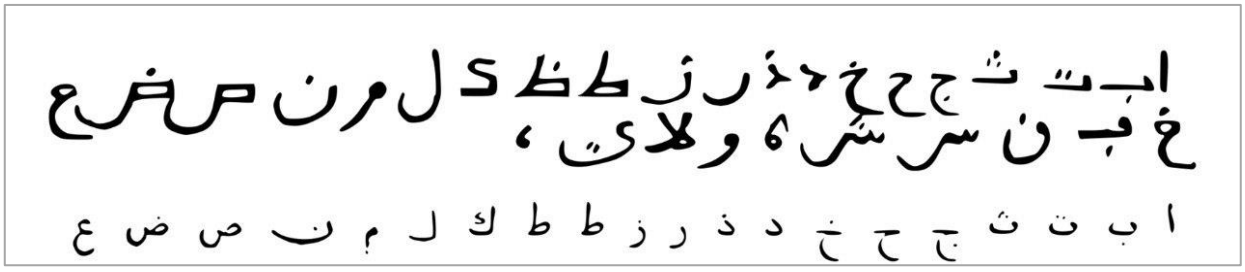
University of Jeddah
Collage of computer science &
Artificial Intelligence

# Deep Learning
# PROJECT



| Team Members: | |
|---|---|
| Raghad Almitairi | 2110409 |
| Renad Alzahrani | 2110292 |
| Rawan Jaafari | 2111481 |

| Instructor: |
|---|
| Nuha Babaker |

# Introduction:

One of the challenges in the field of deep learning and artificial intelligence that needs to be solved is how to recognize handwritten Arabic letters using deep neural network models. Therefore, our project aims to build a deep neural network model that can recognize handwritten Arabic letters. The problem of recognizing handwritten Arabic letters is an important topic that deserves attention. Arabic letters are characterized by geometric complexity and detail that makes them difficult to recognize by computers. Automatic recognition of Arabic letters requires the ability to deal with associated challenges, such as the high similarity between some letters and subtle differences in their composition and presence in words. There are many different types of letters in the Arabic language, and this is one of the reasons why it is considered one of the most difficult and complex languages. Moreover, there may be other challenges that we may have to overcome as well. However, progress in the field of automatic Arabic character recognition can contribute to the development of technology and progress in the fields of... It is important to use methods and techniques in deep neural networks to be able to accurately recognize and differentiate between Arabic letters so that high accuracy can be obtained. As a result of this project, we will be able to provide a comprehensive view of both the dataset that was used as well as the deep neural network that was used. And presenting the results and conclusions that aim to achieve high accuracy of the test data, which makes it possible to develop solutions to Arabic letter recognition problems and enhance deep learning capabilities.

# Details of Dataset used:

The dataset used in this context consists of 16,800 handwritten characters contributed by 60 participants. These letters represent the Arabic script. Participants ranged in age from 19 to 40 years, and the majority (90%) were right-handed. Each participant was assigned to write each letter ten times on pre-determined templates. The data set is divided into two main subsets: the training set and the test set. The training set consists of 13,440 characters, with 480 images for each character class. On the other hand, the test set contains 3,360 characters, with 120 images for each character class. The selection process ensures that there is no overlap between individuals included in the training and testing sets. The dataset provides a valuable resource for training, testing, and benchmarking Arabic character recognition models, and serves as a basis for further research and development in this area.

# Model:

Convolutional Layers:
Layer 1: Conv2D with 32 filters of size (3, 3) and ReLU activation function.
Layer 2: MaxPooling2D with a window size of (2, 2).
Layer 3: Conv2D with 64 filters of size (3, 3) and ReLU activation function.
Layer 4: MaxPooling2D with a window size of (2, 2).
Layer 5: Conv2D with 128 filters of size (3, 3) and ReLU activation function.
Layer 6: MaxPooling2D with a window size of (2, 2).

