Birzeit University
Faculty of Engineering & Technology
Department of Electrical & Computer Engineering

Vision Course Project

Prepared By:
Hadeel Abdellatif - 1190451
Maryam Altawil -1192099

Supervised By:
Dr. Aziz Qaroush

Birzeit
Jan, 2024

Abstract

In machine learning, handwritten character identification is still a difficult issue, especially for languages like Arabic that have complex characters. Effective character recognition faces substantial challenges due to the semi-cursive nature of Arabic script and the wide variance in letter shapes. By utilizing the powerful capabilities of convolutional neural networks (CNNs), this study seeks to enhance the field of Arabic Handwritten Character Recognition (AHCR).

Central to this endeavor is the development of a foundational CNN model specifically tailored to the complexities of Arabic script. In order to improve the model's capacity to generalize from small handwriting sample sizes, the research investigates and applies cutting-edge methods including data augmentation. To further improve recognition accuracy, it integrates pre-trained models to take advantage of features that have been learned from large datasets. Additionally, it incorporates pre-trained models to leverage learned features from extensive datasets, potentially boosting the recognition accuracy. A key component of this project is the thorough visualization of model performance, which provides critical insights into the model's strengths and areas for improvement. The end goal is to develop a reliable and efficient AHCR system .

Table of Contents

List of Figures

Introduction

In recent years, the field of computer vision has undergone a significant transformation, driven by the convergence of two fundamental factors: the rapid growth of machine learning capabilities and the increasing processing power of modern technology. These advancements have paved the way for addressing intricate challenges in areas such as image recognition, object detection, and character recognition.

Arabic Handwritten Character Recognition (AHCR) stands out as a particularly intricate and captivating field among the many challenges it encompasses. This complexity primarily arises from the inherent characteristics of the Arabic script, characterized by its right-to-left orientation and semi-cursive nature. Comprising 28 characters, the script exhibits varying shapes depending on their position within a word, presenting a unique set of challenges for automated recognition systems.

This project embarks on a detailed exploration of AHCR by harnessing the capabilities of Convolutional Neural Networks (CNNs), which have demonstrated exceptional effectiveness in tasks such as image recognition and classification. Our approach is multifaceted, involving the investigation of diverse network architectures, the implementation of efficient loss functions, and the utilization of advanced optimization techniques. We also give significant attention to data augmentation strategies, which play a pivotal role in enhancing the model's robustness, especially in light of the limited size and variability of handwriting datasets. Additionally, our project delves into the domain of transfer learning, leveraging pre-trained models to enhance feature extraction processes and potentially elevate the accuracy and efficiency of character recognition.

Through these combined efforts, our project aims to address the intricacies of Arabic Handwritten Character Recognition, contributing to the broader advancements in the field of computer vision. Our goal is to create a robust and effective system capable of accurately recognizing Arabic handwritten characters.

Background

Contents

In this chapter, we will explore some key concepts that form the foundation of our project. These include the essentials of Convolutional Neural Networks (CNNs), an overview of their various layers and functionalities. Understanding these concepts is crucial for grasping the methodologies and techniques employed in our research. Each section of this chapter is dedicated to unpacking these critical elements, setting the stage for the detailed exploration of our project.

2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a specialized machine learning model primarily used for image recognition and processing. It is designed to automatically and adaptively learn spatial hierarchies of features from input images [1]. CNNs utilize a mathematical operation called convolution and are composed of various layers that process and transform an input to produce a desired output. The layers include convolutional layers, pooling layers, and fully connected layers. Each layer transforms its input to increase the network's "understanding" of the image, culminating in the ability to make predictions or classifications.

2.2 Convolutional Layer

The convolutional layer is the core building block of a CNN. It applies a number of filters to the input to create feature maps. These filters help the network learn specific characteristics of the images, such as edges, textures, or more complex patterns. The convolutional operation effectively captures the spatial and temporal dependencies in an image through the application of relevant filters [2].

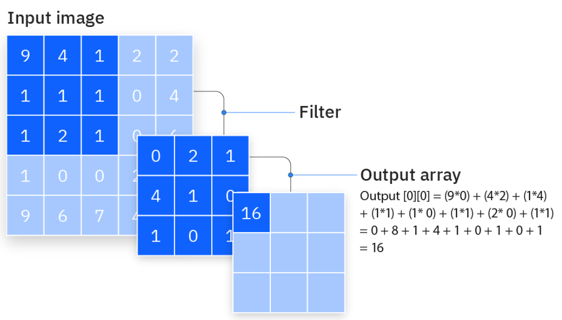

Input image
Filter
Output array
Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1* 0) + (1*1) + (2* 0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16

Figure 2.1: Convolution Layer

2.3 Pooling Layers

Pooling layers follow the convolutional layers and are used to reduce the spatial size of the representation, decreasing the number of parameters and computation in the network. They also help to make the detection of features invariant to scale and orientation changes. Max pooling and average pooling are common types of pooling layers, each providing different benefits [3].

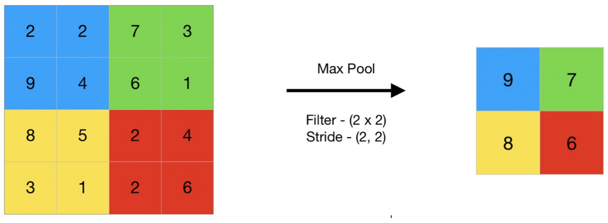


Figure 2.2: Max Pooling

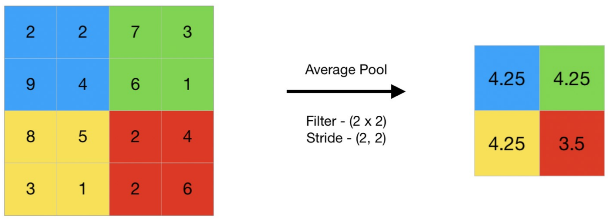


Figure 2.3: Average Pooling

2.4 Fully Connected Layer

After several convolutional and pooling layers, a CNN typically ends with one or more fully connected layers. In these layers, neurons have full connections to all activations in the previous layer, as seen in regular neural networks. Their role is to flatten the high-level features learned by convolutional layers and combine them to form a model output, such as class scores [4].

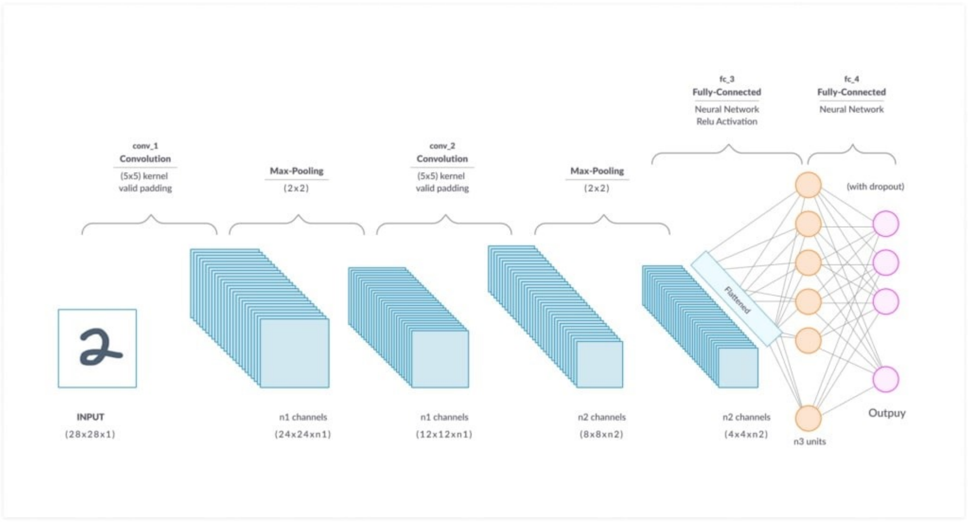


Figure 2.4: Fully Connected Layer

2.5 Transfer Learning

Transfer learning is a powerful approach in deep learning where a model developed for a task is reused as the starting point for a model on a second task. It is popular in deep learning due to the time and resources required to train a model and the large datasets needed. Using a pre-trained model can significantly improve performance when fine-tuned for a specific task [5].

