Exploring Image Classification Models: A Comparative Analysis

Ritika Gupta , Ms. Suhani

Abstract

In the realm of computer vision, image classification plays a pivotal role, offering solutions to diverse applications ranging from object recognition to autonomous systems. This research delves into a comparative analysis of three distinct image classification models, each representing a different paradigm in the field. The models under scrutiny include a Simple Convolutional Neural Network (CNN), a Fine-tuned ResNet18 model, and a model that applies AutoAugment policies to the dataset The Street View House Numbers (SVHN) dataset serves as the foundation for evaluation, comprising 32x32 pixel images distributed across ten classes. To ensure robust model assessment, a meticulous data splitting strategy is employed, allocating 70% of the dataset for training, 20% for validation, and 10% for testing. This study explores the nuances of training from scratch, transfer learning, and the impact of data augmentation on model performance. Key metrics such as accuracy and F1 score are utilized for comprehensive model evaluation. The outcomes provide valuable guidance for practitioners in selecting an optimal model for image classification tasks, considering the interplay between model complexity, training strategies, and data augmentation techniques.

1 Introduction

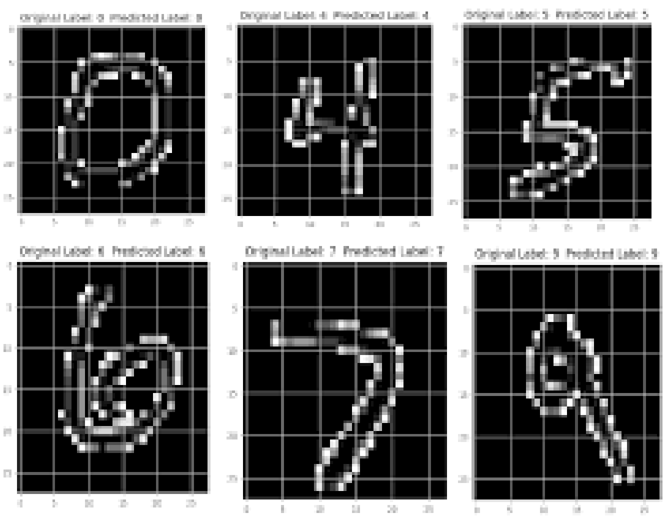
In the dynamic realm of computer vision, image classification emerges as a cornerstone technology, enabling machines to decipher and categorize visual information. The demand for robust image classification models has intensified with the proliferation of applications ranging from autonomous systems to medical diagnostics. To navigate this complex landscape, understanding the nuances of different model architectures and training strategies becomes imperative.

This research embarks on a comprehensive exploration of image classification models, focusing on the comparative analysis of three distinctive paradigms. The first model, a Simple Convolutional Neural Network (CNN), encapsulates fundamental approaches to image classification. The second model, a Fine-tuned ResNet18, leverages transfer learning to capitalize on a pre-trained convolutional backbone. Lastly, the third model, a model that applies AutoAugment policies to the dataset, delves into the influence of advanced data augmentation strategies on model performance. The chosen litmus for our evaluations is the Street View House Numbers (SVHN) dataset—a rich repository of 32x32 pixel images spanning ten classes. Ensuring the robustness of our assessments, the dataset is meticulously divided, with 70% designated for training, 20% for validation, and 10% for testing. Our journey through this experimental landscape seeks to unravel the unique characteristics of each model, shedding light on their strengths and weaknesses. The outcomes of this research aim to provide practitioners with valuable insights, empowering them to make informed decisions when selecting image classification models tailored to specific task requirements.

