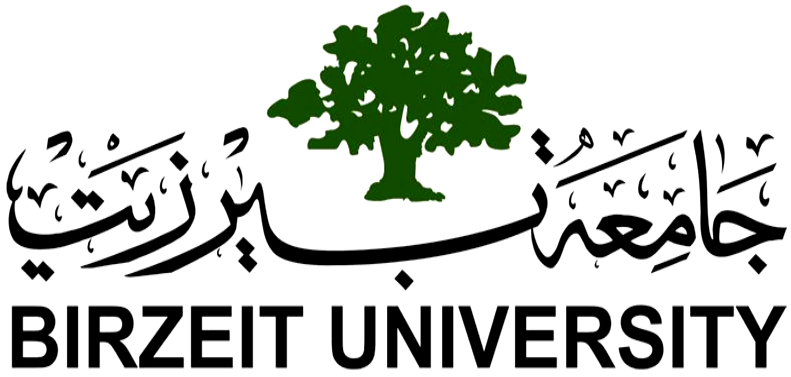

Faculty of Engineering & Technology
Electrical & Computer Engineering Department
COMPUTER VISION - ENCS5343

Course Project: Arabic Handwritten Text Identification Using CNN Networks

Prepared by: Hatem Hussein – 1200894

Nafe Abubaker - 1200047

Instructor: Dr. Aziz Qaroush

Date: January 25, 2025

Abstract

This project investigates the application of deep learning models for Arabic handwritten text recognition, a challenging task due to the complexity and variability of Arabic script. Four models were implemented and evaluated: a basic CNN, a CNN with augmentation, ResNet101 trained from scratch, and ResNet101 fine-tuned with pre-trained weights. The results highlighted the significance of leveraging data augmentation, deep architectures, and transfer learning for improving model performance. The fine-tuned ResNet101 achieved the highest accuracy of 91.85%, demonstrating the efficiency of transfer learning in handling domain-specific tasks with limited datasets. These findings emphasize the critical role of combining advanced architectures, augmentation strategies, and pre-trained models to achieve state-of-the-art results in Arabic handwriting recognition. Future directions include exploring transformer-based architectures, larger datasets, and real-time deployment optimizations.

Table of Contents

List of Figures

List of Tables

1. Introduction

The task of identifying authorship based on Arabic handwritten text presents a challenging problem in the domain of computer vision and deep learning. This study explores the application of Convolutional Neural Networks (CNNs) and transfer learning techniques to address the challenge of recognizing which of 82 distinct users authored an input image. Each user contributed 10 unique Arabic words, written 10 times each, forming a robust dataset designed to test the efficacy of various deep learning approaches.

CNNs are specialized neural networks that excel at extracting spatial hierarchies in image data. They operate by applying convolutional layers to detect local features such as edges and patterns, which are subsequently combined through pooling layers to capture higher-level abstractions. This capability makes CNNs ideal for tasks like handwriting recognition, where local patterns such as curves and strokes play a crucial role in distinguishing individual writing styles.

Transfer learning has emerged as a powerful tool for leveraging pre-trained models developed on large datasets, such as ImageNet. Instead of training a deep network from scratch, transfer learning enables the reuse of learned representations from similar tasks, fine-tuning them for a new dataset. This approach significantly reduces the computational cost and training time while improving model accuracy, particularly when dealing with limited or domain-specific data.

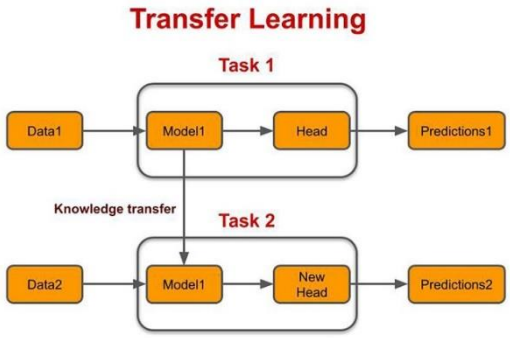

Figure 1.1: Transfer Learning System

By combining CNNs with transfer learning, this project aims to enhance the performance of handwriting recognition systems. Using pre-trained networks like ResNet101, the study focuses on adapting these architectures to classify Arabic handwriting styles effectively.

