

Make sense of Hieroglyphics

Team 6



Agenda

- Problem Statement
- Dataset
- Methods
- Experiments
- Results
- Discussion
- Conclusion
- Future Work
- References
- Appendix

Egyptian hieroglyphs have always been difficult to decipher

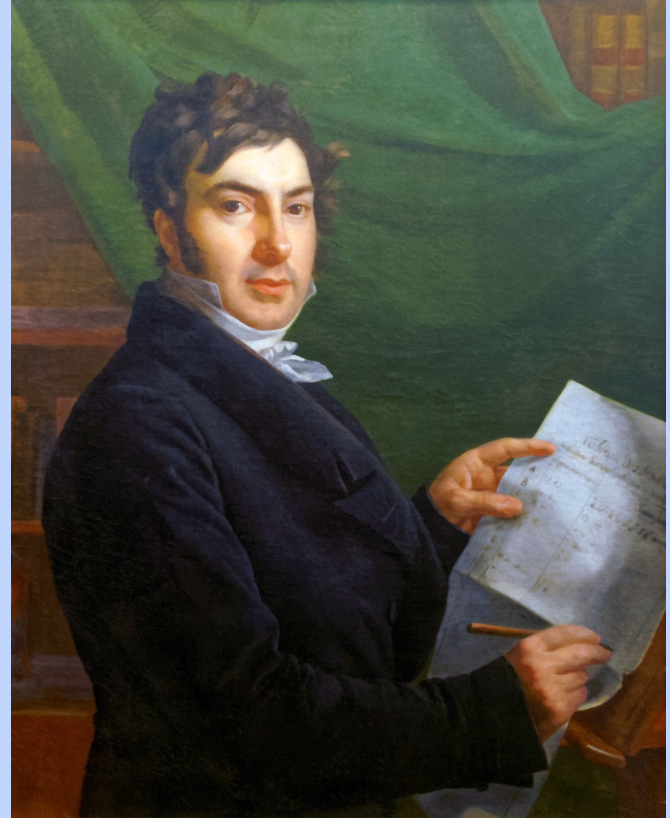
During the period of Medieval to Renaissance, the deciphering of Egyptian hieroglyphs has been slow, and some Islamic scholars have speculated that hieroglyphs may also be a kind of phonetic script.

In the nineteenth century, the deciphering of ancient Egyptian hieroglyphs made a breakthrough, thanks to the discovery of the Rosetta Stone. The Rosetta Stone is a stele written in three languages that express the same content. Through long-term research and comparison, Champollion has come to the conclusion that Egyptian hieroglyphs are phonetic scripts, and each symbol of a hieroglyph represents a letter.



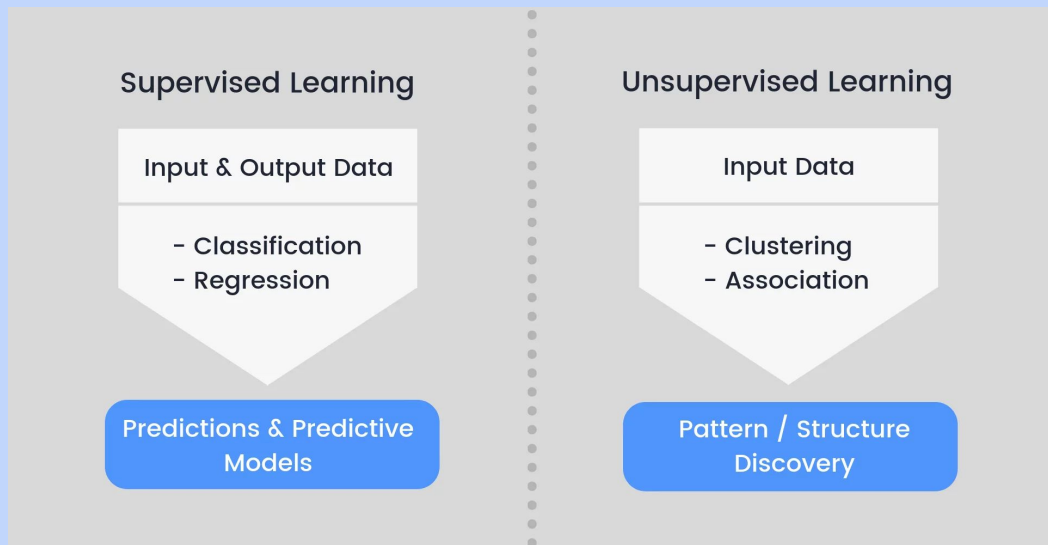
New technology applied to the study of hieroglyphics

When linguists study hieroglyphics, they need to consult a lot of data and make a lot of comparisons. For example, the famous French linguist Champollion spent ten years analyzing the arrangement and appearance of the letters of Egyptian hieroglyphs. However, the identification and classification of the huge numbers of Egyptian hieroglyphs by humans has caused a huge burden on linguists. But in modern times, new machine learning and image recognition technologies can help linguists easily complete these heavy tasks.



