Faculty of Engineering and Technology

Computer Systems Engineering Department

COMPUTER VISION

ENCS5343

Course Project

Prepared By:

Hajar Salah 1191482

Sary Hammad 1192698

Instructor: Aziz Qaroush

Section: (2)

BIRZEIT

January– 2024

Table of Contents

3.1. Introduction

Recognizing handwriting poses a significant computer vision challenge, requiring systems to identify and convert handwritten text from various sources, including documents and touchscreens, into machine-readable formats. The input images may be sourced offline, such as from paper or photographs, or online from digital platforms like touchscreens. The unique patterns and styles inherent in handwritten text, influenced by factors such as age, background, language, and mental state, present a diverse set of challenges.

In the realm of automatic handwritten recognition, extensive exploration has been undertaken, employing diverse machine learning methods like K-nearest neighbors (KNNs), Support Vector Machines (SVMs), transfer learning, and Neural Networks (NNs). Particularly noteworthy in recent studies is the widespread adoption of Convolutional Neural Networks (CNNs), reflecting their efficacy in handling complex image recognition tasks.

Within the context of Arabic script, characterized by semi-cursive writing and a right-to-left orientation featuring 28 alphabet characters, the intricacies of automatic handwritten recognition are magnified. Each character exhibits multiple shapes based on its position in a word, rendering Arabic script recognition more challenging compared to other languages. This report delves into the application of advanced CNN architectures, such as AlexNet and EasyOCR, to address these complexities and enhance the precision of Arabic handwriting recognition. [1]

3.2. Experimental setup and results

Task 1-1: Building a Custom CNN Network for AHCR

The CNN architecture adopted for this task designed for AHCR encompasses various layer types, including Convolutional Layers, Pooling Layers, Fully Connected Layers, and Dropout Layers, as illustrated in Figure 2-1.

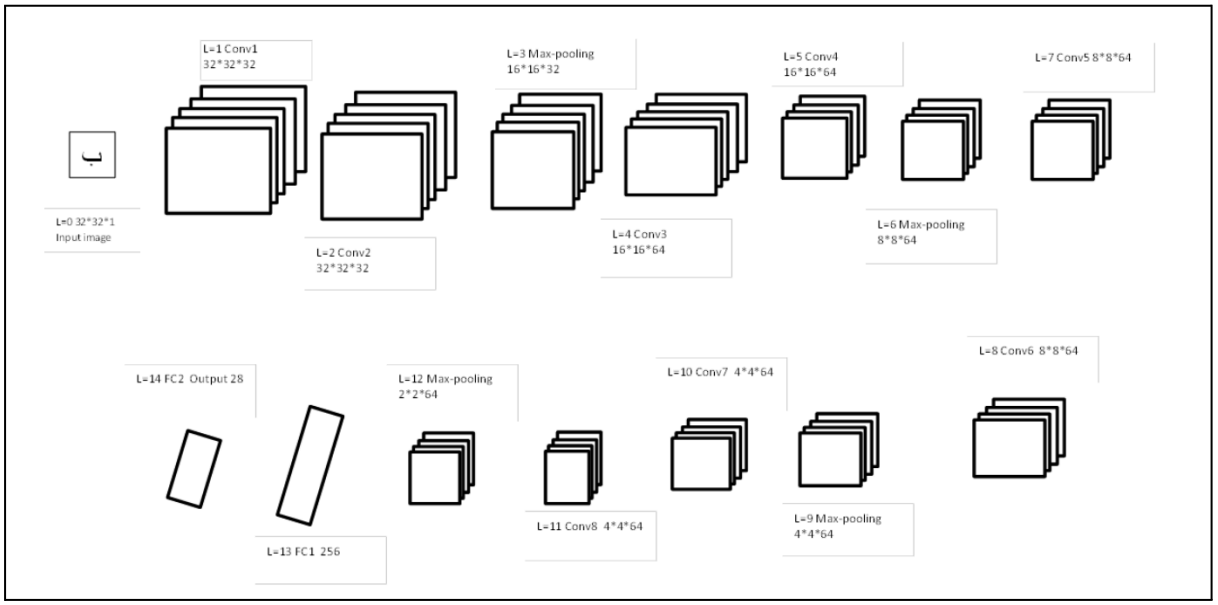

Figure 2-1: CNN-14 Model adopted [2]

Activation functions, specifically ReLU, were employed for both convolutional and fully connected layers. Concerning Convolutional Layer Parameters, the number of filters differed across layers (e.g., 32, 64), with a primary use of 3x3 filters. A default stride of 1 was applied, and padding of 1 was utilized to preserve spatial dimensions.

Pooling Layer Parameters specified a pool size of 2x2. In terms of Fully Connected Layer Parameters, the initial two fully connected layers incorporated 256 neurons each, while the output layer had 28 neurons, assuming a classification task with 28 classes corresponding to the number of Arabic letters.

For Training Hyperparameters, an Adam optimizer with the default learning rate was employed. The batch size for each training step was set to 64. Training extended over 100 epochs, and overfitting mitigation strategies included dropout, implemented in two instances (dropout1 and dropout2). These architectural and hyperparameter choices form the foundation of the model's structure and training characteristics.The dataset shown in appendix (A) was used as an input data.

Figure 2-2 shows the training accuracy vs. validation accuracy across epochs. It is noticed that both start from around 5%, which is probably just random guessing, as that would be 1/28, which is about 3.5%, and both rise up to around 97%.