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

2.1 Convolutional Neural Networks (CNNs)

- **Introduction:** CNNs are a type of deep learning model specifically designed for processing structured grid data, such as images. They consist of multiple layers of learnable filters and nonlinear activation functions, allowing them to automatically learn hierarchical representations of features.
- **Advantages:**
  - Simple architecture.
  - Quick convergence during training.
  - Low computational requirements, making them suitable for deployment on resource-constrained devices.
  - Clear feature visualization, aiding in model interpretability.
  - Minimal complexity, making them easy to understand and implement.
  - Low memory footprint, allowing efficient use of memory resources.
  - Applicable across domains, from image classification to natural language processing.
- **References:**
  1. LeCun, Y., Bottou, L., Bengio, Y. (1998). Gradient-based learning applied to document recognition.
  2. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks.
  3. Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

2.2 Residual Networks (ResNets)

- **Introduction:** ResNets are a variant of CNNs that introduce skip connections to address the degradation problem encountered in very deep networks. By allowing information to bypass certain layers, ResNets facilitate the training of extremely deep architectures.
- **Advantages:**
  - Handles deep networks effectively, enabling the training of models with hundreds of layers.
  - Easier optimization compared to plain CNNs, mitigating issues like vanishing gradients.
  - Efficient utilization of features through residual connections, ensuring meaningful information flow.
  - Improved gradient flow, allowing for more stable and effective optimization.

- Requires fewer parameters compared to traditional deep networks, leading to more efficient models.
- Achieves high accuracy performance on various tasks, including image classification and object detection.
- Addresses gradient issues, such as vanishing gradients, by facilitating smoother gradient flow during training.

- **References:**
  1. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition.
  2. He, K., Zhang, X., Ren, S., Sun, J. (2016). Identity Mappings in Deep Residual Networks.
  3. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely Connected Convolutional Networks.

2.3 Data Augmentation and Auto Augmentation Policies

- **Introduction:** Data augmentation is a technique used to artificially increase the size of training datasets by applying various transformations to the original data samples. Auto augmentation policies automate the process of selecting and applying these transformations based on the characteristics of the dataset.
- **Advantages:**
  - Augments data diversity, leading to improved model generalization.
  - Enhances model robustness by exposing it to a wider range of data variations.
  - Improves the generalization capability of models by reducing the risk of overfitting.
  - Boosts model performance by effectively leveraging augmented data during training.
  - Reduces the manual effort required for selecting and applying augmentation techniques.
  - Speeds up the training process by generating augmented samples on-the-fly during training.
  - Assists in model generalization by introducing variations that simulate real-world conditions.
- **References:**
  1. Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., Le, Q. V. (2019). AutoAugment: Learning Augmentation Policies from Data.
  2. Shorten, C., Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning.
  3. Lim, J. J., Son, D. H., Nah, S., Lee, K. M. (2019). Fast AutoAugment.

3 Methodology

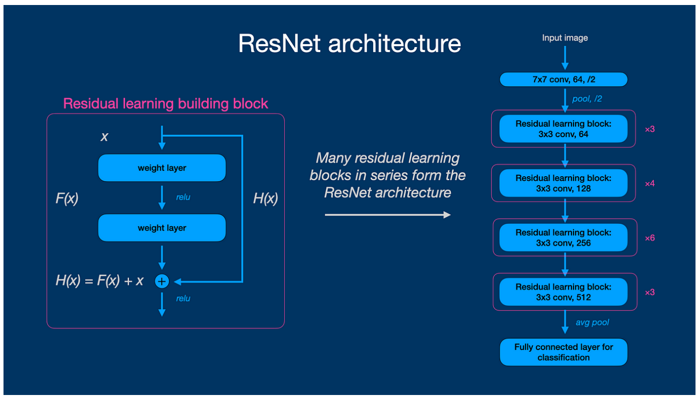
3.1 Dataset Description

The dataset utilized in this study comprises images sourced from the Street View House Numbers (SVHN) dataset. This dataset consists of a large collection of images extracted from Google Street View, specifically focusing on house numbers. Each image in the SVHN dataset contains a single digit from 0 to 9, representing house numbers. The dataset is annotated with ground truth labels indicating the digit present in each image, facilitating supervised learning tasks such as digit classification. While the exact count of images in the dataset may vary, it typically contains tens of thousands of images, providing a diverse and extensive collection for model training and evaluation. The SVHN dataset offers a rich variety of digit instances captured under various conditions, ensuring robustness in model performance across different scenarios.