2.6 Data Augmentation

Data augmentation is a strategy used to increase the diversity of your training set by applying random transformations, such as rotation and shifting. This technique helps improve the generalizability of the model by presenting it with various forms of the input data. In the context of Arabic Handwritten Character Recognition, data augmentation can be crucial due to the variety of handwriting styles and the limited availability of annotated data [6].

Proposed Work

Contents

3.1 Task 1

This task involves creating a custom Convolutional Neural Network (CNN) for Arabic Handwritten Character Recognition (AHCR). This involves careful consideration of architecture, layer types, activation functions, and various layer parameters.These parameters encompass elements like the number of layers, filter specifications, stride, and padding for convolutional layers, pool size and type for pooling layers, and the number of neurons for fully connected layers.
Relevant to our aim, previous studies have demonstrated the ability of Convolutional Neural Networks (CNNs) to recognize Arabic letters penned by hand. For example, Younis created a CNN with the AHCD and AIA9K datasets and achieved impressive accuracy of 94.7% and 94.8%, respectively. El-Sawy et al. made a contribution to this subject as well by introducing a CNN-based algorithm that produced 94.4% classification accuracy. Additionally, utilizing the Hijja dataset, researchers developed CNN-14, a novel model particularly made for handwritten Arabic character recognition, and it obtained 95%accuracy.

3.1.1 Model 1

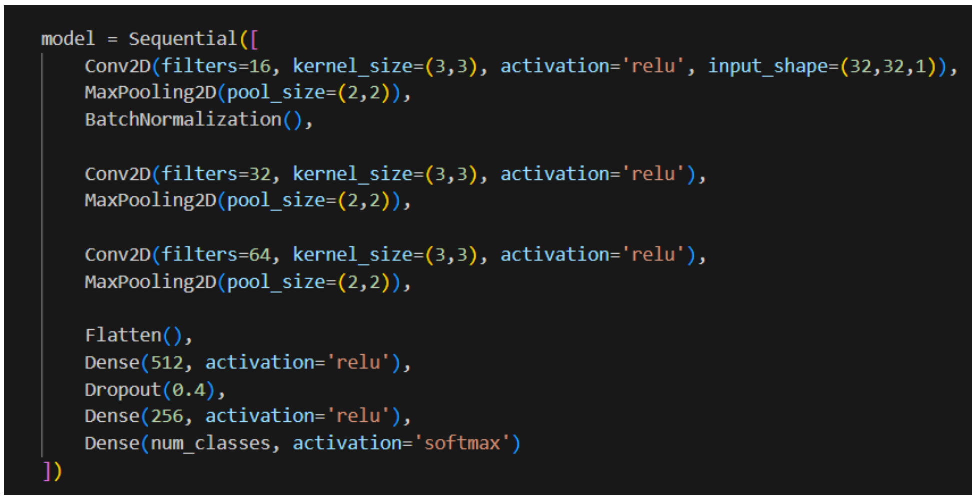
Our first model's architecture is methodically constructed with a series of layers, each playing a pivotal role in processing the input data:

1. Conv2D layers, progressively increasing filters from 16 to 64, employing ReLU activation.
2. MaxPooling2D layers following each Conv2D layer, reducing spatial dimensions.
3. A BatchNormalization layer enhancing model stability.
4. A Flatten layer converting 2D feature maps to a 1D vector.
5. Dense layers with 512 and 256 neurons, integrating Dropout to mitigate overfitting.
6. A final Dense Output Layer for classification, tailored to the number of classes.

Training Configuration and Optimization

- The model undergoes 30 epochs of training, with a batch size of 32.
- An Adam optimizer, set with a learning rate of 0.001, orchestrates the optimization process.
- The loss is computed using 'sparse_categorical_crossentropy'.

This structure, enriched by three convolutional layers paired with max-pooling, is designed to extract and refine features from the input images. The fully connected layers, with a substantial number of neurons, further distill these features into a form conducive to accurate classification. Our training approach, as visually represented in Figure 3.5,demonstrates the model's proficiency in learning and generalizing from the training data, albeit with indications of overfitting as seen in the slight divergence of training and validation accuracies.



```
model = Sequential([
    Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(32,32,1)),
    MaxPooling2D(pool_size=(2,2)),
    BatchNormalization(),

    Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.4),
    Dense(256, activation='relu'),
    Dense(num_classes, activation='softmax')
])
```

Figure 3.5: Model 1 Layers

As Figure 3.6 shows, the first graph shows that both training and validation losses drop sharply at first, but then validation losses stop improving after 5 epochs, indicating overfitting. The second graph shows that both training and validation accuracy rise quickly at first but then level off after 5 epochs, with a small gap, suggesting mild overfitting. The model achieves high accuracy across epochs, implying good generalization to unseen data. However, a slight variance implies some overfitting. The model performs better on new data than on previously seen data. The main findings are: a possible overfitting plateau, a test accuracy above 90%, and consistent accuracy after the initial epochs.s

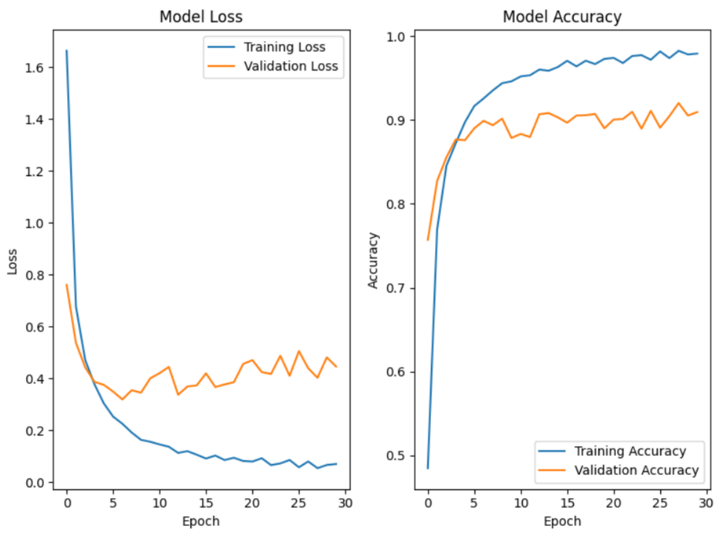


Figure 3.6: Model 1 Plots

The model returns images with their predictions and true labels.

