CS230

Classifying Arabic Calligraphy Styles

Hamza El Boudali and Mohamed Osman
Department of Computer Science
Stanford University
hamza@stanford.edu and laalays@stanford.edu

Abstract

In this paper, we use modern deep learning techniques from computer vision to achieve state-of-the-art results on the task of Arabic Calligraphy Style classification. Arabic calligraphy is the dominant art expression of the Islamic faith and of the Arabic language - one of the most widely spoken languages in the world. While there exists literature on classifying Arabic Calligraphy using traditional machine methods, to our knowledge, we are one of the first to use deep convolutional neural networks on this task on a significant dataset. In particular, we take ResNet18 as a baseline, and propose using a Vision Transformer (ViT) model and an inception model for better results. We also construct a new dataset containing over 2000 Arabic calligraphy samples from 9 various styles and different color schemes, which will be useful for the Arabic Calligraphy+ML community. Our ViT model achieves over 93% accuracy on each class and an F_1 score of 98% which outperforms existing methods in the literature.

1 Introduction

Arabic is one of the most popular languages in the world (over 300 million speakers) and Arabic calligraphy is the predominant art form of the Islamic faith. It is a beautiful art form with great religious, cultural, and historical significance. We aim to contribute to the technical problem of better understanding Arabic calligraphy through machine learning. There are various creative styles of Arabic calligraphy (e.g. Kufi, Farisi, Diwani, etc.), such that the same Arabic letter can be written in various forms, designs, and orientations, making it difficult for even a native Arabic speaker to read some calligraphy pieces.

An ML model that can quickly and accurately classify Arabic calligraphy would be useful for enthusiasts, artists, and historians interested in analyzing old manuscripts from Islamic history. It can be also be the first step in a larger natural language system that seeks to understand Arabic calligraphy, such as an optical character recognition system. There is some existing literature on using ML to classify Arabic calligraphy, however, to our knowledge, this is the first attempt to use deep CNNs and vision transformers for such a task. There is also work done to collect and curate online Arabic

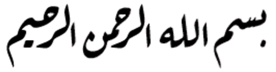
CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

calligraphy samples. We rely on this work in this paper. Identifying Arabic calligraphy styles is a classic CV classification task. The ML architectures used in this paper takes in the images of Arabic calligraphy and classifies the style used in the image into one of nine most popular calligraphy styles. For reference, here is the word in the above image in various calligraphy styles:

Naskh

Thuluth

Dewani


Reqaa

Kufi

2 Related work

There is much literature on using machine learning for recognizing, classifying, and understanding Arabic Calligraphy. Many datasets have been collected through various methods (such as, but not limited to: camera images, manuscripts and books, web scraping) and published in papers for the ML community to use. In one such paper, the researchers provide a dataset called AC which consists of calligraphy samples from 9 different styles [1]. They also use traditional ML approaches to tackle the problem of style classification, including Support Vector Machine (SVM) and K-Nearest-Neighbor (KNN). However there is no use of modern deep learning techniques such as convolutional neural networks (CNN) for classifying styles. We seek to fill this research gap.

Similarly, Ezz et al. attempted to classify just two styles: Reqaa and Naskh using Gaussian Naive Bayes, Decision Tree, Random Forest, and K-Nearest Neighbor classifiers. Ajeghrir et al. [3] utilized CNNs for Arabic calligraphy style recognition [3], however the data they used was scarce and they were also were only able to classify between two styles: Reqaa and Farsi.

An important paper seeking to fix the problem of lack of data is the Calliar paper [2], which provides an online Arabic calligraphy dataset. This dataset was annotated for the stroke, letter, word, and sentence level prediction, which makes it an ideal dataset for an optical character recognition (OCR) system. There are many research papers which use modern deep learning for optical Arabic character recognition [3][4], and, although, as previously mentioned, we believe our work can be utilized towards solving this problem, OCR itself is a different task than image classification. As we will now show, we make use of the images in the Calliar dataset for our own purposes of style classification.

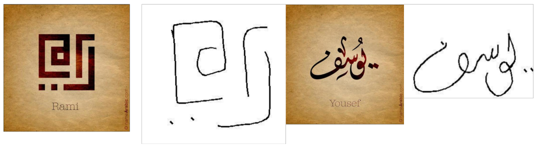
3 Dataset

We created a custom dataset that combines images from the AC dataset and Calliar dataset (both are cited above in the Related Works section). The AC dataset was already labelled, and has 1685 images (training: 1145, val: 270, test: 270) over 9 Classes: Thuluth, Farisi, Diwani, Kufi, Square Kufi, Rekaa, Naskh, Maghribi, Mohakek. These images were taken by a camera with 18 megapixels (5184 pixels wide by 3456 pixels high). There is variation in the source of the data, as it is drawn from books, manuscripts, the web, etc. as well as the phrase lengths.

