16 into our framework. This accelerated the training process and allowed us to transfer valuable features from extensive datasets to our specific problem domain.

We conducted exhaustive experiments to explore the robustness and reliability of our models. These experiments rigorously evaluated the models' ability to handle diverse inputs and ensured consistent performance under varying conditions, increasing the overall reliability and performance of our model.

## 2 Methods

### 2.1 First Approach

In our first approach, we constructed a Convolutional Neural Network (CNN) using Keras. The model begins with a convolutional layer involving 28 filters with a 3x3 kernel size, extracting features from input images. A subsequent max pooling layer reduces feature map dimensions, aiding computational efficiency. The flattened data is then processed by a dense layer with 128 neurons and ReLU activation. A dropout layer with a 20% deactivation rate helps prevent overfitting. Finally, a 10-neuron output layer with softmax activation produces class probabilities for classification tasks. This design is commonly used for image analysis, effectively extracting features and making predictions based on them.



```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10, activation=tf.nn.softmax))
```

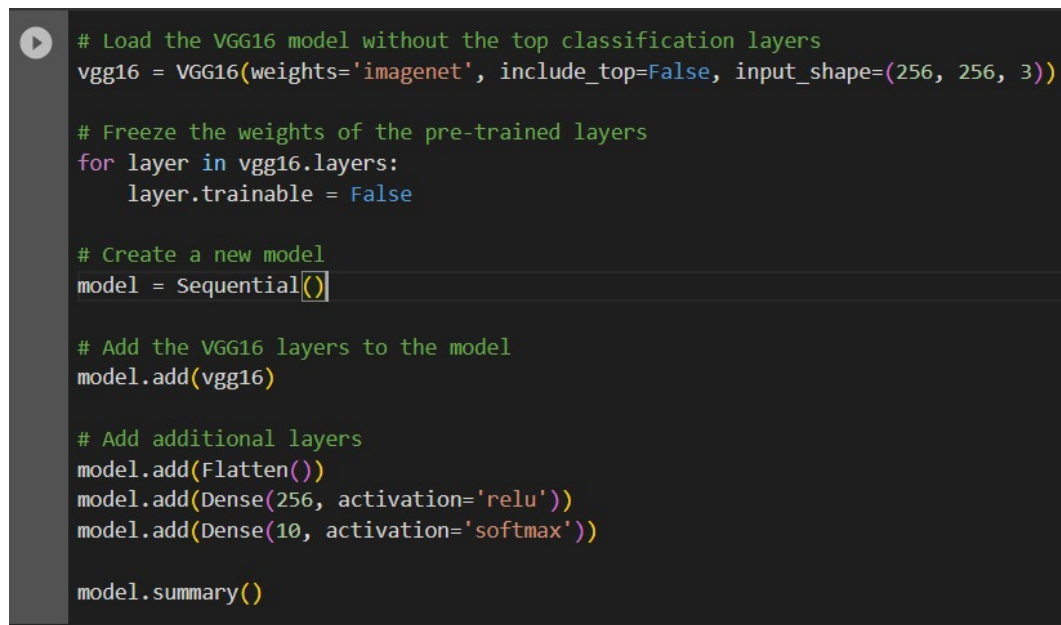
Figure 1:

## 2.2 Second Approach

In our second approach, the initial step involved loading the pre-trained VGG-16 model, excluding its top classification layers. This model, trained on the 'imagenet' dataset, is designed to extract intricate features from images. The input shape was defined as (256, 256, 3), indicating an image size of 256x256 pixels with 3 color channels (RGB).

We then froze the weights of the pre-trained layers to retain the previously learned features. A new sequential model was instantiated using Keras and the pre-trained VGG-16 layers were added to it. That formed the foundation for feature extraction from input images.

To adapt the VGG-16 layers for the specific classification task, extra layers were appended. First, a 'Flatten' layer reshaped the extracted features into a vector. Then, a dense layer with 256 neurons, employing the rectified linear unit (ReLU) activation function, contributed to feature transformation. Another dense layer with 10 neurons and softmax activation facilitated multi-class classification.