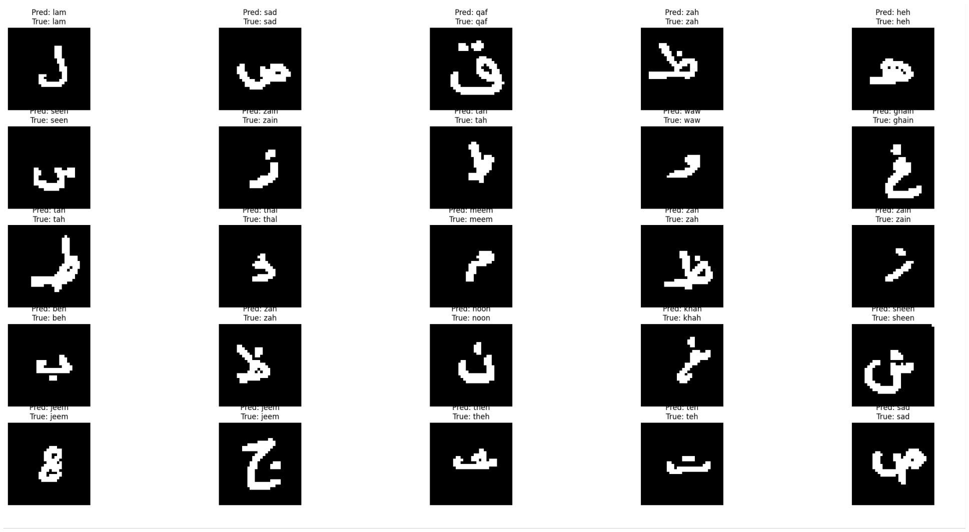


Figure 3.7: Display images with their predictions and true labels

The model returns images with incorrect predictions.

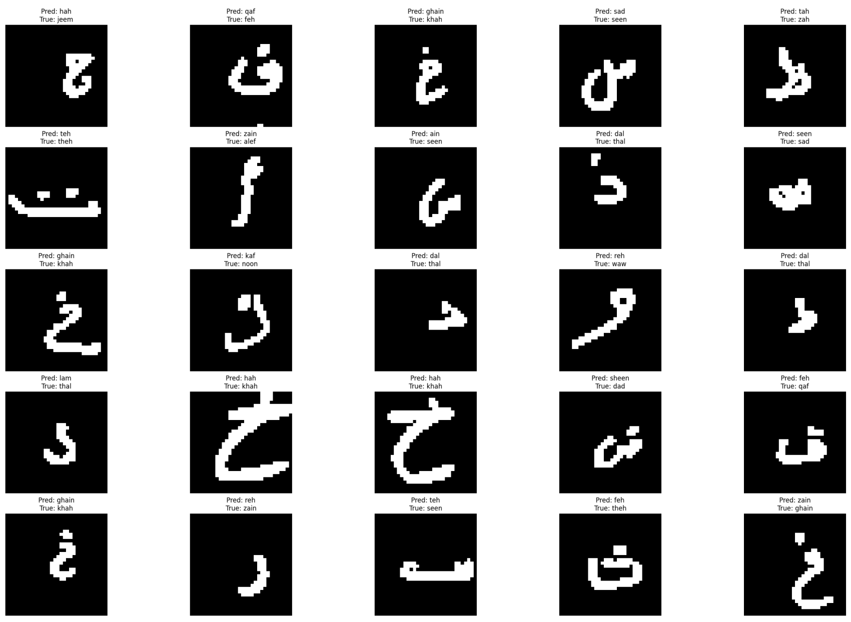


Figure 3.8: Display images with incorrect predictions

After training, the model was evaluated on the testing dataset, yielding the following results shown in Table 3.1:

Table 3.1: Metrics

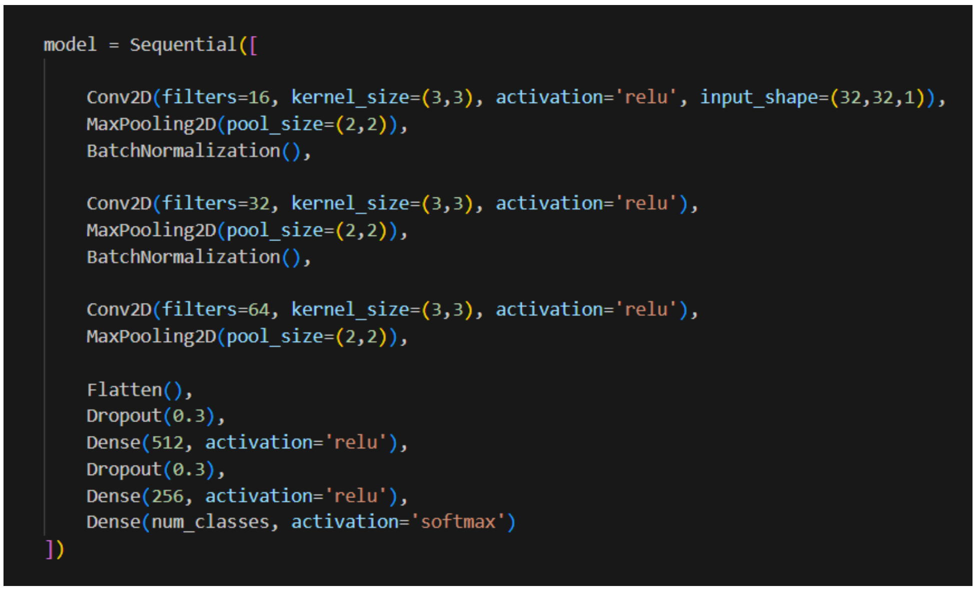
Metric	**Value**
Accuracy	0.9083
Precision	0.9118
Recall	0.9083

3.1.2 Model 2

To enhance our model's performance and elevate its accuracy from 90% to 95%, we implemented several strategic modifications. These adjustments aimed to minimize the discrepancy between training and validation metrics, addressing the issue of overfitting. Key among these changes was the introduction of dropout layers following specific components of the model, notably after the max pooling layers.This addition ensures the model does not overly specialize in the training data, thereby improving its adaptability to new data.

We optimized the dropout rate, reducing it from 0.4 to 0.3. This lower rate results in fewer features being disregarded, potentially enabling the model to more effectively learn meaningful patterns from the training data without succumbing to overfitting. This, in turn, contributes to the increased accuracy.

In our initial model, leaky ReLU layers were implemented solely after the first dense layer. However, in our revised model, we redistributed these leaky layers to different sections, particularly after the fully connected layers. This restructuring allows for more effective prevention of overfitting compared to the original model.



```
model = Sequential([

    Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(32,32,1)),
    MaxPooling2D(pool_size=(2,2)),
    BatchNormalization(),

    Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    BatchNormalization(),

    Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Flatten(),
    Dropout(0.3),
    Dense(512, activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dense(num_classes, activation='softmax')
])
```

Figure 3.9: Model 2 Layers

According to Figure 3.10, In the first graph about model loss, we note that the validation loss, in particular, shows negligible progress after about 5 epochs, with both training and validation losses initially decreasing significantly, which suggests less overfitting. The accuracy of the model appears in the second graph, which shows a sharp increase at first, followed by a plateau after around five epochs. The plateauing of validation accuracy suggests that the model has reached a point where it is not generalizing much better with additional training. Although a slight disparity between training and validation accuracy suggests some degree of overfitting, it is not significant. So, the model is learning effectively from the training data, as shown by the decreasing training loss and increasing training accuracy.