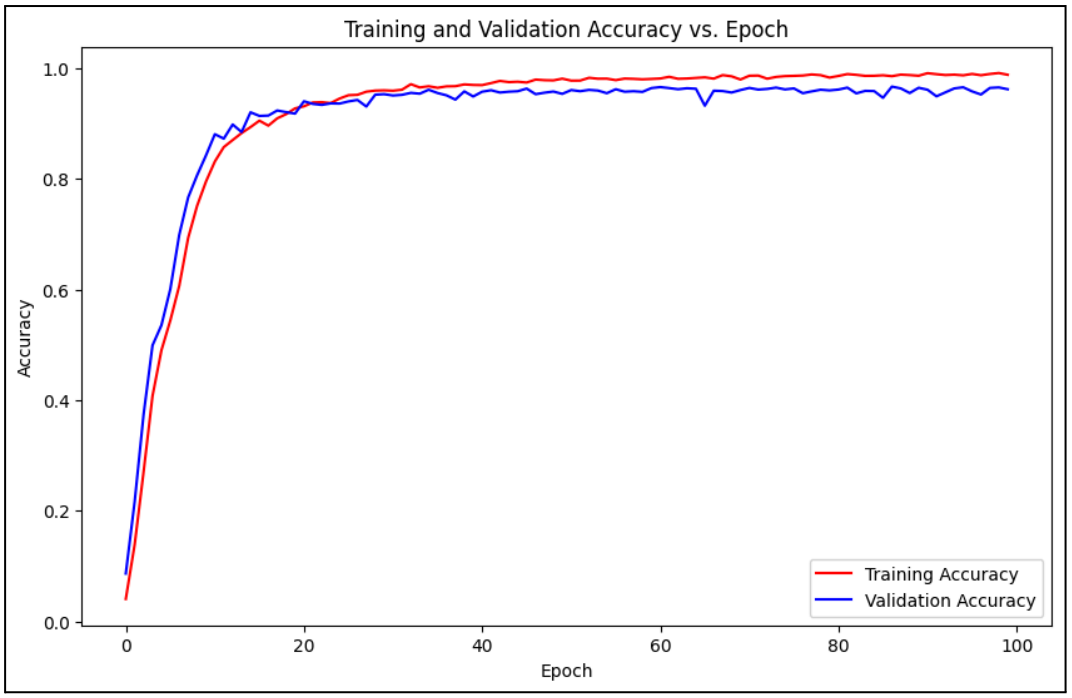

Figure 2-2: Training vs. Validation Accuracy through the epoch

Figure 2-3 shows the validation loss vs. training loss across epochs. It is noticed that the loss starts from around 3, and reaches down to almost 0.2.

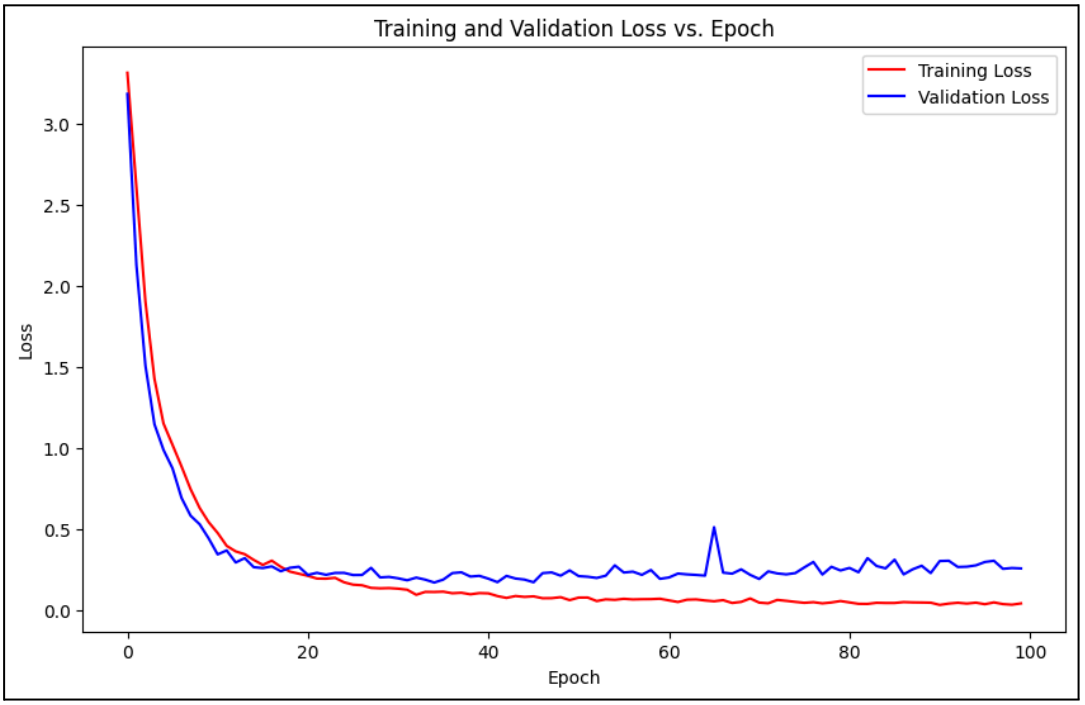

Figure 2-3: Validation Loss vs. Validation Loss through the epoch

For this model, it is noticed that both loss and accuracy curves reach almost a plateau, and both are very close to each other. The test accuracy of this model on the used database was remarkably high. The reported test accuracy stands at 96.19%., with Precision: 0.9631, Recall: 0.9619, and F1 Score: 0.9619.