The dataset comprises 82 users, each contributing 100 images, resulting in 8,200 labeled samples. Each sample is a grayscale image resized to 128×128, with preprocessing steps such as normalization applied to ensure consistency. The challenge lies in accurately predicting the author of an input image based on the unique nuances of Arabic handwriting.

2. Experimental Setup and Results

2.1 Data Loading and Preprocessing

The preprocessing pipeline for the Arabic handwritten text identification task ensures the dataset is efficiently organized, normalized, and prepared for training and evaluation. Below is a detailed breakdown of the steps involved:

The dataset is structured into folders, each representing a unique user with names like userXXX (e.g., user001, user002). These folders correspond to distinct classes or labels. The preprocessing script traverses these directories, extracting the user ID from the folder name and using it as a label for all images within that folder. This organization establishes a clear mapping between the images and their respective classes.

To manage the dataset efficiently, the images within each user folder are grouped by the word they represent. This grouping leverages the naming convention of the image files, which follow the format userXXX_word_index.png. The word is extracted from the filename, enabling the dataset to be organized into word-based groups. This step ensures that the training and testing splits are performed on a word level rather than individual images.

Each image is processed in grayscale to reduce computational complexity while retaining the essential features required for handwritten text recognition. The images are resized to a consistent resolution of 128x128 pixels, ensuring uniformity across the dataset. Additionally, pixel values are normalized to the range [0, 1] by dividing by 255.0. This normalization improves the stability of the training process and accelerates model convergence.

The dataset is divided into training and testing subsets to ensure the model's ability to generalize to unseen data. For each user, approximately 80% of the words (about 8 out of 10) are allocated to the training set, while the remaining 20% (about 2 out of 10) are reserved for testing. This split ensures that the testing set contains words the model has not encountered during training, making the evaluation more robust.

After processing, the images and their corresponding labels are converted into Numpy arrays. These arrays are highly efficient for storage and computational operations. Each image is reshaped to include a channel dimension (e.g., 128x128x1) for compatibility with convolutional neural networks that expect image input in this format.

The class labels are adjusted to be 0-indexed, aligning with the requirements of popular deep learning frameworks like PyTorch and TensorFlow. This step ensures seamless integration with standard loss functions and evaluation metrics.

To save computational resources, the processed datasets are stored as .npy files. These files can be directly loaded in subsequent stages, eliminating the need for repeated preprocessing. Both the training and testing datasets, including their images and labels, are saved for future use.

2.2 Build and Train Custom CNN

2.2.1 Model Architecture

The architecture of the model designed for Task 1 (Model 8) incorporates a sequential convolutional neural network (CNN) structure tailored for Arabic handwritten text identification. The input layer accepts grayscale images of size 128x128, processed through a series of convolutional blocks designed to extract hierarchical features. Each convolutional block applies a 2D convolution with LeakyReLU activations, batch normalization for stability, and max-pooling layers for down-sampling. The first convolutional block uses 64 filters, followed by an increasing number of filters in subsequent blocks (128, 256, and 512), enhancing the model's capacity to learn intricate patterns. Regularization techniques, including `l2` kernel regularizers and dropout layers (0.3), help prevent overfitting by penalizing large weights and reducing co-dependencies among neurons.

After the convolutional blocks, the model transitions into a fully connected dense layer with 512 neurons to consolidate the extracted features. Batch normalization and dropout are applied here as well, ensuring robustness during training. The final layer is a dense layer with 82 neurons, corresponding to the number of users, and utilizes the softmax activation function to predict probabilities for each class. The model is compiled with the Adam optimizer for efficient gradient-based optimization and uses sparse categorical cross-entropy as the loss function, suitable for multi-class classification. Additionally, a learning rate scheduler dynamically adjusts the learning rate based on the validation loss, and class weights balance the training process by compensating for class imbalances. This design effectively captures both local and global features, offering a strong foundation for accurate user identification.