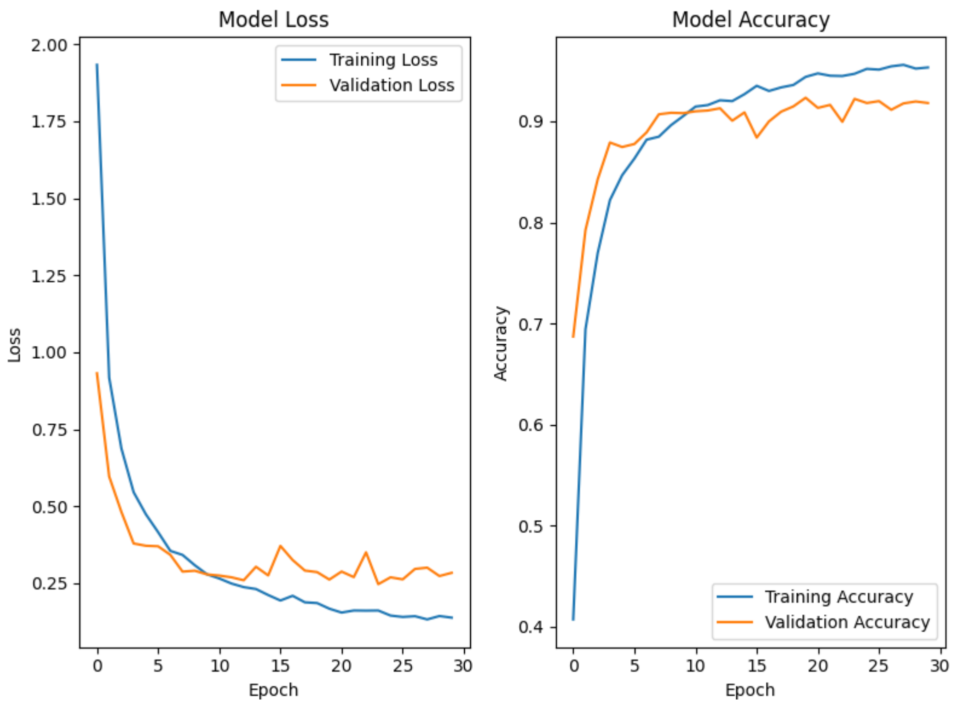


Figure 3.10: Model 2 Plots

After training, the model was evaluated on the testing dataset, yielding the following results shown in Table 3.2:

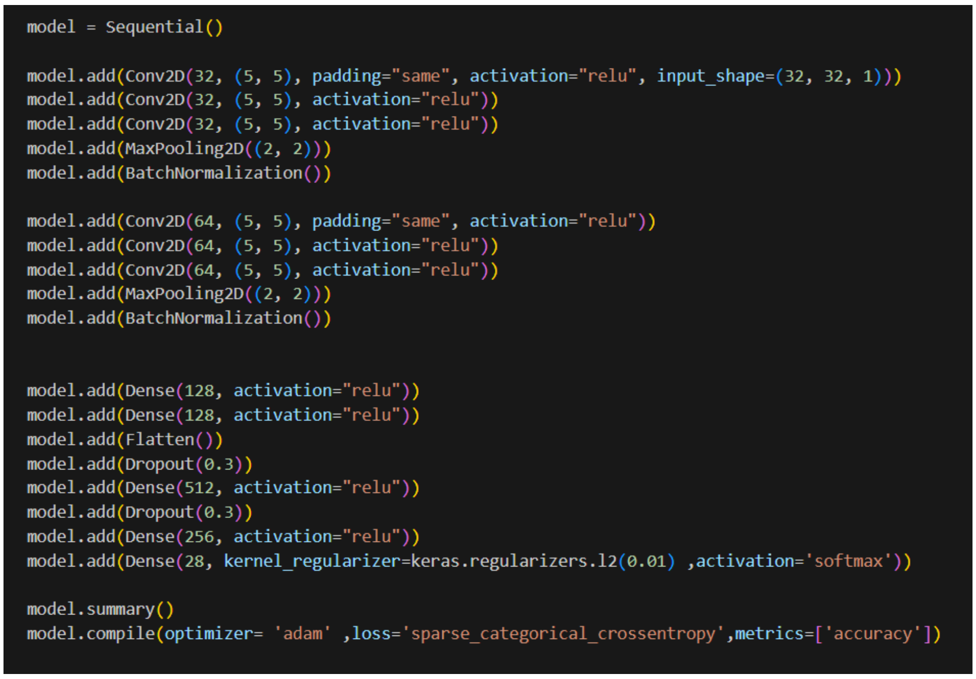
Table 3.2: Model Performance Metrics

Metric	**Value**
Accuracy	0.9280
Precision	0.9301
Recall	0.9280

These results demonstrate the model's capability to excel in tasks that demand accurate classification of handwritten characters, thanks to its high accuracy, low false-positive rate, and strong recall.