```
# Load the VGG16 model without the top classification layers
vgg16 = VGG16(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

# Freeze the weights of the pre-trained layers
for layer in vgg16.layers:
    layer.trainable = False

# Create a new model
model = Sequential()

# Add the VGG16 layers to the model
model.add(vgg16)

# Add additional layers
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

Figure 2:

## 3 Experiments and Results

### 3.1 Used Dataset

#### → MNIST dataset

- 70.000 images
- 28 X 28 Pixels
- Grayscale

#### → Streetnumber Dataset

- 714 images
- 256 X 256 Pixels
- RGB

### 3.2 Experimental Setup

- Experimental setup Model from scratch → Switch to VGG-16
- Learning rate adjustments
- Hyperparameter optimization

This experimental setup aims to enhance the performance of a machine-learning model for a specific task, such as image classification. The process is divided into two phases: creating a model from scratch and then switching to a pre-trained VGG-16 model, a commonly used deep neural network architecture. Initially, a custom model is built by designing layers like convolutional, pooling, and fully connected layers. After evaluating the scratch model's effectiveness, the experiment then shifts to utilizing the VGG-16 model, which has already been trained on a vast dataset for general image recognition tasks. This pre-trained model can potentially offer superior initial features and representations for the specific task. The learning rate is another factor adjusted to optimize the process. The learning rate identifies the step size taken in updating the model's parameters during training. By fine-tuning this hyperparameter, the goal is to strike the right balance between swift convergence and bypassing overshooting the optimal values. This iterative approach of transitioning from a custom model to a pre-trained one and tuning the learning rates represents a purposeful strategy to improve model performance, combining architectural know-how and hyperparameter refinement to achieve superior results.

### 3.3 Hyperparameter Tuning Results

Figure 3 illustrates the iterative process of training our model using 5 epochs and 8 batch sizes. Within each epoch, the model processes batches of 8 data points. It starts with a forward pass, making predictions based on input data. These predictions are compared to actual values to calculate a loss, representing performance. Backpropagation then adjusts model parameters to minimize the loss. This weight update refines predictions, iteratively improving the model's understanding through each batch. As epochs progress, the model's predictions converge towards accurate values, signifying enhanced performance.

```
Epoch 1/5
71/71 [=====] - 462s 6s/step - loss: 3.4927 - accuracy: 0.1441
Epoch 2/5
71/71 [=====] - 455s 6s/step - loss: 2.0794 - accuracy: 0.2544
Epoch 3/5
71/71 [=====] - 455s 6s/step - loss: 1.9379 - accuracy: 0.3310
Epoch 4/5
71/71 [=====] - 457s 6s/step - loss: 1.8449 - accuracy: 0.3203
Epoch 5/5
71/71 [=====] - 458s 6s/step - loss: 1.7349 - accuracy: 0.4021
```

Figure 3:

In Figure 4 below, we maintained the 5 epochs from the prior training procedure, but this time, we scaled up the batch size from 8 to 16.

```
Epoch 1/5
35/35 [=====] - 449s 13s/step - loss: 4.3385 - accuracy: 0.1408
Epoch 2/5
35/35 [=====] - 444s 13s/step - loss: 2.3566 - accuracy: 0.2780
Epoch 3/5
35/35 [=====] - 446s 13s/step - loss: 1.7924 - accuracy: 0.4025
Epoch 4/5
35/35 [=====] - 449s 13s/step - loss: 1.6534 - accuracy: 0.4025
Epoch 5/5
35/35 [=====] - 454s 13s/step - loss: 1.4584 - accuracy: 0.4804
```

Figure 4:

Moving on to Figure 5, we extended our training epochs to 10 and adjusted the batch size to 16. This alteration aimed to gauge the convergence of our model. Our observations align with previous instances where modifying these values impacted the training outcome. Notably, the results confirmed this effect, as these parameter adjustments produced the highest accuracy among the preceding cases.