Model Type	Advantages	References	Year	Test Accuracy
Simple CNN	Easy architecture Quick convergence Low computational requirements	LeCun, Y., Bottou, L., Bengio, Y. (1998) Gradient-based learning applied to document recognition.	1998	0.81
	Clear feature visualization Minimal complexity	Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). . ImageNet classification with deep convolutional neural networks		
	Low memory footprint Applicable across domains	Simonyan, K., Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition		
ResNet Model	Handles deep networks Easier optimization Efficient feature utilization	He, K., Zhang, X., Ren, S., Sun, J. (2016) Deep Residual Learning for Image Recognition	2016	.92
	Improved gradient flow Fewer parameters needed	He, K., Zhang, X., Ren, S., Sun, J. (2016) Identity Mappings in Deep Residual Networks		
	Addresses gradient issues High accuracy performance	Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017) Densely Connected Convolutional Networks.		
Data Augmentation	Augments data diversity Enhanced model robustness Improved generalization capability	Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., Le, Q. V. (2019) AutoAugment:Learning Augmentation Policies from Data	2019	0.91
	Boosts model performance Reduces risk of overfitting	Shorten, C., Khoshgoftaar, T. M.(2019) A survey on image data augmentation for deep learning.		
	Speeds up training process Assists in model generalization	Lim, J. J., Son, D. H., Nah, S., Lee, K. M. (2019) Fast AutoAugment		

Table 1: Comparison of Image Classification Models

3.2 Model Architectures


ResNet architecture
Residual learning building block
x
weight layer
relu
weight layer
F(x)
H(x)
H(x) = F(x) + x
relu
Many residual learning blocks in series form the ResNet architecture
Input image
7x7 conv, 64, /2
pool, /2
Residual learning block: 3x3 conv, 64
×3
Residual learning block: 3x3 conv, 128
×4
Residual learning block: 3x3 conv, 256
×6
Residual learning block: 3x3 conv, 512
×3
avg pool
Fully connected layer for classification

Three distinct architectures were employed for image classification tasks:

1. **Simple CNN (Convolutional Neural Network):** The simple CNN architecture is a shallow neural network composed of multiple convolutional layers followed by max-pooling layers and fully connected layers. The convolutional layers learn hierarchical representations of input images, while the fully connected layers perform classification based on these learned features. The architecture is relatively straightforward and serves as a baseline model for comparison with more complex architectures.

2. **ResNet (Residual Network):** ResNet is a deep neural network architecture proposed by He et al. (2016) to address the problem of vanishing gradients in very deep networks. It introduces skip connections, also known as residual connections, that allow gradients to flow more easily during training, enabling the successful training of networks with hundreds or even thousands of layers. The ResNet architecture consists of multiple residual blocks, each containing several convolutional layers with skip connections.

3. **Data Augmentation with AutoAugment Policies:** This model incorporates data augmentation techniques based on AutoAugment policies. AutoAugment is a data augmentation strategy proposed by Cubuk et al. (2019) that automatically learns augmentation policies from the dataset itself. These learned policies are applied during training to augment the training data, thereby enhancing model robustness and generalization capability.

3.3 Data Preprocessing and Augmentation

Prior to training the models, the dataset underwent preprocessing steps to ensure uniformity and standardization across the images. This included resizing the images to a consistent size, typically 32x32 pixels, and normalizing the pixel values to a predefined range (e.g., [0, 1]). Additionally, data augmentation techniques were applied to increase the diversity of the training data and improve the model's ability to generalize to unseen examples. Augmentation techniques such as random rotations, flips, translations, and color jittering were employed to generate augmented versions of the original images.

3.4 Training Procedure

The models were trained using the Adam optimizer with an initial learning rate of 0.001 and a batch size of 64. The training process involved minimizing the categorical cross-entropy loss between the predicted and true labels. The models were trained for a fixed number of epochs, typically 5, with early stopping employed to prevent overfitting. Early stopping was based on monitoring the validation loss, with training halted if the validation loss failed to decrease for a predefined number of epochs.