3.1.3 Model 3

We significantly overhauled the model architecture to achieve a substantial accuracy improvement, with the goal of reaching more than 95% accuracy. In this update, we introduced a more complex network by adding extra convolutional layers with larger filter sizes and increased depth. These changes were aimed at enabling the model to capture more intricate patterns and features within the input data. Furthermore, we expanded the fully connected layers to enhance the model's capacity for intricate feature mapping. We introduced batch normalization, which allows us to use higher learning rates, speed up training, and reduce the internal covariate shift, which can significantly hamper the training of deep networks. and dropout layers strategically throughout the network, which helps prevent overfitting.



```
model = Sequential()

model.add(Conv2D(32, (5, 5), padding="same", activation="relu", input_shape=(32, 32, 1)))
model.add(Conv2D(32, (5, 5), activation="relu"))
model.add(Conv2D(32, (5, 5), activation="relu"))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(64, (5, 5), padding="same", activation="relu"))
model.add(Conv2D(64, (5, 5), activation="relu"))
model.add(Conv2D(64, (5, 5), activation="relu"))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())


model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(512, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(256, activation="relu"))
model.add(Dense(28, kernel_regularizer=keras.regularizers.l2(0.01) ,activation='softmax'))

model.summary()
model.compile(optimizer= 'adam' ,loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

Figure 3.11: Model 3 Layers

According to figur 3.12, In the first graph, a sharp decline in training and validation loss indicates efficient learning. The second graph shows high training accuracy with a slight divergence in validation accuracy, suggesting minimal overfitting. Overall, the model demonstrates an excellent fit to the data with high predictive accuracy.

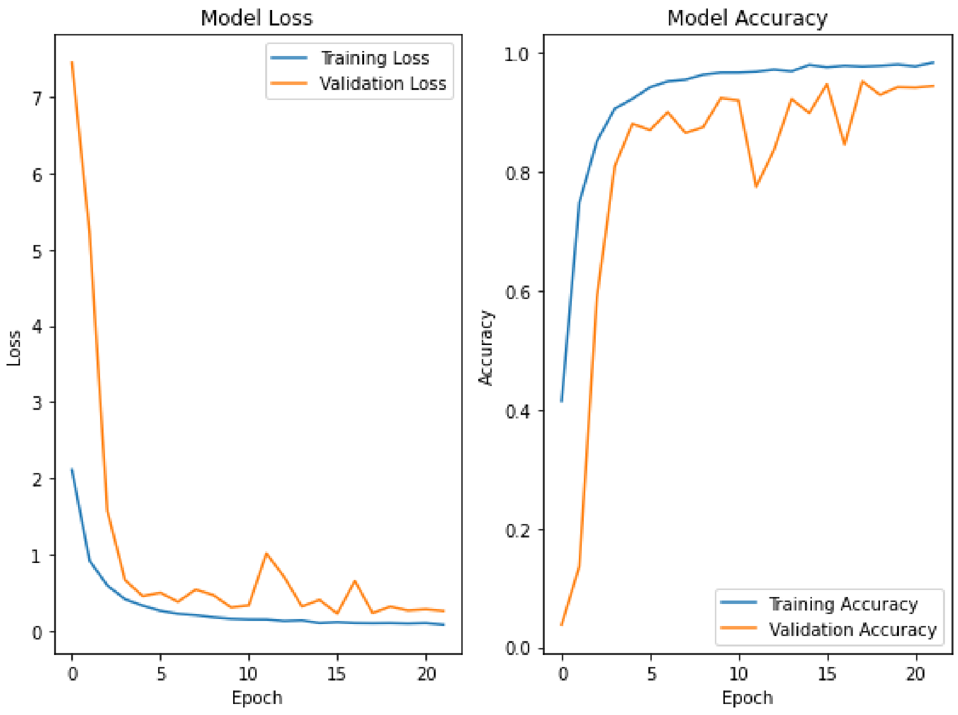


Figure 3.12: Model 3 Plots

After training, the model was evaluated on the testing dataset, yielding the following results shown in Table 3.3:

Table 3.3: Model Performance Metrics

Metric	**Value**
Accuracy	95.92%
Precision	96.08%
Recall	95.92%

Overall, the model's high accuracy, precision, and recall illustrate its robustness for tasks requiring precise recognition of handwritten characters.

3.1.4 Task 1 Results Discussion

After a comprehensive analysis of the performance metrics and architectural details of the three models and comparing the convergence between training and validation, we decided that Model 3 was the optimal choice for our application. Model 3's architecture, which includes more convolutional layers with a larger kernel size and additional batch normalization steps, allows more complex features to be learned from the input images. Strategic use of dropout layers prevents overfitting, as demonstrated by the close match between training and validation accuracy. The success of Model 3 can be attributed to key architectural choices, including augmenting convolutional layers, batch normalization, dropout, and dense layer composition. These design elements enable the model to effectively combine features and perform complex classifications. When comparing Model 3 to Models 1 and 2, we note that all models show rapid learning at initial epochs. However, Model 3 maintains efficient learning without overfitting, as shown by the relatively smoother accuracy and validation loss curves.

Table 3.4: Performance Metrics for Models 1, 2, and 3

Model	**Accuracy**	**Precision**	**Recall**
Model 1	0.9083	0.9118	0.9083
Model 2	0.9280	0.9301	0.9280
Model 3	0.9592	0.9608	0.9592

This table provides a concise summary of the accuracy, precision, and recall scores for Models 1, 2, and 3, highlighting Model 3's superior performance in each category.

After thorough evaluation, we chose Model 3 as our final model due to its superior performance potential. However, we recognized a noticeable difference between the training and validation values, indicating room for improvement. To address this, we implemented some critical modifications. Firstly, we increased the learning rate to facilitate convergence, allowing the model to adapt more effectively. Secondly, we extended the number of training epochs, enabling the model to undergo more iterations and fine-tune its parameters.

The results of these adjustments were highly encouraging, as depicted in the following graph:

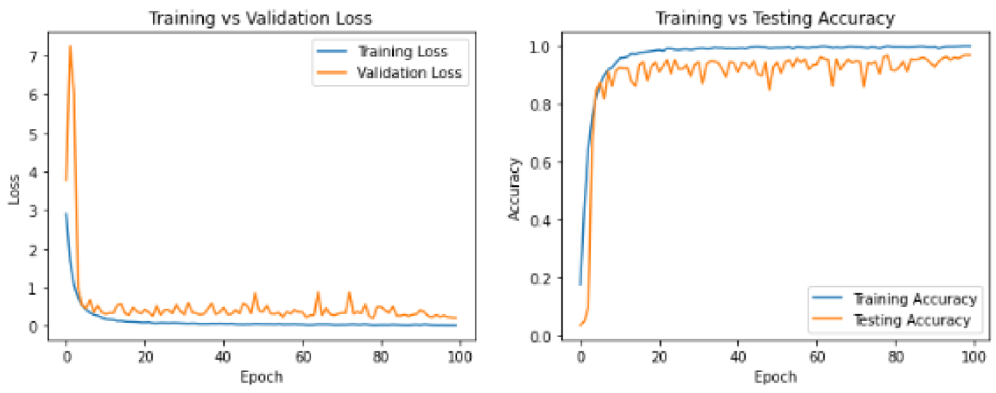


Figure 3.13: Model 3 Plots

After training, the model was evaluated on the testing dataset, yielding the following results shown in Table 3.4:

Table 3.5: Performance Metrics

Metric	**Value**
Accuracy	0.9682
Precision	0.9686
Recall	0.9682

The model demonstrated a significant reduction in the disparity between training and validation values. Moreover, the epoch size was increased to 64, further enhancing the model's ability to generalize effectively.

