|  |  |
| --- | --- |
| **Name** | **id** |
| **ميريام سامح فهيم** | **20210983** |
| **ملك اشرف عبدالحميد الصياد** | **20210955** |
| **نادين طارق محمد** | **20210987** |
| **مونيا ايمن فوزي ذكي** | **20210975** |
| **ندى حاتم مهدى عيسى** | **20210994** |
| **علي ابراهيم عاصم** | **20210572** |

**Project Documentation**

**Title: Automated Optical Character Recognition of Handwritten Arabic Numerals using Artificial Neural Networks**

Introduction and Overview:

**Project Idea and Overview:**

The project aims to develop an Automated Optical Character Recognition system specifically designed for handwritten Arabic numerals. It involves the use of Artificial Neural Networks (ANNs) to analyze and recognize these handwritten characters accurately. The system will involve preprocessing steps, feature extraction, and ANN-based classification to achieve high recognition accuracy.

**Applications and Functionalities:**

Similar applications include:

* **Desktop Applications:** Systems such as Tesseract OCR and Adobe Acrobat for digit recognition but not specifically tailored for handwritten Arabic numerals.
* **Mobile Apps:** Some apps like Google Lens or Microsoft Office Lens provide general OCR functionality but may lack specialized support for handwritten Arabic numerals.

Functionalities include:

* **Image Preprocessing:** Cleaning, resizing, and normalization of input images.
* **Artificial Neural Network:** Implementing ANN models for classification and recognition of the numerals.
* **Output Display:** Displaying recognized numerals or exporting data to other applications.

Literature Review:

**1. Title: "Handwritten Arabic Numeral Recognition using Neural Networks"**

* Authors: A. El-Sawy, M. Loey, et al.
* Description: Discusses various ANN architectures for recognizing Arabic numerals and the challenges involved in handwritten digit recognition.

**2. Title: "A Review of Handwritten Arabic Numeral Recognition Techniques"**

* Authors: M. A. Abushariah, O. M. Alia, et al.
* Description: Provides a comprehensive review of different approaches used in recognizing handwritten Arabic numerals, including ANN-based methods.

**3. Title: "Recognition of Handwritten Arabic Numerals Using Convolutional Neural Network"**

* Authors: N. K. Al-Aidaroos, A. K. Bashier, et al.
* Description: Discusses the application of Convolutional Neural Networks (CNNs) in recognizing handwritten Arabic numerals, presenting comparisons and performance evaluations.

**4. Title: "Preprocessing Techniques for Handwritten Arabic Text Recognition"**

* Authors: S. R. Abdollahi, M. S. Karbasforoushan, et al.
* Description: Focuses on image preprocessing techniques relevant to recognizing handwritten Arabic characters, which could be beneficial in your project.

**5. Title: "Arabic Handwritten Digits Recognition Using Deep Belief Network"**

* Authors: A. D. Alharbi, M. S. Alayed, et al.
* Description: Discusses the application of Deep Belief Networks (DBNs) for recognizing Arabic handwritten digits, providing insights into different ANN architectures.

**Dataset**

<https://www.kaggle.com/datasets/mloey1/ahdd1>

**USE CASE DIAGRAM**

**A diagram of a system

Description automatically generated**

Main Functionalities/Features:

**1. User Input:**

* **Upload Image:** Users can upload images containing handwritten Arabic numerals for recognition.

**2. Preprocessing:**

* **Image Preprocessing:** Automatically clean, resize, and normalize the uploaded images to enhance the quality and standardize the input.

**3. Recognition Process:**

* **Handwritten Numeral Recognition:** Implement Artificial Neural Networks to analyze the preprocessed images and recognize handwritten Arabic numerals accurately.

**4. Display Output:**

* **Display Recognized Digits:** Present the recognized digits to the user with respective confidence scores or probabilities.

**SEQUENCE DIAGRAM**

**A diagram of a process flow

Description automatically generated**

**FLOWCHART DIAGRAM**

**A screenshot of a diagram

Description automatically generated**

**Recognition of Handwritten Arabic Numerals**

**ANN**

The First Step Preparing the data :

We define A variable train and test to load the dataset then we split the data into features and target using so we can put the features in X and the target in y.

After that for training and testing our model and see how it performs we needed to split the data into training and testing with train\_test\_split().

Then we wanted to see some samples of our dataset that’s why we used matplotlib to do some visualization

Then we divided the X\_train and X\_test by 255 to normalize them and by this the computation becomes easier and faster since all the numbers became in range of 0 and 1.