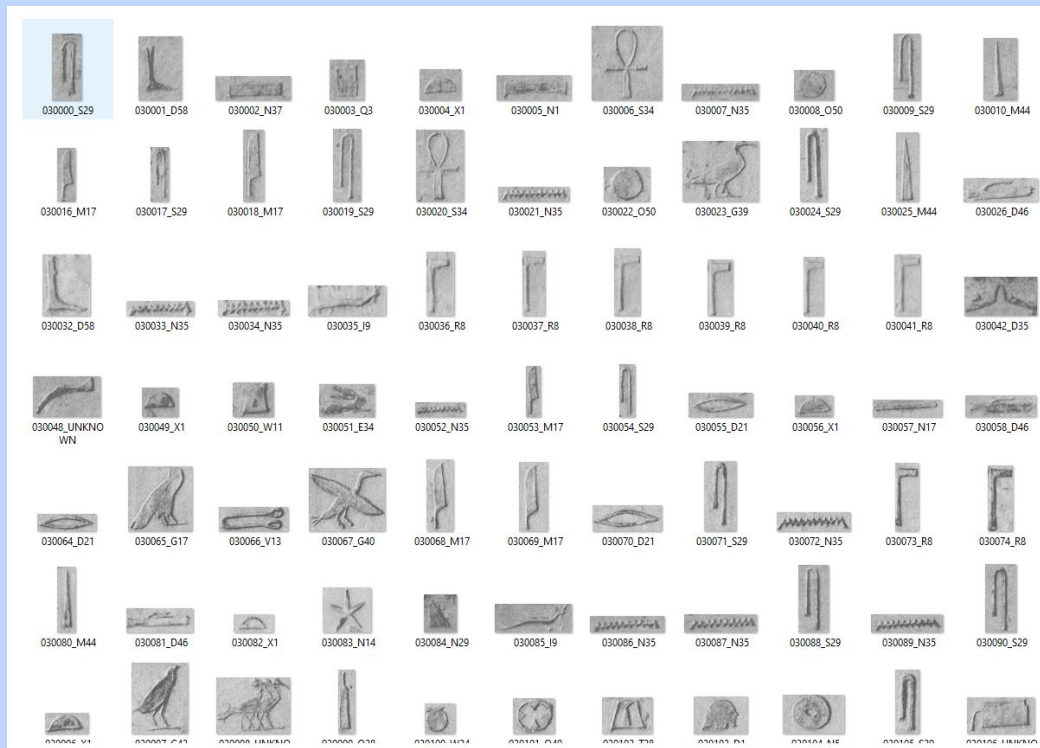
Machine learning for hieroglyphics

In order to help linguists save time and energy on the identification and classification of Egyptian hieroglyphic letters. Our group decided to use machine learning, including supervised learning and unsupervised learning to classify and produce statistical reports of hieroglyph letters.