These improvements culminated in a highly proficient model with minimal discrepancies between training and validation, affirming its robustness and precision in recognizing Arabic Handwritten Characters.

3.2 Task 2

In Task 2 of our project on Arabic Handwritten Character Recognition using a Convolutional Neural Network (CNN), we focus on enhancing the model's performance through advanced data augmentation techniques.

Data Augmentation Techniques

These techniques, including zooming, width and height shifting, shear transformations, and nearest fill mode, aim to diversify the training dataset, thereby increasing the model's generalization ability. We have selected specific augmentation parameters as follows:

- Zoom (0.2)
- Width Shift (0.10)
- Height Shift (0.10)
- Shear (0.1)

These parameters introduce realistic variations that mirror natural handwriting discrepancies like rotations, shifts, and scaling. This approach is supported by research showing the value of data augmentation, which should increase the model's precision and its ability to withstand overfitting, especially when it comes to identifying a variety of complex handwriting styles.

In Figure 3.14, it is evident from the graphs that the Convolutional Neural Network (CNN) used for Arabic Handwritten Character Recognition demonstrates promising results. The model's loss graph shows a notable decrease in both training and validation loss as the number of epochs increases, indicating effective learning. Furthermore, both training and validation accuracy exhibit swift improvement and reach a plateau, with validation accuracy closely aligned with training accuracy. These findings suggest a well-fitting model without significant signs of overfitting.

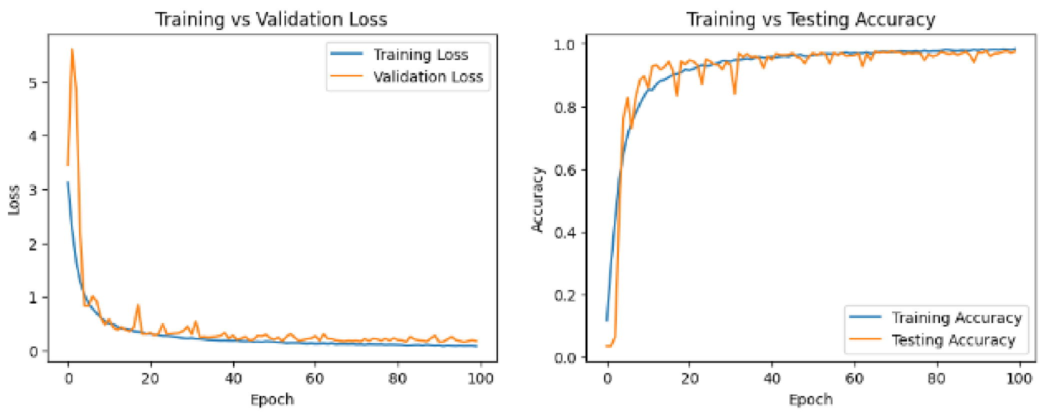


Figure 3.14: Data Augmentation Model Output

The model display 25 images along with their predicted and true labels, as shown in Figure 3.14.

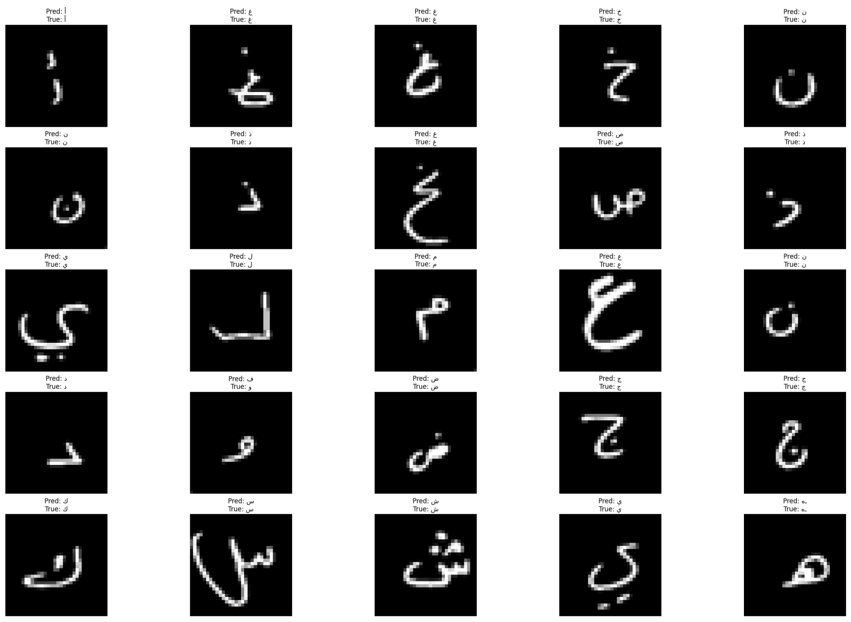


Figure 3.15: Display 25 images along with their predicted and true labels

The model display misclassified images along with their predicted and true labels, as shown in Figure 3.15

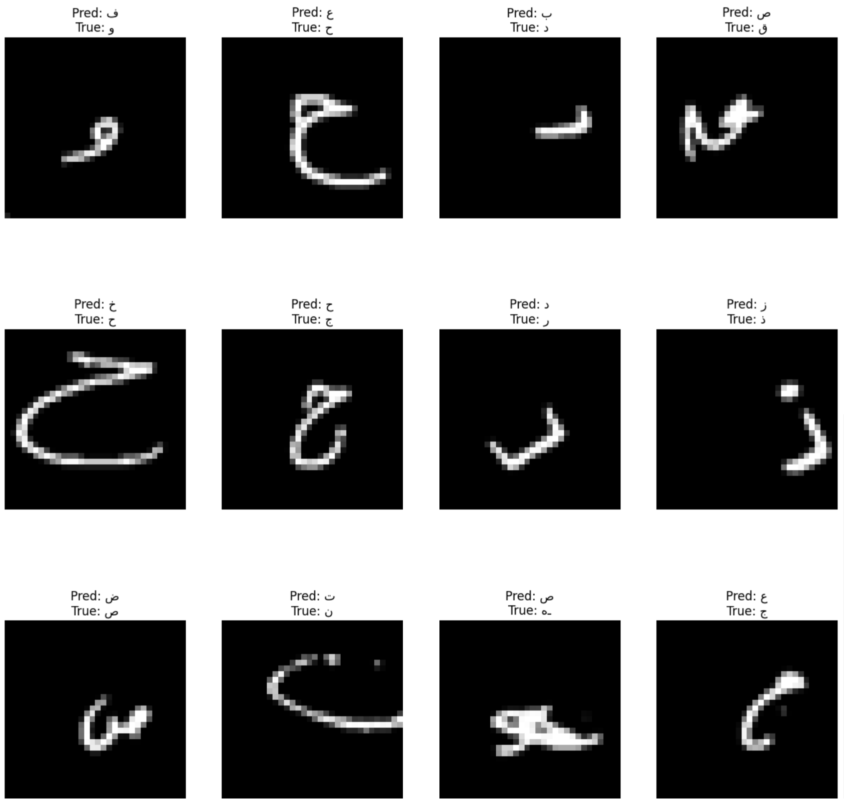


Figure 3.16: Display misclassified images along with their predicted and true labels

The performance measures of the model after applying data augmentation approaches show great promise :

Table 3.6: Performance Metrics

Metric	**Value**
Accuracy	0.9744
Precision	0.9749
Recall	0.9744

All of these numbers show how well data augmentation works to improve the prediction power of the model.

3.2.1 Task 2 Results Discussion

We observed a significant improvement in our model's performance when employing data augmentation techniques. The application of data augmentation has resulted in notable enhancements in model performance metrics. These improvements are summarized in the following table:

Table 3.7: Comparison of Model Performance

Metric	**Without Data Augmentation**	**With Data Augmentation**
Precision	96.82%	97.44%
Accuracy	96.86%	97.49%
Recall	96.82%	97.44%

The table above demonstrates the range of values obtained for precision, accuracy, and recall after applying data augmentation techniques and before . These improvements indicate a more reliable model for character recognition.

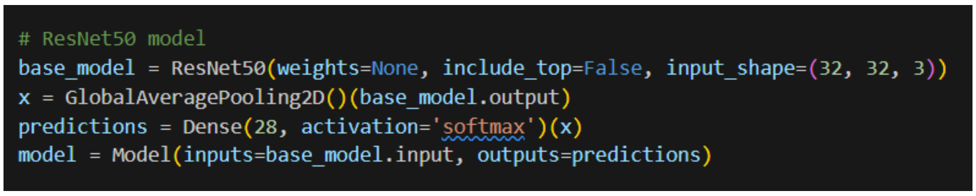
Additionally, the training process with data augmentation exhibits smoother curves in accuracy and loss over epochs, as visually evident in the graphs. Data augmentation has effectively reduced the disparity between training and validation metrics, signifying a model less prone to overfitting and better at generalizing to new data.

These metrics underscore the effectiveness of data augmentation in enhancing the model's ability to generalize and perform reliably across a wide range of handwriting styles.

3.3 Task 3

In the third task of our Arabic Handwritten Character Recognition (AHCR) project, we focused on utilizing ResNet-50, a well-known CNN architecture, to strike a balance between accuracy and complexity tailored to the Arabic script dataset. This involved customizing ResNet-50 and enhancing its generalization from the augmented data implemented in Task 2, with the aim of achieving a robust solution in Arabic handwriting recognition. We critically assessed the model's performance by comparing training and validation losses, along with accuracy metrics across epochs, to ensure significant progress from previous tasks.

According to Figure 3.17, we use ResNet-50 without pre-trained weights, remove the top layers, and change the input shape for our dataset. We add a global average pooling layer to reduce the convolutional output to a single vector per image, capturing key features. We add a fully connected layer with 28 units and softmax activation for the 28 character classes in our dataset. We make the model by defining the input and output layers, and prepare it for compilation and training with ResNet-50 as the backbone.