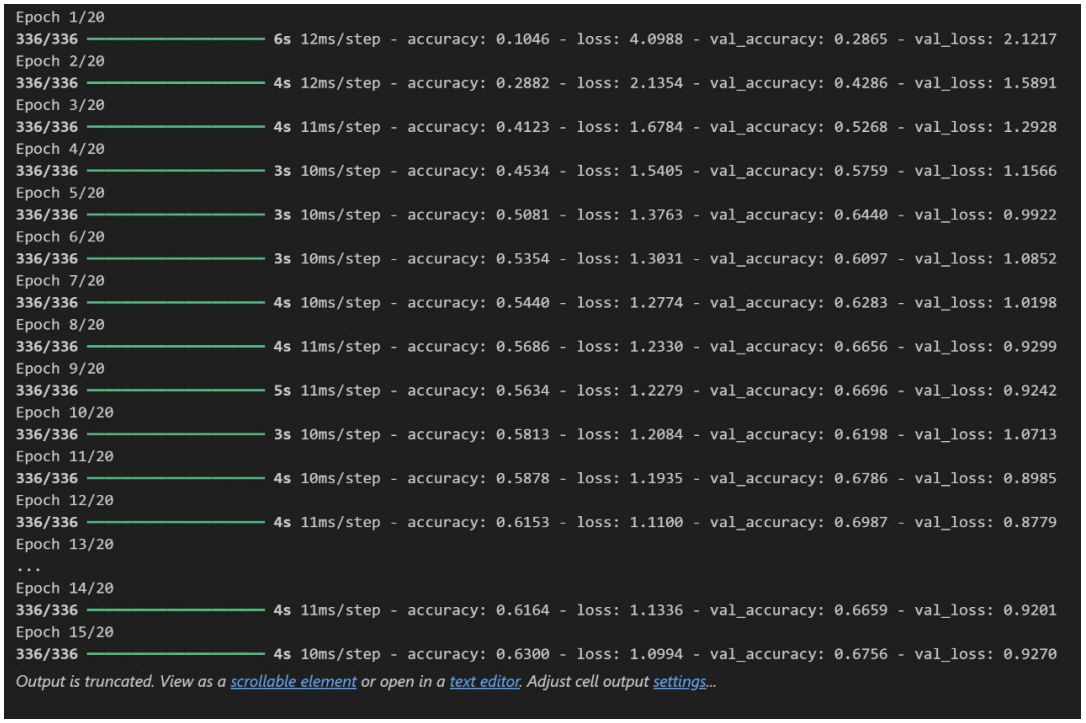
Flattening Layer:
This layer is used to flatten the output from the previous layer into a 1-dimensional vector.

Fully Connected Layers:
Layer 7: Dense layer with 128 units and ReLU activation function.
Dropout: Dropout layer with a rate of 0.5. It helps prevent overfitting by randomly dropping out a fraction of the input units during training.
Layer 8: Dense layer with a number of units equal to the number of classes in the output, and softmax activation function. It produces the final classification probabilities.
The model uses the Adam optimizer with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function. The metric used for evaluation is accuracy.