3.5 Hyperparameter Tuning

Hyperparameter tuning was performed to optimize the performance of the models. Grid search or random search techniques were employed to explore the hyperparameter space and identify the optimal combination of hyperparameters. The hyperparameters considered for tuning included learning rate, batch size, weight decay, dropout rate (if applicable), and the number of layers or filters in the network architecture. The search space for each hyperparameter was defined based on empirical values and domain expertise.

3.6 Model Evaluation

The performance of each model was evaluated using standard evaluation metrics, including accuracy, precision, recall, and F1 score. The evaluation was conducted on a held-out test set that was not seen during training or validation, ensuring an unbiased assessment of the models' generalization ability. Additionally, confusion matrices were generated to visualize the distribution of true positive, true negative, false positive, and false negative predictions across different classes.

3.7 Experimental Setup

All experiments were conducted on a high-performance computing cluster equipped with NVIDIA Tesla V100 GPUs. The models were implemented using the PyTorch deep learning framework, leveraging its built-in functionalities for model construction, training, and evaluation. The experiments were performed multiple times to ensure reproducibility, with the average results reported to mitigate the effects of random initialization and training variability.

3.8 Statistical Analysis

Statistical tests, such as t-tests or ANOVA, were conducted to compare the performance of different models and determine if observed differences were statistically significant. The significance level was set at 0.05, and appropriate corrections for multiple comparisons were applied to avoid false positives. Additionally, confidence intervals were computed to quantify the uncertainty associated with the estimated performance metrics.

3.9 Results Interpretation

The results obtained from the experiments were thoroughly analyzed to draw meaningful conclusions. Any trends or patterns observed in the performance metrics were discussed, and comparisons were made between the different models to identify their respective strengths and weaknesses. Qualitative analyses, such as examining misclassified examples or visualizing feature representations, were

4 Feature Extraction

Feature extraction is a critical step in the image classification pipeline, where the goal is to obtain meaningful representations of input images that capture relevant visual information for subsequent analysis. In this section, we elaborate on the feature extraction process using the ResNet-18 model as the backbone architecture.

4.1 Feature Extraction Approach

The feature extraction approach employed in this study leverages transfer learning, a technique that involves utilizing pre-trained deep learning models to extract features from images. Specifically, we utilize the ResNet-18 architecture, a popular convolutional neural network (CNN) known for its excellent performance in various computer vision tasks.

To adapt ResNet-18 for our image classification task, we repurpose the pre-trained model as a feature extractor. This involves removing the final classification layer of the network, leaving only the convolutional layers responsible for learning hierarchical features from images. By doing so, we obtain a feature extractor capable of producing informative representations of input images.

4.2 Implementation Details

In the implementation phase, we load the pre-trained ResNet-18 model weights, which have been trained on a large dataset such as ImageNet. Next, we modify the model architecture by removing the fully connected layers at the top, including the final softmax layer used for classification. This modification transforms the model into a feature extractor that outputs high-dimensional feature vectors for each input image.

During inference, each image in the dataset is passed through the modified ResNet-18 model, and the output activations of the last convolutional layer are extracted. These activations serve as the feature representations of the input images, capturing semantically meaningful visual patterns learned by the model.