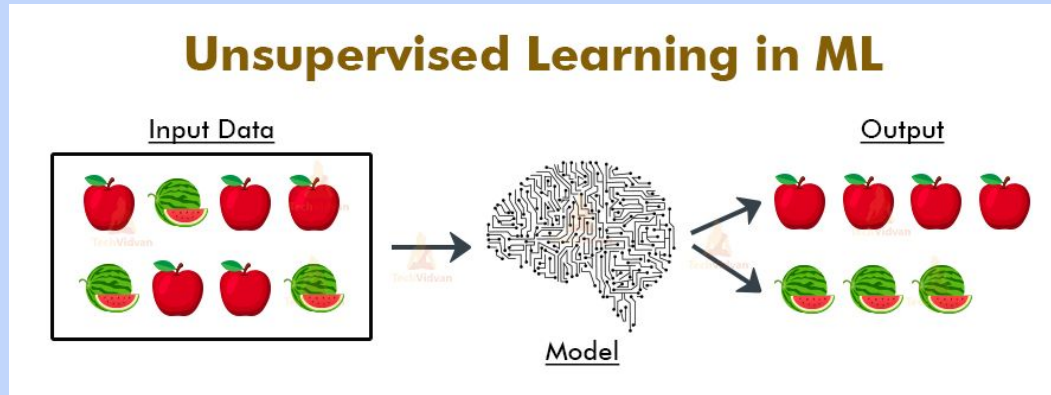
Our dataset

- Github
- <https://github.com/morrisfranken/glyphreader>
- More than 4000 images data with Gardiner Label
- Used in other research paper as well

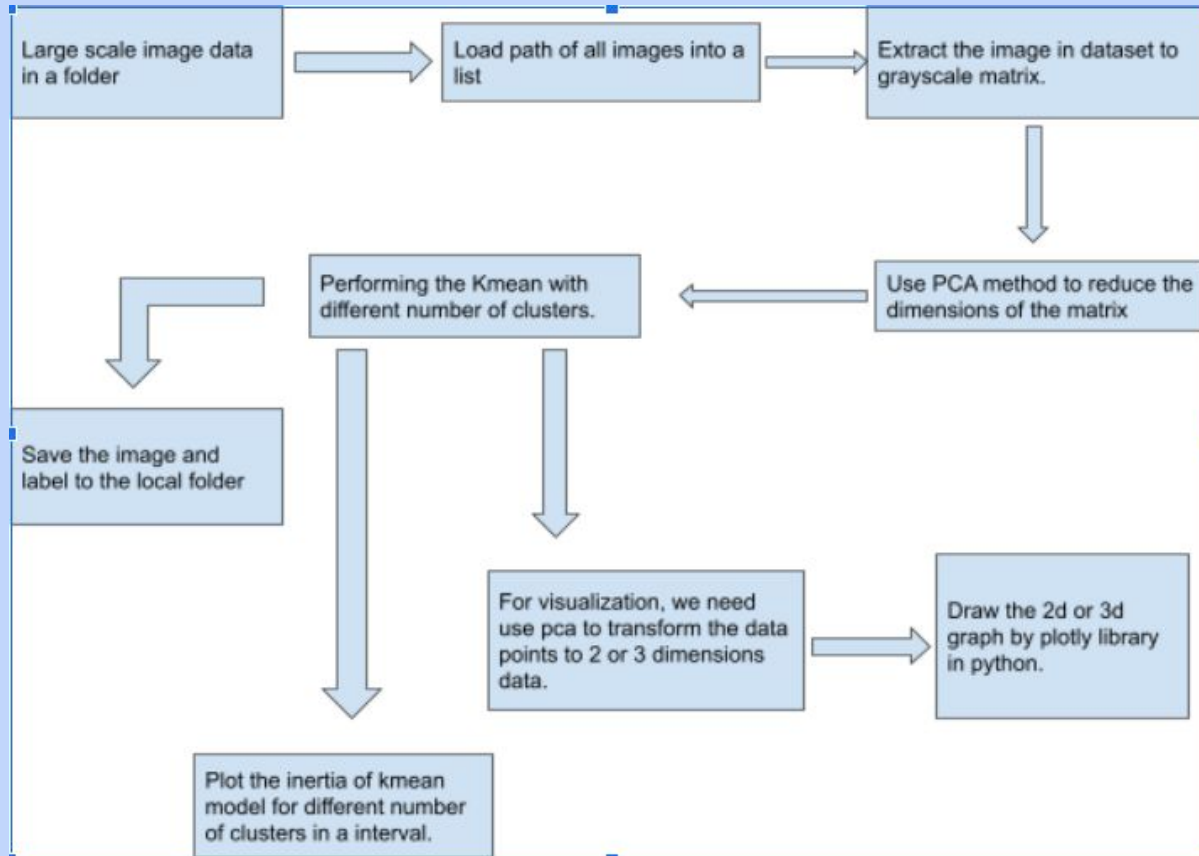


Unsupervised learning to cluster the letters

- aggregate and classify our image data
- does not require a training data set
- Only a sufficient amount of data is needed to build an unsupervised learning model



Road map of unsupervised model (K-Mean)