```
Epoch 1/10
35/35 [=====] - 445s 13s/step - loss: 3.8096 - accuracy: 0.1480
Epoch 2/10
35/35 [=====] - 443s 13s/step - loss: 2.0470 - accuracy: 0.3357
Epoch 3/10
35/35 [=====] - 441s 13s/step - loss: 1.7811 - accuracy: 0.3736
Epoch 4/10
35/35 [=====] - 441s 13s/step - loss: 1.7420 - accuracy: 0.4332
Epoch 5/10
35/35 [=====] - 451s 13s/step - loss: 1.4892 - accuracy: 0.4910
Epoch 6/10
35/35 [=====] - 443s 13s/step - loss: 1.5002 - accuracy: 0.4874
Epoch 7/10
35/35 [=====] - 442s 13s/step - loss: 1.4084 - accuracy: 0.5036
Epoch 8/10
35/35 [=====] - 446s 13s/step - loss: 1.2426 - accuracy: 0.5704
Epoch 9/10
35/35 [=====] - 445s 13s/step - loss: 1.2145 - accuracy: 0.5830
Epoch 10/10
35/35 [=====] - 443s 13s/step - loss: 1.1531 - accuracy: 0.5903
```

Figure 5:

### 3.4 Results

Figure 6 depicts a consistent trend: the accuracy of our model rises with the increasing number of training epochs. This pattern suggests that our model is progressively improving its predictive performance as it undergoes more training iterations. The climbing accuracy signifies that our model is successfully learning and adapting to the dataset, becoming more skilled at making accurate predictions.

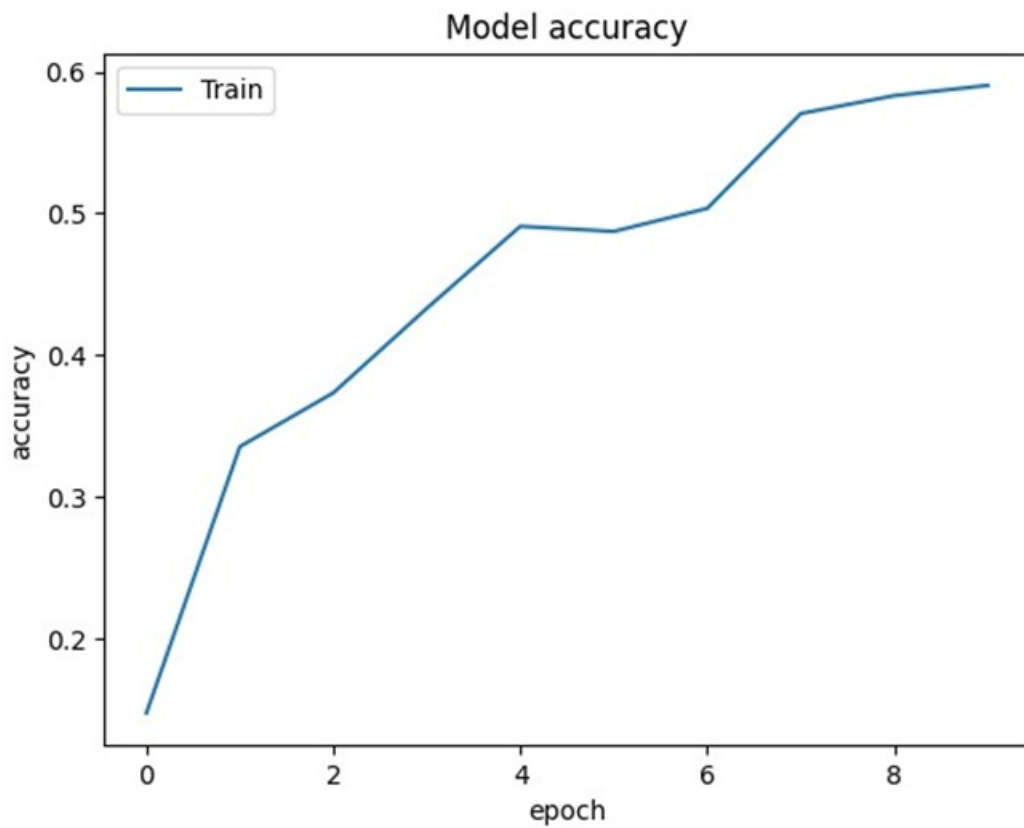


Figure 6:



In Figure 7 below, our model's loss decreases with each epoch, which signifies that the model's predictions are becoming progressively more accurate over time. The loss is a measure of the disparity between the model's predicted values and the actual values in our training data. As the model trains, it adjusts its internal parameters to minimize this disparity, effectively honing its ability to make better predictions. This iterative refinement involves fine-tuning the model's parameters so that it aligns more closely with the underlying patterns in the data. In essence, the decreasing loss indicates that our model is learning from the data and improving its predictive capabilities, which is a positive outcome in the training process.