2.2.2 Model Performance

Throughout the experimentation process, multiple models were designed and evaluated to identify the most effective architecture for the Arabic handwritten text identification task. Each model incorporated unique architectural changes and hyperparameter adjustments to optimize performance. The comparison was conducted by analyzing training and validation loss, as well as accuracy metrics across epochs, as illustrated in the accompanying visual. Notably, the models varied in their convolutional layers, regularization techniques, and dropout configurations, highlighting the iterative effort to balance complexity and generalization.

Among all the models, Model 8 emerged as the best-performing architecture. It achieved a final test accuracy of approximately 55.82% and a test loss of 2.2639. This performance is clearly reflected in the plots, where Model 8 exhibits a steady improvement in both training and validation accuracy while maintaining low loss values. The enhancements in Model 8, such as the adjusted convolutional layers, dropout rates, and optimized kernel regularization, were instrumental in achieving this result. This model's architecture and performance stand out as the most suitable for this task, offering a strong foundation for future improvements and practical deployment.

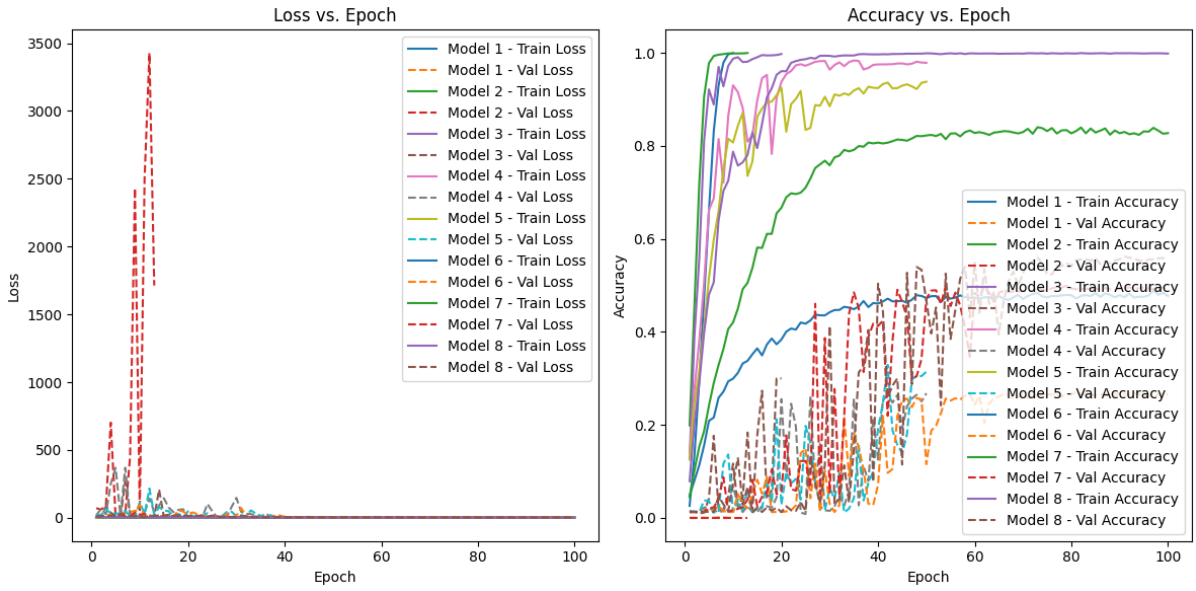

Figure 2: Trying Multiple Models

2.3 Retrain the Previous CNN Model With Data Augmentation

2.3.1 Data Augmentation

To enhance the diversity and robustness of the training dataset, a detailed augmentation strategy was applied using the `ImageDataGenerator` library. This approach introduces variations in the training images to simulate real-world distortions, enabling the model to generalize better. The augmentations include **rotation** (±10 degrees) to account for slight angular variations in handwriting, **width and height shifts** (up to 10%) to address positional inconsistencies, and **shear transformations** (range of 0.1) to mimic natural distortions in handwritten text. Additionally, a minimal **zoom** factor (0.001) was introduced to simulate subtle changes in text size, while the **nearest fill mode** ensured smooth filling of empty spaces created during transformations, maintaining the images' structural integrity.

```
# Define augmentation strategy
train_datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.001,
    fill_mode='nearest'
)
```