Calliar on the other hand has 600 images (training: 400, val: 132, test: 100) and is not labelled by calligraphy style since it was annotated for the task of optical character recognition. We decided to manually annotate the images by getting the help of two experts. We wrote a simple UI script for them to easily label the data. From their labelling, we discovered that the majority of images in the Calliar dataset are of the thuluth style, and only two other styles, Farisi and Diwani, have above 50 images. To avoid introducing class imbalance issues, we made sure to only add 50 images from those three styles to the AC dataset: thuluth, farisi, and diwani. As we will discuss more in depth later, we try training and testing our model on just the AC data as well as on this custom dataset which contains AC and Calliar data.

We perform a data augmentation operation on the Calliar images by splitting them in half to double the size of that dataset. We were able to do this since each Calliar image consists of a black-and-white handwritten image and a corresponding colored, digitized image of the same calligraphy sample. We did this to increase the number of examples, but also so that our model can be trained and tested on both purely handwritten and and purely digitized calligraphy samples.

Examples images from Calliar dataset



This data collection and preprocessing will be useful for the Arabic Calligraphy-ML community in future research. Our dataset can be found in our github repo [6].

4 Methods

As a baseline model for classifying Arabic calligraphy, we chose to use ResNet18. ResNet stands for residual neural network, and it is a popular deep CNN that generally performs well on computer vision tasks. There are several common architectures of ResNet, such as ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. The difference between these is the number of layers, so our model has 18 layers.

In particular, we used Pytorch's torchvision implemention of the ResNet18 architecture, pretrained on ImageNet data. This implementation is already well tested, and given that our task is image classification, it made sense to take advantage of transfer learning from the massive ImageNet dataset which consists of over 14 million annotated images and hundreds of classes. We use the pretrained weights and then finetune our model on our custom dataset by changing the last fully-connected layer in the network to output over our 9 calligraphy style classes instead of the ImageNet classes. We use the classic optimization algorithm for the multi-class classification setting, stochastic gradient descent with momentum (learning rate = 0.001, momentum = 0.9). We also make use of a learning rate decay scheduler whereby the learning rate decreases by a factor of 0.7 every 7 epochs. We use the cross-entropy loss function as that is generally the most appropriate for image classification tasks.

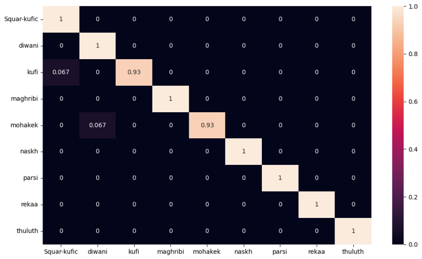
For our first proposed model, we use a Vision Transformer (ViT) model [5]. The motivation for this is that ViT has achieved state-of-the-art results on many similar image classification tasks, and we believed that it would perform very well on our task as well. Vision transformers make use of the attention mechanism (borrowed from transformer models in NLP settings) to focus on certain parts of the input data more than others. Finally, after implementing ViT, we decided to propose another model: inception. Inception, like ResNet, is a deep CNN, and we were interested in seeing how it would perform relative to the baseline and ViT on our experiments.

5 Experiments/Results/Discussion

We started our experimentation with ResNet18 model pretrained on ImageNet and fine tuned on AC dataset as a baseline. After carefully tuning in the hyper-parameters, the baseline model poorformed well on AC dataset. Our baseline model outperforms the results from the AC dataset paper which achieves 96% accuracy.

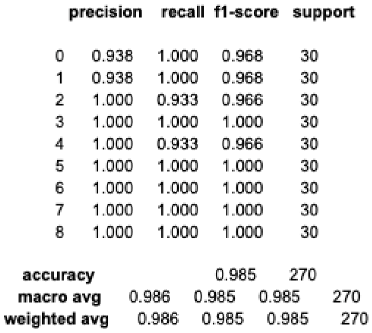
However, we realized the images from AC dataset have very high resolutions and does not contain handwritten calligraphy images. To test the baseline model performance against handwritten calligraphy and images with less resolution, we used our custom dataset (explained more in the dataset section) that contains handwritten calligraphy styles. The baseline model performed poorly on our custom dataset.

