Arabic Handwritten Characters Recognition using Convolutional Neural Network

CCAI 435| Deep Learning Course

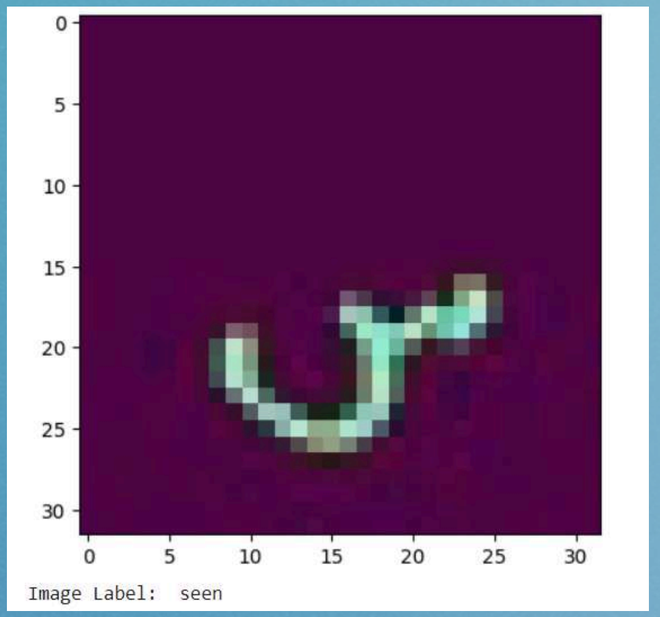
Introduction to problem

The handwritten character recognition problem, like in the well-known MNIST dataset, has been a research focus for years due to its importance in automated text processing. Unfortunately, most of the attention has been on English characters. Recognizing handwritten Arabic characters is tough because of the diversity of writing styles, character shapes, and the nature of the script. This project aims to develop a model to classify handwritten Arabic characters by using advanced deep learning models and the Arabic Handwritten Chars dataset.

Dataset Description

The Arabic Handwritten Characters dataset, introduced in a paper by El-Sawy et al., contains all Arabic characters handwritten by 60 participants of varying ages. The dataset is divided into two splits: a training set with 13,440 images and a testing set with 3,360 images all have the same size (32,32). Each character's label is linked to the corresponding image name, which represents the character's letter name. The following figures show the size of dataset and sample images from the dataset:

```
Train Data shape (13440, 32, 32, 3)
Train Labels shape (13440,)
Test Data shape (3360, 32, 32, 3)
Test Labels shape (3360,)
```



Neural Network Architecture & Results

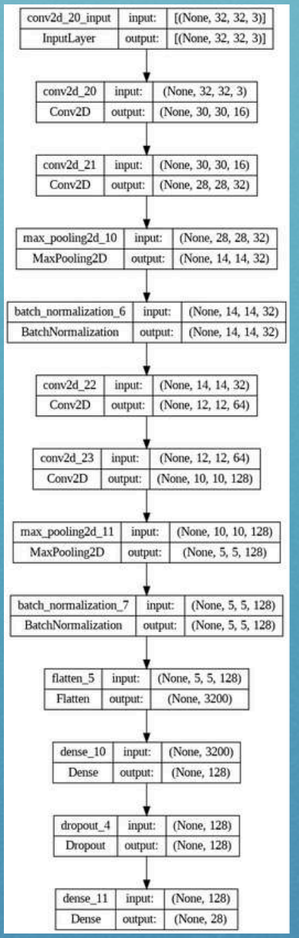
Best Neural Network Architecture

We have constructed a neural network architecture with a multi-layer convolutional neural network (CNN) implemented using TensorFlow's Keras. Below, we will mention the main features of our network:

- **12 layers in total, including:**
- **Feature Extraction:**
  - Conv2D layers with increasing filters (16, 32, 64, 128), all using a filter size of (3,3) and stride of 1
  - MaxPooling and Batch Normalization for regularization and dimension reduction
- **Classification:**
  - Flatten layer to transition from 2D to 1D
  - Dense layers for classification:
    - First layer: 128 units with L1 and L2 regularization (l1=0.0001, l2=0.0001)
    - Second layer: 28 units with softmax activation
  - Dropout at 50% to prevent overfitting

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 30, 30, 16)	448
conv2d_21 (Conv2D)	(None, 28, 28, 32)	4640
max_pooling2d_10 (MaxPooling2D)	(None, 14, 14, 32)	0
batch_normalization_6 (BatchNormalization)	(None, 14, 14, 32)	128
conv2d_22 (Conv2D)	(None, 12, 12, 64)	18496
conv2d_23 (Conv2D)	(None, 10, 10, 128)	73856
max_pooling2d_11 (MaxPooling2D)	(None, 5, 5, 128)	0
batch_normalization_7 (BatchNormalization)	(None, 5, 5, 128)	512
flatten_5 (Flatten)	(None, 3200)	0
dense_10 (Dense)	(None, 128)	409728
dropout_4 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 28)	3612

