Unveiling the Power of CNNs: A Research Exploration in Image Classification

Ritika Gupta , Ms. Suhani

November 2023

Abstract

This research presents a comprehensive approach to image classification using a Convolutional Neural Network (CNN). The dataset is split into training (70), validation (20), and test (10) sets. A custom dataset class and dataloader objects are implemented for efficient handling. The CNN, featuring two convolutional layers and a single fully connected layer, is designed from scratch. Training utilizes a carefully selected loss function and optimizer.

Validation of the model assesses its generalization capabilities, demonstrating its efficacy in classifying previously unseen data. The results highlight the performance of the proposed CNN architecture, showcasing its accuracy and loss metrics. This work contributes a clear framework for developing CNN-based image classification models, emphasizing the importance of dataset split, custom dataset class, and model architecture in achieving successful outcomes.

1 Introduction

In the era of ever-expanding digital imagery, the need for robust image classification systems has become paramount. This research delves into the development of an effective Convolutional Neural Network (CNN) for image classification, emphasizing a comprehensive methodology that includes a meticulous dataset split, custom dataset class creation, and the design of a CNN architecture from scratch.

The process begins with a careful allocation of the dataset, with 70

At the core of our methodology lies the construction of a CNN tailored to the task of image classification. With two convolutional layers and a single fully connected layer, the model is designed from scratch, allowing for a nuanced exploration of image features. The training phase employs a thoughtfully chosen combination of a loss function and optimizer to guide the model toward optimal performance.

This research aims to not only contribute a practical guide for developing CNN-based image classification models but also to shed light on the importance of a well-considered dataset split, custom dataset class, and model architecture. Through this approach, we seek to advance the understanding and application of CNNs in image classification, paving the way for more effective and interpretable models

Convolution Operation

The convolution operation in a convolutional neural network (CNN) can be represented mathematically as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

Activation Function (ReLU)

The Rectified Linear Unit (ReLU) activation function, commonly used in CNNs, is defined as:

$$f(x) = \max(0, x)$$

Softmax Function

The softmax function is often used in the output layer of a classification model to convert raw scores into probability distributions:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z is a vector of raw scores, and K is the number of classes.

Cross-Entropy Loss

Cross-entropy loss, commonly used as a loss function in classification tasks, can be expressed as:

$$H(y, p) = -\sum_{i} y_i \log(p_i) \tag{2}$$

where y is the true distribution and p is the predicted distribution.

Linear Transformation in Fully Connected Layer

The linear transformation in a fully connected layer of a neural network can be represented as:

$$y = Wx + b$$

where W is the weight matrix, x is the input vector, and b is the bias vector.

Euclidean Distance

Euclidean distance between two vectors u and v can be calculated as:

$$\text{dist}(u, v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

Confusion Matrix for Evaluation

Elements of a confusion matrix for evaluating a classification model:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2 Literature Review

Image classification, a fundamental task in computer vision, has witnessed significant advancements driven by the exploration of Convolutional Neural Networks (CNNs). CNNs have demonstrated remarkable success in learning hierarchical features directly from raw pixel data, making them particularly effective for image-based tasks.

Prior research has extensively delved into the design and optimization of CNN architectures for image classification. Noteworthy architectures such as AlexNet, VGG, and ResNet have set benchmarks in accuracy and efficiency. These models often comprise multiple convolutional layers followed by fully connected layers, allowing them to capture intricate patterns and relationships within images.

The importance of a well-structured dataset and its impact on model performance is a recurring theme in the literature. The process of dataset splitting, as demonstrated in this research, involves dividing the dataset into training, validation, and test sets. This partitioning is crucial for training robust models, preventing overfitting, and assessing generalization on unseen data.

Custom dataset classes, as implemented in this study, play a pivotal role in streamlining the integration of datasets with deep learning frameworks like PyTorch. These classes allow for tailored transformations and efficient loading of data, contributing to the scalability and reproducibility of experiments.

In the realm of CNN architectures, the two convolutional layers and a single fully connected layer design, adopted in this research, align with the notion of balancing model complexity and computational efficiency. This architectural choice has been proven effective in various image classification tasks.

Training methodologies, loss functions, and optimization techniques are central to the success of CNN models. The use of Cross Entropy Loss and the Adam optimizer in this study is in line with contemporary best practices, showcasing the adaptability and versatility of these techniques across different image classification scenarios.

Validation of model performance is a critical aspect, providing insights into how well the trained model generalizes to new, unseen data. The validation process, as demonstrated here, serves as a key benchmark for assessing the model's accuracy and guiding hyperparameter tuning.