Baseline (ResNet18) performance on AC dataset



	Squar-kufic	diwani	kufi	maghribi	mohakek	naskh	parsi	rekaa	thuluth
Squar-kufic	1	0	0	0	0	0	0	0	0
diwani	0	1	0	0	0	0	0	0	0
kufi	0.067	0	0.93	0	0	0	0	0	0
maghribi	0	0	0	1	0	0	0	0	0
mohakek	0	0.067	0	0	0.93	0	0	0	0
naskh	0	0	0	0	0	1	0	0	0
parsi	0	0	0	0	0	0	1	0	0
rekaa	0	0	0	0	0	0	0	1	0
thuluth	0	0	0	0	0	0	0	0	1

Figure 1: Confusion matrix



```
              precision    recall  f1-score   support

           0      0.938     1.000     0.968        30
           1      0.938     1.000     0.968        30
           2      1.000     0.933     0.966        30
           3      1.000     1.000     1.000        30
           4      1.000     0.933     0.966        30
           5      1.000     1.000     1.000        30
           6      1.000     1.000     1.000        30
           7      1.000     1.000     1.000        30
           8      1.000     1.000     1.000        30

    accuracy                          0.985       270
   macro avg      0.986     0.985     0.985       270
weighted avg      0.986     0.985     0.985       270
```

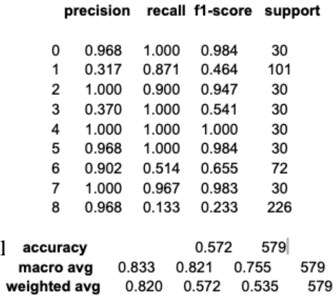
Figure 2: Classification Report

Baseline (ResNet18) performance on custom dataset

Diwani vs Thuluth

Figure 6: Diwani

Figure 7: Thul



```
              precision    recall  f1-score   support

           0      0.968     1.000     0.984        30
           1      0.317     0.871     0.464       101
           2      1.000     0.900     0.947        30
           3      0.370     1.000     0.541        30
           4      1.000     1.000     1.000        30
           5      0.968     1.000     0.984        30
           6      0.902     0.514     0.655        72
           7      1.000     0.967     0.983        30
           8      0.968     0.133     0.233       226

    accuracy                          0.572       579
   macro avg      0.833     0.821     0.755       579
weighted avg      0.820     0.572     0.535       579
```

Figure 3: Classification Report of baseline model performance on custom dataset

As the confusion matrix shows, the model cannot differentiate some calligraphy styles like $thuluth$ vs $maghribi$ and $diwani$ vs $parsi$. As expected, those styles are more similar and requires more sophisticated architecture to differentiate.
Because of its wide popularity in classification tasks, we started experimenting with Vision Transformer model as a potential proposed model. We used pre-trained weights from ImageNet and trained the weights in the last layer of the architecture. We also fine tuned the model hyper-parameters. We first tested the model on just the AC dataset in order to make sure the model has same or better performance of the baseline model on the AC dataset. The model achieved the same performance as the baseline model on the AC dataset. But unlike the baseline model, the vision transformer model performed well on the custom dataset with handwritten calligraphy styles. The huge improvement is due to the model's ability to differentiate very similar styles. Vision transformer is our first proposed model in this paper.

The vision transformer model has a high test accuracy and is our proposed model for this task. However, we experimented with other models to see how much they improve the performance. The most natural next model to try is Inception because of the models deep architecture which is helpful for differentiating similar styles. The inception model increased the accuracy score to 93% due to its ability to differentiate the similar style calligraphies.

Although we were afraid of overfitting due to the very high training accuracy for both the baseline and proposed models in the original experiments with just AC data, we saw that the validation accuracy was similarly high so there did not appear to be a problem with generalizing to unseen examples. This is when we decided to add the Calliar data to our test set to make sure that our models are generalizable.

ViT16 model performance on AC dataset

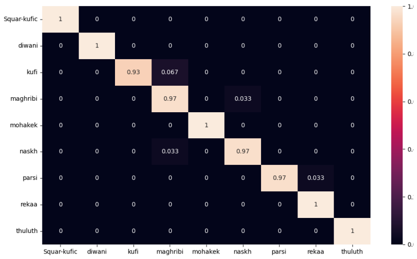
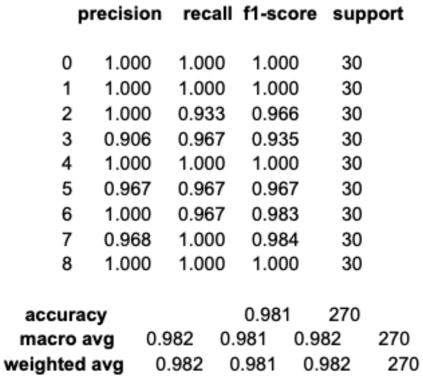