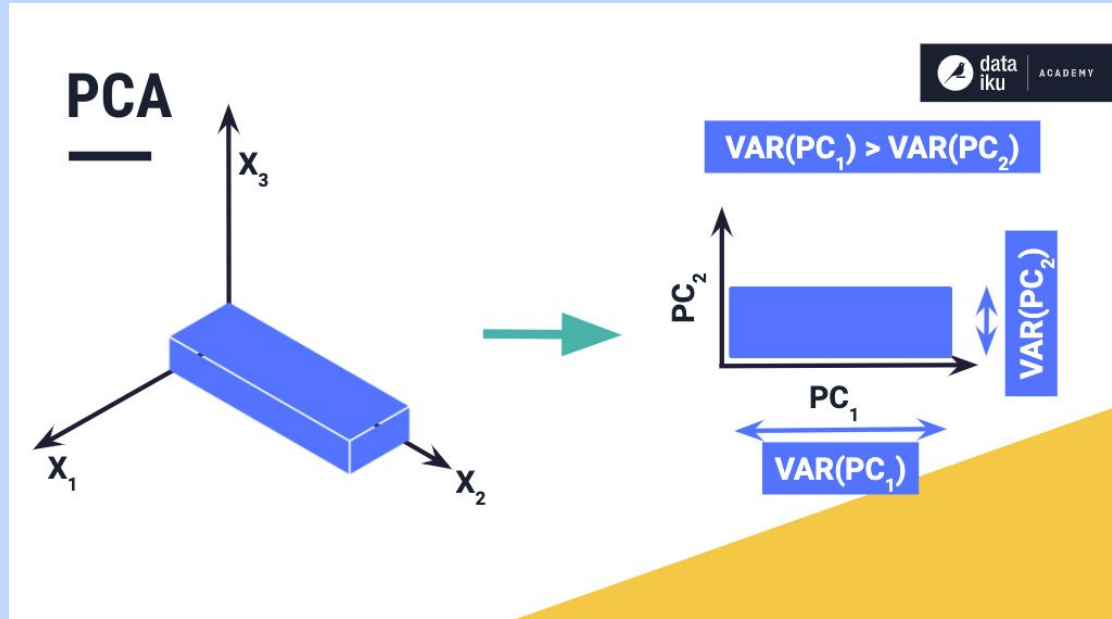


Methods: Feature Extraction and PCA

To extract the information in the picture, we need to convert the picture information into digital information for calculation.

The most popular way is that if I want to extract the features of each image, I need to convert the different pixels of each image into grayscale numbers. Then I need to convert each picture into a matrix of gray values.

But simply using a full-scale matrix for calculation and model building will result in very slow efficiency. I used the PCA method to reduce the dimensionality of my data, which will increase the speed of building a k-means model.