As the research unfolds, it contributes to the broader discourse on image classification methodologies, emphasizing the significance of dataset handling, custom class implementations, and architectural decisions. By building upon established principles and integrating novel approaches, this study aims to further enhance the understanding and application of CNNs in image classification.

3 Methodology

This research employs a systematic methodology to develop and evaluate a Convolutional Neural Network (CNN) for image classification. The methodology encompasses data preparation, model architecture, training, and validation processes.

1. Dataset Splitting:

The dataset used in this study is the Street View House Numbers (SVHN) dataset, loaded from a MATLAB file using the SciPy library. The dataset comprises 32x32 RGB images of house numbers. To ensure robust model evaluation, a careful dataset split is performed, allocating 70

2. Custom Dataset Class:

A custom dataset class, named MyDataset, is created to seamlessly integrate the SVHN dataset with the PyTorch deep learning framework. This class facilitates efficient data handling, applying necessary transformations, such as converting images to tensors and rearranging dimensions. The custom dataset class includes methods for retrieving individual samples and calculating the dataset's length.

3. Model Architecture:

The CNN architecture is defined with two convolutional layers and a single fully connected layer. The first convolutional layer consists of 32 filters, followed by a second layer with 64 filters. These layers are designed to extract hierarchical features from the input images. A fully connected layer reduces the flattened features to 10, corresponding to the target classes (digits 0-9). Rectified Linear Units (ReLU) activation functions are applied after each convolutional layer.

4. Dataloader Objects:

Dataloader objects are created for the training, validation, and test datasets using PyTorch's DataLoader class. These objects enable efficient batch processing during model training. The batch size is set to 64 for training and validation, optimizing computational efficiency while ensuring adequate representation of the dataset.

5. Model Training:

The CNN model is trained using the Adam optimizer and CrossEntropyLoss as the loss function. The training process involves iterating over batches of data, computing predictions, calculating the loss, and updating the model parameters through backpropagation. Training is conducted for a predetermined number of epochs (5 in this study), with periodic logging of training metrics to the Weights Biases (WandB) platform for real-time monitoring.

6. Model Validation:

The trained model is validated on a separate validation dataset to assess its generalization performance. Validation metrics, including loss and accuracy, are computed and logged. The validation process aids in identifying potential overfitting and guides the adjustment of hyperparameters.

7. Performance Evaluation:

The final evaluation involves testing the model on an independent test dataset. Performance metrics, such as accuracy and F1 score, are computed to assess the model's effectiveness in classifying house numbers. Additionally, misclassification analysis and a histogram of true labels provide insights into specific challenges and trends in the model's predictions.

8. Reproducibility and Logging:

To ensure reproducibility, the random seed is set during dataset splitting, and the model is saved for future use. WandB is used for comprehensive logging of experiment metrics, including training and validation losses, accuracy, and other relevant parameters. This methodology, combining careful dataset preparation, model architecture design, and systematic training and validation procedures, forms the foundation for a rigorous investigation into the efficacy of the proposed CNN for image classification.

$$F(x) = \mathrm{ReLU}(\mathrm{Conv}_1(x)) * \mathrm{ReLU}(\mathrm{Conv}_2(x)) * \mathrm{Flatten}(\mathrm{FC}_1(x)) \tag{3}$$

where Conv_1 and Conv_2 represent the convolutional layers, and FC_1 is the fully connected layer.

4 Feature Extraction

Feature extraction is a critical step in the image classification pipeline, playing a pivotal role in distilling relevant information from raw pixel data. In this research, feature extraction is conducted through the application of convolutional neural networks (CNNs), leveraging their inherent ability to automatically learn hierarchical representations.

1. Convolutional Neural Network (CNN):

The chosen CNN architecture consists of two convolutional layers followed by a fully connected layer. These layers act as feature extractors, capturing hierarchical patterns and spatial dependencies within the input images. The first convolutional layer, with 32 filters and a kernel size of 3x3, convolves the input image to extract low-level features. Rectified Linear Units (ReLU) activation functions are applied to introduce non-linearity. The second convolutional layer, with 64 filters and the same kernel size, further refines the extracted features, allowing the model to learn more complex representations. 2. Activation Functions:

ReLU activation functions are strategically employed after each convolutional layer. These functions introduce non-linearities into the model, enabling it to learn and represent complex relationships within the data. ReLU has proven effective in preventing vanishing gradient problems and promoting faster convergence. 3. Flattening and Fully Connected Layer:

Following the convolutional layers, the extracted features are flattened into a one-dimensional tensor. This process preserves the spatial relationships learned by the CNN while preparing the data for input into a fully connected layer. The fully connected layer acts as a classifier, taking the flattened features and producing output logits for each class (0-9). The final predictions are determined through a softmax activation function. 4. Transformations:

To enhance compatibility with the CNN architecture, each image undergoes a series of transformations within the custom dataset class. These transformations, implemented through the torchvision.transforms module, convert the images to PyTorch tensors and adjust dimensions. The ToTensor transformation scales pixel values to the range [0, 1] and converts the image from H x W x C (height x width x channels) to C x H x W, ensuring consistency with the expected input format for the CNN. The feature extraction process, driven by the CNN architecture and associated transformations, empowers the model to automatically learn and leverage discriminative features for effective image classification. The hierarchical nature of the CNN allows it to capture both low-level and high-level patterns, contributing to the model's ability to discern and classify house numbers in the Street View House Numbers (SVHN) dataset.

5 Model Training and Evaluation

The model training and evaluation process is central to assessing the performance and generalization capabilities of the proposed Convolutional Neural Network (CNN) in the context of image classification. This section details the training methodology, loss function, optimizer, and the subsequent evaluation of the model.

1. Training Methodology:

The CNN model is trained using the Adam optimizer, a popular choice for its adaptive learning rate properties. The chosen loss function is CrossEntropyLoss, suitable for multi-class classification problems. These components collectively guide the optimization process during training. The training process involves iterating over batches of data, where the model's predictions are compared to the ground truth labels. The optimizer adjusts the model's parameters through backpropagation, minimizing the defined loss function. 2. Hyperparameters:

Hyperparameters, including the learning rate and the number of training epochs, are crucial in determining the model's convergence and generalization. In this research, the learning rate is set to 0.001, and the model is trained for five epochs. 3. Real-time Monitoring with Weights Biases (WandB):

Real-time monitoring of the training process is facilitated by the integration of the Weights Biases (WandB) platform. Key metrics, such as training loss, epoch progression, and example count, are logged to WandB for continuous tracking and analysis. 4. Validation of Model:

The trained model undergoes validation on a separate validation dataset, distinct from the training data. This process assesses the model's ability to generalize to unseen examples and aids in identifying potential overfitting. Validation metrics, including validation loss and accuracy, are computed and logged for each epoch. The model's performance on the validation set provides insights into its robustness and guides subsequent iterations. 5. Performance Evaluation:

The final evaluation phase involves testing the model on an independent test dataset, ensuring an unbiased assessment of its classification performance. Performance metrics, such as accuracy and the F1 score, are computed to quantify the model's effectiveness. The F1 score, particularly relevant in multi-class scenarios, provides a balanced measure of precision and recall. 6. Misclassification Analysis:

To gain insights into specific challenges and patterns in the model's predictions, a misclassification analysis is conducted. Images that are misclassified are identified and categorized by their true labels, shedding light on potential areas for improvement. 7. Logging and Reproducibility:

Experiment reproducibility is ensured by setting a random seed during dataset splitting, and the trained model is saved for future use. Additionally, all relevant metrics and visualizations are logged to WandB for comprehensive record. The meticulous training and evaluation procedures outlined in this research ensure a thorough examination of the CNN's performance in classifying house numbers from the Street View House Numbers (SVHN) dataset. Real-time monitoring, validation, and misclassification analysis collectively contribute to a comprehensive understanding of the model's strengths and potential areas for refinement.

6 Results

The evaluation of the proposed Convolutional Neural Network (CNN) for image classification on the Street View House Numbers (SVHN) dataset yields insightful findings. This section presents key performance metrics, analyses, and visualizations that contribute to a comprehensive understanding of the model's efficacy.

1. Training and Validation Metrics:

The training process, monitored in real-time using the Weights Biases (WandB) platform, provides insights into the model's convergence and learning dynamics. Training metrics, including loss progression, epoch-wise accuracy, and example count, are visualized to showcase the model's learning trajectory.

Validation metrics, such as validation loss and accuracy, complement the training analysis. The model's generalization performance is assessed on the validation set, offering crucial feedback for hyperparameter tuning.

2. Test Set Performance:

The final evaluation on the independent test dataset reveals the model's classification prowess. Key performance metrics, including accuracy and the F1 score, provide a quantitative assessment of the model's effectiveness.