Task 1-2: Building another Custom CNN Network For AHCR

Another CNN architecture from a different source was implemented in order to see what performs better, figure 2-4 shows the architecture from the source:

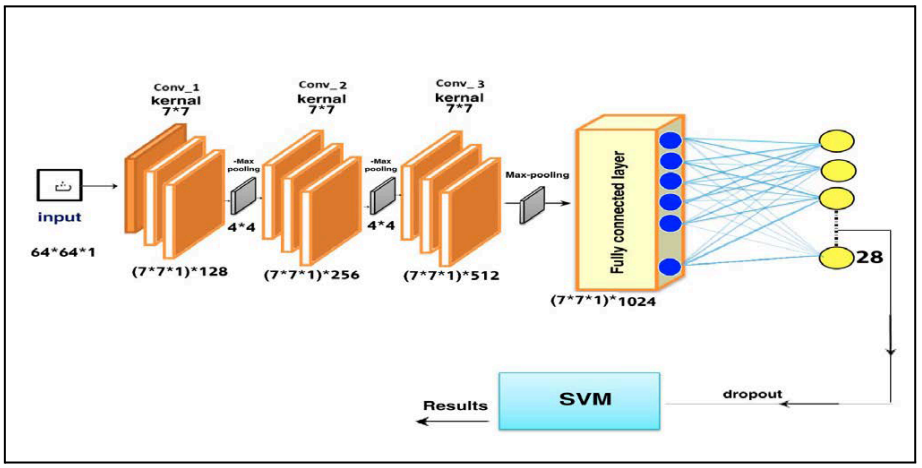

Figure 2-4: Model 2 Architecture [3]

And figure 2-5 shows the architecture after editing it for the dataset:

```
CNNModel2(
  (conv1): Conv2d(1, 128, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(128, 256, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(256, 512, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=8192, out_features=1024, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=1024, out_features=28, bias=True)
)
```

Figure 2-5: Model 2 Parameters

Figure 2-6 shows the training vs validation accuracy for this model:

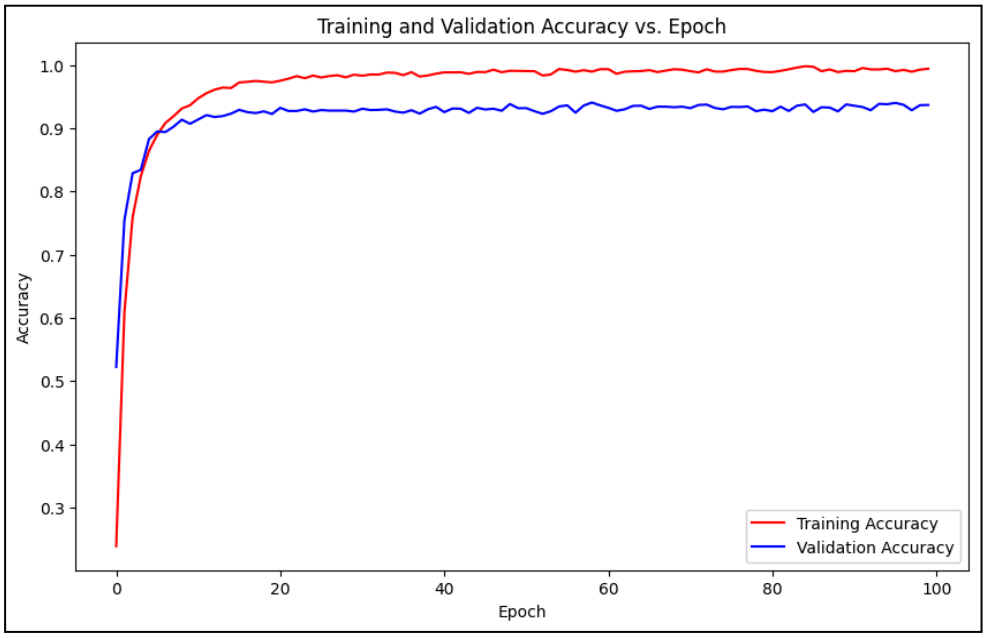

Figure 2-6: Training Accuracy vs. Validation Accuracy through the epoch

And figure 2–7 shows the training vs validation loss for this model:

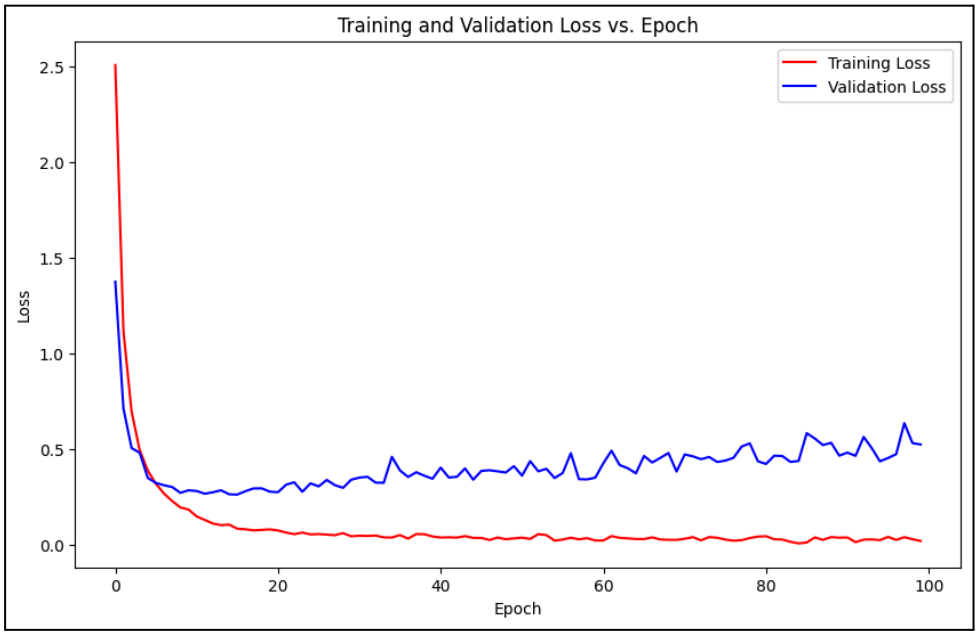

Figure 2-7: Validation Loss vs. Validation Loss through the epoch

It is immediately noticed that the gap between the training and validation has increased from that of model 1, this indicates more overfitting, which is somewhat true if this model's architecture was analyzed, as there is only one dropout layer, while the first model has 2. However, this is not a bad model as it also somewhat reaches a plateau at the end of both loss and accuracy curves, and the reported accuracy for this model is 93.69%, with

Precision: 0.9385, Recall: 0.9369, and F1 Score: 0.9369. But moving forward, model 1 will be adopted as the base model.

Task 2: Retrain The Selected CNN Network After Doing Data Augmentation

In this phase, the same steps as those followed in task 1 were utilized. However, the input images were augmented threefold through three different methods: the contrast of the input images was adjusted, salt noise was applied, and the images were rotated by 15 degrees. Consequently, the final input size was increased threefold compared to the original.

Figure 2-8 shows the training accuracy vs. validation accuracy across epochs. It is noticed that they also increase towards a plateau, and also that the validation accuracy starts higher than the training accuracy from the first epoch.