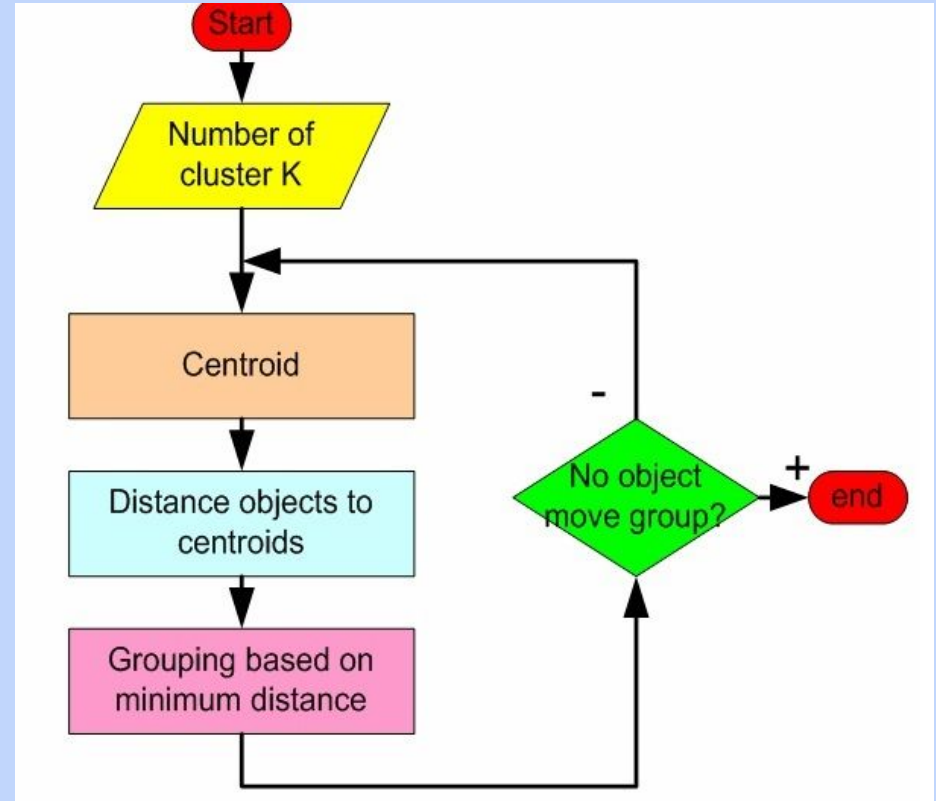


Methods: K-Means Clustering

We used the k-means method to build our model.

k-means clustering is a method of vector quantization that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

There are many parameters that can be set, but in order to better help customers to use, we set other parameters in advance. The user now only needs to set the number of clusters to change the kmean model. The clustered data will be automatically stored in the local folder.

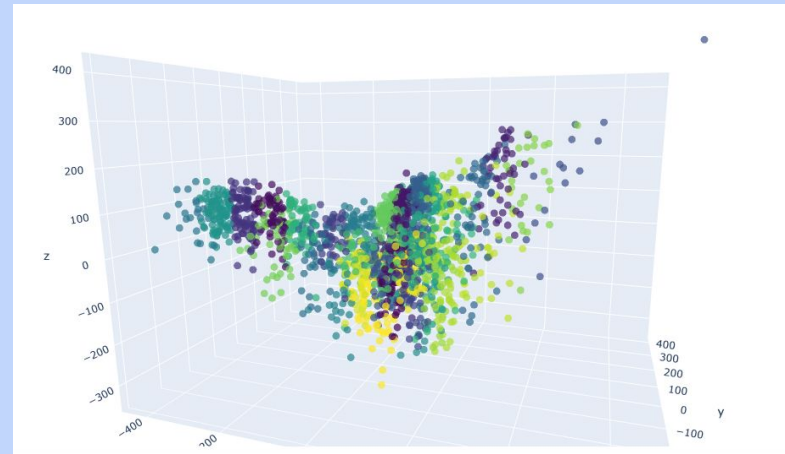
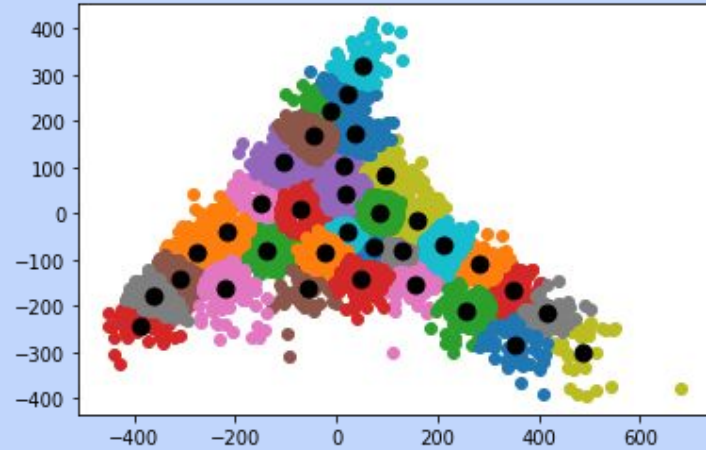


Visualization for validation

For unsupervised learning model, we cannot directly calculate the accuracy of our model. Because we do not have labels for each data to calculate. Therefore, the k-means can use visualization to validate the result.

When each cluster in the visualized image has a clear boundary, and all the data points of each cluster are huddled together, we can consider this k-means model to be relatively successful.

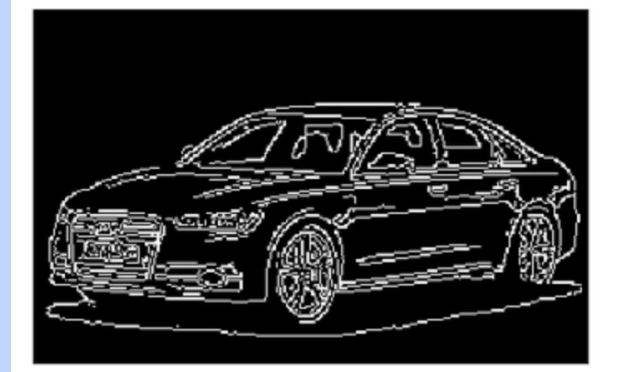
<https://colab.research.google.com/drive/1hD1VJM-2WBANVMXQSPew-iT7WFw4MHVv>



Methods: Feature Descriptor























We wish to cluster the image by its appearances so we used 3 different feature descriptor methods to collect the image's information. Feature descriptor works by representing only the most important information about the image.



