# Results:

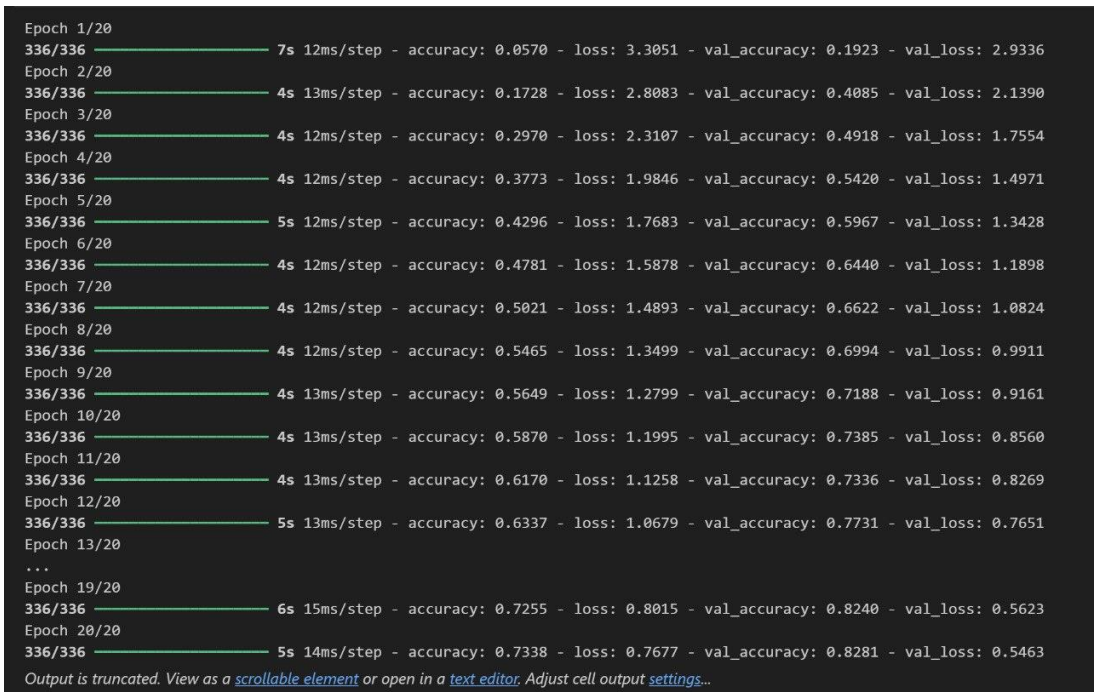
**1:**

| Learning rate | epoch | Dropout | accuracy |
|---|---|---|---|
| 0.01 | 20 | 0.5 | 63 |




**2:**

| Learning rate | epoch | Dropout | accuracy |
|---|---|---|---|
| 0.0001 | 20 | 0.5 | 82 |



**3:**

| Learning rate | epoch | Dropout | accuracy |
|---|---|---|---|
| 0.0001 | 20 | 0.25 | 91 |

```
Epoch 1/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1818 - loss: 2.7703 - val_accuracy: 0.6347 - val_loss: 1.0402
Epoch 2/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.6353 - loss: 1.0710 - val_accuracy: 0.7738 - val_loss: 0.6637
Epoch 3/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.7623 - loss: 0.6978 - val_accuracy: 0.8538 - val_loss: 0.4538
Epoch 4/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.8250 - loss: 0.4976 - val_accuracy: 0.8653 - val_loss: 0.3988
Epoch 5/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.8711 - loss: 0.3813 - val_accuracy: 0.8765 - val_loss: 0.3784
Epoch 6/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8990 - loss: 0.3084 - val_accuracy: 0.9003 - val_loss: 0.3054
Epoch 7/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9089 - loss: 0.2767 - val_accuracy: 0.9129 - val_loss: 0.2630
Epoch 8/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - accuracy: 0.9268 - loss: 0.2209 - val_accuracy: 0.9152 - val_loss: 0.2706
Epoch 9/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9391 - loss: 0.1772 - val_accuracy: 0.9267 - val_loss: 0.2294
Epoch 10/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9507 - loss: 0.1506 - val_accuracy: 0.9074 - val_loss: 0.2741
Epoch 11/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9477 - loss: 0.1418 - val_accuracy: 0.9245 - val_loss: 0.2321
Epoch 12/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9567 - loss: 0.1247 - val_accuracy: 0.9263 - val_loss: 0.2342
```

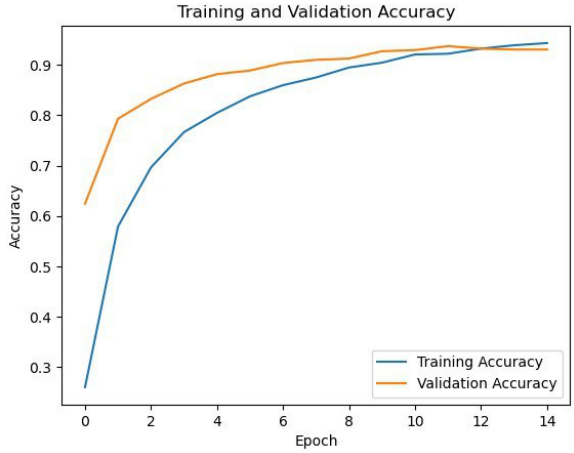
4:

| Learning rate | epoch | Dropout | accuracy |
|---|---|---|---|
| 0.001 | 20 | 0.5 | 93 |