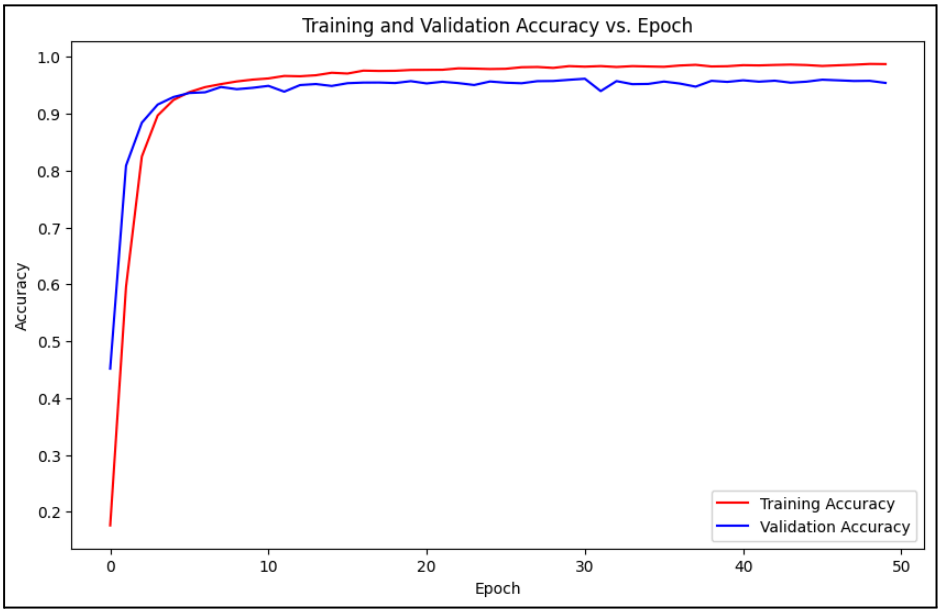

Figure 2-8: Training Accuracy vs. Validation Accuracy through the epoch

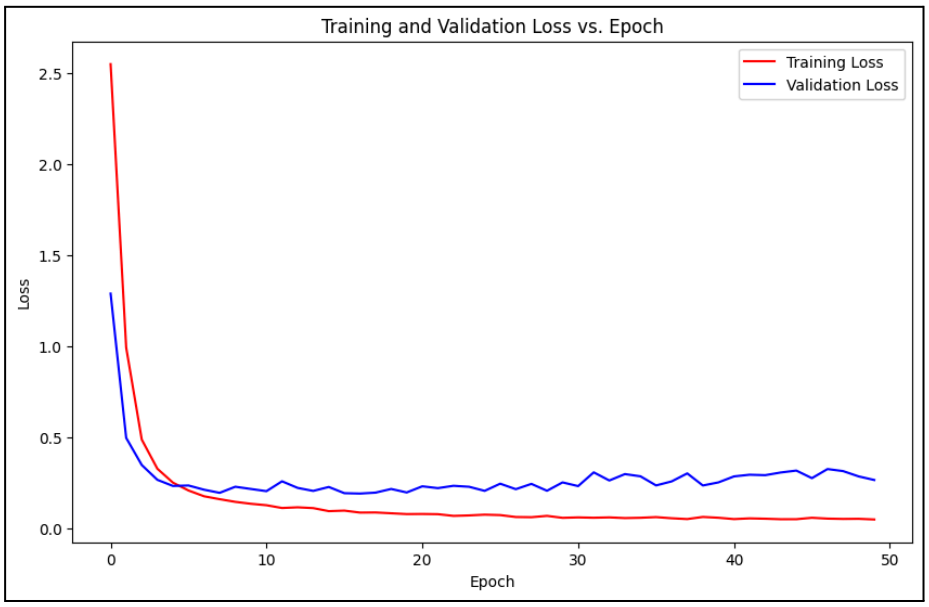

Figure 2-9: Training Loss vs. Validation Loss through the epoch

The same thing is observed regarding the validation loss starting much lower than the training loss when data augmentation was added, which is due to the fact that from the first epoch, the model is trained on a dataset more similar to what the validation set might include, through the augmentation techniques that were employed in this task.
The reported test accuracy stands at 95.41%, with Precision: 0.9551, Recall: 0.9542 and F1 Score: 0.9543.

Results comparison: Task1 vs. Task2

It is noticed that the accuracy of task 2 is a bit lower than the accuracy of the first model in task 1. Generally, data augmentation should improve the performance of the model on the validation set, as the model is exposed to a larger dataset with more variance that should help the model adapt to the unseen validation set. Perhaps the choice of augmentation techniques or epochs was not the best in this case. It is also noticed that the plateau in task 2 is straighter than the one in task 1, indicating steadier training figures, but as mentioned earlier, the final results indicated slightly worse performance on the validation set.

In addition, the accuracy figures in task 2 for both validation and training start higher than the ones in task 1, meaning that the model was learning the patterns better since epoch 1. But this could be explained by the larger dataset used in task 2. As each epoch in task 2, resemble 3 epochs from task 1 (thus our choice of lowering the epochs to 50 instead of 100).

Note that the gap between training and validation accuracy is a bit larger than in the non-augmented model, suggesting that the augmented data may be making the training task harder.

Task 3: AlexNet CNN Network

In this task, the AlexNet architecture was used as a CNN model for Arabic Handwritten Character Recognition (AHCR). The augmented input images produced in task 2 were used in this part as an input. Data loading involved reshaping and resizing images (into 227 x227). The AlexNetModel, representing the AlexNet architecture, is employed for AHCR. Training occurs over 50 epochs using CrossEntropyLoss and Adam optimizer. Overall, the code follows best practices, employing a comprehensive approach to AHCR with AlexNet, image resizing, and data augmentation.

Figure 2-10 shows the training accuracy vs. validation accuracy across epochs.

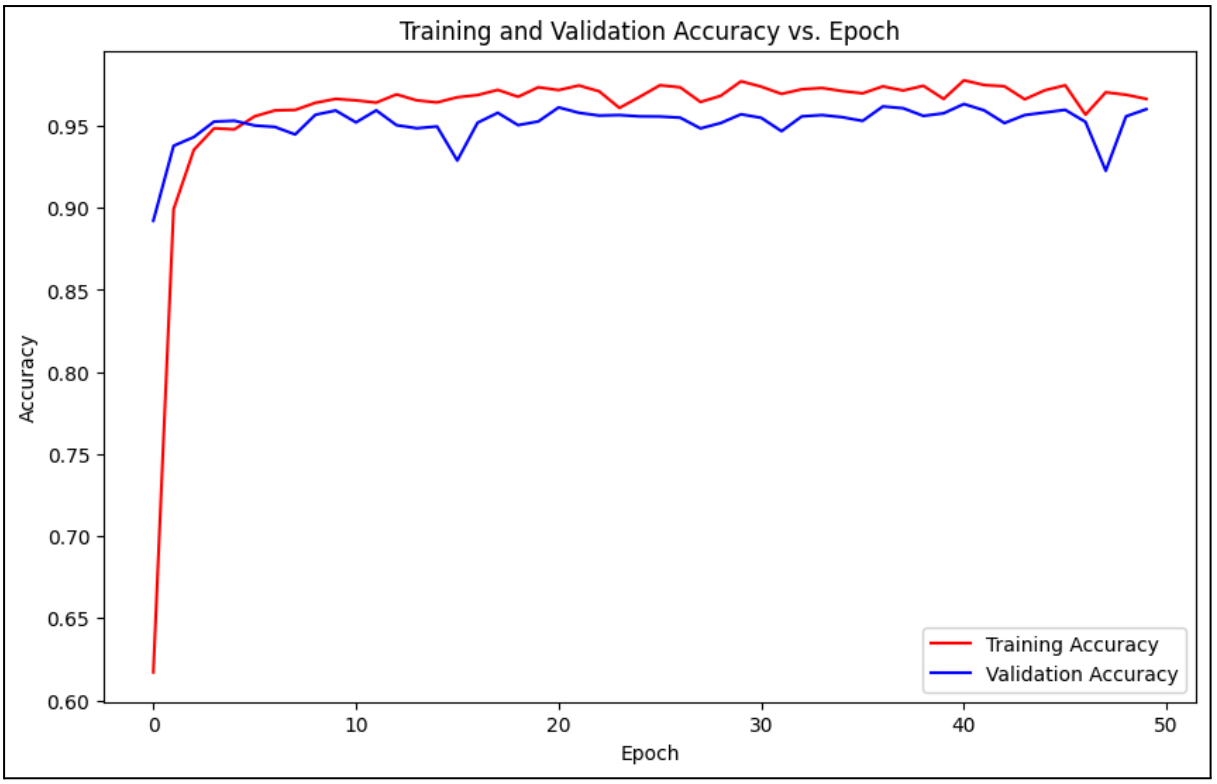

Figure 2-10: Training Loss vs. Training Accuracy through the epoch

Figure 2-11 shows the training loss vs. validation loss across epochs.