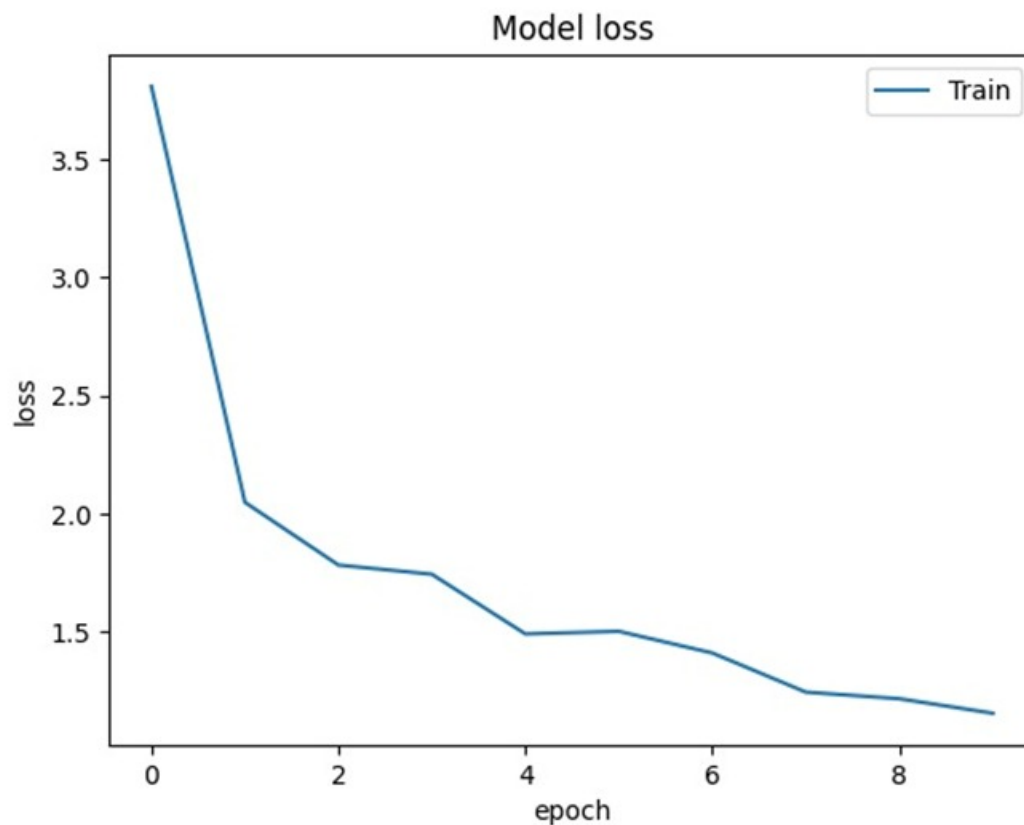


Figure 7:

Figure 8 illustrates the accuracy of the initial phase of our project. During this phase, we developed a CNN model from scratch using the mnist dataset.