Figure 3: Augmentation Code

Each image in the training dataset was augmented twice, effectively increasing the dataset's size and providing more diverse samples for training. For instance, the original dataset size of

`train_images.shape[0]` samples was expanded by an additional `augmented_images.shape[0]` augmented samples, resulting in a final dataset size of `train_images_expanded.shape[0]`. These augmentations were carefully chosen to align with the task's requirements, particularly for Arabic handwritten text identification, ensuring they reflect realistic handwriting variations without introducing distortions that might degrade the dataset's quality. This enhanced dataset not only helps improve the model's performance on unseen data but also mitigates overfitting by offering a richer and more varied training set.

2.3.2 Model Performance After Augmentation

The graphs below illustrate the training and validation loss and accuracy trends across epochs for the augmented model architecture. The left graph depicts the loss curve, where the training loss steadily declines over epochs, indicating that the model effectively minimizes errors during training. The validation loss, although initially unstable, shows significant improvements after early fluctuations, stabilizing towards the later epochs. This behavior highlights the impact of augmentation, which introduces variability in the dataset, thereby increasing the complexity of the validation phase initially but ultimately leading to better generalization.

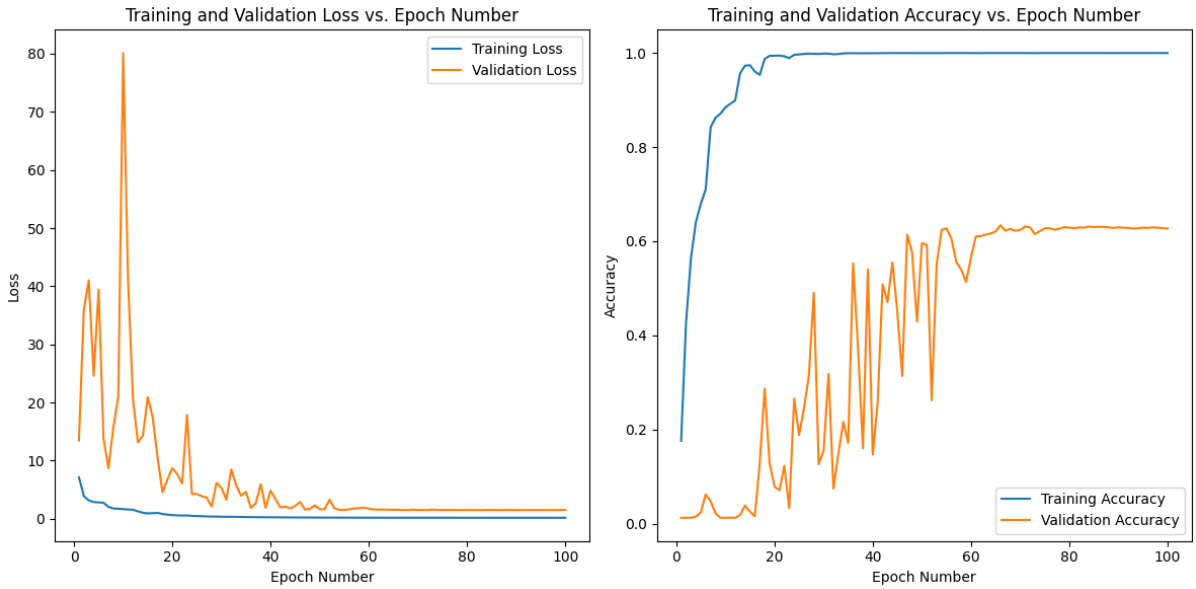

Figure 4: CNN Model with Augmentation Performance Curve

The right graph showcases the training and validation accuracy trends. The training accuracy rapidly converges to near 100%, reflecting the model's capacity to learn from the enriched dataset. The validation accuracy demonstrates noticeable improvements compared to the model's performance without augmentation. While it fluctuates initially, the accuracy stabilizes at a much higher value in later epochs, showcasing the effectiveness of augmentation in enhancing the model's ability to generalize well on unseen data.