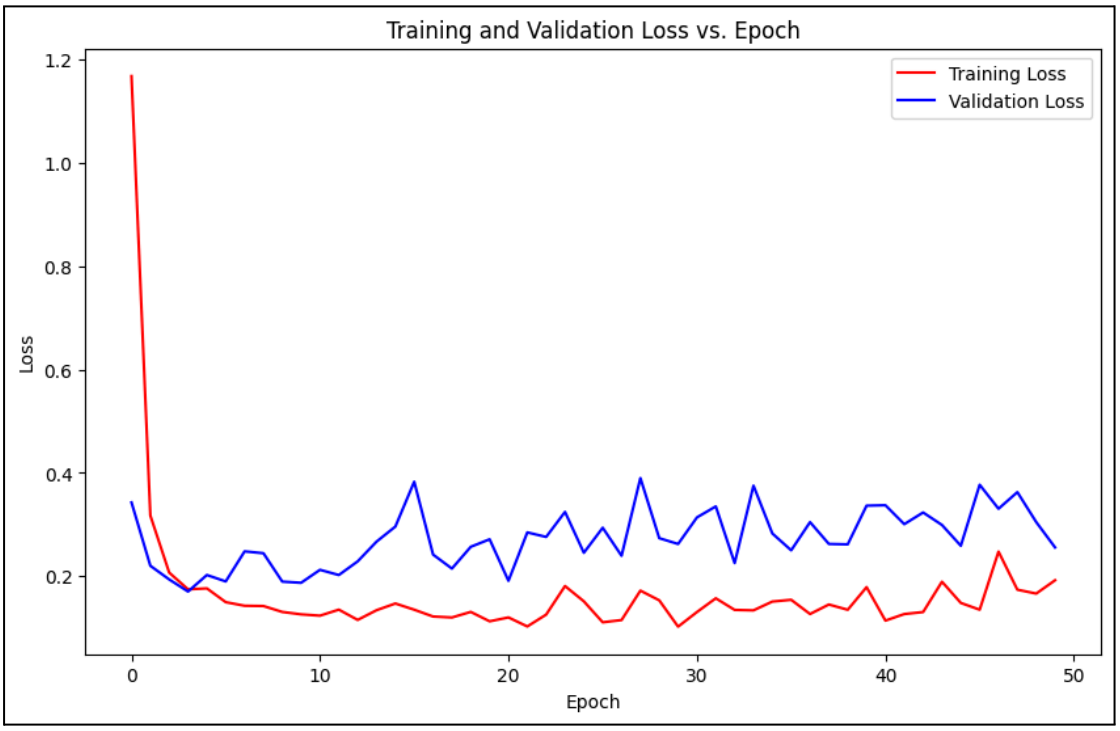

Figure 2-11: Training Loss vs. Validation Loss through the epoch

The reported test accuracy stands at 96.01%, with Precision: 0.9612, Recall: 0.9601, and F1 Score: 0.9601.

The training accuracy initiates at a high level and exhibits a gradual increase over epochs, implying effective learning from the training data. In parallel, the validation accuracy commences at a high value but demonstrates more variability compared to its training counterpart. Notably, the absence of a significant gap between training and validation accuracy is a positive indicator, suggesting minimal overfitting. Despite a slight divergence, the close alignment of the training and validation accuracy lines indicates robust generalization capabilities.

In terms of loss, the training loss experiences an initial rapid decline, followed by a stabilization phase, aligning with typical training patterns. Conversely, the validation loss undergoes an initial decrease but displays more pronounced fluctuations compared to the training loss. Instances of spikes in validation loss signal certain epochs where the model's performance falters on the validation set. Unlike the accuracy trends, the loss graph manifests a more substantial gap between training and validation loss, pointing to a notable degree of overfitting. This observation emphasizes the need for cautious consideration of model complexity and potential regularization techniques.

Results comparison: Task3 vs. Task 1 and Task2

Task 3 exhibits a noteworthy performance, achieving a high accuracy from the outset and maintaining a consistently strong performance across epochs. The final accuracy reaches approximately 96%, a value closely aligned with the peak accuracy observed in the initial model of Task 1. As a result, Task 3 outperforms the model featured in Task 2. It is essential to note, however, that the disparity between training and validation accuracy in Task 3 surpasses that observed in Task 1. This heightened gap implies a degree of overfitting during the training process for Task 3, highlighting the importance of addressing this challenge to enhance model generalization.