- HOG: This is done by extracting the gradient and orientation of the image.
- Canny: An edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.
- InceptionV3



Clustering methods: Kmeans














After combining three feature descriptor data into one file. We did Kmeans clustering and agg clustering models using feature descriptors data. The cluster number we finally chose is 27. And this is the information in each of the clusters.















	A	B	C	D	E	F	G	H	I	J
	CLUSTER	NUMBER OF PICTURES	TOP1 NAME	PERCENT	Symbol Display	Correspond Alphabet	TOP2 NAME	PERCENT	Symbol Display	Correspond Alphabet
1	0	123	D35	0.46			D36	0.293		
2	1	170	Z1	0.288			S29	0.129		
3	2	77	D58	0.455			Q1	0.064		B
4	3	81	G43	0.975		W,O	U7	0.024		mer
5	4	176	S29	0.994			P8	0.006		
6	5	189	M17	0.423		I	R8	0.296		
7	6	201	M17	0.721		I	T22	0.299		
8	7	122	G25	0.221			G39	0.164		
9	8	133	G17	0.624		M	G35	0.12		A
10	9	150	9	0.927		F	F18	0.033		
11	10	171	G35	0.605			G36	0.005		

10	174	G17	0.626		M	G35	0.086		
11	419	N55	0.807			UNKNOWN	0.041		
12	34	M23	0.97		su	M26	0.03		
13	136	V31	0.88		C,K	V30	0.04		
14	186	D21	0.77		R	D4	0.15		
15	108	E34	0.99			UNKNOWN	0.01		
16	104	V13	0.62			UNKNOWN	0.07		
17	434	N55	0.25		N	D21	0.09		R
18	133	M17	0.87		I	M18	0.03		

Clustering methods: Agglomerative Clustering

- Same feature descriptor as the previous k-means
- With elbow method, the best number of cluster is 27
- Compared with the k-means

	A	B	C	D	E	F	G	H
1	Cluster	Total Imag	Top 1	Percent	Image Display	Top 2	Percent	Image Display
2	0	864	N35	51.85		Unknown	5.32	
3	1	204	O50	48.04		W24	14.71	
4	2	200	S29	100				
5	3	230	G17	53.91		G1	14.35	
6	4	55	V28	65.45		S34	14.55	
7	5	54	D54	22.22		G7/Unknown	18.52	
8	6	206	Q3	33.98		Unknown	17.96	

	A	B	C	D	E	F	G	H
9	7	208	X1	85.1		E34	12.98	
10	8	91	G17	76.92		G35	8.76	
11	9	212	M17	81.13		U33/M18	4.25	
12	10	172	Unknown	14.53		O1	11.63	
13	11	63	D58	52.38		Q1	26.98	
14	12	288	M17	28.13		S29	22.57	
15	13	142	G43	100				
16	14	33	M23	100				

Methods: ResNet34 Classification

- A residual learning framework that efficiently trains networks that are deep and more complex than previous layers
- Use repetitive reformulating of layers with previous reference layer inputs instead of learning with new functions
- Implemented drop out
- SGD optimizer
- Used to classify data into 27 classes

```
res_mod = models.resnet34(pretrained=True)
```

```
num_fts = res_mod.fc.in_features  
res_mod.fc = nn.Linear(num_fts, 27)
```

```
res_mod
```

```
ResNet(  
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)  
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (relu): ReLU(inplace=True)  
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)  
  (layer1): Sequential(  
    (0): Bottleneck(  
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)  
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)  
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (relu): ReLU(inplace=True)  
      (downsample): Sequential(  
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)  
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      )  
    )  
  )  
)
```

Methods: AlexNet Classification

- Five convolutional layers and three fully connected layers
- Activation used is Rectified Linear Unit (ReLU) to help with vanishing gradient problems
- SGD optimizer
- Used to classify data into 27 classes

```
alexnet_mod = models.alexnet(pretrained=True)
alexnet_mod

AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
```