```python
# ResNet50 model
base_model = ResNet50(weights=None, include_top=False, input_shape=(32, 32, 3))
x = GlobalAveragePooling2D()(base_model.output)
predictions = Dense(28, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

Figure 3.17: Task 3 Model 1

We selected ResNet-50 for our AHCR project because of its deep architecture, which is excellent at learning from complex patterns in data and is perfect for identifying the subtle curves and shapes of Arabic handwritten letters. To improve its resistance to handwriting changes, this model is trained using augmented data. It was specifically designed for our 32x32-pixel images by stacking channels to fulfill its 3-channel input requirement. In order to ensure that the model works well on both training and unseen test data, it must be highly accurate while limiting its complexity. This requires careful consideration of the trade-off between computing efficiency and performance.

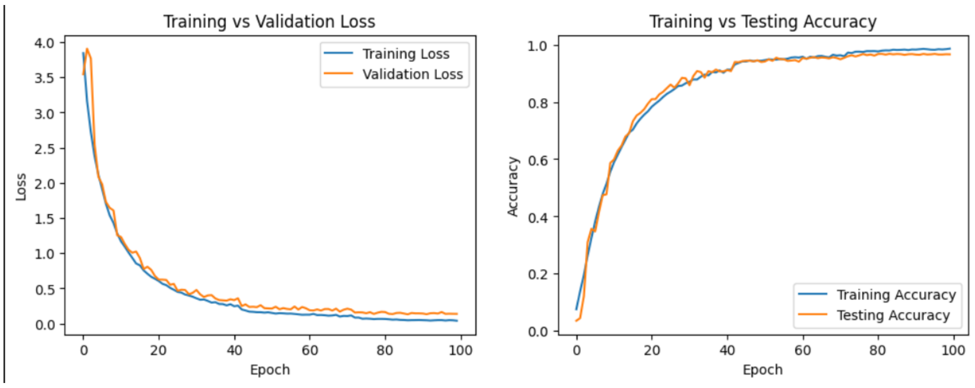


Figure 3.18: Task 3 Model 1 Graphs

According to Figure 3.18, the model's improved ability to recognize characters was shown by the training and validation loss graphs, which first fell quickly before leveling off as the number of epochs increased. Similarly, the accuracy graphs for training and testing increased slowly to almost match at high accuracy, indicating that the model is not overfitting and is capable of good generalization. Finally, the model gives us great results, with a loss of 0.0410 and an accuracy of 0.9862.

3.3.1 Task 3 Results Discussion

We find that Task 3 shows excellent results from the ResNet-50 model, with a high accuracy of 98.62%, an increase above Task 1's accuracy of 96% and Task 2's accuracy of 97.44%. This trend shows the model's improved capacity for generalization and its ability to adapt to character recognition. The ResNet-50 model's effective learning and predictive accuracy are shown by the notable decrease in loss and the convergence of training and testing accuracy, exceeding the already impressive results from the previous tasks.

3.4 Task 4

Using transfer learning, we utilize a pre-trained Convolutional Neural Network (CNN) in this task for Arabic Handwritten Character Recognition (AHCR). This extends our previous work on retrained and customized networks. Our goal is to fine-tune the pre-trained model to improve AHCR performance, comparing it to earlier tasks to assess the benefits and limitations of transfer learning in computer vision for handwriting recognition.

We tested two different pre-trained models in this task.

3.4.1 First Pre-trained Model

First, we used a model that had already been trained to Recognize Handwritten Digits in order to categorize digits [7] into the appropriate classifications

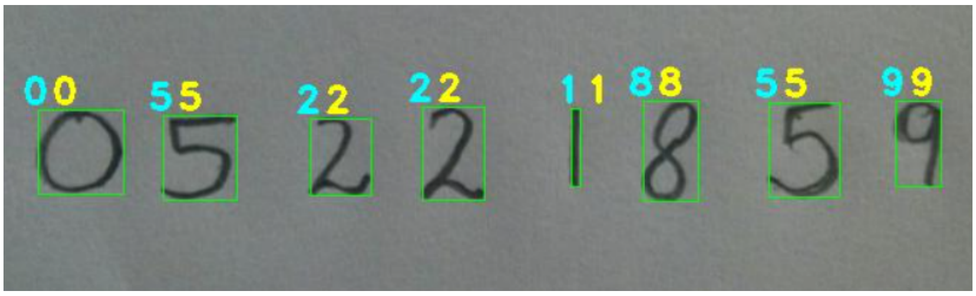


Figure 3.19: Pre-trained Model 1

In our transition from training a digit handwriting recognition system to a character handwriting recognition system, we follow a structured approach. Firstly, we carefully shape and normalize the data as part of our preparation process. We then apply carefully chosen data augmentation techniques, as outlined in task 2, to enhance the model's ability to generalize effectively. Subsequently, we adapt the pre-trained model by incorporating a new output layer designed specifically for Arabic character classification. This streamlined process capitalizes on the pre-trained model's existing knowledge, resulting in a considerable acceleration in the construction of an accurate and reliable Arabic character recognition system.

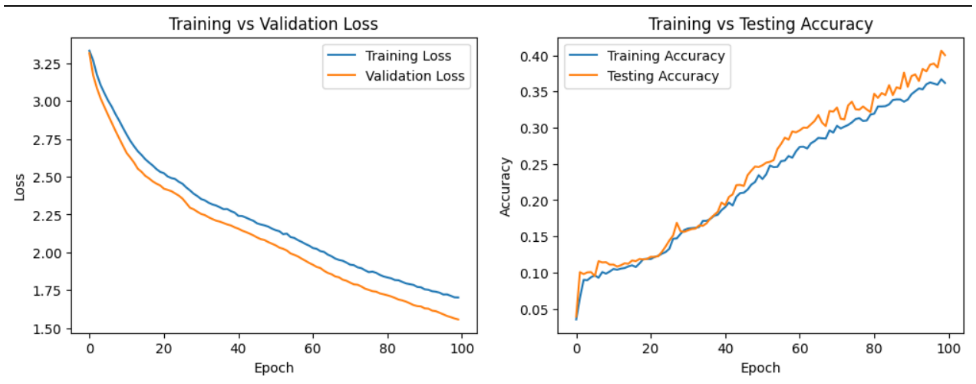


Figure 3.20: Pre-trained Model 1 Graphs

According to Figure 3.20, the training and validation loss graphs demonstrate a consistent decrease in the model's loss, indicating effective learning from the training data. A lack of an increase in validation loss further suggests that the model is not overfitting significantly. This is supported by the steady rise in both training and testing accuracy over time, with the testing accuracy closely tracking the training accuracy, indicative of the model's ability to generalize well to new, unseen data. However, the model's accuracy, reaching only around 40% by the 100th epoch, implies that it may not have reached an optimal level for practical application in handwritten character recognition.

3.4.2 Second Pre-trained Model

We decided to try a different approach based on English handwriting recognition models [8] since the previous model only achieved 40% accuracy. We used a pre-trained model that was originally made for recognizing English handwritten characters and adjusted it for our task of recognizing Arabic handwritten characters. We carefully prepared the data by shaping and normalizing it, and used data augmentation techniques from task 2 to improve the model's generalization. We changed the pre-trained model by adding a new output layer that was designed for Arabic character classification. This method aimed to use the advantages of a model that learned from a similar task and fine-tune it for our recognition needs, hoping for a more accurate and reliable Arabic character recognition system.