After applying augmentation to the dataset, the model achieves a test accuracy of **62.70%** with a loss of **1.5192**, compared to **55.82%** accuracy without augmentation. This significant improvement underscores the role of data augmentation in introducing variations that mimic real-world scenarios, enabling the model to become more robust. Augmentation strategies

such as rotation, width, and height shifts effectively capture diverse handwriting styles and variations, particularly in Arabic handwritten text. This enhances the model's learning capacity and contributes to a higher validation accuracy, demonstrating the importance of augmentation in improving the model's overall performance.

2.4 Well-Known CNN Architecture (ResNet101)

2.4.1 Model Architecture

The model used in Task 3 is based on the ResNet101 architecture, a well-established convolutional neural network (CNN) designed for deep learning tasks. ResNet101 employs residual blocks, which mitigate the vanishing gradient problem often encountered in very deep networks by enabling the model to learn identity mappings. This architecture is particularly effective for extracting complex features from data, making it suitable for the challenging task of Arabic handwritten text identification.

For this task, the ResNet101 model was adapted to handle grayscale images by modifying the first convolutional layer to accept a single-channel input instead of the usual three-channel RGB input. The fully connected (FC) layer at the end of the network was replaced with a custom layer that outputs probabilities for 82 classes, corresponding to the 82 unique users in the dataset. To improve training efficiency and prevent overfitting, the remaining layers of the pre-trained ResNet101 were frozen during training, allowing the network to focus on learning task-specific patterns from the fully connected layers. The model was trained using the AdamW optimizer, a OneCycleLR scheduler for dynamic learning rate adjustments, and gradient scaling with AMP (automatic mixed precision) to optimize performance on GPUs. These enhancements made the model robust for user identification based on handwritten Arabic text. Below is a table illustrating the structure of the last layers of the **ResNet101Arabic** model used:

Table 2.4.1.1: ResNet101Arabic Last Layers

Layer	Type	Output Shape	Description
fc	Fully Connected (FC)	(batch_size, 512)	Linear layer that reduces the feature dimensions from resnet101.fc.in_features to 512.
activation	ReLU	(batch_size, 512)	Applies the ReLU activation function for non-linearity.
dropout	Dropout	(batch_size, 512)	Dropout with a probability of 0.4 to prevent overfitting.
output_layer	Fully Connected (FC)	(batch_size, 82)	Final layer that outputs probabilities for 82 classes (users) using softmax.

2.4.2 Model Performance

The training and validation performance of the Task 3 model is depicted in the below plots, showcasing the progression of both loss and accuracy metrics over 30 epochs. The **loss curves** illustrate a consistent decline for both training and validation sets, with a slight divergence toward the later epochs indicating a minor generalization gap. Simultaneously, the **accuracy curves** display a steady improvement, culminating in a final **test accuracy of 82.90%**, which highlights the model's strong performance on unseen data.