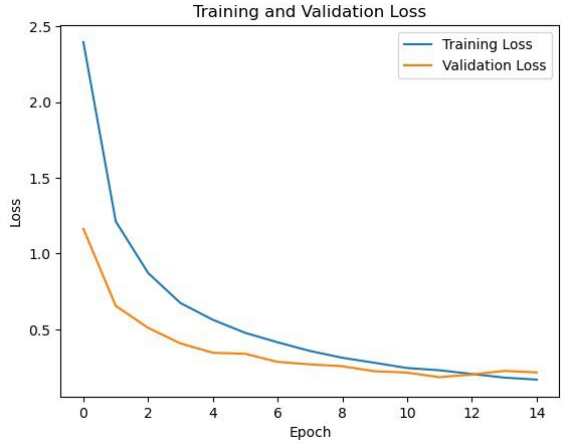
```
Epoch 1/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.1384 - loss: 2.9177 - val_accuracy: 0.6243 - val_loss: 1.1636
Epoch 2/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.5354 - loss: 1.3393 - val_accuracy: 0.7928 - val_loss: 0.6544
Epoch 3/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.6832 - loss: 0.9089 - val_accuracy: 0.8322 - val_loss: 0.5093
Epoch 4/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.7630 - loss: 0.6849 - val_accuracy: 0.8627 - val_loss: 0.4065
Epoch 5/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8062 - loss: 0.5720 - val_accuracy: 0.8813 - val_loss: 0.3447
Epoch 6/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8352 - loss: 0.4779 - val_accuracy: 0.8884 - val_loss: 0.3379
Epoch 7/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8627 - loss: 0.4108 - val_accuracy: 0.9033 - val_loss: 0.2845
Epoch 8/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.8732 - loss: 0.3656 - val_accuracy: 0.9096 - val_loss: 0.2681
Epoch 9/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8954 - loss: 0.3053 - val_accuracy: 0.9122 - val_loss: 0.2555
Epoch 10/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9060 - loss: 0.2724 - val_accuracy: 0.9267 - val_loss: 0.2227
Epoch 11/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9197 - loss: 0.2437 - val_accuracy: 0.9289 - val_loss: 0.2137
Epoch 12/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9283 - loss: 0.2152 - val_accuracy: 0.9368 - val_loss: 0.1834
Epoch 13/20
...
Epoch 14/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9412 - loss: 0.1708 - val_accuracy: 0.9301 - val_loss: 0.2253
Epoch 15/20
336/336 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9405 - loss: 0.1733 - val_accuracy: 0.9301 - val_loss: 0.2148
```

**We were able to achieve excellent results when we achieved an epoch value of 20 and a learning rate of 0.001. As a result of this case, The model obtained high accuracy**

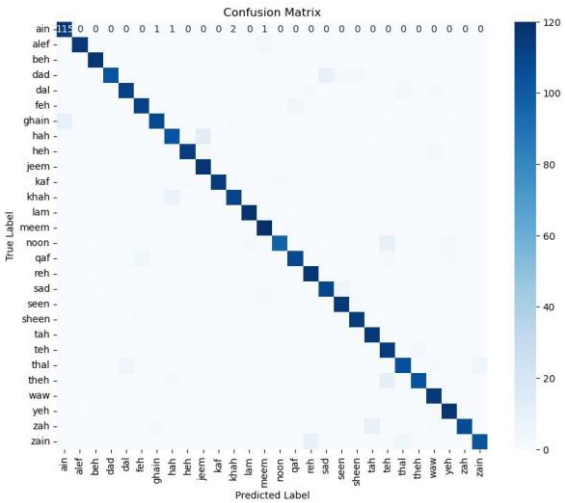
Training and validation accuracy:




Training and validation loss :




confusion matrix:



# Conclusion:

In conclusion, the described dataset represents a valuable resource for training and testing models for Arabic handwritten character recognition. The dataset has shown promising progress in training a Convolutional Neural Network (CNN) model to recognize handwritten Arabic letters. With a test accuracy of 0.93, the CNN model can be relied upon to accurately recognize Arabic letters in different writing styles. The promising results confirm the success of the training and the CNN model's ability to accurately recognize handwritten Arabic letters.