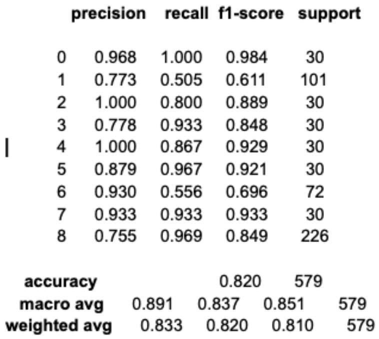
Figure 4: Confusion Matrix



	precision	recall	f1-score	support
0	1.000	1.000	1.000	30
1	1.000	1.000	1.000	30
2	1.000	0.933	0.966	30
3	0.906	0.967	0.935	30
4	1.000	1.000	1.000	30
5	0.967	0.967	0.967	30
6	1.000	0.967	0.983	30
7	0.968	1.000	0.984	30
8	1.000	1.000	1.000	30
accuracy			0.981	270
macro avg	0.982	0.981	0.982	270
weighted avg	0.982	0.981	0.982	270

Figure 5: Classification Report

ViT16 model performance on custom dataset



	precision	recall	f1-score	support
0	0.968	1.000	0.984	30
1	0.773	0.505	0.611	101
2	1.000	0.800	0.889	30
3	0.778	0.933	0.848	30
4	1.000	0.867	0.929	30
5	0.879	0.967	0.921	30
6	0.930	0.556	0.696	72
7	0.933	0.933	0.933	30
8	0.755	0.969	0.849	226
accuracy			0.820	579
macro avg	0.891	0.837	0.851	579
weighted avg	0.833	0.820	0.810	579

Figure 6: Classification Report

Inception model performance on custom dataset

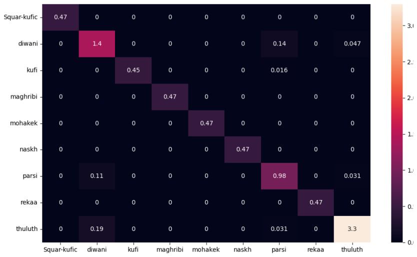
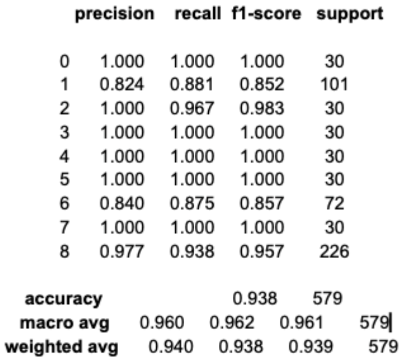


Figure 7: Confusion Matrix



	precision	recall	f1-score	support
0	1.000	1.000	1.000	30
1	0.824	0.881	0.852	101
2	1.000	0.967	0.983	30
3	1.000	1.000	1.000	30
4	1.000	1.000	1.000	30
5	1.000	1.000	1.000	30
6	0.840	0.875	0.857	72
7	1.000	1.000	1.000	30
8	0.977	0.938	0.957	226
accuracy			0.938	579
macro avg	0.960	0.962	0.961	579
weighted avg	0.940	0.938	0.939	579

Figure 8: Classification Report

6 Conclusion & Future Work

Our proposed ViT and Inception algorithms achieved the best performance, as expected. Although ResNet18 did considerably well as a baseline, the vision transformer achieved higher F_1 scores on average across the various classes. All deep learning models surpassed the traditional ML approaches from [1], which shows that modern deep learning techniques are indeed the state-of-the-art for Arabic calligraphy style classification.

For future work, we suggest using a bigger dataset with more examples and more variety by font, text size, and color. Arabic calligraphy is extremely diverse so the more comprehensive and representative a dataset is, the better our models will perform on real-world examples.

Furthermore, if we had more time we would have liked to spend it on evaluating our model's performance by producing saliency maps of some example images to see how the ResNet18 and ViT models are actually learning important features from the data. We also would have liked to analyze the AUC ROC curves for more qualitative interpretations of our results. Finally, other researchers aiming to build on our work may try other state-of-the-art image classification models, like CoCa, and compare it to the performance of our models. Deeper models like VGG are also an area of interest.

7 Contributions

Both team members contributed to reading research papers, implementing and evaluating the baseline and proposed models in Python and PyTorch, and writing this paper.

References

If you do not use one of these formats, each reference entry must include the following (preferably in this order): author(s), title, conference/journal, publisher, year. If you are using TeX, you can use any bibliography format which includes the items mentioned above.

[1] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8807829tag=1

[2] https://arxiv.org/abs/2010.11929

[3] https://mustapha-ajeghrir.me/assets/pdf/Rapport_arabic_calligraphy.pdf

[4] https://d-nb.info/1216415676/34

[5] https://arxiv.org/abs/2010.11929

[6] https://github.com/mohamedlaalays/CS230_project