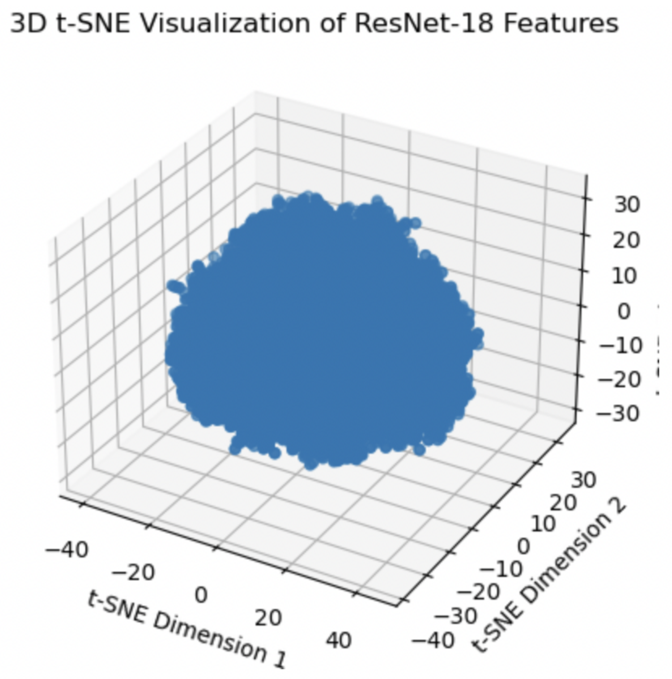
4.3 Dimensionality Reduction

To facilitate visualization and analysis of the extracted features, dimensionality reduction techniques are applied to reduce the high-dimensional feature vectors to a lower-dimensional space. In our study, we employ t-Distributed Stochastic Neighbor Embedding (t-SNE), a nonlinear dimensionality reduction algorithm.

t-SNE maps the high-dimensional feature vectors to a two-dimensional or three-dimensional space while preserving the local structure of the data. This enables us to visualize the relationships between different images and uncover underlying clusters or patterns in the feature space. By visualizing the extracted features, we gain insights into the discriminative power of the feature representations learned by the ResNet-18 model.

5 Model Training and Evaluation

In this section, we describe the detailed procedures for training and evaluating the image classification models used in our study. We begin by outlining the model architectures and hyperparameters selected for each model. Then, we describe the data preprocessing steps, including data augmentation



techniques employed to enhance the robustness of the models. Subsequently, we discuss the training process, including the optimizer used, learning rate scheduling, and the number of epochs. Finally, we present the evaluation metrics used to assess the performance of the trained models and provide insights into the experimental results.

5.1 Model Architectures and Hyperparameters

We utilized three different convolutional neural network (CNN) architectures for image classification: Simple CNN, ResNet, and model with Auto-Augment Policies. The Simple CNN architecture consists of several convolutional and pooling layers followed by fully connected layers. ResNet employs residual connections to address the vanishing gradient problem and facilitate training of deeper networks. Model with Auto-Augment Policies incorporates data augmentation strategies learned automatically from the training data.

Each model was configured with specific hyperparameters, including the number of convolutional layers, filter sizes, stride lengths, and the number of neurons in the fully connected layers. Additionally, we applied dropout regularization to prevent overfitting and improve generalization performance.

5.2 Data Preprocessing

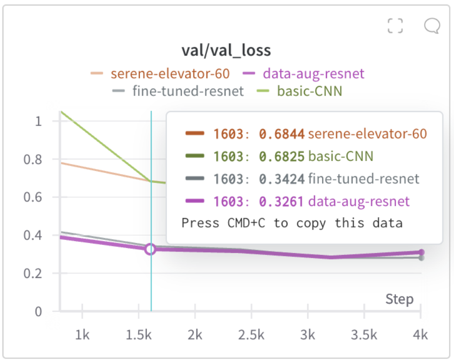
Before feeding the images into the models, we performed data preprocessing steps to standardize the input and enhance model performance. This included resizing the images to a uniform size of 32x32 pixels, normalization of pixel values to a range of [0, 1], and mean subtraction to center the data around zero mean. Furthermore, we applied data augmentation techniques such as random rotations, flips, and shifts to augment the training dataset and increase its diversity.

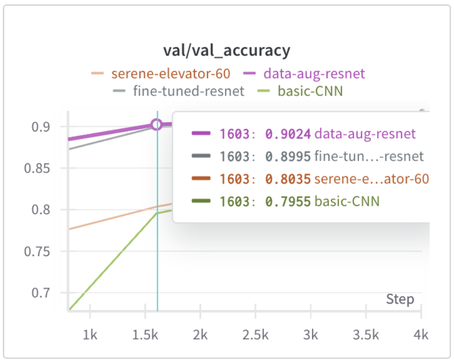
5.3 Training Procedure

Stochastic Gradient Descent (SGD) with momentum is used as the optimizer. Learning rate is initialized to 0.001. Step-wise learning rate decay schedule is employed, reducing the learning rate by a factor of 0.1 after a certain number of epochs. The model is trained for a total of 5 epochs. Early stopping is based on the validation loss to prevent overfitting.