Best Neural Network Architecture




```
1 his_6 = model_6.fit(train, train_labels, epochs=10, validation_split=0.2, batch_size=32)
```

```
Epoch 1/10
336/336 [==============================] - 5s 7ms/step - loss: 2.9316 - accuracy: 0.3899 - val_loss: 7.3247 - val_accuracy: 0.0368
Epoch 2/10
336/336 [==============================] - 2s 6ms/step - loss: 1.5677 - accuracy: 0.7397 - val_loss: 2.0505 - val_accuracy: 0.5945
Epoch 3/10
336/336 [==============================] - 2s 7ms/step - loss: 1.1559 - accuracy: 0.8453 - val_loss: 1.5237 - val_accuracy: 0.7165
Epoch 4/10
336/336 [==============================] - 2s 7ms/step - loss: 0.9503 - accuracy: 0.8922 - val_loss: 1.0590 - val_accuracy: 0.8423
Epoch 5/10
336/336 [==============================] - 2s 6ms/step - loss: 0.8267 - accuracy: 0.9115 - val_loss: 1.1662 - val_accuracy: 0.8006
Epoch 6/10
336/336 [==============================] - 2s 6ms/step - loss: 0.7222 - accuracy: 0.9327 - val_loss: 0.8289 - val_accuracy: 0.8943
Epoch 7/10
336/336 [==============================] - 2s 6ms/step - loss: 0.6880 - accuracy: 0.9348 - val_loss: 1.0195 - val_accuracy: 0.8348
Epoch 8/10
336/336 [==============================] - 2s 6ms/step - loss: 0.6662 - accuracy: 0.9381 - val_loss: 0.7760 - val_accuracy: 0.9077
Epoch 9/10
336/336 [==============================] - 2s 6ms/step - loss: 0.6487 - accuracy: 0.9423 - val_loss: 0.6551 - val_accuracy: 0.9457
Epoch 10/10
336/336 [==============================] - 3s 7ms/step - loss: 0.6111 - accuracy: 0.9523 - val_loss: 0.6330 - val_accuracy: 0.9505
```

```
1 loss, acc = model_6.evaluate(test, test_labels)
```

```
105/105 [==============================] - 0s 3ms/step - loss: 0.6352 - accuracy: 0.9482
```