| # | Team          | Members   | Score   |
|---|---------------|---|---------|
| 1 | KuchBhiRakhDo |  | 0.99889 |

Figure 8:

Figure 9 presents the model accuracy achieved during the second phase of our project. In this stage, we employed a pre-trained VGG-16 model with its previous weights and biases, utilizing the Adam optimizer. The training data used for this phase came from the StreetNumbers dataset.



Figure 9:

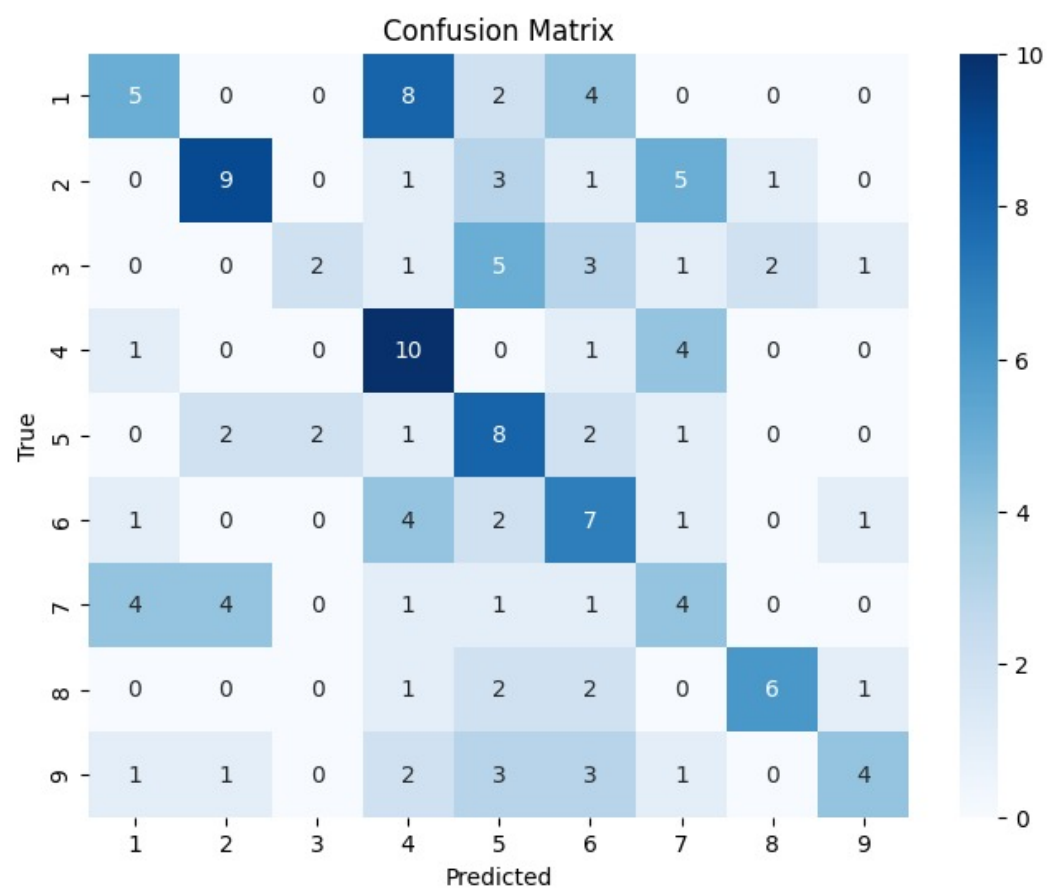


Figure 10:

## 4 Discussion

### 4.1 Results Discussion

In the context of neural network training, using a more aggressive learning rate entails utilizing a larger step size when updating the model's parameters during each optimization iteration, potentially leading to faster convergence but also an increased risk of overshooting optimal values. Employing a medium batch size of 16 involves dividing the training dataset into subsets of 16 samples each, and updating the model based on the average gradient computed from these subsets, striking a balance between computational efficiency and noise in gradient estimation. Opting for the Adam optimizer involves utilizing a more sophisticated optimization algorithm compared to traditional stochastic gradient descent (SGD), as Adam incorporates adaptive learning rates for each parameter, which can lead to improved convergence in complex optimization landscapes. Overall, these choices collectively aim to accelerate training by utilizing a larger learning rate and leveraging the benefits of both a medium batch size and the advanced Adam optimizer, potentially facilitating quicker convergence and enhanced optimization outcomes.