Task 4: Pre Trained GoogLeNet CNN Network

In this task, a research for public CNN based models that were pre trained on tasks similar to the AHCR task in order to apply transfer learning techniques was performed. The CNN model selected was GoogLeNet, a model specifically devoted to recognizing handwritten and machine printed characters. First, the pretrained model was loaded, then feature extraction was performed through removing the last two layers (fully connected and dropout layers) and using the remaining layers as feature extractors. Then the model was fine tuned as a new fully connected layer was added, and the last fully connected layer was modified to match the number of classes in the target dataset (28). This model was ran with CrossEntropyLoss and AdamOptimizer and the following results were achieved:

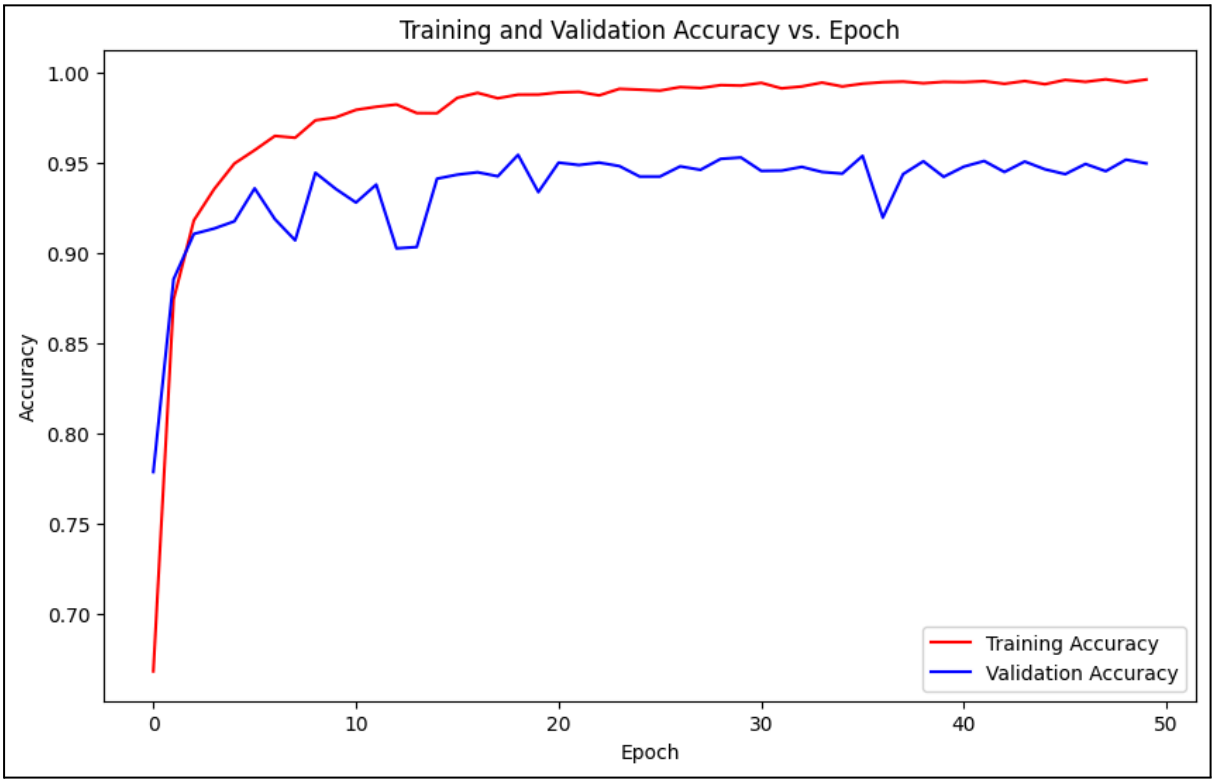

Figure 2-12: Training Accuracy vs. Validation Accuracy through the epoch

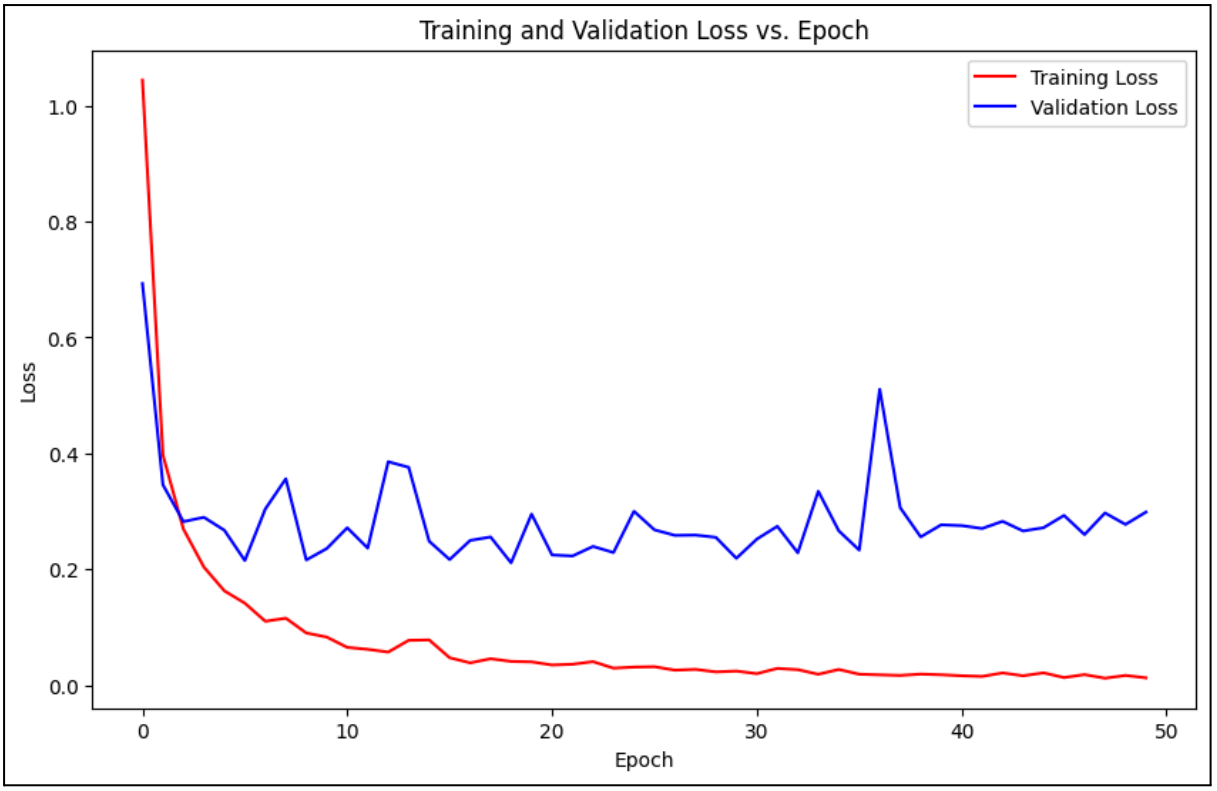

Figure 2-13: Training Loss vs. Validation Loss through the epoch

The reported test accuracy stands at 94.99%, with Precision: 0.9521, Recall: 0.9499, and F1 Score: 0.9498

The training accuracy starts off high and continues to increase slightly over time, indicating the model is learning from the training data. The validation accuracy also starts high but shows more fluctuation compared to the training accuracy. It's a good sign that it's not drastically lower than the training accuracy, which would indicate overfitting. The training and validation accuracy lines are close together, which suggests that the model is generalizing well. The slight gap indicates that there might be a bit of overfitting, but it's not significant.

The training loss decreases rapidly at first and then levels off, which is typical in the training process. The validation loss decreases initially but shows more volatility compared to the training loss. The spikes in validation loss suggest the model encounters some epochs where it doesn't perform as well on the validation set. Unlike the accuracy graph, the loss graph shows a larger gap between training and validation loss, which again points to some overfitting.

Results comparison: Task4 vs. all Previous Tasks:

It is noticed that the GoogLeNet model was able to achieve the highest starting accuracy in comparison with all the previous models, which is expected as it was employed on similar OCR tasks, making it somewhat easier for the model to get started and employ its previous learning from the first epoch.

The custom CNN architecture from task 1 seems to outperform the fine-tuned GoogLeNet model in terms of both accuracy and loss. It shows better stability (less fluctuation in validation metrics), closer performance between training and validation sets (indicating better generalization), and no significant signs of overfitting.