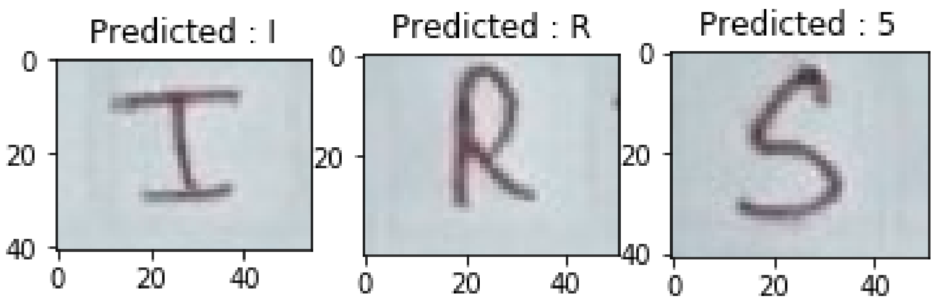


Figure 3.21: Pre-trained Model 2

Acording to figure 3.22, the model's performance improves steadily without overfitting, as shown by the decreasing training and validation losses in the first graph. The second graph displays a large improvement in accuracy for both datasets, indicating the model's good generalization. The epoch logs reveal a remarkable jump in testing accuracy from 14.8% to 76.6% by epoch 100, with training accuracy following suit. The final testing loss is much lower than the initial one, implying the model's high precision.

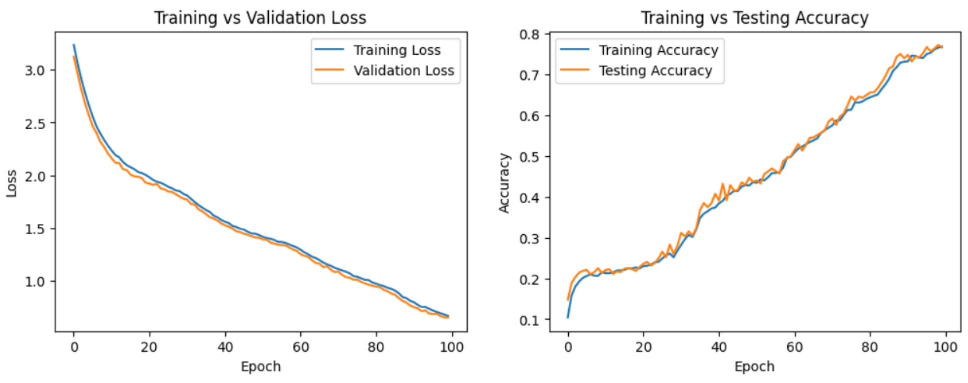


Figure 3.22: Pre-trained Model 2 Graphs

3.4.3 Improvement of Second Pre-trained Model

The model's performance increased greatly with a more intricate design and improved training. The initial model was improved with convolutional layers, max pooling, batch normalization, and a customized training rate, yielding an accuracy of 0.7680 and a loss of 0.6703. Its metrics significantly improved as a result of these adjustments, showing a loss of 0.0832 and an accuracy of 0.9752. This development demonstrates the impact of fine-tuning training parameters and employing a deeper neural network architecture. Batch normalization increases the speed and stability of training, while convolutional layers aid in extracting the spatial patterns from the images. These adjustments together make the model more reliable and precise, allowing it to recognize and classify images with higher accuracy.

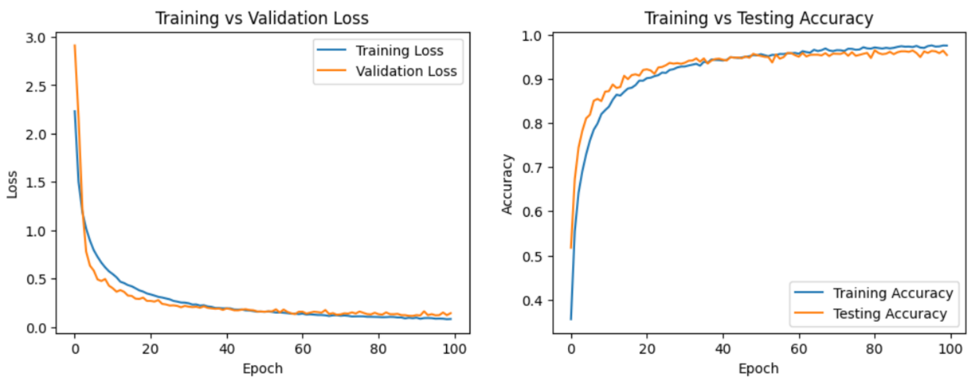


Figure 3.23: Improvement of Pre-trained Model 2 Graphs

According to Figure 3.23, the training and validation loss curves are closer to each other, showing better generalization and less overfitting. Also, both the training and testing accuracy curves not only meet at a higher value than before but also have a similar shape, implying that the changes in the model's structure and training method have resulted in a more reliable model that does well on new data. The testing accuracy, especially, has a sharper rise and reaches a higher level, highlighting the enhanced prediction ability of the improved model. Finally, the model gives us great results, with a loss of 0.0832 and an accuracy of 0.9752.

3.4.4 Task 4 Results Discussion

Compared to the previous tasks, Task 4 has produced a more refined model, as shown by the closer training and validation loss curves, indicating better generalization skills. The model shows a faster reach in the learning curve and achieves a high testing accuracy of 97.44%, which is somewhat lower than the 98.62% seen in Task 3. This indicates not only a higher rate of learning but also a resilience that exceeds the results of Tasks 1 and 2. The model's ability to handle unknowing data is shown by the constancy of its training and testing accuracy, which makes it an excellent resource for Arabic Handwritten Character Recognition.

Conclusion

Our project has consistently demonstrated notable enhancements in the accuracy and robustness of our Arabic Handwritten Character Recognition (AHCR) model. This success is largely attributed to our phased, iterative methodology. In Task 1, we laid the groundwork with an initial accuracy of 96.82%, which we further improved in Task 2 through data augmentation, resulting in greater stability and enhanced accuracy. The implementation of ResNet-50 in Task 3 was a pivotal development, propelling our accuracy to an exceptional 98.62%. Task 4 maintained this positive momentum, refining the model to achieve 97.44% accuracy, marked by better generalization capabilities and accelerated learning.

These developments underscore our systematic and progressive strategy, culminating in a robust AHCR system adept at deciphering various Arabic handwriting styles. Our work significantly contributes to the field of computer vision, particularly in addressing the intricacies of Arabic Handwritten Character Recognition.

Moreover, the reduced gap between training and validation metrics and the lower loss values further validate the strength and resilience of our model.

Bibliography

[1] R. Awati, “convolutional neural network (cnn),” 2021. Accessed on 14.1.2024 at 5:36 AM. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.

[2] “What are convolutional neural networks?,” 2021. Accessed on 18.1.2024 at 2:16 AM. https://www.ibm.com/topics/convolutional-neural-networks.

[3] “Cnn | introduction to pooling layer,” 2023. Accessed on 18.1.2024 at 2:21 AM. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[4] “What do the fully connected layers do in cnns?,” 2018. Accessed on 18.1.2024 at 2:21 AM. https://stats.stackexchange.com/questions/182102/what-do-the-fully-connected-layers-do-in-cnns.

[5] J. Brownlee, “A gentle introduction to transfer learning for deep learning,” September 16, 2019. Accessed on 18.1.2024 at 2:21 AM. https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[6] R. Maalej, “New mdlstm-based designs with data augmentation for offline arabic handwriting recognition,” 14 February 2022. Accessed on 18.1.2024 at 2:50 AM. https://link.springer.com/article/10.1007/s11042-022-12339-8.

[7] H. Digits, “New mdlstm-based designs with data augmentation for offline arabic handwriting recognition.” Accessed on 22.1.2024 at 2:50 AM. https://github.com/ehsanmqn/handwritten-digit-recognition/tree/master.

[8] E. handwriting, “New mdlstm-based designs with data augmentation for offline arabic handwriting recognition.” Accessed on 22.1.2024 at 2:55 AM. https://github.com/srijan14/keras-handwritten-character-recognition/blob/master/models/model.h5.