Figure 11 showcases the three most successful iterations of our model training, providing insights into our training accuracy, training loss in relation to epochs and batch size, as well as the optimizer used.

| S/N | Epoch | Batch Size | Optimizer | Train acc. | Train loss |
|-----|-------|------------|-----------|------------|------------|
| 1   | 5     | 8          | adam      | 0.4021     | 1.7349     |
| 2   | 5     | 16         | adam      | 0.4804     | 1.4584     |
| 3   | 10    | 16         | adam      | 0.5903     | 1.1531     |

Figure 11:

### 4.2 Positives and Key Findings

The neural network has successfully acquired the ability to learn from the data it has been exposed to, demonstrating effective generalization as it avoids overfitting, wherein it captures noise instead of genuine patterns. The process of hyperparameter optimization, involving the systematic tuning of parameters that influence the network's behavior, has produced positive results for our system. This is evidenced by a substantial performance gain of 10 percentage points, elevating the network's output from a 49% accuracy level to an improved 59% accuracy, showcasing the efficacy of the chosen hyperparameter configurations in enhancing the model's overall performance.

→ Network learns

- Network does not overfit
- hyperparameter optimization has proven effective for us (from 49% to 59% performance gain)

### 4.3 Limitations

The dataset used for training and testing the model is constrained by a finite number of images, potentially affecting the model's performance due to a limited variety of examples to learn from. This scarcity can hinder the network's capacity to generalize effectively to new, unseen data, as it may struggle to recognize patterns and variations that would be present in a more extensive dataset.

In addition to these challenges, the quality of the dataset itself poses a concern as it has not undergone thorough cleaning procedures. This means that the dataset includes images with multiple numbers present in a single frame, making it difficult for the model to correctly identify individual digits. Furthermore, the dataset contains images captured from a distance, introducing potential distortion and variations that are not representative of close-up images, and low-resolution images that lack the clarity necessary for accurate recognition. These suboptimal data quality issues collectively undermine the model's potential performance, as it struggles to learn from noisy, unclear, and unconventional data representations, thus highlighting the critical importance of data curation and preprocessing in achieving desirable outcomes.

## 5 Conclusion

In summary, our semester-long journey through deep learning has been nothing less than enlightening and fulfilling. Our exploration of the intricacies of neural networks, convolutional neural networks (CNNs), and their applications in computer vision has provided us with a deep understanding of the underlying mechanisms that underlie these innovative techniques.

Through diligent study and practical hands-on experience, we not only grasped the theoretical foundations but also translated this knowledge into practical know-how expertise. The Python library Keras became our canvas and allowed us to brush strokes of innovation as we implemented various neural network architectures. Engaging in these practical exercises has strengthened our understanding of the inner workings of neural networks, thereby enhancing our expertise in the subject matter.

A significant milestone in our journey was our exploration of transfer learning, a technique that significantly boosted the accuracy of our models. Leveraging pre-trained models like VGG-16 allowed us to showcase our versatility and grasp the immense potential of harnessing existing knowledge to address unique and challenging tasks.

In this dynamic and rapidly evolving field, the importance and significance of data augmentation cannot be understated. As we delved into this process, we recognized how manipulating and diversifying our data facilitated the robustness and generalization of our models, an indispensable capability skill in our pursuit of excellence.