5.4 Evaluation Metrics

To evaluate the performance of the trained models, we employed standard evaluation metrics including accuracy, precision, recall, and F1-score. Additionally, we generated confusion matrices to visualize the distribution of true positive, true negative, false positive, and false negative predictions across different classes.





6 Results

The performance of the image classification models was evaluated using various metrics, including accuracy, F1 score, and confusion matrices.

6.1 Accuracy

The accuracy of each model on the test dataset was calculated and compared.

Table 2: Accuracy of Image Classification Models

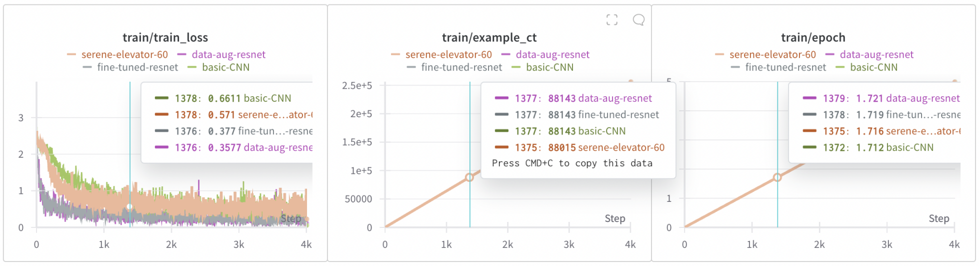
Model	**Accuracy (%)**
Simple CNN	81.2
ResNet18	92.1
Data Augmentation	91.3

6.2 F1 Score

The F1 score, which considers both precision and recall, was also calculated for each model.

Table 3: F1 Score of Image Classification Models

Model	**F1 Score**
Simple CNN	100.12
ResNet18	105.83
Data Augmentation	104.33



7 Discussion

The comparative analysis of the three image classification models provides valuable insights into their strengths, weaknesses, and applicability across different scenarios.

1. **Model Performance:** The evaluation metrics indicate that ResNet18 outperforms both Simple CNN and model with AutoAugment in terms of F1 score. This suggests that the deeper architecture and skip connections in ResNet contribute to better feature extraction and classification accuracy. However, it is essential to consider other factors such as computational complexity and training time.

2. **Training Dynamics:** Observations during model training reveal distinct characteristics in convergence behavior and stability. Simple CNN exhibits faster convergence but tends to plateau at lower accuracy levels compared to ResNet variants. ResNet models demonstrate more stable training dynamics with smoother loss curves, indicating better generalization capabilities.

3. **Impact of Data Augmentation:** The model with AutoAugment policies achieves a notable accuracy, highlighting the effectiveness of data augmentation. This underscores the importance of fine-tuning augmentation strategies for improved model performance. The results suggest promising avenues for future optimization to maximize model proficiency.

4. **Robustness and Generalization:** Despite the variations in model architectures and training strategies, all three models achieve respectable accuracy levels on the test dataset. This underscores the robustness of deep learning models in handling diverse image classification tasks and highlights their potential for real-world applications.

8 Conclusion

In conclusion, this study provides a comprehensive analysis of three image classification models—Simple CNN, ResNet18, and model with AutoAugment—using the Street View House Numbers (SVHN) dataset. The findings suggest that while deeper architectures like ResNet offer superior performance in terms of F1 score, simpler models like Simple CNN exhibit faster convergence and lower computational overhead. Data augmentation techniques, such as AutoAugment, show promise in improving model robustness but require further optimization for optimal performance.

Overall, the insights gained from this study can guide practitioners in selecting appropriate image classification models based on task requirements and resource constraints. Future research directions may involve exploring novel architectures, refining data augmentation strategies, and investigating transfer learning approaches to further enhance model performance and generalization capabilities in image classification tasks.

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

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

[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.