In the GoogLeNet results, there was a slight but consistent gap between the training and validation accuracy, indicating potential overfitting, albeit minor. The AlexNet results show a closer match between training and validation accuracy, which might suggest better generalization compared to the GoogLeNet model.

3.3. Conclusion

Throughout this project, the use of different CNN architectures was employed for the Arabic Handwritten Character recognition task. Among these models were two different custom architectures designed for AHCR, AlexNet, and GoogLeNet, yet the highest performer was the first custom CNN before the data augmentation. This was surprising as data augmentation is one of the most widespread methods of enhancing a model's generalization and performance.

It appears that the custom CNN from task 1 was able to attain a higher validation accuracy without relying on data augmentation techniques, which suggests that the model was well tuned and sufficiently complex to capture the important patterns of the data, achieving an optimal balance between learning performance and generalization. Perhaps the augmentation introduced a level of complexity that was not effectively captured by the validation data, or perhaps led to overfitting a little bit.

In regards to GoogLeNet and AlexNet, both demonstrated competent learning behaviors, yet they did not match the performance of the first custom CNN from task 1. Both models showed higher overfitting than the custom model, which was shown by the larger gap between the training and validation curves. In addition, these two models had fluctuating plateaus, meaning that on some epochs, these models were performing worse than expected in a fluctuating manner.

In conclusion, the custom CNN's superior performance without augmentation implies the importance of specific architectural designs for specific tasks, as a well-architectured model can surpass the benefits typically provided by augmentation techniques and by other widespread architectures that are known to generally perform well on certain general datasets. It also highlights the necessity of matching the complexity of the model to the pattern complexity of the dataset.

3.4. Appendix

Appendix (A)

https://drive.google.com/file/d/1ZQ8fSD6WgkXFBKIxMRBMn0-gTwzFjUvz/view?usp=sharing

3.5. References

[1] (PDF) handwriting recognition using artificial intelligence neural ... (n.d.-a). https://www.researchgate.net/publication/343345535_Handwriting_Recognition_using_Artificial_Intelligence_Neural_Network_and_Image_Processing

[2] Researchsquare. (n.d.). https://assets.researchsquare.com/files/rs-3141935/v1_covered_d5ccbfc9-e258-4ce8-bf21-d8549925d0cd.pdf?c=1691185945

[3] Arabic handwritten character recognition based on convolution neural ... (n.d.-a). https://www.researchgate.net/publication/344411338_Arabic_Handwritten_Character_Recognition_based_on_Convolution_Neural_Networks_and_Support_Vector_Machine