As we complete our research, we realize that we have the technical know-how to create, train, and improve neural networks as well as an undying curiosity to learn more. Our continued development in the field of deep learning will be fueled by the foundations we've built and the insights we've obtained. We leave this chapter with the ability to decipher intricacies and the zeal to explore uncharted territory in the always changing field of artificial intelligence.

## **Submission of project works xAI-Proj**

### **Team Contribution**

#### **Shehroz Shafiq Khan**

- In-depth research and usage of different Python libraries for data manipulation and data visualization.
- Explored various Keras modules and layers, selecting those that best suited the project's requirements.
- Implemented a range of CNN architectures using Keras, tailoring each model to the specific requirements for both MNIST and StreetNumbers datasets.
- Integrated Keras's transfer learning capabilities, incorporating pre-trained CNN models like VGG16 to assess its suitability for the street number recognition task.
- Implemented data augmentation techniques to artificially expand the training dataset and conducted experiments to assess the impact of data augmentation on reducing overfitting.
- Led the team's efforts in physically visiting various neighborhoods and recording images of street house numbers for the dataset.

#### **Hafiz Ahmer Saeed Khan**

- Research on different CNN models and in-depth study of their architectures that led to the selection of specific CNN models used in the project.
- Implemented hyperparameter tuning to optimize the model's learning rate, batch size, and other parameters crucial for achieving high accuracy.
- Compiled the final project report, summarizing the methodology, results, and insights from our CNN model analysis.
- Put together the presentations and deliverables in all phases of the project.
- Visited different neighborhoods to collect images of street house numbers for the dataset.

**Muhammad Junaid**

- Research on different Computer Vision based problems and their solutions.
- Developed custom evaluation metrics specific to the street number recognition task, such as precision-recall curves and confusion matrices, to gain deeper insights into model performance.
- Explored different cloud-based technologies for model training, such as GCP and Google Colab to utilize cloud-based GPUs and TPUs for efficient training of the diverse CNN models, optimizing resource utilization.
- Collaborated with other team members to optimize model efficiency.
- Compiled the final project documentation.
- Visited different neighborhoods to collect images of street house numbers for the dataset.

**GIT repository :** <https://github.com/shehrozshafiqkh/xAI-deep-learning>

## References

1- Tyler Folkman: How To Get Started with Deep Learning

(<https://towardsdatascience.com/how-to-get-started-with-deep-learning-1f4e9f9b221e>)

2- Purva Huilgol : Getting into Deep Learning

(<https://www.analyticsvidhya.com/blog/2020/03/deep-learning-5-things-to-know/>)

3- Jason Brownlee : Transfer Learning in Keras with Computer Vision Models

(<https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>)

4- Jason Brownlee : Your First Deep Learning Project in Python with Keras Step-by-Step

(<https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>)

5- Derrick Mwit : A Practical Tutorial With Examples for Images and Text in Keras

(<https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras>)

6- Adrian Rosebrock : Transfer Learning with Keras and Deep Learning

(<https://pyimagesearch.com/2019/05/20/transfer-learning-with-keras-and-deep-learning/>)

7- Introduction to Keras

([https://keras.io/getting\\_started/](https://keras.io/getting_started/))

8- Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He : A Comprehensive Survey on Transfer Learning

(<https://arxiv.org/abs/1911.02685>)

## **Declaration of Authorship**

All final papers have to include the following ‘Declaration of Authorship’:

## **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, August 31, 2023

---

(Place, Date)

Shehroz Shafiq Khan

---

(Signature)

Bamberg, August 31, 2023

---

(Place, Date)

Hafiz Ahmer Saeed Khan

---

(Signature)

Bamberg, August 31, 2023

---

(Place, Date)

Muhammad Junaid

---

(Signature)