Best Neural Network Training and Results

- **Configuration:**

  Loss: Sparse categorical crossentropy

  Optimizer: Adam

  Metric: Accuracy

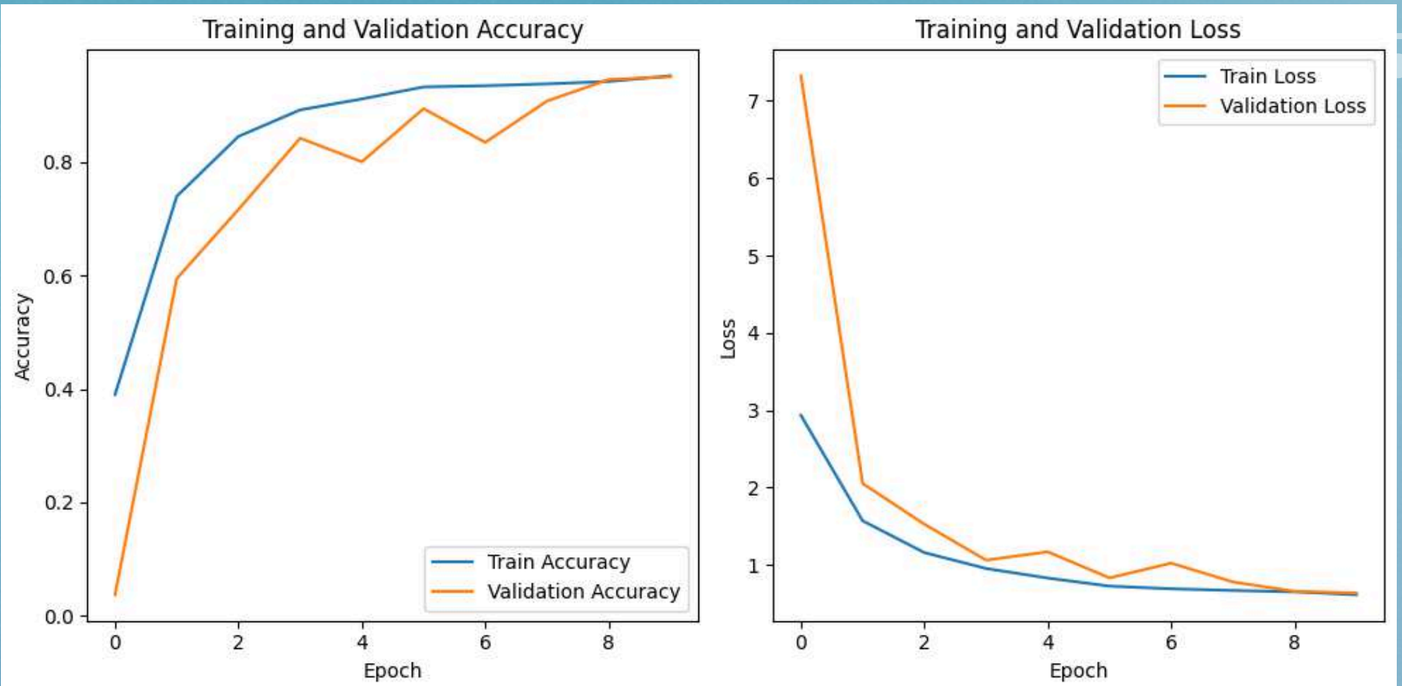
- **Training:**

  Epochs: 10

  Validation split: 20%

  Batch size: 32

- **Results**

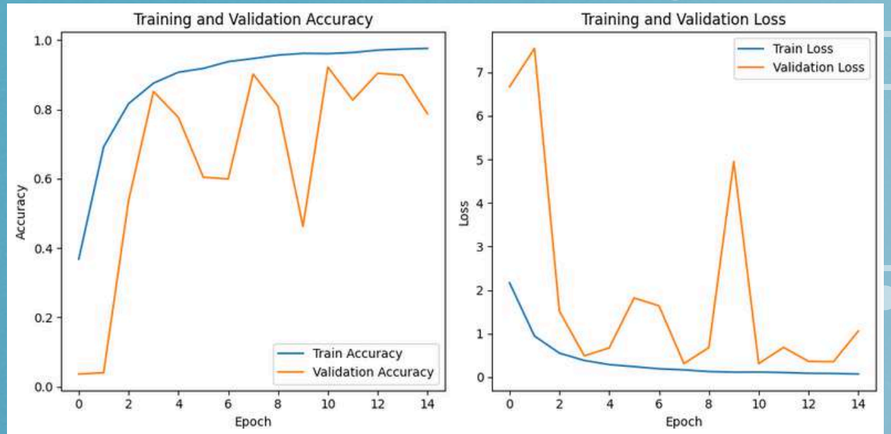



```
loss: 0.6111 - accuracy: 0.9523 - val_loss: 0.6330 - val_accuracy: 0.9505
```

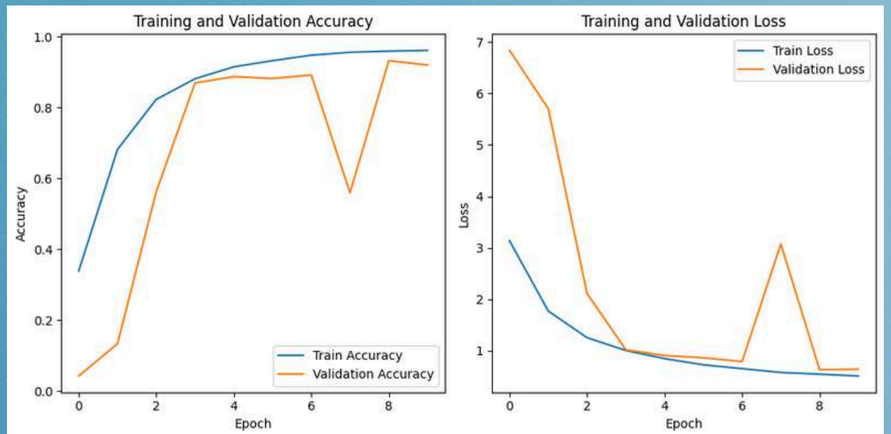
```
loss: 0.6352 - accuracy: 0.9482
```

Our Experiment

We trained our final model using over 6 different architectures and hyperparameters. In the process of training, we have had issues like overfitting and low validation accuracy. We have tried to overcome these challenges through dropout and regularization techniques. The following figure shows a sample of our trail to reach the best-performing model architecture and the most recent one for the dataset.




```
loss: 0.0706 - accuracy: 0.9764 - val_loss: 1.0587 - val_accuracy: 0.7876
105/105 [==============================] - 1s 4ms/step - loss: 1.0636 - accuracy: 0.7851
```




```
loss: 0.5051 - accuracy: 0.9619 - val_loss: 0.6384 - val_accuracy: 0.9204
105/105 [==============================] - 0s 3ms/step - loss: 0.6445 - accuracy: 0.9176
```

Conclusion

After testing different neural network architectures on the Arabic Handwritten Characters dataset, we saw significant variations in performance.The last Trial , achieved the best results with a testing accuracy of 0.948% , a training accuracy of 0.952% and a validation of 0.950%.

Reference

- Elsawy, Ahmed & Loey, Mohamed & El-Bakry, Hazem. (2017). Arabic Handwritten Characters Recognition using Convolutional Neural Network. WSEAS TRANSACTIONS on COMPUTER RESEARCH. 5. 11-19.