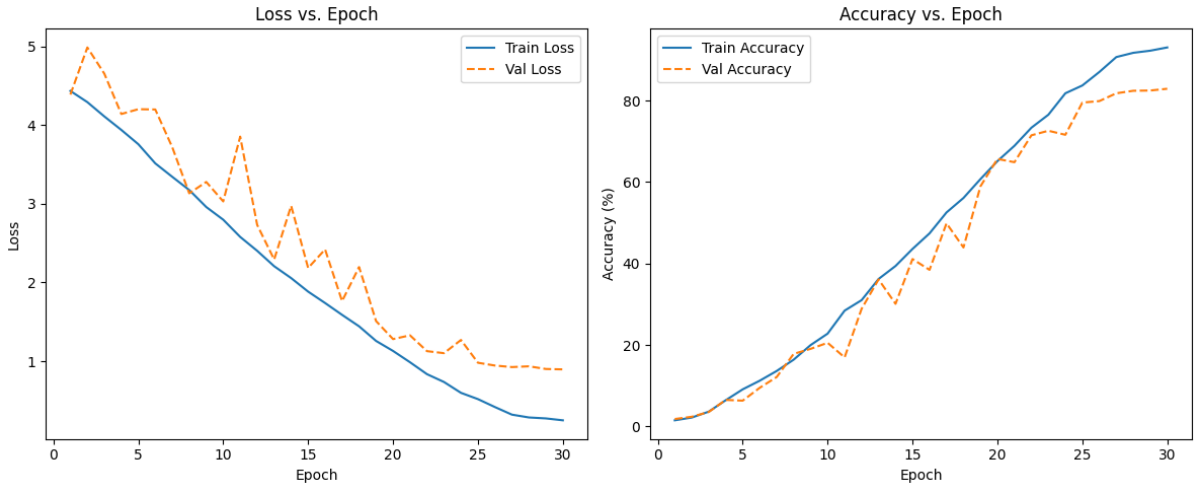

Figure 5: ResNet101Arabic Performance Curve

The model started with an initial **training accuracy of 1.53%** and progressively improved over successive epochs, reaching a final accuracy of **93.04%** on the training set. Similarly, the validation accuracy started at **1.85%** and concluded at **82.90%,** signifying effective generalization. The validation loss consistently decreased, further validating the model's ability to capture patterns in the data without overfitting. Importantly, the final **test loss of 0.8960** is relatively low, indicating the robustness of the model during evaluation.

When compared to Task 2, the Task 3 model demonstrates significant improvements in both training and validation accuracy. The higher validation accuracy of **82.90%** and the lower generalization gap reflect better fine-tuning strategies and architecture modifications in Task 3. These results emphasize that leveraging the ResNet101 architecture with task-specific adaptations and proper hyperparameter tuning substantially enhances the model's capability to identify Arabic handwritten text users.

Overall, Task 3's results confirm that the model is well-suited for the problem, achieving a notable balance between high accuracy and minimal overfitting.

2.5 Pre-trained CNN Model (ResNet101 with imagenet as weights)

2.5.1 Model Architecture

The architecture of the model is built on the robust **ResNet101**, a widely used convolutional neural network initially trained on the **ImageNet** dataset. The model leverages transfer learning by retaining the pre-trained weights from ImageNet, which encapsulate general-purpose feature representations learned from millions of diverse images. This approach significantly reduces the computational cost and training time while enhancing performance, especially when working with smaller datasets like Arabic handwritten text.

Key Modifications to the Base Model:

1. **Input Layer Adaptation**:
    - The original input layer of ResNet101, designed for RGB images with 3 channels, was modified to accept grayscale images (1 channel). The first convolutional layer (conv1) was reconfigured to process grayscale data by changing the input channel size to 1 while retaining the ability to extract low-level features.
2. **Freezing Pre-Trained Layers**:
    - All layers in the pre-trained ResNet101 model were **frozen**, preventing their weights from being updated during training. This ensures the preserved feature extraction capabilities of the ImageNet weights remain intact while focusing computational resources on fine-tuning the added layers.
3. **Custom Fully Connected (FC) Layer**:
    - The original fully connected (FC) layer of ResNet101 was replaced with a custom architecture tailored for the task. This layer stack includes:
        - A **fully connected layer** with 512 neurons to condense the extracted features.
        - A **ReLU activation** to introduce non-linearity and enhance feature learning.
        - A **dropout layer** with a rate of 0.4 to prevent overfitting by randomly deactivating neurons during training.
        - A final **softmax layer** with 82 output neurons, corresponding to the number of users in the dataset, to perform multi-class classification.

The model was not trained from scratch. Instead, it utilized the pre-trained weights of ResNet101 from ImageNet as a starting point. This allowed the model to begin with a strong foundation of generic visual features, such as edges, shapes, and textures. The task-specific adaptation involved fine-tuning only the added FC layers while keeping the rest of the network frozen. This approach reduces the risk of overfitting, especially given the relatively limited size of the dataset.