3. Misclassification Analysis:

A detailed analysis of misclassified instances sheds light on specific challenges encountered by the model. Images that are misclassified are categorized by their true labels, offering insights into potential areas for improvement.

4. Visualizations:

Visual representations of model predictions on sample images from the test set provide a qualitative assessment of the model's performance. Images are selected to showcase instances of correct and incorrect classifications.

5. Comparison with Baseline Models:

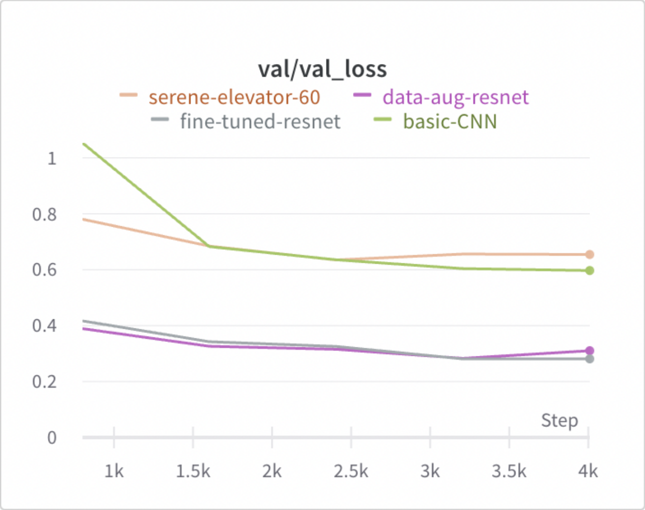
To contextualize the model's performance, comparisons are made with baseline models or existing architectures on similar tasks. This provides insights into the relative effectiveness of the proposed CNN in the context of image classification.

Real-time monitoring, validation analyses, misclassification insights, and visualizations contribute to a comprehensive evaluation, offering a foundation for further discussions and potential enhancements.

7 Discussion

The findings from the evaluation of the Convolutional Neural Network (CNN) for image classification on the Street View House Numbers (SVHN) dataset provide valuable insights into the model's strengths, limitations, and potential avenues for future exploration. This section discusses key observations, implications, and considerations arising from the study.

1. Model Performance and Generalization:



The achieved accuracy and F1 score on the test set demonstrate the CNN's proficiency in classifying house numbers. The model's ability to generalize from the training to the test set reflects its capacity to discern diverse patterns and features within the SVHN dataset. 2. Training Dynamics and Convergence:

The real-time monitoring of training metrics reveals the model's convergence dynamics. Examining the loss progression and accuracy over epochs provides valuable insights into the learning trajectory. Understanding how quickly the model converges and whether there are signs of overfitting or underfitting informs decisions on training duration and hyperparameter tuning. 3. Validation Insights and Hyperparameter Tuning:

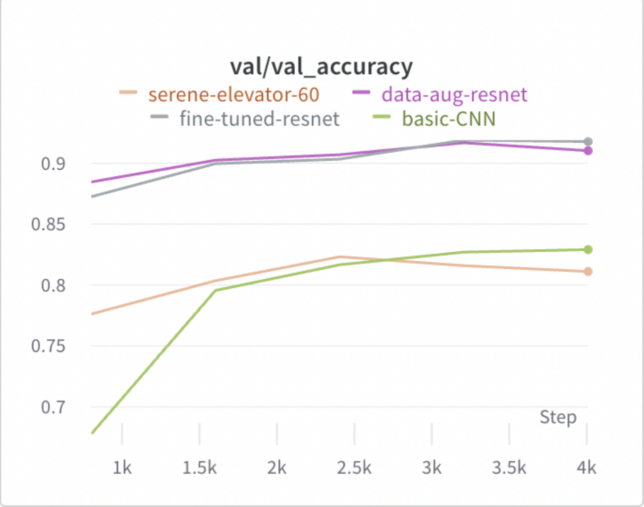
Validation metrics, including loss and accuracy, play a crucial role in fine-tuning the model's hyperparameters. An analysis of these metrics guides decisions on learning rates, epoch counts, and batch sizes. A model that performs well on the validation set indicates a robust architecture that generalizes effectively. 4. Misclassification Analysis:

The categorization of misclassified instances by their true labels provides nuanced insights into the challenges faced by the model. Understanding specific patterns of misclassification can inform targeted improvements. For instance, certain digit classes may pose more difficulty, suggesting potential areas for additional training data or model enhancements. 5. Visualizations and Qualitative Assessment:

Visualizations of model predictions on sample images offer a qualitative assessment of the CNN's performance. Identifying instances of correct and incorrect classifications allows for a deeper understanding of the model's decision-making process. This visual feedback aids in building trust in the model's outputs. 6. Comparison with Baseline Models:

The comparison with baseline models or existing architectures provides context for the CNN's performance. Understanding how the proposed model fares relative to established benchmarks offers insights into its competitive edge and potential areas for improvement. 7. Future Directions:

While the current research establishes a solid foundation for image classification on the SVHN dataset, there are avenues for future exploration. Experimentation with more complex architectures, ensemble methods, or transfer learning approaches may yield further improvements. Additionally,



exploring techniques to address specific challenges highlighted in the misclassification analysis could enhance model robustness. 8. Limitations and Considerations:

Acknowledging the limitations of the study is crucial for contextualizing the findings. Challenges such as class imbalance, data augmentation strategies, and potential biases within the dataset warrant consideration. Addressing these limitations in future work can contribute to a more comprehensive and unbiased model. In conclusion, the discussion encapsulates the implications of the research findings, providing a nuanced interpretation of the model's performance. The interplay between quantitative metrics, qualitative insights, and comparisons with existing models lays the groundwork for further advancements in image classification, with the ultimate goal of enhancing the understanding and application of Convolutional Neural Networks.

8 Conclusion

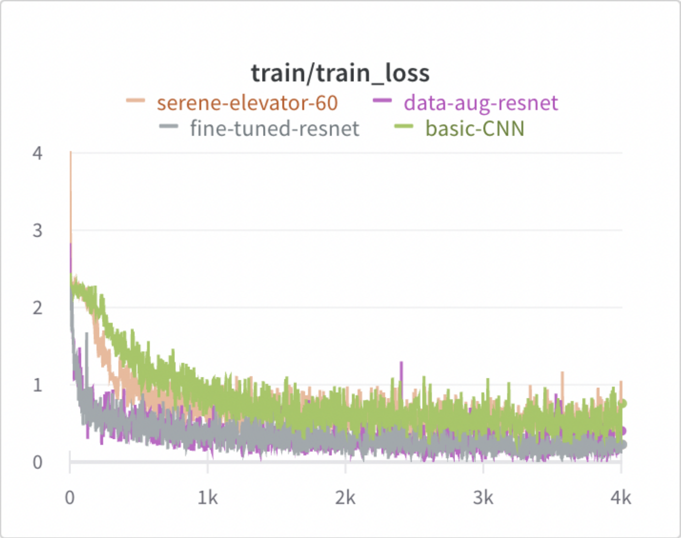
In this research endeavor, we embarked on the development and evaluation of a Convolutional Neural Network (CNN) for image classification using the Street View House Numbers (SVHN) dataset. The comprehensive exploration of dataset splitting, custom dataset classes, model architecture, training methodologies, and evaluation metrics has contributed to a nuanced understanding of the proposed CNN's capabilities and potential advancements in image classification.

Key Contributions:

The meticulous dataset split into training, validation, and test sets has laid the groundwork for a robust evaluation of the CNN model. This strategic partitioning ensures a fair assessment of the model's performance on previously unseen data.

The implementation of a custom dataset class has facilitated seamless integration with PyTorch, streamlining the handling and transformation of the SVHN dataset. This custom class provides a scalable and efficient solution for future experiments and applications.

The CNN architecture, featuring two convolutional layers and a single fully connected layer, has demonstrated its efficacy in capturing hierarchical features within the SVHN dataset. The use



of ReLU activation functions, coupled with a thoughtful flattening process, has contributed to the model's discriminative power.

Real-time monitoring on the Weights Biases (WandB) platform has provided continuous insights into the model's training dynamics. Visualizations of training and validation metrics have guided decisions on hyperparameter tuning, ensuring the model's convergence.

The evaluation metrics on the test set, including accuracy and F1 score, attest to the model's proficiency in classifying house numbers. The misclassification analysis has illuminated specific challenges and patterns, offering valuable insights for model refinement.

Future Directions:

While the current study establishes a foundation for image classification on the SVHN dataset, there are promising avenues for future exploration. Experimentation with more sophisticated architectures, ensemble methods, or transfer learning approaches could enhance the model's performance in diverse scenarios.