Then we reshape the data before entering the model because NN train faster on small images A larger input image requires the neural network to learn from four times as many pixels, and this increase the training time for the architecture.

Second Step Building the model:

Sequential() is to create a sequential model because we have many layers that

neurons of each layer are connected to the neurons of the next layer.

First Layer (Dense):

Dense(512, activation='relu', input\_shape=(28\*28,))

This adds a fully connected layer (Dense) with 512 units (neurons).

activation='relu': Applies Rectified Linear Unit (ReLU) activation function, which introduces non-linearity.

input\_shape=(28\*28,): Defines the input shape of the layer. Here, it indicates a flattened input of 28x28 pixels (MNIST image size).

Dropout Layer:

model.add(Dropout(0.5))

Adds a Dropout layer to prevent overfitting by randomly dropping 50% of the neurons during training. It helps in improving generalization.

Second Layer (Dense):

Dense(256, activation='relu')

Another fully connected layer with 256 neurons and ReLU activation.

Output Layer (Dense):

Dense(10, activation='softmax')

This is the output layer with 10 neurons, representing 10 classes (assuming it's a classification task).

activation='softmax': Utilizes the softmax activation function to output probabilities for each class. It's commonly used in multi-class classification problems as it ensures the sum of output probabilities is 1.

This ANN architecture consists of three layers: two fully connected (Dense) hidden layers with ReLU activation and Dropout regularization, followed by an output layer with softmax activation for classification tasks where the model predicts one of ten classes

Third Step: Training a network

It is a process of finding weights in fully connected layers which minimize differences between output predictions and given ground truth labels on a training dataset.

A model performance under weights is calculated by a loss function through forward propagation on a training dataset, and learnable parameters weights, are updated according to the loss value through an optimization algorithm called backpropagation and gradient descent, among others loss function, measures the compatibility between output predictions of the network through forward propagation and given ground truth labels.

We compile the model with 'categorical\_crossentropy’ loss function is used for multiclass classification model where there are two or more output labels.

The optimizer is an algorithm that is used to change the attributes of the neural

network such as weights and bias. Adam optimization is a stochastic gradient descent

method that is based on adaptive estimation of first-order and second-order moments.

The optimizer generates the hyperparameters and reached the best values for the best

accuracy, save them in neural\_network3.h5 verbose is 1 to see the output progress bar while saving the weights.

we store all the data about model training in history variable to visualization it using

matplotlib.pyplot library or we can use it to know any information about model training

like loss, val\_loss and accuracy.

A graph of a training and training loss

Description automatically generated with medium confidence

Forth step: Save & load model:

We save the model after generating the weights and checkpoints, to use it at any time

without training from scratch. To load the model we load it into variable through

models.load\_model

Plots: we use plots to show the history of loss during training accourding to the epoch.

That show we have reached the minimum loss and save it.

Model Evaluation:

We followed some steps in our model during the design to reach the best results.

1. Regularization: with the early stopping, we added this feature to the model so it will stop training the model in a point which where there is no point to continue because this will result in a better accuracy.
2. Rescaling: we rescaled all the images by dividing the training and the testing data by 255 this will make our machine learning model train faster with smaller images and result in saving time and resources
3. Reshaping: Reshaping basically means, changing the shape of an array. And the shape of an array is determined by the number of elements in each dimension. Reshaping allows us to add or remove dimensions in an array. We can also change the number of elements in each dimension, and we had to do it so the model can train well
4. hyperparameter tuning: we used hyperparameters so we can get that accuracy and result.
5. Visualization: we used many plots for better understanding and visualization of the model and to see all the progress.

Development platforms:

We used google Kaggle and Jupyter notebook to write and execute arbitrary python code.

• Numpy: is used to convert the image into an array and to deal with it.

.Sklearn: we used it for splitting the dataset.

• Keras: deep learning API written in Python, running on top of the machine learning platform

TensorFlow.

• Tensorflow: TensorFlow is the open-source library for several various tasks in machine learning.

• Matplotlib: Matplotlib is a comprehensive library for creating static and interactive visualizations in

Python to virtualize data.

• Tkinter: Tkinter is the standard GUI library for Python.

Pandas: we used this library to read the csv file.

**Github link:**

**https://github.com/meriamsameh/Arabic-handwritten-digits-recognition**

**Testing The Model**

**https://github.com/meriamsameh/Arabic-handwritten-digits-recognitionA screen shot of a computer

Description automatically generated**