The model was trained for 30 epochs using a batch size of 64. The **OneCycleLR** scheduler dynamically adjusted the learning rate to achieve faster convergence. The training process utilized **automatic mixed precision (AMP)** to optimize GPU memory usage. The final evaluation showed that the model effectively transferred its pre-trained knowledge and adapted to the specific task of Arabic handwritten text identification, achieving a strong test accuracy while maintaining efficient resource usage.

This combination of transfer learning and fine-tuning demonstrates how pre-trained models like ResNet101 can be effectively repurposed for specialized tasks, achieving high performance with limited computational overhead.

2.5.2 Model Performance

This model achieved a **test accuracy of 91.85%**, which is a significant improvement compared to the result from Task 3, where the accuracy reached **82.90%**. This improvement showcases the benefits of leveraging transfer learning with fine-tuning on the ResNet101 architecture, which was pre-trained on ImageNet weights. The higher accuracy indicates that the model was able to generalize better to unseen data. This success stems from the strong feature extraction capabilities of the pre-trained layers, which were further refined for the specific task of Arabic handwritten text recognition through domain-specific fine-tuning.

Examining the training and validation curves below, the **loss curves** show a steady decrease across epochs, reflecting effective learning without overfitting. Both training and validation losses converge toward the end, indicating that the model has balanced its performance on the training and validation datasets. The **accuracy curves** demonstrate consistent improvement, with the validation accuracy surpassing 90% by the 25th epoch. Notably, the gap between training and validation accuracy remains narrow, suggesting the model's strong generalization and regularization.

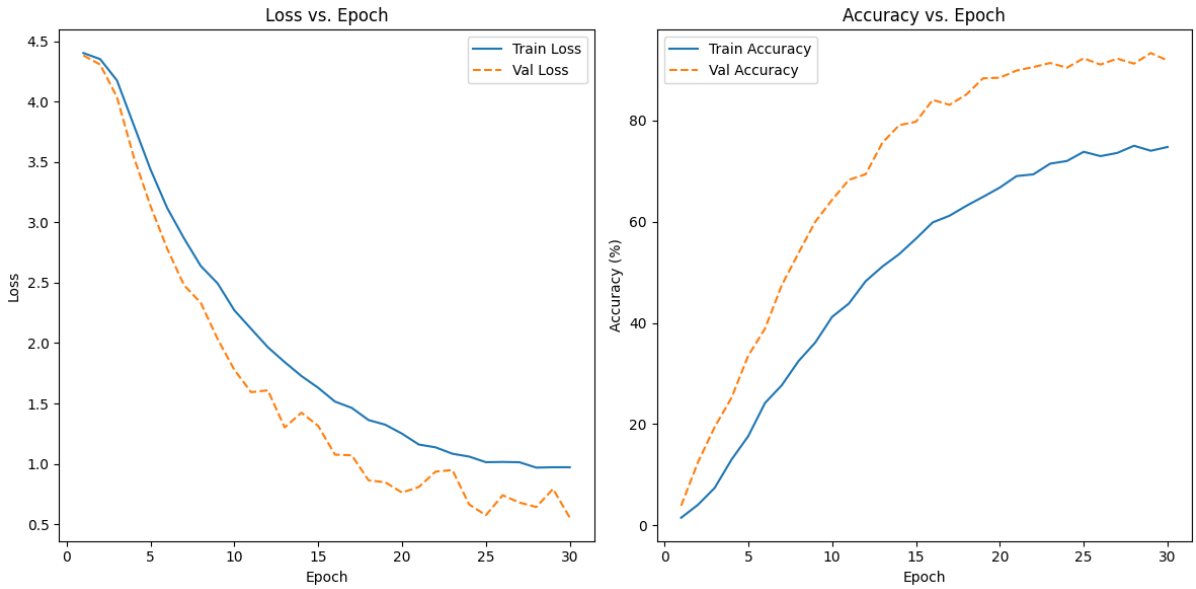

Figure 6: ResNet101 with Transfer Learning Performance Curve

Several factors contributed to this performance improvement. **Transfer learning and fine-tuning** played a pivotal role, as the pre-trained ImageNet weights provided a robust starting point by leveraging features learned on a diverse dataset. Fine-tuning allowed the model to adapt these features to the task-specific nuances of Arabic handwriting. Furthermore, the **OneCycleLR scheduler** helped accelerate convergence during early training while ensuring precise weight updates in later epochs. Although the augmentation strategy was less aggressive than in Task 3, it remained appropriate for the task, adding variability to the training data without compromising its quality.