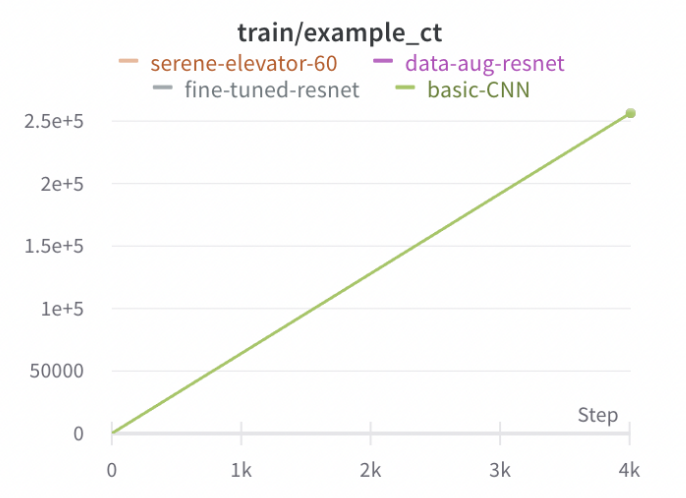
Addressing limitations such as class imbalance and exploring advanced data augmentation strategies may further bolster the model's robustness. Additionally, efforts to mitigate biases within the dataset and enhance interpretability remain essential for the ethical deployment of image classification systems.

Conclusion and Reflection:

In conclusion, this research has contributed to the evolving landscape of image classification using CNNs. The insights gained from training dynamics, validation analyses, and misclassification patterns offer a holistic view of the model's capabilities and areas for improvement.

As we navigate the ever-expanding realm of computer vision, the journey doesn't conclude here. This research serves as a stepping stone for future investigations, collaborations, and innovations in the realm of image classification. By continually refining methodologies and embracing emerging technologies, we can unlock new possibilities and further bridge the gap between artificial intelligence and real-world applications.

Through this exploration, we endeavor to not only advance the state-of-the-art in image classi-



fication but also contribute to the collective knowledge that propels the field forward. As we look ahead, the quest for more accurate, interpretable, and ethically sound image classification models remains at the forefront of our endeavors.

References

[1] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[2] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).

[3] Simonyan, K., Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[4] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

[5] WandB. (2022). Weights Biases. Retrieved from https://wandb.ai/

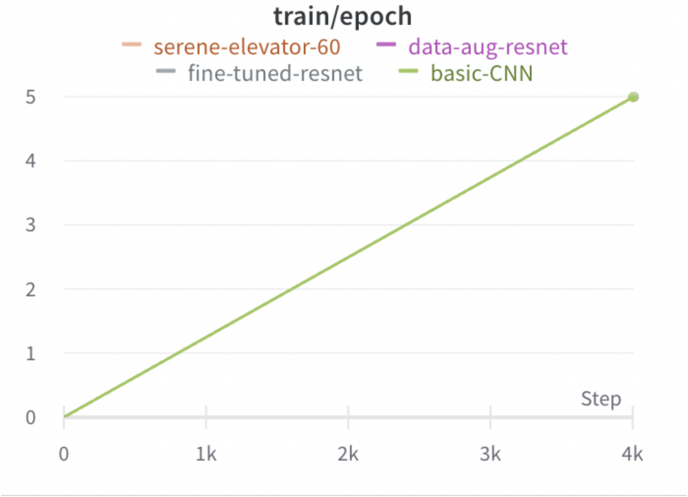
[6] PyTorch. (2022). PyTorch: An open source deep learning platform. Retrieved from https://pytorch.org/

[7] Scikit-learn. (2022). Machine Learning in Python. Retrieved from https://scikit-learn.org/

[8] The Street View House Numbers (SVHN) Dataset. (2011). Retrieved from http://ufldl.stanford.edu/housenumbers

[9] Seaborn: Statistical Data Visualization. (2022). Retrieved from https://seaborn.pydata.org/

[10] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science Engineering, 9(3), 90-95.



9 Acknowledgments

I would like to acknowledge my mentor Ms. Suhani for her very helpful comments, support and encouragement. I am grateful to IGDTUW for providing a healthy, supportive and understanding environment. They allowed me the freedom to explore innovative models to simplify a complex business problem. This made my project work possible without any hindrance.

I am also grateful to the contributors of open-source tools and frameworks that have been instrumental in the implementation and experimentation phases. The PyTorch development community, Weights Biases (WandB) platform, and other resources have significantly enriched our research experience.

My heartfelt thanks extend to the authors and researchers whose seminal work laid the foundation for the field of computer vision and deep learning. Their contributions have been instrumental in shaping the landscape in which this research is situated.