Experiments

Method	Evaluation
Feature Descriptor :K-Means Clustering	Best cluster number 27, among 100, 50, 30
Visualization	Draw the 2D and 3D graph of kmean model.
ResNet34	Learning rate = 0.001, momentum = 0.9, step_size = 7, 11 epochs
AlexNet	Learning rate = 0.001, momentum = 0.9, step_size = 7, 8 epochs

Results

Method	Estimated or calculated accuracy
Feature Extraction	K-means impurity:4.5
K-Means Clustering	The accuracy (for first three clusters) is 0.76
Visualization	Each cluster has a clear boundary and center
ResNet34	Accuracy of 69.4290%
AlexNet	Accuracy of 59.1028%

Discussion

- Feature Extraction
 - Limitations on the cluster's performance
- ResNet34 & AlexNet
 - Limitations to computing power
 - All data had to be manually placed in directories which was not optimal
 - Import errors and version types of R and Python packages
 - Black and white images versus colored

Discussion

When we designed these programs and functions, we realized that our customers are linguists or historians. They don't have much programming knowledge, so for them, we must try our best to make our programs easy to understand and operate. On this basis, we use jupyter notebook to run our program, which makes it easy for our customers to run our code. In addition, we have indexed and detailed comments for the program, which allows our customers to understand the each step of machine learning. In addition, we have recommended each parameter, so customers can easily customize their own machine learning model.

Table of contents

Import the packages for future use

Set the model, but don't suggest user to change it.

The input and output path, user can change it freely

Extract the feature to the list

Do the PCA to reduce the dimension of matrix, I suggest user set n_digits between 10-20

Do the Kmeans training, user can change the number_clusters freely

Save the image and the label to target location

If user want to see the inertia for a specific range of number_clusters, they can use this loop to generate a graph. User can change the range freely.

Draw the 2D graph of clustering, user can change number_clusters freely

draw the 2D graph below, user don't need to change anything

Preparing the data for 3D graph, user don't need to change anything.

plot the 3D graph, user don't need to change anything.

+ Code + Text

▼ Do the PCA to reduce the dimension of matrix, I suggest user set n_digits between 10-20

```
[ ] n_digits=10

pca = PCA(n_components=n_digits).fit_transform(featurelist)
```

▼ Do the Kmeans training, user can change the number_clusters freely

```
[ ] number_clusters=27

kmeans = KMeans(n_clusters=number_clusters, random_state=0)
kmeans.fit(pca)

KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
        n_clusters=27, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=0, tol=0.0001, verbose=0)
```

▼ Save the image and the label to target location

```
[ ] # Copy images renamed by cluster
    # Check if target dir exists
    try:
```

Conclusion

- Our work aims to processing thousands of hieroglyphic images to gain insights into how the images can be clustered and classified
- Our classification techniques had about 69% accuracy which can be improved with even more images
- Our clustering techniques explain how the data relates to the 27 class centers with impurities around 4.5
- By continuing to tune and process parameters, this can even gain accuracy and reduce impurity

Future Work

- Transferable to other image processing clustering and classification techniques with any dataset
- Future researchers can take our work and continue to tune parameters
- Researchers working with hieroglyphics can now input data to our notebooks to gain insights into how their photos break down with clustering and classification

References

Tensorflow team. “Build from Source : Tensorflow,” May 15, 2017. <https://www.tensorflow.org/install/source>.

Patel, Khush. “Architecture Comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet.” *Medium*, Towards Data Science, 8 Mar. 2020, towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d.

Real Python. “Three Ways of Storing and Accessing Lots of Images in Python.” *Real Python*, Real Python, 19 Mar. 2021, realpython.com/storing-images-in-python/.

“ResNet, AlexNet, VGGNet, Inception: Understanding Various Architectures of Convolutional Networks.” *CV*, 9 Aug. 2017, cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/.

“Scribes in Ancient Egypt.” *Ancient Egypt Online*, ancientegyptonline.co.uk/scribe/.

Shetty, Badreesh. “An in-Depth Guide to Supervised Machine Learning Classification.” *Built In*, builtin.com/data-science/supervised-machine-learning-classification.

Appendix

Kmean, PCA and visualization code in jupyter notebook: <https://github.com/yedkk/kmean-model>

ResNet34 and AlexNet Google Colab:

<https://colab.research.google.com/drive/1jqP8RmVEeXl0ATujQFmOWH8JQOIImE5w7?usp=sharing>

Any Question ?