In comparison to Task 3, this model exhibited **higher accuracy, faster convergence, and reduced overfitting**. While Task 3's model required more epochs to reach 80% validation accuracy, the current model achieved this milestone within 15 epochs, underscoring the effectiveness of transfer learning. Additionally, the validation and training curves in this task show minimal divergence, highlighting better generalization and less overfitting than in Task 3. In conclusion, this comparison demonstrates the efficacy of pre-trained architectures and fine-tuning techniques in achieving superior results for specialized tasks like Arabic handwritten text recognition.

2.6 All Models Comparison

The performance progression across the four models highlights the importance of leveraging augmentation, deeper architectures, and transfer learning for domain-specific tasks. **Model 2** illustrated the immediate benefits of augmentation, improving generalization on unseen data. However, **Model 3**, despite being a much deeper architecture, underperformed due to the absence of pre-trained weights, emphasizing the inefficiency of training deep networks from scratch. **Model 4**, which fine-tuned a pre-trained ResNet101, demonstrated the optimal balance between leveraging prior knowledge and task-specific training, achieving the highest accuracy of **91.85%**. This progression illustrates that deep architectures and transfer learning are essential for achieving state-of-the-art results, especially in tasks with limited datasets.

Table 2.6.1: All Models Comparison

Model	Architecture	Key Features	Training Method	Test Accuracy	Remarks
Model 1	Simple CNN	Basic architecture, Dropout for regularization	Trained from scratch	54.73%	A baseline model, simple and fast to train but limited in performance due to lack of complexity.
Model 2	CNN with Augmentation	Same as Model 1, but with data augmentation to improve generalization	Trained from scratch	62.70%	Augmentation improved generalization, demonstrating its effectiveness for small datasets.
Model 3	ResNet101 (Trained from Scratch)	Deep architecture with skip connections, initialized randomly	Trained from scratch	82.90%	Training from scratch yielded better performance but required significant time and computational power.
Model 4	ResNet101 (Pre-trained on ImageNet)	Same as Model 3, but initialized with ImageNet weights and fine-tuned	Fine-tuned with transfer learning	91.85%	Most efficient model, leveraging pre-trained weights reduced training time and improved performance.

Model 4 demonstrated remarkable efficiency and effectiveness by fine-tuning a pre-trained ResNet101 on ImageNet, achieving the highest accuracy of 91.85%. Unlike Model 3, which trained the same architecture from scratch and underperformed, Model 4 leveraged prior knowledge from a large, diverse dataset, significantly reducing training time and computational resources while improving performance. This highlights the strength of transfer learning in efficiently adapting deep architectures to domain-specific tasks, especially when data is limited.

3. Conclusion

In this project, we explored the application of deep learning techniques for Arabic handwritten text recognition using a series of models, ranging from custom CNN architectures to pre-trained ResNet101. The key findings demonstrated the importance of data augmentation, deeper architectures, and transfer learning in achieving superior performance. Model 2 showed how augmentation enhances generalization, while Model 3 revealed the challenges of training deep networks from scratch with limited data. The fine-tuned ResNet101 (Model 4) emerged as the most efficient and effective, leveraging pre-trained weights to achieve the highest accuracy of 91.85%. This progression highlights the value of combining prior knowledge and task-specific adjustments in achieving state-of-the-art results.

For future work, the focus can shift toward integrating advanced techniques such as attention mechanisms or transformer-based architectures to further enhance performance. Additionally, collecting a larger, more diverse dataset could enable training deeper models from scratch without relying solely on transfer learning. Exploring domain adaptation techniques and optimizing deployment for real-time applications are also potential directions for extending the scope and impact of this research.